DIVISION OF FLEXIBLE LEARNING

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

IIIT Hyderabad

Master of Science in Data Science

(Online Mode)

Curriculum and Syllabus

5th Jul 2025

[1] Maintained by **Academic Programme Coordinator,** Division for Flexible Learning, IIIT Hyderabad

[2] Send comments to **associate-dean-dfl@iiit.ac.in**

TABLE OF CONTENTS

I Introduction

This document outlines the design of the Master of Science in Data Science Online programme. It is designed to comply with the UGC norms for online courses.

II Overall Structure of MS Data Science Programme

The programme consists of 8 terms. Each term is designed to be done in 12 weeks. A student aiming to complete the programme in 2 years is expected to take two courses in a term. Each course is worth 5 credits. The student is expected to devote 12.5 hours per week towards a course. These details are summarised below:

- **Term:** A term is 12 weeks (11 weeks of teaching + 1- week exam preparation plus exam) with a 1- week break between terms.
- **Course:** Something that is designed to run for 1 term
- **Student effort per course**: 12.5 hours per week x 12 weeks = 150 hours
- **1 Credit** = 30 student hours
- **No. credits per course** = 5
- **No. courses per term** = 2
- **No. terms per year** = 4
- **Total Credits per programme** = 80

III Weekly Student Effort

The following table lists the weekly effort expected of a student per course in this programme.

No.	Activity	Student Effort /week (h)	Graded	Remarks
1	Watch e-content videos (60min)	1.0	No	Self-study
2	Practice Quizzes	0.5	No	Self-study
3	Study Reading Material	3.0	No	Self-study
4	Attend Live Session	1.0	No	Live (Ref. UGC)
5	Practical / Online Lab Session	2.0	Yes	
6	Interact on forum	1.0	No	Asynchronous
7	Complete assignment	4.0	Yes	Self-study
Hrs/week		12.5		
No. weeks	11			
Total Hrs		137.5		
7	Prepare for the final exam	9.5	No	
8	Attend Exam Prep Session (live)	1.0	No	Live
9	Write Final Exam	2.0	Yes	Live, proctored
	Total Hours per course	150.0		

Note: Courses without Lab practicals, the time (hours/wk) will be adjusted towards self–study.

IV Core and Elective Courses along with Practical Component

The following table lists the courses for the programme:

No.	Course	Type	Online Lab
1.	Math for Data Science I: Basics	Core	-
2.	Systems, Computation and Programs	Core	✓
3.	Data Structures and Algorithms	Core	✓
4.	Computer Systems	Core	✓
5.	Database Management Systems	Core	✓
6.	Software Systems	Core	✓
7.	Algorithm Analysis and Design	Core	-
8.	Statistical Methods in AI	Data Science Core	✓
9.	Data Analytics	Data Science Core	✓
10.	Maths of Data Science II: Advanced Topics	Data Science Core	-
11.	Optimization Methods	Data Science Core	✓
12.	Computer Vision	Data Science Elective	✓
13.	Reinforcement Learning	Data Science Elective	✓
14.	Generative Models	Data Science Elective	✓
15.	Spatial Informatics	Data Science Elective	✓
16.	Introduction to NLP	Data Science Elective	✓
17.	Algorithms for Data Science	Data Science Elective	-
18.	Speech Signal Processing	Data Science Elective	✓
19.	Data Visualization	Data Science Elective	✓
20.	Human-Computer Interaction for Data Science	Data Science Elective	✓

All the lab practical and graded assignment components will be done either on the learner's computers or online. Submissions will be made to an online portal or using GitHub classroom. For lab practicals, we have a plethora of tools and platforms: Virtual Labs (https://vlab.co.in), GitHub classroom, AutoLab, Jupyter Notebooks, online coding and evaluation websites like Codechef, and many other open-source programming languages (Python, Javascript) and editing tools like VSCode. Most of the Virtual Lab experiments are highly interactive and are already being used in many colleges to conduct practicals. Tools and platforms will be course specific.

(IIIT Hyderabad is a participating institute of the Virtual lab project funded by Ministry of Education, Govt. of India since 2009.)

V Detailed Syllabus along with Practical Components

1. Math for Data Science I: Basics

A foundational course covering basic mathematics required for computer science.

Unit 1: Sets, Relations, Functions, Permutations and Combinations, Basic logical expressions.

Unit 2: Graph Theory including basic concepts like degree, path, cycles etc, Graph representations, properties, and special types of graphs.

Unit 3: Basics of Probability Theory and Applications Introduction to probability concepts, random variables, and their applications in computing.

Unit 4: Basics of Linear Algebra, solving linear systems of equations, Vector Spaces, Basis, change of basis, linear transformations.

Course Outcomes

CO1: Understand the basic terminology of Sets, Relations, Functions & Graphs.

CO2: Ability to count the size of sets using permutations and combinations.

CO3: Model uncertain experiments using probability and compute probabilities of events.

CO4: Solve linear equations using matrix transformations. Understand the concept of Vector Spaces.

CO5: Apply linear algebraic concepts to real-world problems.

2. Systems, Computation and Programs

An introductory course focusing on the principles of computation and programming. The course looks at programming from the point of view of data types, control structures and namespaces. The student will learn fundamental principles of programming, and algorithmic problem-solving using iteration and recursion. The programming used will be Python. In addition, the student will gain practical skills in software development through version control and unit testing.

Unit 1: Basic notion of systems and computation. Review of partial functions sets and relations. Version control. Variables, Data Types, and Basic Expressions Introduction to variables, data types, and expressions in programming languages.

Unit 2: Control Structures and Functions Exploration of control flow mechanisms (if statements, loops). Iterative problem-solving. Functions and procedural control. Environments and stores. Closures.

Unit 3: Recursion and Memory Management. Understanding recursion, stack memory, and dynamic memory allocation.

Unit 4: Namespaces, Modules, packages, Object-oriented programming, Inheritance. Interaction: Events and handlers, files and I/O. Web programming basics.

Course Outcomes

CO-1: Explain the syntax of programming language constructs and their semantics and describe a program structure and its execution model.

CO-2: Choose appropriate primitive data types and design new composite data types to model the relevant data in each computation problem and also discover the algorithmic logic required to solve well-defined computational problems.

CO-3: Compare and contrast the performance of different algorithmic approaches for simple computational problems (iterative vs. recursive solutions).

CO-4: Apply object oriented programming concepts to modularize programs and build a large-scale application.

CO-5: Building a web application that can serve a large number of users.

Online Labs

There will be weekly programming assignments in Python programming language. These will need to be submitted by students using the GitHub classroom. The programs will be mostly autograded and feedback will be given to students.

