MASTER OF COMPUTER SCIENCE

2-YEAR FULL-TIME PROGRAMME

Post Graduate Curriculum Framework

under

NEP 2020

(w.e.f 2025)

Department of Computer Science
Faculty of Mathematical Sciences
University of Delhi
Delhi-110007

MSc Computer Science Programme Details:

- **Affiliation**

  The proposed programme shall be governed by the Department of Computer Science, Faculty of Mathematical Sciences, University of Delhi, Delhi-110007.

- **Programme Structure and Objectives**

  The M.Sc. Computer Science programme is a four-semester program spanning two years. It has three structures - **MSc Computer Science with Coursework, MSc Computer Science with Coursework and Research, and MSc Computer Science with Research.** The first year (Semesters I and II) are common to all three structures. In the second year, the student can choose the structure he/she wish to follow.

  The Programme objectives of the M. Sc Computer Science with coursework are to

  - Equip the students with comprehensive knowledge of the current trends in computer science.
  - Enable the students to follow the career path of their choice by choosing courses from a wide list of specialised courses with progression.
  - Prepare the students to take up a career in the highly competitive **IT industry with development skills.**

  The Programme objectives of M. Sc Computer Science with Coursework and Research are to

  - Equip the students with comprehensive knowledge of the current trends in computer science and **introduce them to computer science research.**
  - Enable the students to follow the career path of their choice by choosing courses from a wide list of specialised courses with progression.
  - Prepare the students to take up a career
    - in the highly competitive IT industry **with research/ development skills**.
    - computer science research.

  The Programme objectives of M. Sc Computer Science with Research are to

  - Equip the students with comprehensive knowledge of the current trends in **computer science research**.
  - Enable the students to follow the research path of their choice by choosing courses from a wide list of specialised courses with progression.
  - Prepare the students to take up a career in
    - the highly competitive IT industry **with research**
    - computer science research.

Semester I

Semester I					
	Number of core courses				3
Course Code	**Course Title**	**Credits in each core course**			
		Theory	Tutorial	Practical	**Total**
DSC101	Beyond Polynomial-Time-Algorithms	3	0	1	**4**
DSC102	Machine Learning	3	0	1	**4**
DSC103	Mathematical Foundations of Computer Science	3	0	1	**4**
SEC101	Scientific Writing and Computational Analysis Tools	0	0	2	**2**
	Total credits in core course				14
	Number of DSE/GE courses				2*
	DSE1	3	0	1	**4**
	DSE2/ GE1	3	0	1	**4**
	Total credits in DSE/GE				8
	Total credits in Semester I				**22**

***Select two DSEs or one DSE and one GE**

List of DSEs for Semester I		
Course Code	**Course Title**	**L-T-P**
DSE101	Network Science	3-0-1
DSE102	Distributed and Parallel Computing	3-0-1
DSE103	Internetworking with TCP/IP	3-0-1
DSE104	Internet of Things	3-0-1
DSE105	Graph Theory	3-0-1
DSE106	Soft Computing	3-0-1

List of GEs for Semester I		
Course Code	**Course Title**	**L-T-P**
GE101	Data Analysis and Visualization	3-0-1
GE102	Programming with Python	3-0-1

Semester II					
	Number of core courses	3			
Course Code	**Course Title**	**Credits in each core course**			
		Theory	Tutorial	Practical	**Total**
DSC201	Information Security	3	0	1	**4**
DSC202	Deep Learning	3	0	1	**4**
DSC203	Wireless and Mobile Communications	3	0	1	**4**
(Skill)	Ethics for Responsible AI	1	0	1	**2**
	Total credits in core course				14
	Number of DSE/GE courses				2*
	DSE3	3	0	1	**4**
	DSE4/GE2	3	0	1	**4**
	Total credits in DSE/GE				8
	Total credits in Semester II				**22**

***Select two DSEs or one DSE and one GE**

List of DSEs for Semester II		
Course Code	**Course Title**	**L-T-P**
DSE201	Social Networks Analysis	3-0-1
DSE202	Combinatorial Optimization	3-0-1
DSE203	Cyber Security	3-0-1
DSE204	Information Retrieval	3-0-1
DSE205	Digital Image Processing	3-0-1
DSE206	Advanced Classification Methods	3-0-1
DSE207 (Only for structure 3)	Computer Vision	3-0-1

List of GEs for Semester II		
Course Code	**Course Title**	**L-T-P**
GE201	Data Mining	3-0-1
GE202	Data Science using Python	3-0-1

Semester III

Structure 1 Coursework

Semester III (Structure 1 Coursework)					
	Number of core courses	2			
Course Code	**Course Title**	**Credits in each core course**			
		Theory	Tutorial	Practical	**Total**
DSC301	Optimization Methods	3	0	1	**4**
DSC302	Natural Language Processing	3	0	1	**4**
	Skill	0	0	2	**2**
	Total credits in core course	10			
	Number of DSE/GE courses	3**			
	DSE5	3	0	1	**4**
	DSE6	3	0	1	**4**
	DSE7/GE3	3	0	1	**4**
	Total credits in DSE/GE	12			
	Total credits in Semester III	**22**			

**** Select three DSEs or two DSEs and one GE**

List of DSE for Semester III (Structure 1 Coursework)		
Course Code	**Course Title**	**L-T-P**
DSE301	Special Topics in Theoretical Computer Science	3-0-1
DSE302	Link Prediction	3-0-1
DSE303	Recommender System	3-0-1
DSE304	Time Series Data Analysis	3-0-1
DSE305	Quantum Computing and Applications	3-0-1
DSE306	Digital Forensic and Incident Response	3-0-1
DSE307	Human Computer Interaction	3-0-1
DSE308	Influence Maximization	3-0-1

Semester IV (Structure 1 Coursework)					
	Number of core courses	2			
Course Code	**Course Title**	**Credits in each core course**			
		Theory	Tutorial	Practical	**Total**
DSC401	Reinforcement learning	3	0	1	**4**

DSC402	Computer Vision	3	0	1	**4**
	Skill	0	0	2	**2**
	Total credits in core course				10
	Number of DSE/GE courses				3**
	DSE8	3	0	1	**4**
	DSE9	3	0	1	**4**
	DSE10/GE4	3	0	1	**4**
	Total credits in DSE/GE				12
	Total credits in Semester IV				**22**

**** Select three DSEs or two DSEs and one GE**

Structure 2 Course Work and Research

Semester III (Structure 2 Coursework and Research)					
	Number of core courses				2
Course Code	**Course Title**	**Credits in each core course**			
		Theory	Tutorial	Practical	**Total**
DSC301	Optimization Methods	3	0	1	**4**
DSC302	Natural Language Processing	3	0	1	**4**
	Research-1	0	0	6	**6**
	Total credits in core course				14
	Number of DSE/GE courses				2*
	DSE5	3	0	1	**4**
	DSE6/GE3	3	0	1	**4**
	Total credits in DSE/GE				8
	Total credits in Semester III				**22**

*** Select two DSEs or one DSE and one GE**

List of DSE for Semester III (Structure 2 Coursework and Research)		
Course Code	**Course Title**	**L-T-P**
DSE301	Special Topics in Theoretical Computer Science	3-0-1
DSE302	Link Prediction	3-0-1
DSE303	Recommender System	3-0-1
DSE304	Time Series Data Analysis	3-0-1
DSE305	Quantum Computing and Applications	3-0-1
DSE306	Digital Forensic and Incident Response	3-0-1
DSE307	Human Computer Interaction	3-0-1
DSE308	Influence Maximization	3-0-1

*** Select two DSEs or one DSE and one GE**

Semester III (Structure 3 Research)					
	Number of core courses				1
Course Code	**Course Title**	**Credits in each core course**			
		Theory	Tutorial	Practical	**Total**
DSC301	Optimization Methods	3	0	1	**4**
	Advanced Research Methodology	1	0	1	**2**
	Tools for Research	0	0	2	**2**
	Research-1	0	0	10	**10**
	Total credits in core course				18
	Number of DSE courses				1
	DSE5	3	0	1	**4**
	Total credits in DSE				4
	Total credits in Semester III				**22**

Semester IV (Structure 2 Coursework and Research)					
	Number of core courses				2
Course Code	**Course Title**	**Credits in each core course**			
		Theory	Tutorial	Practical	**Total**
DSC401	Reinforcement learning	3	0	1	**4**
DSC402	Computer Vision	3	0	1	**4**
	Research-2	0	0	6	**6**
	Total credits in core course				14
	Number of DSE/GE courses				2*
	DSE7	3	0	1	**4**
	DSE8/GE4	3	0	1	**4**
	Total credits in DSE/GE				8
	Total credits in Semester IV				**22**

Structure 3 Research

List of DSE for Semester III (Structure 3 Research)		
Course Code	**Course Title**	**L-T-P**
DSE301	Special Topics in Theoretical Computer Science	3-0-1
DSE302	Link Prediction	3-0-1
DSE303	Recommender System	3-0-1
DSE304	Time Series Data Analysis	3-0-1
DSE305	Quantum Computing and Applications	3-0-1
DSE306	Digital Forensic and Incident Response	3-0-1
DSE307	Human Computer Interaction	3-0-1
DSE308	Influence Maximization	3-0-1
DSE309	Natural Language Processing	3-0-1

Semester IV (Structure 3 Research)					
	Number of core courses	0			
Course Code	**Course Title**	**Credits in each core course**			
		Theory	Tutorial	Practical	**Total**
	Techniques for Research Writing	1	0	1	**2**
	Research-2	0	0	16	**16**
	Total credits in core course				18
	Number of DSE courses				1
		Theory	Tutorial	Practical	**Total**
	DSE6	3	0	1	**4**
	Total credits in DSE				4
	Total credits in Semester IV				**22**

List of DSE for Semester IV (To be decided)		
Course Code	**Course Title**	**L-T-P**

SEMESTER - I

DSC101: Beyond Polynomial-Time-Algorithms [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSC101	4	3	0	1	A UG level course in Design and Analysis of Algorithms

Course Objectives

This course is designed to introduce some problems that are faced in real life but are too hard to admit fast solutions. Reduction technique to prove the hardness of a problem is discussed. Some techniques that give fast approximate solutions to these problems are also explored. The solutions provide provable guarantees on the cost relative to the optimum.