No.	Experiment Title	Short Description	Software Requirements
1	Version control with Git	Install git. Manage versioning of files using git commands: init, clone, commit, pull, push and merge. Report should include screenshots of the commands and output.	Github classroom (online).
2	VS Code and Python Installation	Install Python and VS Code. Open, edit and save files. Run simple Python programs. Report should contain steps you followed to install Python and VS code locally on your system.	Python, VS Code
3	Debugging Simple Programs in the VSCode environment	Write a simple (straight line) program and debug it. Report should include screenshots of each step of the debugger.	VS Code (installed or online)
4	Conditionals and Iteration in Python	Compute the nth Hemachandra-Fibonacci number, compute the nth prime. Report should include code and outputs.	Python (installed or online)
5	Functions in Python	Compute the conversion of fahrenheit->celsius, celsius->kelvin and fahrenheit->kelvin. The last function should use the composition of the previous two functions. Compute Factorial using recursion and iteration. Compute the number of vowels in a string.	Python (installed or online)
6	Higher-Order Functions and Closures	Define the following higher order functions: compose, iterate, derivative and integral. Report should include code listings and sample outputs besides something about the design of your program.	Python (installed or online)
7	Exception mechanism in Python	Build a basic command-line calculator that uses try, except, raise and finally to handle errors and exceptions.	Python (installed or online)
8	Object-Oriented Programming in Python	Design and implement Library Management System where users can add, borrow and return books. (Subtasks will explore the concepts of attributes, methods, encapsulation, inheritance, etc.)	Python (installed or online)
9	Aggregate Data types: Lists and Dictionaries in Python	Build a program to manage a group of students, their subjects, and their scores. They will use lists to store multiple data items and dictionaries to map keys (such as student names or subjects) to values (such as scores).	Python (installed or online)
10	Files and I/O in Python	Develop a program that allows users to manage a To-Do list by adding tasks, viewing the list, and marking tasks as completed. The tasks are stored in a text file, and the program interacts with the file to read, write, and update the data.	Python (installed or online)
11	Numerical Computing in Python using Numpy, Matplotlib, and Panda's	Work with the given dataset related to stock prices. Manipulate and analyze the data using NumPy and Pandas and create visualizations using Matplotlib. Compute mean, variance and coefficient of correlation between two different data sets.	Python (installed or online) Numpy, Matplotlib and Pandas

	libraries		libraries
12	Web Data extraction using Python: BeautifulSoup package	Create a Python program that scrapes data from IMDb's Top 250 Movies page. Extract information such as movie titles, release years, ratings, and directors. Finally, store the extracted data in a structured format.	Python (installed or online) BeautifulSoup

3. Data Structures and Algorithms

This course reviews basic data structures and algorithms, focusing on problem-solving techniques and their applications in computing.

Unit 1: Introduction to fundamental data structures (arrays, linked lists, stacks).

Unit 2: Introduction to basic algorithms (sorting and searching).

Unit 3: Algorithms on Trees and Graphs Examination of tree and graph data structures, and algorithms for traversal and manipulation.

Unit 4: Advanced Data Structures and Algorithms Study of advanced data structures (heaps, hash tables) and their associated algorithms.

Course outcomes

CO-1: Understanding fundamental and advanced Data Structures including linked lists, trees, binary search trees, AVL trees, stacks, queues, heaps, hash-table, tries and suffix trees.

CO-2: Ability to program data structures and use them in implementations of abstract data types.

CO-3: Ability to devise novel solutions to small-scale programming challenges involving data structures and recursion.

CO-4: Understand basic algorithms including recursion, searching, hashing, dynamic programming, and traversal.

CO-5: Understanding basic algorithmic complexity. Ability to perform simple inductive proofs and proofs by contradiction and reason for program correctness and invariants.

CO-6: Given a real-world problem, one can sensibly select appropriate data structures and algorithms for solving the problem and be able to implement the solution.

Online Labs

There will be weekly programming practical assignments which will involve implementing algorithms discussed in class. The practical assignments will need to be done in the Python programming language. These will need to be submitted by students using the GitHub classroom. The programs will be mostly autograded and feedback will be given to students.

Lab	Experiment Title	Short Description	Software Requirements
1	Arrays	Basic operations like insertion, deletion, search on lists of numbers	Python, Github
2	Strings	Basic operations like string reversal, palindrome checking, substring search	Python, Github
3	Stacks and queues	Usings arrays to create and maintain data in linear data structures	Python, Github
4	Linked lists	Using pointers to create and maintain data in non-linear data structures	Python, Github
5	Range query	Performing operations like prefix sums, mean/median/mode/majority	Python, Github
6	Suffix trees	Performing operations like autocomplete by implementing tries	Python, Github
7	Priority queues	Min-heaps and max-heaps (binary heaps, binomial heaps, Fibonacci heaps)	Python, Github
8	Graphs	Maintaining graphs using adjacency matrix, adjacency list, incidence list	Python, Github
9	Tree traversal	In-order, pre-order, post-order traversals of binary trees, binary search trees	Python, Github
10	Graph traversal	Depth-First Search (DFS) and Breadth-First Search (BFS) graph traversals	Python, Github
11	Sorting	Bubble sort, selection sort, insertion sort, merge sort, radix sort, bucket sort	Python, Github
12	Hashing	Maintaining hash table with collision resolution (chaining, open addressing)	Python, Github

4. Computer Systems

A comprehensive course on computer organization, instruction set architecture, processor design, and memory management, including operating systems.

Unit 1: Basic Computer Organization and Architecture Overview of computer architecture, including components and their interactions.

Unit 2: Introduction to Operating Systems Overview of operating system concepts, including processes, threads, and resource management.

Unit 3: Process Control and Management Understanding process control, scheduling, and inter-process communication.

Unit 4: Memory Management Techniques Exploration of memory management, including paging, segmentation, and caching.

Unit 5: Computer Networking principles, Layered architecture of network protocols.

Course outcomes

After completion of this course successfully, the students will be able to:

CO-1: Describe the basic functionality of an operating system.

CO-2: Clearly explain the system call interface, its design and implementation.

CO-3: Build systems akin to bash shell, file server etc. using system calls. Describe the basics of process control and management.

CO-4: Describe the principles of virtual memory management. Analyse various memory management schemes for process isolation and physical memory utilization across multiple processes.

CO-5: Understand different layers of protocols in computer networks.

Online Labs

Lab	Experiment Title	Short Description	Software Requirements
1	Introduction to shell	The objective of the experiment will be to get the learners introduced to shell. The learners will be expected to execute some linux commands	Any Unix based OS
2	Shell command implementation	Implementing of a couple of basic shell commands using C programming language.	Unix based OS ,+ gcc, + any editor that supports C
3	Fork and Execute	Development of a program that can emulate the working of commands with a pipe operator in linux shell using fork() and exec() calls	Unix based OS + gcc and any editor that supports C
4	Building a scheduler	Implement a simple Round robin and FIFO scheduler	Python
5	Manage the memory	Program to check the memory of different processes	Python
8	Introduction to multi-threading	Development of a program that makes use of multiple threads to process elements in an large array	C
7	Readers Writers	Implement the readers writers problem using threads	C
8	Error detection code	Implement parity bit-based error detection code	Python
9	Shortest path algorithm	Implement Dijkstra's algorithm to compute the shortest path between a pair of nodes in a network	Python
10	Sliding window protocol	Implement TCP sliding window protocol	Python
11	Client server connection	Implement Echo server and client	Python
12	Subnetting	Implement a program to calculate subnet address for a given IP and subnet mask	Python

5. Database Management Systems

This course teaches students how to model, design, and implement databases, including relational databases and SQL.

Unit 1: Data, Database, Database System

Unit 2: Data models, Conceptual Data Modeling, ER Models

Unit 3: Relational Data Model, Relational Algebra, Tuple Relational Calculus

Unit 4: SQL, Constraints, Triggers, Database Connectivity, Applications

Unit 5: Normalization, Relational Database Design

Course outcomes

CO-1: Data Requirements for Applications Understanding the data needs for different applications and systems.

CO-2: Conceptual Modelling and ER Diagrams Introduction to conceptual data modelling and the use of entity-relationship diagrams.

CO-3: Relational Data Model and Integrity Constraints Overview of the relational data model and the importance of integrity constraints.

CO-4: Mapping Conceptual Models to Relational Databases Techniques for transforming ER diagrams into relational schema.

CO-5: Querying and Manipulating Databases with SQL Introduction to SQL for database querying and manipulation.

CO-6: Application Development for Database Access Overview of application development techniques for accessing and interacting with databases.

Online Labs

Lab	Experiment Title	Short Description	Software Requirements
1	Data Definition in SQL	Defining tables, fields, and constraints in SQL.	MySQL
2	Basic SQL Queries	Writing SELECT queries to retrieve data from single and multiple tables.	MySQL
3	ER Diagram to Relational Schema Mapping	Converting an ER diagram into relational schema.	MySQL, draw.io
4	Inserting and Updating Data	Using INSERT, UPDATE, and DELETE operations on relational tables.	MySQL
5	Joins and Subqueries in SQL	Performing INNER JOIN, LEFT JOIN, RIGHT JOIN, and writing subqueries.	MySQL
6	Aggregation and Grouping	Using aggregate functions like COUNT, SUM, AVG, and GROUP BY in SQL queries.	MySQL
7	Implementing Integrity Constraints	Applying primary keys, foreign keys, and unique constraints on tables.	MySQL
8	Writing Triggers in SQL	Creating and testing BEFORE and AFTER triggers for data manipulation.	MySQL
9	Normalization of a Database	Applying normalization rules (1NF, 2NF, 3NF) to optimize a given database design.	MySQL
10	Database Connectivity in Applications	Connecting a database to an application using Python.	MySQL, Python
11	Indexing and Query Optimization	Creating indexes and analyzing query performance for optimization.	MySQL
12	Backup and Recovery in SQL	Implementing backup strategies and recovery techniques for databases.	MySQL

6. Software Systems

An overview of software systems, including Linux, shell scripting, web programming, and basic networking concepts.

Unit 1: Software Development Life Cycle and importance of architecture and design in the life cycle, Process models; Modelling using UML.

Unit 2: Anti-patterns; Metrics and Measurement; Reverse Engineering and Refactoring.

Unit 3: Design Principles and Classification of Patterns,
Structural patterns: Adapter, Composite, Façade, Proxy, Decorator.
Behavioral patterns: Iterator, Observer, Mediator, Command, Memento, State,
Strategy, Chain of Responsibility
Creational patterns: Abstract Factory, Builder, Singleton, Factory Method

Unit 4: Software architecture and Architectural business cycle; Quality attributes and Tactics for achieving attributes; Architectural styles and Techniques; Designing Architectures, Case studies.

Course outcomes

CO-1: Demonstrate familiarity with various process models, design patterns, architecture patterns and the characteristics of good software architectures

CO-2 Apply principles of user interface design, and sub-system design and analyses the designs for good Software Engineering principles

CO-3: Demonstrate the use of tools to quantitatively measure and refactor existing software systems

CO-4: Compare design trade-offs between different patterns and/or different implementations of the same pattern

CO-5: Design the major components and user interface for a small-scale software system using modelling approaches such as UML class diagrams, and sequence diagrams

CO-6: Critique the quality of a software design and use product quality metrics to assess the quality of delivered software

Online Labs

Lab	Experiment Title	Short Description	Software Requirements
1	Advance Git Commands	Introduce version control with advance git commands - diff, reset, branch, checkout, switch, merge, fetch, pull.	Git 2.46 GUI Client and a GitHub account
2	BASH Commands	Basic BASH Commands - find, locate, head, tail, grep, awk, bc, wc, read, export etc.	BASH Shell in LINUX OS
3	BASH Control Statements and Loops	Variables, Arrays, Loops - FOR-WHILE-DO-WHILE, CASE, SWITCH, IF-ELSE, Some program examples.	BASH Shell in LINUX OS
4	SQL Commands	CRUD Operations, Sub Queries, UNION, EXISTS, INTERSECT, JOINs.	MySQL database
5	SQL Programming	CTE, Cursors, Stored Procedures.	MySQL database
6	NoSQL Commands	CRUD Operations - Inserts, Updates, Deletes.	MongoDB
7	HTML, CSS	HTML tags, CSS - Basics, Classes, Selectors, Combinators.	Browser with internet connectivity
8	DOM & Browser Events	DOM Events, Browser Events, Mouse Events.	Browser with internet connectivity
9	ReactJS + ExpressJS + NodeJS Setup	Configure Basic MERN Stack Front-End with NodeJS backend setup.	NodeJS, ExpressJS, ReactJS
10	ReactJS with MongoDB (MERN)	Sample MERN Stack website with Mongo Backend.	NodeJS, ExpressJS, ReactJS, MongoDB
11	Basic Python Commands	Basic Commands, Control Statements, Loops and Sample programs.	Python
12	REST API and Flask Setup	REST API Methods, API using Flask and Data access using SQLAlchemy.	Flask, SQLAlchemy

7. Algorithm Analysis and Design

This course focuses on various algorithmic techniques, including graph algorithms, greedy algorithms, and dynamic programming.

Unit 1: Basic Graph Algorithms Examination of algorithms for graph traversal (BFS, DFS) and shortest path problems.

Unit 2: Greedy Algorithms Understanding the greedy approach to problem-solving and examples of its applications.