Course Learning Outcomes:

Upon successful completion of this course, students will

1. be able to appreciate that certain problems are too hard to admit fast solutions and be able to prove their hardness.
2. be able to explain what an approximation algorithm is, and the advantage of using approximation algorithms.
3. be familiar with some techniques to design approximation algorithms.
4. be familiar with some techniques to analyse the approximation factor of an algorithm.

Syllabus

Unit I **(11 hours)**

Introduction to Classes P, NP, NP-Hard, NP-Complete: Verifiability and Reduction. Constraint Satisfaction: Satisfiability Problem (SAT), 3SAT.

Unit II **(9 hours)**

Graph and Set Problems: Clique, Vertex Cover, Independent Set, Hamiltonian Cycle Problem, Travelling Salesman Problem. Sets and Partitions: Set partition, Set Cover, Subset Sum and Knapsack Problem.

Unit III **(10 hours)**

NP Hardness of Clustering Problems: k-means clustering, k-centre clustering and k-median clustering. Machine Learning Algorithms: Introduction to Clustering Algorithms, learning the predictors and fitting a model to data using gradient descent.

Unit IV **(15 hours)**

Techniques to design approximation algorithms: Introduction to Linear Programming and Integer Programming. Introduction to LP-rounding, Primal-Dual, Dual Fitting, Greedy and Local Search techniques to design approximation algorithms.

Readings:

1. Kleinberg, J. and Tardos,E. *Algorithm Design*. 1st Edition, Pearson Education India, 2013.
2. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms*, 4th Edition, The MIT Press, 2022.
3. Vazirani, Vijay V., *Approximation Algorithms*, Springer, 2013.
4. Williamson, David P., and David B. Shmoys. *The Design of Approximation Algorithms,* Cambridge University Press, 2011.

Practicals

General Instructions:

1. All programs to be developed/uploaded immediately on Github.
2. Data sets (synthetic/real) created for Practical 1 are to be used for Practical 2 and 4 also. So save them.
3. Data sets (synthetic/real) created for Practical 3 are to be used for Practical 4, 5 also. So save them.
4. Results obtained in a practical will be required in subsequent practicals. So save them.
5. A program must be completely automated with no manual intervention - from taking the input values like n, m to generating the records of the results. Generating data sets can be internal to the program. It will be better to save it in a file also, in case it is required to be re-used.
6. Time should not include the time for reading the input and writing the output.
7. If GenAI tools are used to create a part of the code, prior permission must be sought in writing and it must be recorded as documentation at the top of the program.
8. Libraries for Sorting, Searching, basic data structures like Stacks, Queues, Binary Search trees, Hashmaps, basic graph algorithms like DFS, BFS, MST (Prim's Kruskal), Djikstra's algorithm can be used. For any other specialised part, prior permission must be sought.
9. Standard LP solvers are to be used to solve Linear Programs.
10. For synthetic data sets:
    a. For graph problems:
        i. Create at least 10 data sets randomly for each pair of (n, m) where n is the number of vertices and m is the number of edges..

n	m	
10	10, 20, …100	
20	20, 40 …..100, 200, 300, 400	

List of Practicals

1. Implement/Use the brute force algorithm for the Vertex Cover problem for n = 10, m = 10, 20, …100. Record the size of the computed vertex cover and the running time for each (n, m) pair. **(10 hours)**

2. Implement/Use the greedy 2-factor approximation algorithm for the Vertex Cover problem for n = 10, m = 10, 20, …100 **(4 hours)**
   a. Record the size of the computed vertex cover and the running time for each (n, m) pair.
   b. Compute the Approximation factor for each (n, m) pair using the results obtained in Practical 1 above.

3. For n = 20 for each data set: **(2 hours)**
   a. Run the brute force algorithm for the Vertex Cover problem. Record the size of the computed vertex cover and the running time.
   b. Run the greedy approximation algorithm for the Vertex Cover problem. Record the size of the computed vertex cover and the running time
   c. Compute the Approximation factor of greedy by comparing the results with those of brute force.

4. Write a program to approximate Vertex Cover by 2 factor using LP- rounding and run it for n = 10, 20. **(4 hours)**
   For each (n, m) pair, compute the Approximation factor of greedy algorithm (obtained in Practical 2 and 3(b)) by comparing the results with LP optimal.

5. Take 5 real data sets. For each data set: **(5 hours)**
   a. Use a library and run a (heuristic)algorithm for the k-centre problem. Record the cost and the time. Run on toy data first.
   b. Implement/Use and run a greedy 2-approximation algorithm for the k-centre algorithm. Record the cost and the time, and compare with the observations in part (a). Run on toy data first.

6. As per the recent development on approximation algorithms for clustering problems. For instance, a recent approximation algorithm for k-centre with outliers. **(5 hours)**

DSC102: Machine Learning [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSC 102	4	3	0	1	NIL

Course Objectives:

This course provides a foundational understanding of machine learning (ML) concepts, covering both supervised and unsupervised learning techniques. It aims to equip students with

the knowledge required to develop, analyze, and evaluate ML models. Emphasis is placed on practical applications and the theoretical underpinnings of ML algorithms.

Course Learning Outcomes

Upon successful completion of this course, students will be able to

1. Understand fundamental concepts of machine learning, including hypothesis space, bias-variance trade-off, dimensionality reduction, and model evaluation metrics.
2. Design and implement regression and classification models such as linear regression, logistic regression, decision trees, SVMs, and neural networks using real-world datasets.
3. Apply clustering techniques such as k-means, hierarchical, and density-based clustering, and evaluate the quality of clustering results.
4. Perform model tuning and selection using appropriate validation techniques and performance metrics.

Syllabus

Unit-I **(10 hours)**

Introduction to Machine Learning: Hypothesis and target class, bias-variance tradeoff, Occam's razor, approximation and estimation errors, curse of dimensionality, dimensionality reduction, feature scaling, feature selection methods.

Unit-II **(10 hours)**

Regression: Linear regression with one variable, linear regression with multiple variables, gradient descent, logistic regression, polynomial regression, over-fitting, regularization. performance evaluation metrics, validation methods.

Unit-III **(12 hours)**

Classification: Decision trees, Naive Bayes classifier, support vector machine, kernel functions, perceptron, multilayer perceptron, neural network, back-propagation algorithm, validation methods.

Unit-IV **(13 hours)**

Clustering: Introduction, clustering paradigms, similarity and distance, density, characteristics of clustering algorithms, centre-based clustering techniques, hierarchical clustering, density-based clustering, other clustering techniques, scalable clustering algorithms, methods for clustering validation.

Readings

1. Alpaydin, Ethem. *Introduction to machine learning*. MIT press, 2020.
2. Mitchell, Tom M., and Tom M. Mitchell. *Machine learning*. Vol. 1, no. 9. New York: McGraw-hill, 1997.
3. Bishop, Christopher M., and Nasser M. Nasrabadi. *Pattern recognition and machine learning*. Vol. 4, no. 4. New York: springer, 2006.

4. Shalev-Shwartz, Shai, and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
5. Michalski, Ryszard S. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

List of Practical's:

1. Perform feature engineering and selection on a dataset by applying feature scaling techniques such as Min-Max scaling and standardisation, using PCA for dimensionality reduction, and implementing at least two feature selection methods like correlation-based filtering and chi-square test. **(4 hours)**
2. Simulate the bias-variance trade-off using polynomial regression on synthetic data and explore the impact of high-dimensional feature spaces on model performance using PCA. **(4 hours)**
3. Implement linear regression from scratch using one and multiple variables, and visualise and interpret the resulting models. **(2 hours)**
4. Implement logistic regression to classify binary data, demonstrate the effect of overfitting, and apply L1 and L2 regularisation techniques. **(2 hours)**
5. Evaluate regression models using performance metrics such as MSE, RMSE, and R^2, and apply k-fold cross-validation to assess the generalisation ability of the models. **(2 hours)**
6. Implement decision trees and Naive Bayes classifiers on a real dataset, and visualise the resulting decision boundaries and tree structures. **(2 hours)**
7. Train support vector machine (SVM) models using both linear and non-linear (e.g., RBF) kernel functions, and visualise the resulting classification boundaries. **(4 hours)**
8. Build a neural network model using a small dataset using existing libraries, and explain and visualise the learning process, including weights and activations. **(4 hours)**
9. Apply k-means and hierarchical clustering on a dataset such as customer segmentation, visualise the resulting clusters, and validate the results using silhouette scores. **(4 hours)**
10. Apply the DBSCAN clustering algorithm to a dataset with non-globular clusters, compare its performance with k-means, and visualise the differences in clustering behaviour. **(2 hours)**

DSC103: Mathematical Foundations of Computer Science [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSC103	4	3	0	1	NIL

Course Objectives:

This course will discuss fundamental concepts and tools in discrete mathematics with emphasis on their applications to computer science. The objectives of this course comprise providing students with knowledge of logic and Boolean circuits, sets, functions, relations, deterministic

and randomised algorithms. Furthermore, the students will learn analysis techniques based on counting methods, recurrence relations, trees and graphs.

Course Learning Outcomes:

At the end of the course, students will be able to

1. perform operations on vectors; represent vectors geometrically; apply vector algebra to solve problems in sub-disciplines of computer science.
2. perform operations on matrices and sparse matrices; compute the determinant, rank and eigenvalues of a matrix; apply matrix algebra to solve problems in sub-disciplines of computer science.
3. perform data analysis in a probabilistic framework.
4. visualise and model the given problem using mathematical concepts covered in the course.

Syllabus:

Unit-I (8 Hours)

Vectors: Definition of Vectors, Vector Addition, Dot and Cross Products, Span, Norm of vectors, Orthogonality, geometry of vectors, Application of vectors in document analysis.

Unit-II (12 Hours)

Matrix Algebra: Matrices as vectors; Matrix-vector, vector-matrix and matrix-matrix multiplications; Inner and outer products, triangular matrix, diagonal matrix, systems of linear equations, linear independence, determinant, rank of matrix, Eigen values and Eigen vectors, matrix transformations, geometry of transformations, Applications of matrix algebra in image representation and transformations.

Unit-III (11 Hours)

Probability Theory and Basic Statistics: Sample Space and Events, Probability axioms, Conditional Probability, Bayes' law, Introduction to Descriptive and Inferential Statistics, Describing Data Sets as Frequency tables, Relative frequency tables and graphs, Scatter diagram, Grouped data, Histograms, Ogives; Percentiles, Box Plot, Coefficient of variation, Skewness, Kurtosis.

Unit-IV (14 Hours)

Distributions: Continuous and Discrete random variables, probability density function, probability mass function, distribution function and their properties, mathematical expectation, conditional expectation, Uniform (continuous and discrete), Binomial, Poisson, Exponential, Normal, $\chi 2$ distributions, weak Law of Large Numbers, Central Limit Theorem, Chebyshev's inequality. Stochastic Processes Introduction to stochastic process, Markov Chain, Transition probabilities, Birth-Death process

Readings:

1. Trivedi, Kishor S, Probability & statistics with reliability, queuing and computer science applications. John Wiley & Sons, 2008.
2. Ross Sheldon M, Probability models for computer science. Academic Press, 2001.
3. Davis, Ernest. Linear algebra and probability for computer science applications. CRC Press, 2012.

4. Norm Matloff, From Algorithms to Z-Scores: Probabilistic and Statistical Modeling in Computer Science, University of California, Davis (Creative CommonLicence)https://www.cs.ucdavis.edu/~matloff/matloff/public_html/ProbStatBook.pdf

List of Practical's:

1. Write a modular program to create a term-frequency matrix using the extracted text. Each row of the matrix corresponds to a unique term (word) and each column represents one of the selected paragraphs. The value in each cell of the matrix should indicate how frequently the term appears in the respective paragraph. With this matrix, compute two separate paragraph similarity matrices of size n × n: one using cosine similarity and the other using dot product. These matrices will capture the pairwise similarity between all paragraphs based on their word usage patterns**. (4 hours)**

a) Once the two similarity matrices are generated, analyze the results to determine which two paragraphs are the most similar and which two are the most dissimilar under each method. Compare the findings from the cosine similarity matrix and the dot product matrix to see if they highlight the same paragraph pairs as being similar or different.

b) After completing this analysis, prepare a brief report (no more than three pages) summarizing your observations. The report should clearly state the most and least similar paragraph pairs according to both methods, and whether the results align across similarity measures. Additionally, identify the five terms that contribute most significantly to the similarity and dissimilarity between the selected paragraphs. Conclude the report with insights into why these terms might be influential and any notable patterns observed during the comparison.

2. Take a picture with your phone camera, transfer it to your computer without applying any filters or edits. Read the image file and convert it to a grayscale format. Resize the image to a manageable dimension to suit the specifications of your machine (maintain the aspect ratio). Save the grayscale image as a 2D matrix. Design a menu-driven application that allows the user to choose from the following geometric transformations: rotation, scaling, and translation. Prompt the user to enter the required parameters (angle for rotation, scaling factor, translation distances) for each transformation, and validate input appropriately. Display the original grayscale image alongside the transformed versions accompanied by a descriptive caption indicating the type of transformation and parameters used. Also, write technical specifications of the camera and compile the report. **(4 hours)**

3. Download a dataset consisting of N grayscale human face images. **(8 hours)**

I. Design a modular program to perform the following:

a) Read each image, convert it to grayscale if needed, and resize it to 32 × 32 pixels for uniformity.

b) Flatten each image into a 1D vector of size 1024 (i.e., 32 × 32)

c) Stack all vectors to form a 2D array of shape N × R, where R = 1024. In this array, each row corresponds to a different face image in vector form.

d) Apply Principal Component Analysis (PCA) to the matrix and retain only the top eigenvectors that explain at least 95% of the total variance.

e) Reshape the top 5 eigenvectors into 32 × 32 images and plot them. (Note: These visualisations are called eigenfaces, and they represent the most significant facial features in the dataset.)

II. Next, take a selfie using your smartphone, ensuring that your face is well-lit, in focus, and centred. Transfer this selfie to your computer, convert it to grayscale, and resize it to 32 × 32 pixels. Display the image, then project it onto the eigenface space using the eigenfaces extracted in step (d). Reconstruct and render your face using a linear combination of the top eigenfaces, and display the result side-by-side with the original for comparison.

III. Start with N=50, and if your system resources allow, increase the dataset to 100 faces and, if possible, resize all images, including your selfie, to 64 × 64 pixels. Apply PCA again to this higher-resolution dataset and plot the top 5 eigenfaces. Resize and project your face similarly, then display the reconstructed result.

IV. Finally, compile your observations into a report. Place at most two pairs of images per page (e.g., original vs reconstructed), side by side, and add brief, insightful observations beneath each pair. These observations should compare the quality of reconstruction, differences between 32×32 and 64×64 results, and how well the eigenfaces capture facial features.

4. Solving systems of linear equations. Compute rank, determinant, eigenvalues and eigenvectors of a matrix. **(2 Hours)**

5. Central limit theorem demonstration: Simulate sums/averages of random samples and show convergence to normal distribution. **(2 Hours)**

6. Simulating 1D and 2D random walks for a stochastic process, plot multiple paths, compute average position and variance over time. **(2 Hours)**

7. Construct and validate the Markov Chain Transition probability matrix on real-world examples (e.g., weather: Sunny, Rainy). **(2 Hours)**

8. Compute n-step transition probabilities to find long-term transitions: Define transition matrix P. Compute P^2, P^3, ... P^n using matrix multiplication. Show the probability of moving from state i to j in n steps. **(2 Hours)**

9. Stationary Distribution of a Markov Chain: Identify steady-state behaviour. Solve: $\pi P = \pi$, subject to sum(π) = 1. Simulation for frequencies converges to π. Analyse the structure of the transition graph. Check if all states are reachable or not. Check periodicity and irreducibility. **(2 Hours)**

10. Simulate the birth-death process for a queue or population models M/M/1 or M/M/M//∞, like customer service (bank, call centre, etc.), network packet queues, and hospital patients (arrival/departure) systems. Calculate birth rate (λ), rate (μ), and other performance metrics. **(2 Hours)**

SEC101: SCIENTIFIC WRITING AND COMPUTATIONAL ANALYSIS TOOLS [0-0-2]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
SEC101	2	0	0	2	NIL

Course Objective:

The objective of this course is to develop proficiency in the use of software tools required for project development.

Course Learning Outcomes:

Upon successful completion of this course, students will be able to

1. create a basic LaTeX document, format text with different styles, and apply fundamental document structures such as sections, paragraphs, and formatting options.

2. create and manipulate matrices and arrays in MATLAB, understanding their importance in numerical computation and data analysis.

3. apply optimisation techniques and perform curve fitting tasks in MATLAB, including fitting data to a model and minimizing error.

Syllabus:

Unit-I

LaTex : Basic Document Setup, and Formatting Text, Lists and Tables, Multi-Column Layouts, Mathematical Equations, Figures and Graphics, Customizing Fonts and Colors, References, Citations, and Headers/Footers/Page Numbering, Creating a Resume or CV.

Unit-II

MATLAB: Basic MATLAB Operations, Matrices and Arrays**,** Plotting and Visualization**,** Loops and Conditional Statements, Functions and Scripts**,** File Handling**,** Signal Processing**,** Image Processing**,** Optimization and Curve Fitting**,** Simulink Basics

Readings:

1. https://guides.nyu.edu/LaTeX/sample-document
2. Brian R. Hunt, Ronald L. Lipsman, Jonathan M. Rosenberg. *A Guide to MATLAB: For Beginners and Experienced Users*, Cambridge University Press, 2017.

List of Practical's

1. Basic Document Setup and Formatting Text: Create a new document with a specific page size, orientation, and margins, Format a paragraph using different font styles (bold, italic, underline) and sizes, Apply alignment options (left, right, center, justified) to different text blocks, Use styles to format headings and body text consistently. **(4 hours)**

2. Lists and Tables, Multi-Column Layouts: Create a bulleted and numbered list for an agenda or shopping list, design a table to represent student grades or inventory details, merge and split cells in the table, and apply borders and shading. Create a newsletter-style document using two or three-column layouts.**(4 hours)**

3. Mathematical Equations: Insert and format basic math expressions, such as fractions, superscripts, and subscripts. Write complex equations, like the quadratic formula, matrices, and integrals, using equation tools or LaTeX syntax (if applicable), and align multiple equations properly using equation editors or tab alignment. **(4 hours)**

4. Figures and Graphics, Customising Fonts and Colours: Insert an image or figure into the document with a caption, Resize and position the image using wrap text and alignment options, Change font family and colour scheme for different document elements, Create a cover page with custom fonts, colours, and images. **(4 hours)**

5. References, Citations, and Headers/Footers/Page Numbering: Insert a bibliography and add citations using a reference manager or built-in citation tools, Apply consistent header and footer designs across the document, Insert and format page numbers (e.g., Roman numerals for intro, Arabic for content), Create a Table of Contents and update it automatically. **(4 hours)**

6. Creating a Resume or CV: Use a template or design a CV layout from scratch with appropriate sections (Education, Experience, Skills), Insert a profile photo, format contact details, and add social media links, Apply proper use of whitespace, bullets, and alignment to ensure clarity and readability, Export the CV as a PDF and check formatting consistency. **(4 hours)**

7. Introduction to MATLAB Interface and Basic Commands: Using command window, editor, and workspace, Arithmetic operations, using help, clc, clear, who, whos. Variable Assignment and Data Types: Creating scalars, vectors, complex numbers, Type conversion, and precision handling, Matrix Creation and Manipulation: Defining matrices, transposition, reshaping, Indexing, slicing, concatenation, Matrix multiplication, inversion, determinant, eigenvalues. **(4 hours)**

8. Plotting and Visualization 2D and 3D Plotting: *plot(), subplot(), title(), xlabel(), ylabel(), legend(), 3D plots: mesh(), surf(), contour()*. Data Visualisation with Customisation: Bar graphs, pie charts, histograms, Line styles, markers, colour changes. **(4 hours)**

9. Loops and Conditional Statements: Using for, while, and if-else, writing loops to sum a series, the Fibonacci series, and Conditionals to check prime numbers or grading systems, Functions and Scripts, Creating User-Defined Functions, writing a function to compute factorial, and standard deviation. **(4 hours)**

10. Using Scripts for Automation: Writing scripts to read input, process, and display output, File Handling: Reading and Writing Files, reading data from .txt, .csv using *fopen, fscanf, textscan, readmatrix()*. Writing output to files. **(4 hours)**

11. Signal Processing: Basic Signal Generation and Analysis, generating sine, square, and triangular waves, plotting signals, and performing FFT. Filtering Signals: Applying FIR and IIR filters, using filter (), butter (), and freqz(). **(4 hours)**

12. Image Processing: Image Reading and Display, Read and display an image using *imread(), imshow(), and rgb2gray()*. Image Enhancement and Filtering: Histogram equalization, edge detection, smoothing**. (8 hours)**

13. Optimisation and Curve Fitting: Curve Fitting using *polyfit()* and *fit()*. Fit a polynomial or custom function to data, Evaluate goodness of fit, optimisation using fminsearch, fmincon: Minimise a nonlinear function with constraints. **(8 hours)**

List of DSEs for Semester I

DSE101: NETWORK SCIENCE [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	

DSE101	**4**	**3**	**0**	**1**	**Basic knowledge of probability theory**

Course Objectives:

The course aims to acquaint the students with the graph theory concepts relevant for network science. The students learn the dynamics of and on networks in the context of applications from disciplines like biology, sociology, and economics

Course Learning Outcomes:

At the end of the course, students will be

1. able to appreciate ubiquity of graph data model.
2. able to understand the importance of graph theoretic concepts in social network analysis.
3. able to understand the structural features of a network.
4. familiar with the theoretical graph generation models.
5. identify community structures in networks.
6. able to write programs to solve complex network problems.