Unit 3: Divide and Conquer Techniques Introduction to the divide and conquer strategy and its applications in algorithms.

Unit 4: Dynamic Programming and Network Flows Study of dynamic programming techniques and network flow problems.

Unit 5: Computational Complexity and NP Problems Overview of computational complexity theory and classification of problems.

Course outcomes

CO-1: Demonstrate the ability to fully understand the analysis of various known algorithms.

CO-2: Identify problems where various algorithm design paradigms can be applied.

CO-3: Understand the notions of computational intractability and learn how to cope with hardness.

CO-4: Understand the notion of approximation and randomised algorithms. If time permits, introduction to quantum algorithms.

8. Statistical Methods in AI

A course covering statistical techniques and their applications in machine learning, including basic algorithms and advanced models.

Unit 1: Representation and Challenges in Machine Learning Overview of machine learning representation and common challenges in the field.

Unit 2: Basic Machine Learning Algorithms Introduction to fundamental machine learning algorithms such as regression and classification.

Unit 3: Popular Algorithms and Model Selection Exploration of popular machine learning algorithms and techniques for model selection.

Unit 4: Neural Network Learning and Deep Learning Concepts Introduction to neural networks and foundational concepts in deep learning.

Course Outcomes

CO-1: Data processing: process raw data and convert it into a machine-exploitable format

CO-2: Problem formulation: formulate a practical problem as a machine learning problem (classification, clustering etc.)

CO-3: Classical algorithms: In-depth investigation of theory and practice of classical algorithms in supervised and unsupervised learning (e.g. SVM, Kmeans, decision trees).

CO-4 Deep Learning: Introduction to theory, practice of deep learning and recent advances

CO-5 System building: design practical systems incorporating basic machine learning

Online Labs

Lab	Experiment Title	Short Description	Software Requirements
1	Data Preprocessing and Visualization	Explore techniques for handling missing data, normalization, and visualizing datasets.	Python, Pandas, Matplotlib, Seaborn
2	Linear Regression and Gradient Descent	Implement Linear Regression and Gradient Descent for a regression task.	Python, NumPy, Matplotlib
3	Logistic Regression and Binary Classification	Apply Logistic Regression to solve a binary classification problem.	Python, NumPy, Scikit-learn
4	K-Nearest Neighbors (KNN) Classifier	Implement KNN algorithm for classification tasks and analyze its performance.	Python, Scikit-learn
5	Support Vector Machines (SVM)	Train a Support Vector Machine on a classification dataset and tune hyperparameters.	Python, Scikit-learn
6	Decision Trees and Random Forests	Build Decision Trees and Random Forests for classification and regression problems.	Python, Scikit-learn
7	K-Means Clustering	Implement the K-Means algorithm and explore the impact of the number of clusters.	Python, Scikit-learn
8	Principal Component Analysis (PCA)	Use PCA for dimensionality reduction and visualize transformed data.	Python, Scikit-learn, Matplotlib
9	Naive Bayes Classifier	Implement Naive Bayes for text classification tasks like spam detection.	Python, Scikit-learn
10	Neural Networks (MLP)	Train a simple neural network using a Multi-Layer Perceptron (MLP) for classification.	Python, TensorFlow/PyTorch
11	Model Evaluation and Cross-Validation	Evaluate models using techniques like cross-validation, confusion matrix, and ROC curves.	Python, Scikit-learn
12	Hyperparameter Tuning	Perform hyperparameter tuning using Grid Search and Random Search.	Python, Scikit-learn

9. Data Analytics

A course on data mining techniques and their applications in knowledge discovery from large datasets.

Unit 1: Introduction, data summarization through characterization, discrimination and Data warehousing techniques

Unit 2: Concepts and algorithms for mining patterns and associations

Unit 3: Concepts, algorithms related to classification and regression

Unit 4: Concepts and algorithms for clustering the data

Unit 5: Outlier analysis and future trends.

Course Outcomes

CO-1. describe the concepts of data summarization, data warehousing, pattern mining, classification and clustering approaches.

CO-2. perform the task of data summarization, pattern mining, classification and clustering based on the requirements.

CO-3. prescribe a single or a combination of data summarization, pattern mining, classification and clustering approaches for the problem scenario of a business/organization.

CO-4. construct the improved data analytics methods for existing services.

CO-5. formulate new data mining problems for creating new services and design the corresponding solutions.

Online Labs

Lab	Experiment Title	Short Description	Software Requirements
1	Summarization	Given a table of data with about 5 attributes and 100 rows, derive a summarized table by applying generalization techniques.	Weka, Python
2	Data Preprocessing - Handling Missing Values	Given a large table of data about 10 attributes and 1000 rows, compare the performance of two approaches to fill missing values: mean-based technique vs. most probable technique.	Weka, Python
3	Data Preprocessing - Handling Noise	Given a large table of data about 10 attributes and 10000 rows, compare the performance of noise filling techniques: Binning and regression.	Weka, Python
4	Summarization with Attribute Oriented Induction	Given a large table of data about 10 attributes and 1000 rows, derive a summarized table by extending attribute-oriented induction technique.	Weka, Python
5	Scalable Generalization with Data Cube	Given a large table of data about 30 attributes and 100000 rows, extract interesting cubes by applying BUC algorithm.	Weka, Python
6	Extracting Association Rules	Given a dataset of users and movie ratings of 100000 users, build an association rule-based recommendation algorithm.	Weka, Python
7	Decision Tree Based Classification	Given a dataset of loan applications in banking, build a decision tree classifier and compute the accuracy, precision, recall.	Weka, Python
8	Naive Bayes Based Classification	Given a dataset of loan applications in banking, build a Naive Bayes classifier and compute the accuracy, precision, recall.	Weka, Python
9	K-Nearest Neighbor Based Classification	Given a dataset of loan applications in banking, build a k-nearest neighbor classifier and compute the accuracy, precision, recall.	Weka, Python
10	Ensemble-Based Classification	Given a dataset of loan applications in banking, build an ensemble classifier of decision tree, Naive Bayes, and k-nearest neighbor and compute the accuracy, precision, recall.	Weka, Python
11	K-Means Clustering	Given a dataset of loan applications in banking, compare the quality of clusters based on silhouette coefficient by applying k-means and k-medoids algorithm.	Weka, Python
12	Data Preprocessing Based Clustering	Given a dataset of loan applications in banking, compare the quality of clusters based on silhouette coefficient by applying k-means and considering data reduction techniques: Principal Component Analysis and attribute subset selection (decision tree induction).	Weka, Python

10. Math for Data Science II: Advanced Topics

This course introduces concepts of probability and linear algebra relevant to computer science, focusing on random variables, distributions, and linear transformations.