Syllabus:

Unit-I **(5 hours)**

Introduction: Introduction to complex systems and networks, modelling of complex systems, review of graph theory.

Unit II **(10 hours)**

Network properties: Local and global properties, clustering coefficient, All-pair-shortest path-based properties, centrality measures for directed and undirected networks, degree distribution, and clustering coefficient.

Unit III **(15 hours)**

Graph models: Random graph model, degree distribution, small world network model, power laws and scale-free networks, Barabasi-Albert (preferential attachment) network model, measuring preferential attachment.

Unit IV **(15 hours)**

Community structure in networks: Communities and community detection in networks, Hierarchical algorithms for community detection, Modularity-based community detection algorithms, label propagation algorithm, multi-level graph.

Readings:

1. Pósfai, Márton, and Albert-László Barabási, *Network Science*, Cambridge University Press, 2016.
2. Newman, MEJ. *Networks: An Introduction*, Oxford University Press, 2010.
3. Easley David, and Jon Kleinberg, *Networks, crowds, and markets: Reasoning about a highly connected world*, Cambridge university press, 2010.
4. Meira Jr, W. A. G. N. E. R., and M. J. Zaki, *Data mining and analysis, Fundamental Concepts and Algorithms*, 2014.

List of Practical's:

1. Create and Visualize a Simple Network and understand nodes, edges, and basic network structure. Use NetworkX (Python), Gephi **(4 hours)**
2. Visualize Networks Using User Attributes. Use node color, size, or layout to reflect attributes like age or interest. **(2 hours)**
3. Implement Local and Global Network Properties. Also, measure average path length, diameter, and network density. **(2 hours)**
4. Calculate Node Centrality Measures and compare degree, closeness, betweenness, and eigenvector centrality. Use social graph, e.g., Zachary's Karate Club and dolphin datasets. **(8 hours)**
5. Plot and analyze degree distributions in real-world and synthetic networks. **(2 hours)**
6. Implement Erdős–Rényi random graphs and assess their structural properties. **(2 hours)**
7. Generate Small World Networks using Watts-Strogatz model and measure clustering and path length. **(2 hours)**
8. Implement Barabási–Albert Model and Generate preferential attachment networks and observe scale-free properties. **(2 hours)**
9. Implement and apply Hierarchical Community Detection algorithms to discover community structures. **(4 hours)**
10. Use the Louvain algorithm to find communities and optimize communities using modularity. **(2 hours)**

DSE102: Distributed and Parallel Computing [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSE102	**4**	**3**	**0**	**1**	**Knowledge of Computer Architecture**

Course Objectives:

The course aims to provide comprehensive knowledge about Distributed and Parallel Computing. It includes understanding of Parallel Computing & Parallel Programming concepts, design and analysis of distributed & shared memory programming models, design and analysis of accelerated computing models, performance comparison of sequential and parallel programming models and mapping of applications to parallel computing systems.

Course Learning Outcomes:

On completing this course, students will be able to

1. understand the Parallel Computing and Parallel Programming concepts.
2. analyse the need for Parallel Computing.
3. design and analyse the shared memory programming models.
4. design and analyse the distributed memory programming models.
5. design and analyse the accelerated computing models.
6. demonstrate performance analysis of sequential and parallel programming models.
7. evaluate the performance of modern parallel computing systems.

Syllabus

Unit-I **(8 hours)**

Introduction: Introduction to Parallel Programming concepts, Parallel Architectures, Pipelining, Introduction to high-performance computing (HPC) and scientific computing. The Need for HPC. Processor performances, Memory hierarchy, multi-core processing and Vector computing. Models (SIMD, MIMD, SIMT, SPMD, Dataflow Models, Demand-driven Computation).

Unit-II **(7 hours)**

Performance Measures- speedup, execution time, efficiency, cost, scalability, effect of granularity on performance, scalability of parallel systems, Amdahl's Law, Gustafson's Law. Performance bottlenecks. Introduction to Linux for parallel programming, bash scripting. HPC cluster access and environment setup, applications of High-Performance Computing.

Unit-III **(15 hours)**

Shared and Distributed Memory Programming Models: Concept of Decomposition, Tasks, Dependency Graphs, Granularity, Decomposition Techniques, Characteristics of Tasks and Interactions, Mapping Techniques for Load Balancing, Parallel Algorithm Models. OpenMP and thread programming, Message Passing Interface (MPI), Introduction to GPGPU/Vector programming. Effective use of debuggers and parallel programming.

Unit-IV **(15 hours)**

Accelerated Computing models: GPU (Graphics Processing Unit), Introduction to GPU Architectures, Clock speeds, CPU / GPU comparisons, CUDA (Compute Unified Device Architecture), CUDA Programming, CUDA Terminology - Kernels, Threads, Blocks, Memory Management, Built-in Variables and Functions, Thread Scheduling, CUDA Memory Model, Thread Synchronization. GPU – OpenACC. Analyzing and Parallelising with OpenACC,

OpenACC Optimizations. Performance analysis of parallel programming. Use of toolkits such as BLAS, LAPACKz, PETSC. Advanced scientific visualisation.

Readings:

1. Pacheco, Peter. *An introduction to parallel programming*. Elsevier, 2011.
2. Kumar, Vipin, Ananth Grama, Anshul Gupta, and George Karypis. *Introduction to parallel computing*. Vol. 110. Redwood City, CA: Benjamin/Cummings, 1994.
3. Quin, M. *Parallel programming in C with MPI and OpenMP*. Tata McGraw Hills edition (2000).
4. Cook, Shane. *CUDA programming: a developer's guide to parallel computing with GPUs*. Newness, 2012.
5. Sanders, Jason. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional, 2010.

References

1. Hager, Georg, and Gerhard Wellein. *Introduction to high-performance computing for scientists and engineers*. CRC Press, 2010.
2. HPC Tutorials (https://hpc-tutorials.llnl.gov/).

List of Practicals:

1. Write Python Programs for Computing speedup using multiprocessors. **(2 hours)**
2. Installation of OpenMP environment setup **(4 hours)**
3. Using OpenMP, write a multithreaded program where each thread prints “hello world”. **(2 hours)**
4. Write a Parallel program that should print the series of 2 and 4. Make sure both should be executed by different threads. **(2 hours)**
5. Consider a scenario where you have to parallelise a program that performs matrix multiplication using OpenMP. Your task is to implement parallelization using both static and dynamic scheduling, and compare the execution time of each approach. **(4 hours)**
6. Consider a scenario where you have a shared variable total_sum that needs to be updated concurrently by multiple threads in a parallel program. However, concurrent updates to this variable can result in data races and incorrect results. Your task is to modify the program to ensure correct synchronization using OpenMP's critical and atomic constructs. **(4 hours)**
7. Write an MPI program to send and receive Hello World to a Root process and print the received messages. **(2 hours)**
8. Write an MPI program to send two numbers (array elements) per process to a Root process and print the received messages. **(2 hours)**
9. Write an MPI program to find the sum of ranks of all the processes. Implement using point-to-point communication as well as Collective communication routines. **(4 hours)**
10. Write an MPI program to parallelize serial code on Pi calculation. **(4 hours)**

DSE103: Internetworking with TCP/IP [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSE103	**4**	**3**	**0**	**1**	**Knowledge of Computer Networks**

Course Objectives:

This course provides an in-depth understanding of the architecture, design and behaviour of the Internet through the TCP/IP suite of protocols. This course will enable students to test and troubleshoot IP-based communications systems. Furthermore, this course will explore various flow control and congestion control mechanisms of TCP, along with Quality of Service (QoS) techniques that enhance network performance.

Course Learning Outcomes:

Upon successful completion of this course, students will be able to:

1. explain the TCP/IP architecture, IPv4 and IPv6 addressing, NAT, and address resolution mechanisms like ARP, RARP and DHCP.
2. evaluate various unicast and multicast routing protocols (RIP, OSPF, BGP, MPLS, DVMRP, MOSPF) and choose appropriate routing methods
3. explain transport layer protocols, TCP/UDP mechanisms, and congestion control algorithms
4. apply networking principles to design, configure and optimize network performance using TCP/IP protocols and QoS mechanisms in real-world scenarios.

Syllabus

Unit-I: (15 hours)

Introduction: TCP/IP Protocol Suite, Addressing. IPv4 Addresses – Classful addressing, classless addressing, special addresses, ARP: Address Mapping, network address translation (NAT), ARP Protocol, DHCP operation; Transport Layer: Transmission Control Protocol-UDP and TCP, TCP- Connection Establishment and Closure (3-way handshake), Flow Control and Congestion Control, Congestion control algorithms- Leaky bucket and Token bucket algorithms.

Unit-II: (10 hours)

Quality of Service in TCP/IP networks: Application requirements and performance metrics (bandwidth, delay, jitter, loss), traffic shaping and traffic policing, packet scheduling algorithms – first-in, first-out queuing, priority queuing, fair queuing, weighted fair queuing, admission control, Integrated services and resource reservation protocol (RSVP), differentiated services (Expedited forwarding, assured forwarding)

Unit-III: (10 hours)

IP version 6: Introduction, IPv6 features overview, IPv6 header format, extension headers, IPv6 addressing, traffic class, flow labels, IPv6 security, packet sizes, DNS in IPv6- format of IPv6 resource records, DHCP in IPv6 – DHCPv6 messages, Internet Transition: Migrating from IPv4 to IPv6, Dual IP stack implementation-IPv6/IPv4 node, tunnelling, interoperability

Unit-IV: (10 hours)

Routing Protocols: static vs. dynamic routing, unicast routing protocols, intra and interdomain routing-Routing Information Protocol (RIP), Open Shortest Path First (OSPF), BGP (Border Gateway Protocol) and MPLS (Multi-Protocol Label Switching); Multicast Routing Protocols: multicasting vs multiple unicasting, MOSPF (Multicast Link State Routing), DVMRP (Multicast Distance Vector).

Readings:

1. Forouzan, Behrouz A. *Data communications and networking*. McGraw Hill, 2012.
2. Comer, Douglas E. *Internetworking with TCP/IP Principles, Protocol, and Architecture.* 6th edition, Pearson Education, 2013.
3. Parziale, Lydia, Wei Liu, Carolyn Matthews, Nicolas Rosselot, Chuck Davis, Jason Forrester, and David T. Britt. T*CP/IP tutorial and technical overview.* (IBM Redbook), 2006 http://www.redbooks.ibm.com/abstracts/gg243376.html
4. Kurose, James, and Keith Ross. *Computer Networking: A Top-Down Approach*. Pearson, 2016.
5. Tanenbaum, Andrew S. *Computer Networks*. Sixth edition. Pearson Education, 2022.

List of Practical's:

1. Find the physical and logical address of your machine by checking your computer's network settings. Further, execute the following computer networking command in Linux/Windows/Mac OS using the command prompt/terminal
1. ipconfig 2. tracert 3. ping 4. netstat 5. nslookup 6. arp 7. route 8. Pathping **(4 hours)**

2. Download ARP trace from here: https://kevincurran.org/com320/labs/wireshark/trace-arp.pcap and open the file downloaded in Wireshark. Set a display filter for packets with the Ethernet address **00:25:64:d5:10:8b.** Find and select an ARP request for the default gateway and examine its fields. **(4 hours)**

3. You are an IT admin for ABC company. You had a report that Ram (a new employee) cannot browse or email with his laptop. After researching, you found that Sita, sitting next to Ram, can browse without any problem. Compare the capture file from both machines and find out why Ram's machine is not online. (Files provided: File: Sita.pcap, Ram.pcap) **(2 hours)**

4. Download the Cisco 3600 Series Router and load the IOS Image in the GNS3 Emulator. Assign IP addresses to the routers and their interfaces. Show the static routing using two routers. **(4 hours)**

5. Design a topology with 2-3 routers and multiple LAN segments (subnets). Configure IPv4 addresses, subnet masks, and default gateways on routers and virtual PCs (VPCS). Further, verify connectivity between hosts in different subnets using ping. Capture traffic using Wireshark to observe source/destination IP addresses and ICMP packets. **(2 hours)**

6. Download the Cisco 7200 Series router and load the IOS Image in the GNS3 emulator. Assign IP addresses to PCs via DHCP configuration **(4 hours)**

7. Implement and observe one-to-one static NAT in the GNS3 Emulator **(2 hours)**

8. Implement unicast routing protocol RIP in GNS3 Emulator. **(2 hours)**

9. Implement the OSPF routing protocol in the GNS3 Emulator and then capture its packets using Wireshark. **(4 hours)**

10. Configure IPv6 global unicast addresses and link-local addresses statically on routers and VPCS. Capture traffic to analyse IPv6 headers. **(2 hours)**

DSE104: Internet of Things [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSE104	**4**	**3**	**0**	**1**	**Knowledge of Computer Network concepts**

Course Objectives:

To introduce the terminology, technology of IoT, and its applications. Introduce the concept of M2M (machine to machine) with necessary protocols, introduce the Python Scripting Language, which is used in many IoT devices, introduce the Raspberry PI platform, which is widely used in IoT applications and implementation of web-based services on IoT devices.

Course Learning Outcomes:

Upon successful completion of this course, the student will be able to

1. understand IoT value chain structure (device, data cloud), application areas, and technologies.
2. understand IoT sensors and the technological challenges IoT devices face, focusing on wireless, energy, power, and sensing modules.
3. market forecast for IoT devices with a focus on sensors.
4. explore and learn about the Internet of Things with the help of preparing projects designed for Raspberry Pi

Syllabus:

Unit-I **(10 Hours)**

Introduction to Internet of Things, definition and characteristics, physical design of IoT, communication models, communication APIs, Wireless Sensor networks, cloud computing, embedded systems, IoT applications: home, smart city, environment, energy, agriculture, and industry

Unit-II **(10 Hours)**

IoT and M2M- software-defined networks, network function virtualization, the difference between SDN and NFV for IoT, basics of IoT system management with NETCONF, YANG-NETCONF, YANG, SNMP NETOPEER

Unit-III **(11 Hours)**

Introduction to Arduino and raspberry Pi- installation, interfaces (serial, SPI, I2C), connecting LED, buzzer, switching high power devices with transistors, controlling AC power devices with Relays, controlling servo motor, speed control of DC Motor, unipolar and bipolar stepper motors

Unit-IV **(14 Hours)**

Sensors- light sensor, temperature sensor with thermistor, voltage sensor, ADC and DAC, temperature and humidity sensor DHT11, motion detection sensors, wireless Bluetooth sensors, level sensors, USB sensors, embedded sensors, distance measurement with ultrasound sensor.

IoT physical servers, cloud storage models, communication APIs web server, web server for IoT, cloud for IoT, python web application framework designing a RESTful web API.

Readings:

1. Bahga Arshdeep, *Internet of Things: A Hands-On Approach*, Bahga & Madissetti, 2014.
2. Misra Sudip, Anandarup Mukherjee, and Arijit Roy, *Introduction to IoT*, Cambridge University Press, 2021.
3. Wallace, Shawn, Matt Richardson, and Wolfram Donat. *Getting started with raspberry pi*. Maker Media, 2021.
4. Monk, S., *Raspberry Pi cookbook: Software and hardware problems and solutions*, O'Reilly Media, 2016.
5. Waher, Peter, *Learning internet of things*, Packt publishing, 2015.

List of Practical's:

1. Introduction to Arduino / NodeMCU / ESP32 in terms of basic GPIO programming. **(2 Hours)**
2. Connect and read data from a PIR sensors or ultrasonic sensors. **(2 Hours)**
3. Read temperature from LM35 or potentiometer using ADC. **(2 Hours)**
4. Wi-Fi Connectivity in NodeMCU / ESP32]. **(2 Hours)**
5. Sending sensor data to the cloud using ThingSpeak or Blynk APIs. **(2 Hours)**
6. Control an LED or relay module via Blynk app or HTTP APIs. **(2 Hours)**
7. Collect and visualize temperature and humidity data using sensors. **(2 Hours)**
8. Publish/subscribe sensor data using MQTT protocol. **(2 Hours)**
9. Build a webpage hosted on the microcontroller to show live sensor data. **(2 Hours)**
10. Automate light/fan control using IR sensor and relay. **(2 Hours)**

11. IoT-based weather monitoring station**. (2 Hours)**
12. Display data on LCD or upload to cloud. **(2 Hours)**
13. Design IoT-based smart parking system. **(2 Hours)**
14. Use ultrasonic sensors to detect parking availability. **(2 Hours)**
15. Read moisture level and send alert via app or email **(2 Hours)**

DSE105: Graph Theory [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSE105	**4**	**3**	**0**	**1**	**Basic understanding of graphs**

Course Objectives:

This course will thoroughly introduce the basic concepts of graph theory, graph properties and formulations of typical graph problems. The student will learn to model diverse applications in many areas of computing, social and natural sciences.

Course Learning Outcomes:

Upon successful completion of this course, the student will be able to

1. model problems using different types of basic graphs like trees, spanning tree, bipartite and planar graphs.
2. understand and identify special graphs like Euler graphs and Hamiltonian graphs.
3. have increased ability to understand various forms of connectedness in a graph.
4. appreciate different graph-coloring problems, matching problems and their solutions.

Syllabus:

Unit-I **(9 Hours)**

Fundamental Concepts: Definitions, examples of problems in graph theory, adjacency and incidence matrices, isomorphisms, paths, walks, cycles, components, cut-edges, cut-vertices, bipartite graphs, Eulerian graphs, Hamiltonian graphs, vertex degrees, reconstruction conjecture, extremal problems, degree sequences, directed graphs, de Bruijn cycles, orientations and tournaments, The Chinese postman problem.

Unit-II **(9 Hours)**

Trees: Trees and forests, characterizations of trees, spanning trees, radius and diameter, enumeration of trees, Cayley's formula, Prüfer code, counting spanning trees, deletion-contraction, the matrix tree theorem, graceful labeling, minimum spanning trees (Kruskal's algorithm), shortest paths (Dijkstra's algorithm).

Unit-III **(16 Hours)**

Matching and Covers: Matchings, maximal and maximum matchings, M-augmenting paths, Hall's theorem and consequences, Min-max theorems, maximum matchings and vertex covers, independent sets and edge covers, Connectivity, vertex cuts, Edge-connectivity.

Connectivity and Paths: Blocks, k-connected graphs, Menger's theorem, line graphs, dual graphs, network flow problems, flows and source/sink cuts, Ford-Fulkerson algorithm, Max-flow min-cut theorem.

Unit-IV **(11 Hours)**

Graph Coloring: Vertex colorings, bounds on chromatic numbers, Chromatic numbers of graphs constructed from smaller graphs, chromatic polynomials, properties of the chromatic polynomial, the deletion-contraction recurrence.

Planar Graphs: Planar graphs, Euler's formula, Kuratowski's theorem, five- and four-color theorems.

Readings:

1. West, Douglas B, *Introduction to Graph Theory*, 2nd Edition, Pearson, 2017.
2. Chartrand, Gary and Zhang, Ping, *Introduction to Graph Theory*, Tata McGraw Hill, 2017.
3. Gross, Jonathan L., Yellen, Jay and Anderson, Mark, *Graph Theory and Its Applications*, 3rd Edition, Taylor & Francis, 2024.

References:

1. Deo, Narsingh, *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall India Learning Private Limited, New edition, 1979.

List of practicals:

You may choose a suitable programming language and all necessary/relevant inputs/ problems.