Unit 1: Probability Fundamentals and Distributions Basic concepts of probability, random variables, and common probability distributions.

Unit 2: Random Variables and Expectation Exploration of discrete and continuous random variables and their expected values.

Unit 3: Central Limit Theorem and Bayesian Statistics Introduction to the central limit theorem and Bayesian statistics for decision making.

Unit 4: Linear Algebra Concepts Overview of vectors, matrices, determinants, and linear transformations, Matrix Decomposition Theorems.

Course outcomes

CO-1: Understanding the basic probability concepts sample space, events, probability mass function, conditional probability, Bayes Rule, Random Variables, Probability Mass and Density functions, Cumulative distribution function, Expectation, Variance, Bernoulli Binomial, Gaussian, Geometric, Exponential, Poisson distributions.

CO-2: Demonstrate familiarity with the use of Linearity of Expectation, Markov's and Chebyshev's Inequalities, the Law of Large Numbers, Central Limit Theorem.

CO-3: Apply principles of Tail bounds and Central Limit Theorem to real-world problems in Estimation, Randomized Algorithms, etc.

CO-4: Derive formulas for finding Maximum Likelihood Estimates (MLE) and Maximum Apriori Estimates (MAE) for Probability Models.

CO-5: Create mathematical models using principles of Probability and analyze them.

11. Optimization Methods

This course provides a comprehensive introduction to optimization techniques—including linear, integer, unconstrained, and constrained methods.

Unit 1: Linear Programming, Geometric Interpretation, Simplex Method, Duality, Primal-Dual Method, Interior Point Methods, Ellipsoidal Methods, Computational Issues.

Unit 2: Integer Programming, LP Relaxation, Examples from Combinatorial Optimization, Shortest Paths, Network Flows, and Matchings, Convex Sets and Functions, Need for Constrained Methods in Solving Constrained Problems.

Unit 3: Unconstrained Optimization, Optimality Conditions, Gradient Descent, Newton Method, Quasi-Newton Methods, Trust Region Methods, Conjugate Gradient Methods, Least Squares Problems.

Unit 4: Constrained Optimization, Optimality Conditions and Duality, Convex Programming Problem, Quadratic Programming, Dual Methods, Penalty and Barrier Methods, Interior Point Methods. Linear Equations, Solutions based on Matrix Factorization, Singular Value Decomposition.

Course Outcomes

CO-1: To provide a foundational understanding of Linear Programming and its geometric and algorithmic aspects, including the Simplex Method, Duality theory.

CO-2: To introduce Integer Programming and its applications to combinatorial optimization.

CO-3: To develop competency in unconstrained optimization techniques, including Gradient Descent, Newton, Quasi-Newton, and Conjugate Gradient methods, and to apply them to least squares and related problems.

CO-4: To equip students with the ability to solve constrained optimization problems, by understanding KKT conditions, duality principles.

CO-5: To build mathematical and algorithmic proficiency in handling linear systems and matrix decompositions, especially in the context of optimization.

Online Labs

Lab	Experiment Title	Short Description	Software Requirements
1	Linear Programming	Implement linear programming problems and visualize the feasible region.	Python, PuLP
2	Integer Programming	Solve integer programming problems using branch-and-bound methods.	Python, PuLP
3	Convex Optimization	Apply convex optimization methods to real-world data.	Python, CVXPY
4	Gradient Descent	Implement gradient descent for linear regression and visualize convergence.	Python, Matplotlib
5	Newton's Method	Compare Newton's method with gradient descent for optimization problems.	Python, Matplotlib
6	Singular Value Decomposition	Perform SVD on matrices and interpret results.	Python, NumPy

12. Computer Vision

An exploration of computer vision principles, including image processing, feature extraction, and object recognition.

Unit 1: Image Processing Techniques Introduction to image processing methods for enhancing and analyzing images.

Unit 2: Feature Extraction and Matching Overview of techniques for feature extraction and matching in images.

Unit 3: Object Recognition and Scene Understanding Study of object recognition methods and techniques for scene understanding.

Unit 4: Deep Learning for Vision Applications: Introduction to convolutional neural networks (CNNs), architectures for image classification and detection, and applications of deep learning in advanced vision tasks such as segmentation and pose estimation.

Course Outcomes

CO-1 Understand & implement basic graphics pipeline for game design.

CO-2 Understand various geometric transformations and projections for graphics.

CO-3 Implement rasterization pipeline features using the OpenGL library.

CO-4 Design virtual environments using popular libraries like Unity and Unreal Engine.

CO-5: Apply deep learning models, particularly convolutional neural networks, to solve real-world computer vision problems such as classification, detection, and segmentation.

Online Labs

Lab	Experiment Title	Short Description	Software Requirements
1	Green Screen Replacement	Capture your image/video in front of a green background and replace the green portion with another image/video.	C/C++/Python, OpenCV
2	Video Shot Detection	Use image difference to detect significant changes in the scene between consecutive frames of a video.	C/C++/Python, OpenCV
3	Camera Calibration	Capture a known 3D object in multiple views and calibrate the camera.	C/C++/Python, OpenCV
4	Feature Matching	Compute SIFT descriptor for representing interest points and find matches.	C/C++/Python, OpenCV
5	Image Mosaicing	Estimate homography between two images robustly and stitch them together.	C/C++/Python, OpenCV
6	Stereo Rectification	Understand stereo calibration and rectification.	C/C++/Python, OpenCV
7	Face Detection	Implement the Viola and Jones algorithm for face detection.	C/C++/Python, OpenCV
8	GrabCut Segmentation	Implement GrabCut for interactive segmentation of images.	C/C++/Python, OpenCV
9	Image Retrieval	Develop an image retrieval system using colors and color moments.	C/C++/Python, OpenCV
10	Optical Flow and Tracking	Compute optical flow and track objects in a video sequence.	C/C++/Python, OpenCV
11	YoLo Inference	Use the Yolo-V8 code base to implement object detection from images.	C/C++/Python, OpenCV, PyTorch
12	ResNet Inference	Use ResNet-50 for classification of a set of images from the ImageNet dataset.	C/C++/Python, OpenCV, PyTorch

13. Reinforcement Learning

This course provides a comprehensive introduction to reinforcement learning, covering foundational concepts, dynamic programming, Monte Carlo and temporal-difference methods, and modern function approximation techniques for decision-making in complex environments.

Unit 1: Review of Probability and Stochastic Processes, Markov Chains, Introduction to Optimization, Introduction to Dynamic Programming and Markov Decision Processes.