1. Write a program to check whether the graph is: Bipartite, Eulerian, Hamiltonian or not. **(4 Hours)**

2. Write a program to find the minimum spanning trees of the graph. **(2 Hours)**

3. Write a program to find the shortest paths of the graph. **(2 Hours)**

4. Write a program to implement the Prüfer code of the graph. **(2 Hours)**

5. Write a program to implement the maximum and maximal matching of the graph. **(4 Hours)**

6. Write a program to implement the Ford-Fulkerson algorithm, Max-flow, and Min-cut theorem of the graph. **(6 Hours)**

7. Write a program to implement the Vertex colouring of the graph. **(2 Hours)**

8. Write a program to find the chromatic numbers of the graph. **(2 Hours)**

9. Write a program to check whether the graph is planar or not. **(2 Hours)**

10. Write a program to solve the Chinese postman problem. **(4 Hours)**

DSE106: Soft Computing [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSE106	4	3	0	1	NIL

Course Objectives:

The objective of the course is to understand and apply different domains of soft computing techniques like neural networks, fuzzy logic, genetic algorithm and swarm optimization. This course provides insights of soft computing frameworks applicable to bring its precision solutions for wide range of complex scientific applications.

Course Learning Outcomes:

Upon successful completion of this course, students will be able to

1. understand different soft computing techniques.
2. design hybrid soft techniques over conventional computing methods.
3. design robust and low-cost intelligent machines to solve real-world problems.

Syllabus:

Unit-I **(16 Hours)**

Fundamental Concepts: Introduction of soft computing, soft computing vs. hard computing, various types of soft computing techniques, applications of soft computing.

Artificial Neural Networks and Paradigms: Introduction to neuron model, neural network architecture, learning rules, perceptrons, single layer perceptrons, multilayer perceptrons, back propagation networks, Kohonen's Self organizing networks, Hopfield network, applications of neural networks.

Unit-II **(9 Hours)**

Fuzzy Logic: Introduction of fuzzy sets and fuzzy reasoning, basic functions on fuzzy sets, relations, rule-based models and linguistic variables, fuzzy controls, fuzzy decision making, inferencing, defuzzification, fuzzy clustering, fuzzy rule-based classifier, applications of fuzzy logics.

Unit-III **(12 Hours)**

Evolutionary Algorithms: Introduction to evolutionary algorithms, basic principles of evolutionary algorithms, evolutionary strategies, genetic algorithm, fitness computations, cross-over, mutation, evolutionary programming, classifier systems, genetic programming parse trees, variants of genetic algorithm, genetic algorithm applications.

Unit-IV **(8 hours)**

Swarm Optimizations: Ant Colony Optimization, Particle Swarm Optimization, Artificial Bee Colony Optimization, concept of multi-objective optimization problems (MOOPs), Multi-Objective Evolutionary Algorithm (MOEA), Non-Pareto approaches to solve MOOPs, Pareto-based approaches to solve MOOPs, Some applications with MOEAs.

Readings:

1. Jang, Jang-Shing Roger, Sun, Chuen-Tsai, and Eiji, Mizutani, *Neuro-Fuzzy and Soft Computing*: A *Computational Approach to Learning and Machine Intelligence*, 1st edition, Pearson Education, 2015.
2. Haykin, Simon O., *Neural Networks: A Comprehensive Foundation*, 2nd edition, Pearson, 1998.
3. Zimmermann, Hans-Jurgen, *Fuzzy Set Theory - and its Applications*, 2nd edition, Kluwer Academic Publishers, 1991.
4. Goldberg, David E., *Genetic Algorithms in Search, Optimization, and Machine Learning* Addison Wesley, 1989.

References:

1. Yegnanarayana, B., Artificial Neural Networks, PHI, 2004.
2. Zurada, Jacek M., Introduction to Artificial Neural Systems, 1st edition, Jaico Publishing House, 1994.
3. Ross, Timothy J., Fuzzy Logic with Engineering Applications, 3rd edition, Wiley, 2011

List of practicals:

You may choose a suitable programming language and all necessary/relevant inputs/ problems.

1. Write a program to implement single-layer and multi-layer neural networks. **(2 Hours)**

2. Write a program to implement back-propagation neural networks. **(2 Hours)**

3. Write a program to implement Kohonen's self-organising and Hopfield networks. **(3 Hours)**

4. Write a program to implement fuzzy: membership functions, controller systems, defuzzification, and rules. **(3 Hours)**

5. Write a program to implement/design the working of (i) Anti-lock Braking System and (ii) FAN speed control of Room Fan using a fuzzy system. **(6 Hours)**

6. Write a program for $Max : f(x) = x_1^3 - 2x_2^2 + 3x_3^3 + 4x_4$ that use Genetic Algorithms to solve the problem of Maximization function. ***Subject to:*** $1 \leq x_1 \leq 5; 2 \leq x_2 \leq 4; 3 \leq x_3 \leq 6; 1 \leq x_4 \leq 10$, Genetic Algorithm will be used to find the value of x_1, x_2, x_3, x_4 and $f(x)$ that satisfy the above equation. **(2 Hours)**

7. Write a program to implement the Ant Colony Optimization technique. **(3 Hours)**

8. Write a program to implement the Particle Swarm Optimization technique. **(3 Hours)**

9. Write a program to implement the Artificial Bee Colony Optimization technique. **(3 Hours)**

10. Write a program to implement the Multi-objective evolutionary algorithm. **(3 Hours)**

List of GEs for Semester I

GE101: DATA ANALYSIS AND VISUALIZATION [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
GE101	**4**	**3**	**0**	**1**		**Basic understanding of statistics and familiarity with Python programming**

Course Objectives:

The course develops students' competence in cleaning and analyzing data related to a chosen application. It also aims to develop skills in using various tools for data visualization and choosing the right tool for given data.

Course Learning Outcomes:

On completing the course, students will be able to:

1. use data analysis tools with ease.
2. load, clean, transform, merge, and reshape data.
3. create informative visualisations and summarise data sets.
4. analyse and manipulate time series data.
5. solve real-world data analysis problems.

Syllabus

Unit-I **(10 hours)**

Introduction: Introduction to Data Science, Exploratory Data Analysis and Data Science Process. Motivation for using Python for Data Analysis, Introduction of Python shell iPython and Jupyter Notebook. Essential Python Libraries: NumPy, pandas, matplotlib, SciPy, scikit-learn, stats models.

Unit-II **(10 hours)**

Getting Started with Pandas: Arrays and vectorized computation, Introduction to pandas Data Structures, Essential Functionality, Summarizing and Computing Descriptive Statistics. Data Loading, Storage and File Formats. Reading and Writing Data in Text Format, Web Scraping, Binary Data Formats, Interacting with Web APIs, Interacting with Databases, Data Cleaning and Preparation. Handling Missing Data, Data Transformation, String Manipulation

Unit-III **(15 hours)**

Data Wrangling: Hierarchical Indexing, Combining and Merging Data Sets Reshaping and Pivoting. Data Visualization Matplotlib: Basics of Matplotlib, plotting with pandas and seaborn, and other Python visualization tools. Data Aggregation and Group operations: Data grouping, Data aggregation, General split-apply-combine, Pivot tables and cross-tabulation.

Unit-IV **(10 hours)**

Time Series Data Analysis: Date and Time Data Types and Tools, Time series Basics, Frequencies and Shifting, Time Zone Handling, Periods and Periods Arithmetic, Resampling and Frequency Conversion, Moving Window Functions.

Readings

1. McKinney, Wes. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.", 2012.
2. O'Neil, Cathy, and Rachel Schutt. *Doing data science: Straight talk from the frontline*. " O'Reilly Media, Inc.", 2013.

List of Practical's:

1. Install and explore Jupyter Notebook and iPython shell. Write Python code snippets using basic data types, control structures, and functions. Import and demonstrate the use of libraries like NumPy, pandas, matplotlib, and scikit-learn with simple examples. **(4 hours)**
2. Perform array creation, indexing, slicing, broadcasting, and vectorized operations using NumPy. Apply mathematical functions, linear algebra routines, and random number generation. **(4 hours)**
3. Create and manipulate Series and DataFrames. Demonstrate operations like indexing, filtering, sorting, reindexing, and applying functions across rows/columns. Use pandas for basic descriptive statistics and summaries. **(4 hours)**
4. Read data from CSV, Excel, JSON, and HTML files using pandas. Demonstrate web scraping using requests and BeautifulSoup. Interact with Web APIs and connect to SQLite databases. Save and load data in binary formats (e.g., pickle, HDF5). **(4 hours)**
5. Handle missing values, duplicate data, and inconsistent formats. Apply string operations and regular expressions for text data cleaning. Demonstrate data transformations such as normalization, binning, and type conversion**. (2 hours)**
6. Merge and join multiple DataFrames. Apply hierarchical indexing and reshaping techniques like stack, unstack, melt, and pivot. Create pivot tables and perform cross-tabulations. **(2**

hours)

7. Use groupby() to aggregate and summarize data. Demonstrate split-apply-combine strategy. Compute custom aggregation functions and perform multi-level group operations. **(2 hours)**
8. Create line, bar, histogram, and scatter plots using matplotlib. Use pandas and seaborn for advanced plots like boxplots, pair plots, heatmaps, and violin plots. Customize plots with titles, labels, legends, and annotations. **(4 hours)**
9. Work with date and time objects in pandas. Parse date strings, index data with datetime, and perform slicing and selection. Resample data to different frequencies and demonstrate upsampling/downsampling. **(2 hours)**
10. Apply moving window functions (rolling, expanding). Handle time zones and perform shifting, lagging, and differencing. Create and manipulate period objects and apply arithmetic on time series data. **(2 hours)**

GE102: PROGRAMMING WITH PYTHON [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
GE 102	**4**	**3**	**0**	**1**		**NIL**

Course Objectives:

The course aims to develop the student's problem-solving skills. The course also focuses on debugging skills. The student learns to develop well-documented modular wode.

Course Learning Outcomes:

At the end of the course, students will be able to

1. select a suitable programming construct and in-built data structure for a given problem.
2. design, develop, document, and debug modular programs.
3. use recursion as a programming paradigm for problem-solving.
4. apply object-oriented paradigm for problem-solving.

Syllabus:

Unit-I **(9 hours)**

Introduction: Notion of class, object, identifier, keyword, and literal; basic data types: int, float, string, Boolean; basic operators (arithmetic, relational, logical, assignment), standard libraries.

Unit-II **(10 hours)**

Program Development: Modular program development, input and output statements, control statements: branching, looping, exit function, break, continue, and switch-break; use of mutable and immutable structures. strings, lists, sets, tuples and dictionary, and associated operations testing, and debugging a program.