Unit 2: Infinite horizon discounted MDP, Bellman Optimality Criteria, Value Iteration & Policy Iteration, Average cost criteria

Unit 3: Introduction to R, Monte Carlo methods, TD Learning, Q-learning and Bootstrapping

Unit 4: Systems with continuous state-action space, Controllability and stability, Linear Quadratic Regulator (LQR), Policy Iteration (PI) and Value Iteration (VI) methods

Unit 5: Function approximation techniques – DQN, Actor-Critic methods, Integral reinforcement learning, Policy gradient methods

Course outcomes

(CO's): After completion of the course, the students will be able to

CO-1. Analyze, understand and apply the theory of Markov Decision processes

CO-2. Analyze, understand and apply the theory of Reinforcement learning

CO-3. Implement reinforcement learning algorithms using Python

CO-4. Implement RL projects in group demonstrating use cases for topics learnt.

CO-5. Provide hands-on experience with advanced function approximation methods, including Deep Q-Networks (DQN), Actor-Critic models, and policy gradient approaches for solving real-world RL problems.

Online Labs

Lab	Experiment Title	Short Description	Software Requirements
1	Modelling of a Dynamical System	Use differential equations to model RLC circuit, spring mass damper system, ground robot and simulate the response for different initial conditions for the states and control inputs.	Python
2	Controllability of a Linear Dynamical System	Verify the controllability of the RLC circuit, spring mass damper system, ground robot for a given set of system parameters.	Python

Lab	Experiment Title	Short Description	Software Requirements
3	Dynamic Programming for a Network Problem	Compute the Dynamic programming based optimal solution for a network problem.	Python
4	Value Iteration (VI) and Policy Iteration (PI) Methods	Optimal control synthesis for a discrete-time, continuous state-action dynamical system using Policy Iteration (PI) and Value Iteration (VI) methods.	Python
5	Backward Recursion Based Dynamic Programming	Solve the finite time optimal control problem for a first order system using Backward Recursion-based Dynamic Programming.	Python
6	Hamilton-Jacobi-Bellman (HJB) Equation	Hamilton-Jacobi-Bellman (HJB) equation.	Python
7	Q-Learning and SARSA	Implement Q-Learning and SARSA algorithms for robot navigation problem.	Python
8	Deep Q Learning	Implement Deep Q-Learning algorithm for robot navigation problem.	PyTorch/TensorFlow, Conda
9	Actor-Critic Method	Implement Neural Network based function approximation for Critic and Actor to solve optimal control for inverted pendulum.	PyTorch/TensorFlow, Conda
10	Integral Reinforcement Learning	Implement Integral Reinforcement Learning (IRL) based optimal control for inverted pendulum.	PyTorch/TensorFlow, Conda
11	DDPG Algorithm	Implement Deep Deterministic Policy Gradient (DDPG) based RL-algorithm for inverted pendulum and cart-pole system.	OpenAI Gym, PyTorch/TensorFlow, Conda
12	PPO Algorithm	Implement Proximal Policy Optimization (PPO) based RL-algorithm for inverted pendulum and cart-pole system.	OpenAI Gym, PyTorch/TensorFlow, Conda

14. Generative Modelling

This course provides an in-depth study of generative modeling techniques, covering variational autoencoders, generative adversarial networks, normalizing flows, energy-based models, and diffusion models, along with the mathematical foundations underlying them.

Unit 1. Brief review of Probability and Random processes, Ordinary differential equations, and optimization methods.

Unit 2. Variational Autoencoders: The Gaussian VAE, ConvNets and ResNets, Posterior collapse, Discrete VAEs.

Unit 3. Generative Adversarial Networks: f-GANs, Wasserstein GANs, Solvers for min-max, Generative Sinkhorn Modeling: KR-duality, Optimal Transport,Sinkhorn algorithm.

Unit 4. Generative Flow: Autoregressive flows, Invertible networks, Neural Ordinary Differential Equations.

Unit 5. Energy based Models: Stein's method and score matching, Langevin dynamics. Diffusion Models: Stochastic calculus, Diffusion Process, Stochastic differential equations, Simulated Annealing, Denoising diffusion models.

Course Outcomes:

CO-1. Learn the extensive mathematical foundations required for generative models.

CO-2. Learn to build mathematical models for a generative task.

CO-3. Analyze and solve complex optimization models and solvers.

CO-4. Analyze the obtained results with various benchmarks and scores.

CO-5. Learn to program basic generative model applications.

Online Labs

Lab	Experiment Title	Short Description	Software Requirements
1	Variational Autoencoders	Build a Gaussian VAE for a given dataset.	Python, TensorFlow
2	GAN Implementation	Implement a basic GAN for image generation.	Python, TensorFlow
3	Conditional GAN	Build a conditional GAN to generate images based on class labels.	Python, TensorFlow
4	Energy Based Models	Implement a simple energy-based model for image denoising.	Python, TensorFlow
5	Diffusion Models	Create a diffusion model for generating samples from a learned distribution.	Python, TensorFlow

15. Spatial Informatics

This course introduces the principles and technologies of spatial informatics, covering geospatial data representation, spatial analysis, web-based GIS, and real-world applications across environmental, urban, and scientific domains.

Course Structure:

Unit 1. Geographical Information Systems (GIS), Fundamental concepts of Space

Unit 2. Geospatial data and its Digital representation – Vectors and Raster, GIS Data collection, Editing and Data formats. Data structures for Spatial data and Spatial data management (Geospatial database)

Unit 3. Spatial Data Query and Analysis – Spatial Analysis, Network Analysis. Data compatibility - Projections and Georeferencing. Spatial reasoning and uncertainty

Unit 4. Web-GIS, GML and Map services. Geospatial applications in few areas like in Hydrology (Water flows and floods); Ecology and Environment; Land use and Land cover; Urban planning and Transportation; etc.

Unit 5. Topics in Spatial Informatics: 3DGIS, Open-Source Initiatives in GIS/RS

Course Outcomes:

CO-1: Describe how Spatial Data Science helps uncover patterns

CO-2: Apply Geospatial techniques to Prepare the data for analysis

CO-3: Analyze the spatial and temporal data and interpret its outcomes

CO-4: Assessment of application of Spatial data science in key domain areas

CO-5: Design research projects that helps synthesize the learning into an application

Online Labs

Lab	Experiment Title	Short Description	Software Requirements
1	Discrete Spatial Object Handling	Import and display vector data like points, lines, and polygons. Understand formats and CRS.	QGIS
2	Continuous Spatial Objects Handling	Import and display raster or image data. Understand spatial resolution and classification.	QGIS
3	Map Creation and Layout	Understand map elements, layout design, and exporting maps.	QGIS
4	Map Projections	Define and understand CRS, integrate thematic layers.	QGIS
5	Vector Data Operations and Analysis - 1	Vector data processing, operations, and transformations.	QGIS
6	Vector Data Operations and Analysis - 2	Spatial modeling based on vector data.	QGIS
7	Raster Data Operations and Analysis - 1	Raster data processing using different operators.	QGIS
8	Raster Data Operations and Analysis - 2	Raster-based spatial modeling and combined data processing.	QGIS
9	Creation of Spatial Database	Import geometries to spatial databases, ensuring data consistency.	PostgreSQL with PostGIS
10	Spatial Queries Using SQL	Query processing and spatial data indexing.	PostgreSQL with PostGIS
11	Web Map Service	Publish a map on the web using WMS.	Geoserver, OpenLayers
12	Web Feature Service	Build interactive queries for map generation.	Geoserver, OpenLayers

16. Introduction to NLP

This course provides a foundational understanding of Natural Language Processing, covering key concepts from lexical and syntactic analysis to distributed and contextual semantics using modern machine learning approaches.