Unit-III **(18 hours)**

Recursion: Use of recursion as a programming paradigm for problem solving.
Object Oriented Programming: Use of classes, inheritance, and operator overloading in problem solving. Exception Handling and File Handling: Reading and writing text and structured files, errors and exceptions.

Unit-IV **(8 hours)**

Visualization using 2D and 3D graphics: Visualization using graphical objects like point, line, histogram, 3D objects, animation.

Readings:

1. Allen Downey, *Think Python: How to Think Like a Computer Scientist*, 3rd edition, O'REILLY Publishers, 2024.
2. J.V. Guttag, *Introduction to Computation and Programming Using Python: With Application to Understanding Data*, MIT Press, 2016.
3. Robert Sedgewick , Kevin Wayne, Robert Dondero, *Introduction to Programming in Python: An Interdisciplinary Approach, Addison-Wesley Professional, 2015*
4. Tony Gaddis, *Starting Out with Python*, Pearson, 2021

List of Practical's:

1. Write a program that accepts height and weight from the user and calculates the Body Mass Index (BMI), displaying the BMI value and its category (underweight, normal, overweight, or obese). Use appropriate data types and arithmetic operators. **(1 hours)**
2. Write a program that takes a list of numbers as input from the user and provides options to find the maximum, minimum, average, and sorted version of the list using built-in list functions. **(1 hours)**
3. Write a menu-driven program using modular functions that performs the following operations on sets: union, intersection, difference, and symmetric difference. Allow users to input two sets of integers. **(1 hours)**
4. Develop a program that simulates a basic dictionary-based glossary. Allow the user to add, update, delete, and retrieve definitions of terms using a dictionary data structure. **(1 hours)**

5. Write a menu-driven program that accepts a string input from the user and provides the following options: (a) check whether the given string is a palindrome, and (b) count the number of vowels and consonants in the string. The program should continue to display the menu until the user chooses to exit. Ensure that the implementation handles both uppercase and lowercase characters and ignores spaces and punctuation where appropriate. **(1 hours)**

6. Write a recursive program for the following: **(5 hours)**

   - To compute the factorial of a given non-negative integer.
   - To find n^{th} number in the Fibonacci series
   - To add multiplies two number without using multiplication operator.

- To count number of characters in a string
- To add items of a list

7. Write a program to implement a Student Record System using object-oriented programming concepts. Create a class called Student that includes attributes such as student ID, name, age, department, and marks. Define methods to input student details, store them in an appropriate data structure and display the details of a student based on their student ID. The program should allow the user to add multiple students and retrieve information of any specific student by searching with the ID. Use constructors for initializing student objects and demonstrate encapsulation by keeping attributes private and accessing them through getter methods. **(3 hours)**
8. Write a program to implement a Bank Account Management System using object-oriented programming principles. Begin by creating a base class called BankAccount that includes attributes such as account number, account holder name, and balance. Define member functions to perform common operations like deposit(amount), withdraw(amount), and check_balance() for displaying the current balance. **(3 hours)**
9. Write a program to implement a Library Management System. Create classes Book, User, and BorrowedBook. Use inheritance and method overriding for different user types (Student, Faculty) with borrowing limits. Include features to borrow and return books. **(3 hours)**
10. Write a program that reads a text file containing names and marks of students, calculates average marks, and writes the result into another file. Handle exceptions for file not found, read/write errors, and invalid data. **(3 hours)**
11. Develop a contact management system that stores contacts in a structured text file (CSV format). Implement the functionalities to add, update, delete, and search contacts. Use try-except blocks to handle runtime errors like missing fields or invalid input. **(3 hours)**
12. Write a program using a graphics library (e.g., matplotlib or turtle) to draw basic 2D shapes such as points, lines, rectangles, and circles based on user input coordinates and dimensions. **(1 hours)**
13. Write a program that visualizes data using a histogram. Accept a set of numbers (e.g., student marks) and plot the distribution of scores using matplotlib. Label axes and add a title. **(1 hours)**
14. Write a program that plots a 3D surface or wireframe using matplotlib or a suitable 3D plotting library. Allow the user to input a mathematical function and visualize its graph. **(1 hours)**
15. Create a data visualization dashboard that takes user-defined numeric data and shows a bar chart, line graph, and pie chart on demand. Allow toggling between different chart types. **(2 hours)**

SEMESTER - II

DSC201: INFORMATION SECURITY [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSC201	**4**	**3**	**0**	**1**	**NIL**

Course Objectives:

This course aims to provide a comprehensive understanding of security principles, threats, and cryptographic techniques used in modern computing systems. It covers security models, authentication mechanisms, and key management protocols to safeguard data and communication. The course also emphasizes real-world applications of security frameworks, to enhance secure communication and data protection.

Course Learning Outcomes:

Upon successful completion of this course, students will be able to:

1. analyze the distinction between system protection and security, emphasizing their role in safeguarding digital assets
2. implement symmetric and asymmetric encryption algorithms
3. describe the role and implementation of digital signatures.
4. understand how cryptographic methods and security models are applied in practical systems like secure communication and e-commerce

Syllabus:

Unit-I **(4 hours)**

Overview of Security: Protection versus security; aspects of security– confidentiality, data integrity, availability, privacy; user authentication, access controls. Security Threats and Models: Program threats, worms, viruses, Trojan horse, trap door, stack and buffer overflow; system threats- intruders; communication threats- tapping and piracy.

Unit-II **(20 hours)**

Cryptography: Substitution, transposition ciphers, symmetric-key algorithms: Data Encryption Standard, Advanced Encryption Standard, IDEA, block cipher operation, stream ciphers: RC-4. Public key encryption: knapsack, RSA, El Gamal, Elliptic curve cryptography.

Unit-III **(15 hours)**

Integrity and authentication: Message Integrity - MDC, MAC, cryptographic hash function - iterated hash functions, compression functions, SHA-512, Digital signatures -RSA digital signature scheme, ElGamal digital signature scheme, digital signature standard (DSS), elliptic curve digital signature scheme, Entity authentication.

Unit-IV **(6 hours)**

Key Management and Intrusion Detection & Prevention: Symmetric key distribution, Kerberos, Diffie-Hellman key agreement, public key distribution, public key infrastructures, Intrusion Detection and Prevention Systems.

Readings:

1. Stalling, William. *Cryptography and Network Security Principles and Practices*. 8th edition, Pearson education, 2023
2. Forouzan, Behrouz A., and Debdeep Mukhopadhyay. *Cryptography and Network Security.* 3rd edition. McGraw-Hill education, 2015.
3. Elbirt, Adam. *J Understanding and Applying Cryptography and Data Security*. CRC Press, Taylor Francis Group, 2015.
4. Pfleeger, Charles P. SL Pfleeger, Jonathan Margulies. *Security in Computing*. 5th edition. Prentice-Hall of India, 2018.

List of practical's:

Students may choose a suitable programming language to do the following lab exercises.

1. Implement the extended Euclidean algorithm **(2 hours)**
2. Implement Encryption and decryption using the Additive Cipher **(2 hours)**
3. Cryptanalysis of the Additive Cipher **(2 hours)**
4. Implement Encryption and decryption using the Multiplicative Cipher **(2 hours)**
5. Cryptanalysis of the Multiplicative Cipher **(2 hours)**
6. Implementation of columnar transposition cipher for encryption and decryption of messages. **(2 hours)**
7. Encryption and decryption of files using python's cryptography library to demonstrate symmetric encryption and asymmetric encryption algorithms discussed in the class **(10 hours)**
8. Computation of SHA-512 Hash and HMAC for file integrity verification using a Shared Secret Key. **(4 hours)**
9. Verification of file integrity and displaying "Integrity Verified" or "Integrity Compromised" Based on various hashing algorithms. **(4 hours)**

DSC202: DEEP LEARNING [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSC202	4	3	0	1	NIL

Course Objectives:
The student learns various state-of-the-art deep learning algorithms and their applications to solve real-world problems. The student develops skills to design neural network architectures and training procedures using various deep learning platforms and software libraries.

Course Learning Outcomes:
On completing this course, students will be able to:

1. describe the feedforward and deep networks.
2. design single and multi-layer feed-forward deep networks and tune various hyper-parameters.
3. analyze the performance of deep networks.

Syllabus:

Unit-I **(14 hours)**

Introduction: Historical context and motivation for deep learning; deep feedforward neural networks, regularizing a deep network, model exploration, and hyperparameter tuning. Convolution Neural Networks: Introduction to convolution neural networks: stacking, striding, and pooling, applications like image and text classification.

Unit-II **(17 hours)**

Introduction to Natural Language Processing (NLP), Traditional NLP Techniques, Sequence Modeling: Recurrent Nets: Unfolding computational graphs, recurrent neural networks (RNNs), bidirectional RNNs, encoder-decoder sequence to sequence architectures, deep recurrent networks. Large Language Models: Transformer Architecture, Pre-training and Fine-tuning Language Models, Ethical Considerations and Bias in Language Models, Applications of Large Language Models (Text Generation, Sentiment Analysis, Question Answering)

Unit-III **(10 hours)**

Autoencoders: Undercomplete autoencoders, regularized autoencoders, sparse autoencoders, denoising autoencoders, representational power, layer, size, and depth of autoencoders, stochastic encoders and decoders. Generative Adversarial Networks (GANs): Introduction to Generative Adversarial Networks, GAN Architectures (DCGAN, CycleGAN), Applications of GANs (Image Generation, Style Transfer)

Unit-IV **(4 hours)**

Structuring Machine Learning Projects: Orthogonalization, evaluation metrics, train/dev/test distributions, size of the dev and test sets, cleaning up incorrectly labelled data, bias and variance with mismatched data distributions, transfer learning, multi-task learning.

Readings:

1. Ian Goodfellow, *Deep Learning,* MIT Press, 2016.
2. Jeff Heaton, *Deep Learning and Neural Networks*, Heaton Research Inc, 2015.
3. Mindy L Hall, *Deep Learning*, VDM Verlag, 2011.
4. Li Deng (Author), Dong Yu, *Deep Learning: Methods and Applications (Foundations and Trends in Signal Processing),* Now Publishers Inc, 2009.