Unit 1: Stages of NLP: from lexical to semantic. Fundamental Language processing: Tokenization, Language modeling, Text classification.

Unit 2: Morphology, POS Tagging, Chunking, Discriminative vs generative modes, HMM and CRF.

Unit 3: Syntax parsing: Constituency and Dependency, PCFG, projectivity Arc-eager.

Unit 4: Distributed semantics: SVD, Word2Vec, RNN, LSTM, Unit 5: Contextual Distributed semantics: ElMO, BERT.

Course Outcomes:

After completion of this course successfully, the students will be able to :

CO-1. Demonstrate the knowledge of stages and fundamental building blocks of NLP

CO-2. Apply NLP machine learning algorithms for classification, representation, and parsing

CO-3. Demonstrate the knowledge of Dense vector representation for NLP

CO-4. Explain the concepts behind distributed semantics

CO-5. Discuss the approaches to global and contextual semantic representation

CO-6. Apply the above concepts for fundamental NLP tasks.

Online Labs

Lab	Experiment Title	Short Description	Software Requirements
1	Tokenization	Write a Python program to tokenize a given text into sentences and words.	Python, NLTK, Colab/Jupyter Notebook
2	N-gram Language Model	Build a N-gram language model and calculate the perplexity on a test dataset.	Python, NLTK, Colab/Jupyter Notebook
3	Text Classification	Implement a text classifier using Naive Bayes and evaluate its accuracy.	Python, Scikit-learn, Colab/Jupyter Notebook
4	Morphological Analysis	Perform morphological analysis using a finite-state transducer.	Python, Colab/Jupyter Notebook, Apertium, lttoolbox
5	POS Tagging	Implement POS tagging using Hidden Markov Models and CRFs.	Python, NLTK, Colab/Jupyter Notebook
6	Chunking	Perform chunking on a given text using HMM and CRFs.	Python, NLTK, Colab/Jupyter Notebook
7	Constituency Parsing	Perform constituency parsing using the PCFG model and visualize the parse tree.	Python, NLTK, Colab/Jupyter Notebook
8	Dependency Parsing	Implement dependency parsing using a transition-based Arc-eager algorithm.	Python, SpaCy, Colab/Jupyter Notebook
9	Word2Vec	Train a Word2Vec model on a corpus and analyze semantic similarities between words.	Python, Gensim, Colab/Jupyter Notebook
10	LSTM-based Text Generation	Implement text generation using an LSTM model and evaluate its performance.	Python, Keras, Jupyter Notebook
11	BERT for Text Classification	Fine-tune a pre-trained BERT model for text classification.	Python, Hugging Face, Jupyter Notebook
12	Named Entity Recognition using BERT	Implement NER using a pre-trained BERT model on a given dataset.	Python, Hugging Face, Jupyter Notebook

17. Algorithms for Data Science

This course introduces fundamental algorithmic techniques and theoretical foundations for data science, including statistical estimation, dimensionality reduction, scalable algorithms for large datasets, and the theory of supervised learning.

Unit 1: Estimation from Random Samples and Confidence Intervals, Random Walks in Graphs and PageRank,
Unit 2: Best fit subspace or PCA using SVD.
Unit 3: Algorithms for Large Datasets: Streaming Algorithms, Property testing, Hashing, Approximate Nearest Neighbours using Locality Sensitive Hashing
Unit 4: Theory of Supervised Learning: PAC Learning, Sample Complexity, VC Dimension, Learning Half spaces, Juntas

Course Outcomes:

CO1: Understand how probability, statistics and graph theory can be used to model data science problems
CO2: Design efficient algorithms for Data Science problems with provable guarantees on runtime and accuracy
CO3: Study how to draw good samples efficiently and how to estimate statistical and linear algebra quantities, with such samples
CO4: Enable students to analyze higher dimensional data using abstract methods
CO5: Learn the theory for understanding when optimization over training samples can be expected to lead to good performance on new, unseen data.

18. Speech Signal Processing

This course introduces the fundamentals of speech processing, focusing on audio signals and machine learning applications.

Unit 1: Overview of signal processing, speech production, speech perception, types of speech, and LTI model of speech production.

Unit 2: Pitch, formants, epochs and vowel region extraction.

Unit 3: Speech analysis: STFT analysis, Linear prediction analysis and cepstral analysis.

Unit 4: Prosody analysis and excitation source analysis of speech.

Unit-5: Applications of speech processing such as speech recognition, speaker recognition and speech synthesis.

Course Outcomes

CO-1: Explaining the speech production and modelling of it.

CO-2: Analyzing the algorithms for speech events extraction.

CO-3: Applying mathematical foundations of signal analysis for speech feature Extraction.

CO-4: Analysing the speech signals using excitation source and prosody.

CO-5: Explaining the basics of speech applications.

CO-6: Designing the algorithms for speech events detection and speech application building.

Online Labs

Lab	Experiment Title	Short Description	Software Requirements
1	Manual Speech Signal-to-Symbol Transformation	Manually segment a speech signal into its constituent symbols.	Praat/Audacity
2	Speech Production Mechanism	Express characteristics of speech waveform in terms of production characteristics.	Praat/Audacity
3	Nature of Speech Signal	Study the time-varying nature of the speech signal in the time domain and frequency domain.	Praat/Audacity
4	Basics of DSP	Study the effects of sampling and quantization on speech signals.	Matlab/Python
5	Short-Time Analysis of Speech	Compute short-time energy and zero-crossing rate.	Matlab/Python
6	Pitch Analysis of Speech	Perform autocorrelation-based pitch extraction.	Matlab/Python
7	Spectrographic Analysis of Speech	Observe the spectrographic characteristics of voiced, unvoiced, and plosive sounds.	Matlab/Python
8	Time/Pitch Scale Modification	Study the effects of time and pitch changes in speech signals.	Matlab/Python
9	Cepstral Analysis of Speech	Compute the cepstrum for various voiced and unvoiced sounds.	Matlab/Python
10	Linear Prediction Analysis of Speech	Observe the LP spectrum and LP residual for voiced and unvoiced segments.	Matlab/Python
11	Formant Synthesis	Study the relation between formant frequencies of a vocal tract system and the perception of sounds.	Matlab/Python
12	Analysis by Synthesis of Speech	Break down the speech signal into different components.	Matlab/Python

19. Data Visualization

This course explores data visualization principles, techniques, and tools, equipping learners with the skills to design clear and impactful visual representations of data. The course will cover essential theory, best practices, and hands-on projects using popular data visualization tools and programming languages.

Unit 1: Introduction to Data Visualization. Data Visualization Design Principles

Unit 2: Data Cleaning and Preprocessing for Visualization. Basic Chart Types and Their Applications

Unit 3: Advanced Visualizations. Color Theory and Accessibility

Unit 4: Interactive Dashboards. Storytelling with Data. Time-Series Data Visualization

Course Outcomes

By the end of the course, students will:

CO-1. Understand fundamental principles and theories of data visualization.

CO-2. Analyze data sets to determine appropriate visual representations.

CO-3. Use industry-standard tools like Tableau, Power BI, and Python libraries (Matplotlib, Seaborn) to create visualizations.

CO-4. Develop dashboards and interactive visualizations for storytelling and decision-making.

CO-5. Critically assess and redesign existing visualizations to improve clarity and communication.

Online Labs

Lab	Experiment Title	Short Description	Software Requirements
1	Exploring Data Visualisation Tools	Familiarise with popular data visualisation tools and environments.	Tableau, Power BI
2	Applying Design Principles to Simple Visuals	Redesign a poor visualization by applying contrast, alignment, and spacing principles.	Design Tools like Powerpoint, Canva and Tableau or Python (Seaborn/Matplotlib).
3	Redesigning a Poor Visualization Using Alignment and Spacing	Redesigning a Poor Visualization Using Alignment and Spacing	Design Tools like Powerpoint, Canva and Tableau or Python (Seaborn/Matplotlib).
4	Applying Color Theory and Contrast for Effective Visuals	Applying Color Theory and Contrast for Effective Visuals	Design Tools like Powerpoint, Canva and Tableau or Python (Seaborn/Matplotlib).
5	Data cleaning and preprocessing	Cleaning and formatting the data. Visualise the cleaned data using basic charts (line, bar) to compare the raw vs. cleaned data.	Python
6	Building Basic Visualizations in Python	Learn to create basic visualisations in Python using libraries like Matplotlib and Seaborn.	Matplotlib and Seaborn
7	Building Simple Dashboards in Tableau	Create a basic interactive dashboard using Tableau	Tableau
8	Creating a Data Story with Visualisations	Build a compelling data story using visualisations and annotations.	Tableau or Python (Seaborn/Matplotlib)
9	Visualising Time-Series Data	Visualise temporal trends and patterns in data	Tableau or Python (Seaborn/Matplotlib)
10	Redesigning Misleading Visualisations	Identify common mistakes in visualisations and correct them.	Tableau or Python (Seaborn/Matplotlib)
11	Building Interactive Dashboards in Power BI -Part 1	Import a dataset and create visualisations in Power BI. Add interactivity using slicers, drill-downs, and filters. Customize the design to make it user-friendly and visually appealing.	Power BI
12	Building Interactive Dashboards in Power BI -Part 2	Import a dataset and create visualisations in Power BI. Add interactivity using slicers, drill-downs, and filters. Customize the design to make it user-friendly and visually appealing.	Power BI

20. Human-Computer Interaction for Data Science

This course introduces students to the fundamental concepts and techniques of Human-Computer Interaction (HCI) in the context of data science. The course focuses on how users interact with data-driven systems, the role of interface design in data exploration, and the importance of user-centered design in data-driven applications. Students will learn how to create effective and intuitive data visualization tools, build user-friendly systems, and apply HCI techniques to improve the user experience in data science projects.

Unit 1: Overview of HCI and its relevance to Data Science. Understanding Users and User-Centered Design including Cognitive models and human information processing

Unit 2: Data Cleaning and Preprocessing for Visualisation. Basic Chart Types and Their Applications

Unit 3: Advanced Visualisations. Color Theory and Accessibility. Interactive Dashboards

Unit 4: Storytelling with Data. Time-Series Data Visualization

Course Outcomes

By the end of the course, students will:

CO-1. Understand the core principles of Human-Computer Interaction.

CO-2. Design user-centered interfaces for data-driven applications.

CO-3. Apply HCI methodologies to design data visualization systems.

CO-4. Understand the cognitive aspects of human interaction with data visualisations and systems.

CO-5. Create effective and accessible data visualizations and dashboards that communicate insights clearly and support data-driven storytelling.

Online Labs

Lab	Experiment Title	Short Description	Software Requirements
1	Exploring Data Visualization Tools	Familiarize yourself with popular data visualization tools and environments.	Tableau, Power BI
2	Applying Design Principles to Simple Visuals	Redesign a poor visualization by applying contrast, alignment, and spacing principles.	Design tools like PowerPoint, Canva, Tableau or Python (Seaborn/Matplotlib)
3	Redesigning a Poor Visualization Using Alignment and Spacing	Redesign a poor visualization using alignment and spacing principles.	Design tools like PowerPoint, Canva, Tableau or Python (Seaborn/Matplotlib)
4	Applying Color Theory and Contrast for Effective Visuals	Apply color theory and contrast to improve the effectiveness of visualizations.	Design tools like PowerPoint, Canva, Tableau or Python (Seaborn/Matplotlib)
5	Data Cleaning and Preprocessing	Clean and format the data. Visualize the cleaned data using basic charts (line, bar) to compare the raw vs. cleaned data.	Python
6	Building Basic Visualizations in Python	Learn to create basic visualizations in Python using libraries like Matplotlib and Seaborn.	Matplotlib and Seaborn
7	Building Simple Dashboards in Tableau	Create a basic interactive dashboard using Tableau.	Tableau
8	Creating a Data Story with Visualizations	Build a compelling data story using visualizations and annotations.	Tableau or Python (Seaborn/Matplotlib)
9	Visualizing Time-Series Data	Visualize temporal trends and patterns in data.	Tableau or Python (Seaborn/Matplotlib)
10	Redesigning Misleading Visualizations	Identify common mistakes in visualizations and correct them.	Tableau or Python (Seaborn/Matplotlib)
11	Building Interactive Dashboards in Power BI - Part 1	Import a dataset and create visualizations in Power BI. Add interactivity using slicers, drill-downs, and filters. Customize the design for usability and visual appeal.	Power BI
12	Building Interactive Dashboards in Power BI - Part 2	Continue building on the interactive dashboard in Power BI, adding further interactivity and visual enhancements.	Power BI