List of Practical's:

1. Implement a deep neural network using PyTorch or TensorFlow on a simple dataset. Apply techniques for regularization, such as dropout and early stopping. Tune hyperparameters like learning rate, number of layers, and batch size. **(4 hours)**
2. Design and train a convolutional neural network on an image dataset. Apply stacking, striding, pooling, and normalization techniques. Evaluate model performance using accuracy and confusion matrix. **(4 hours)**
3. Preprocess text data, tokenize and vectorize inputs, and implement a CNN-based model for sentiment classification. **(2 hours)**
4. Implement a basic RNN to model sequences, such as character-level language modeling. Visualize unfolding of the computational graph and train the model on a small text corpus. **(4 hours)**
5. Build a sequence-to-sequence model using an encoder-decoder architecture with bidirectional RNNs for tasks like machine translation. Evaluate output quality using BLEU score or accuracy. **(2 hours)**
6. Use Transformers to load a pre-trained BERT or GPT model. Fine-tune it on a downstream task such as sentiment analysis or question answering datasets. **(4 hours)**
7. Analyze generated outputs from a large language model for biased or unethical responses. **(2 hours)**
8. Implement different types of autoencoders (undercomplete, sparse, denoising) on the MNIST dataset. Visualize reconstructed images and evaluate reconstruction error. **(4 hours)**
9. Build and train a simple GAN on MNIST or CelebA dataset. Generate synthetic images and compare visual quality with real images. Track generator and discriminator loss over time. **(2 hours)**
10. Use transfer learning with a pre-trained CNN (e.g., ResNet) to classify a small custom dataset. Discuss best practices for train/dev/test splits, evaluation metrics, and handling label noise. Experiment with multi-task learning if time permits. **(2 hours)**

DSC203: Wireless and Mobile Communications [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSC203	4	3	0	1	NIL

Course Objectives:

This course aims to provide a comprehensive understanding of wireless communication principles, including propagation mechanisms, fading effects, and multiple access techniques. It explores IEEE 802.11 WLANs and cellular communication concepts such as frequency reuse, handoff strategies, and interference management. It provides insights into ad-hoc and sensor networks, regulatory aspects, and network design challenges, preparing students to apply these concepts in real-world scenarios and emerging wireless technologies.

Course Learning Outcomes:

On completion of the course, students will be able to:

1. analyze and simulate wireless communication scenarios to observe propagation effects, fading impacts, and the performance of multiple access techniques.
2. understand and evaluate cellular network design by analyzing frequency reuse, handoff strategies, interference management, and capacity enhancement techniques
3. compare GSM, CDMA, LTE, WiFi, and WiMAX while understanding regulatory and spectrum management
4. analyze the architecture and key factors influencing the deployment of wireless sensor networks (WSNs) and vehicular ad-hoc networks (VANETs)

Syllabus:

Unit-I **(10 hours)**

Introduction: Wireless communication systems, history and evolution (1G to 5G and beyond), Radio wave propagation mechanisms (reflection, diffraction, scattering), large-scale path loss models, small-scale fading and multipath propagation, doppler effect and delay spread, spread spectrum techniques, multiple access and duplexing techniques

Unit-II **(10 hours)**

Cellular mobile communication: frequency re-use and channel assignment strategies, handoff strategies, types, priorities, practical considerations, interference and system capacity, co-channel and adjacent channel interference, power control measures, grade of service, definition, standards, coverage and capacity enhancement in cellular network, cell splitting, sectoring, microcells

Unit-III **(10 hours)**

Wireless systems and standards: Global System for Mobile (GSM) - services and features, system architecture, radio sub-system, channel types (traffic and control), frame structure, signal processing, CDMA standards-frequency and channel specifications, Forward and Reverse CDMA channels, WiFi, WiMAX, UMB, UMTS, LTE, and recent trends, Regulatory issues (spectrum allocation, spectrum pricing, licensing, tariff regulation and interconnection issues)

Unit-IV **(15 hours)**

IEEE 802.11(Wireless LAN) Standards: IEEE 802.11 standards overview (802.11a/b/g/n), physical layer, medium access control layer-CSMA/CA mechanism, RTS/CTS mechanism for

collision avoidance, hidden node and exposed node problems, QoS enhancements in IEEE 802.11, future trends in IEEE 802.11, IEEE 802.11 and IEEE 802.3 (Ethernet) interoperability, IEEE 802.11 in cellular networks (5G and Wi-Fi Offloading); Ad-hoc networks and sensor networks: introduction, challenges and issues, AODV, DSR, DSDV routing protocols; architecture and factors influencing the sensor network design; concept of MANET and VANET

Readings:

1. Rappaport, Theodore S. *Wireless communications: principles and practice*. Cambridge University Press, 2024.
2. Murthy, C. Siva Ram, and B. S. Manoj. *Ad hoc wireless networks: Architectures and protocols*. Pearson education, 2008.
3. Stallings, William. *Wireless communications & networks*. Pearson Education India, 2009.
4. Goldsmith, Andrea. *Wireless communications.* Cambridge university press, 2013.
5. Garg, Vijay. *Wireless communications & networking*. Elsevier, 2010.
6. Carlos,M.d. and Agrawal, D P. *Ad Hoc and Sensor Networks: Theory and Applications*, World scientific publishing company, 2011.

List of Practical's

1. Write a program to: **(4 hours)**
   a) determine the free-space path loss and the power received.
   b) endorse the statement: "Free Space Attenuation increases by 6 dB whenever the length of the path is doubled. Similarly, as frequency is doubled, free space attenuation also increases by 6 dB".
   c) plot a chart between PL (dB) and distance (Km) with varying frequencies.

2. Write a program to: **(4 hours)**
   (a) determine the two-ray path loss and the power received.
   (b) Endorse the statement: "The total attenuation increases by 12 dB when the separation distance is doubled" in the two-ray model.
   (c) plot a chart between PL(dB) and distance (Km) with varying frequencies for two ray path loss model.
   (d) Further, plot a chart between PL(dB) and distance (Km) comparing the free space path loss model and two-ray model with varying frequencies.

3. Write a program to generate a PN (Pseudo-Noise) sequence using a Linear Feedback Shift Register (LFSR). **(2 hours)**

4. Write a program to: **(4 hours)**
   a) Calculate the received power at a distance d using a log-normal shadowing model for

both indoor and outdoor environments.

b) plot a graph for Log normal shadowing model between distance and path loss/received power
   i. varying path loss exponent
   ii. varying standard deviation of shadowing parameters.

c) Compare friss-space path loss model and log normal shadowing model

5. Implement a function to create a hexagonal grid with a specified cluster size and cell size. Visualise the hexagonal grid using matplotlib. **(2 hours)**

6. Simulate a cellular network with a given number of base stations (BS) and mobile users randomly distributed in the hexagonal cells. Assign frequencies to each base station. Implement a function to calculate the signal strength at each mobile user's location based on the distance from the serving base station. Consider the free space path loss model for signal propagation. **(4 hours)**

(a)Plot a heatmap or contour plot of signal strength across the cellular network.

(b) Observe how signal strength varies within and between cells.

7. Capture WiFiTraffic on your device using the tool Wireshark. Analyse the interframe gaps for DIFS, SIFS and EIFS. Further, for each data frame, identify the corresponding ACK frame. Check whether the ACK frame is immediately following the data frame and has the SIFS timing gap. If not, determine if there is any delay and discuss possible reasons. **(2 hours)**

8. Write a Python script to simulate cell splitting. Divide some existing cells into smaller cells to demonstrate how this process increases system capacity. **(2 hours)**

9. Simulate a wireless network with three nodes (A, B, C) where A and C are hidden terminals. Implement the RTS/CTS protocol to prevent collisions at B when both A and C attempt to send data simultaneously. **(4 hours)**

10. Capture Wi-Fi frames using Wireshark on your machine's Wi-Fi interface and observe basic WLAN information. Filter for beacon frames and identify the BSSID and SSID, then find and analyze at least one Data frame and one ACK frame. Note the packet's source and destination MAC addresses. **(2 hours)**

Skill: Ethics for Responsible AI [1-1-0]

Course title	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	

Ethics for Responsible AI	**2**	**1**	**0**	**1**	**Knowledge of AI and ML concepts**

Course Objectives:

The objective of this course is to equip students with foundational concepts of ethics and relate them to the ethical concerns in the rapidly advancing field of Artificial Intelligence. The course also imparts practical skills to the students to address ethical challenges in AI development and deployment through important case studies. Taught using recent research publications, the course aims to develop competence to adapt to new ethical challenges that arise due to rapid advancements in AI.

Course Learning Outcomes:

At the end of this course, students will be able to

1. explain the fundamental theories of ethics.
2. relate the theories to the ethical concerns regarding Artificial Intelligence.
3. understand the practical implications of the ethical concerns.
4. apply mitigation strategies using available tools.
5. equipped to understand and handle new ethical challenges that arise due to swiftly advancing AI technology.

Syllabus

Unit-I **(10 hours)**

Foundations of Ethics & Ethical AI: Introduction to Ethics; Ethical theories - Utilitarianism, Deontology, Virtue Ethics; Case Study - The Trolley Problem in autonomous vehicles; AI-Specific Ethical Challenges - Bias and Fairness, Transparency and Explainability, Accountability, Privacy. Case studies for Ethical concerns and Mitigation: Bias and Fairness, Transparency and Explainability, Accountability, Privacy (hiring, criminal justice, healthcare, autonomous driving etc.), Technical challenges for addressing ethical concerns; Mitigation strategies and techniques - Federated learning, differential privacy, Algorithmic manipulation, Data Curation; Explainability tools (SHAP and LIME).

Unit-II **(5 hours)**

AI Governance: Societal risks - Job displacement, misinformation, environmental costs, deepfake; IEEE guidelines; ACM Code of Ethics for development and deployment of AI; Introduction to "right to explanation" in EU's GDPR, EU AI Act, Indian scenario; Future challenges - AGI Ethics, autonomous weapons.

Reading:

1. Jobin, A., Ienca, M. & Vayena, E. (2019). *The global landscape of AI ethics guidelines.* Nat Mach Intell 1, 389–399, 2019.

2. Radanliev, P., Santos, O., Brandon-Jones, A., & Joinson, A., *Ethics and responsible AI deployment*. Frontiers in Artificial Intelligence, 7, 1377011, 2024.

3. Pant, A., Hoda, R., Spiegler, S. V., Tantithamthavorn, C., & Turhan, B., *Ethics in the age of*

AI: An analysis of AI practitioners' awareness and challenges. ACM Transactions on Software Engineering and Methodology, 33(3), 1-35, 2024

4. Willem, T., Fritzsche, M. C., Zimmermann, B. M., Sierawska, A., Breuer, S., Braun, M., ... & Buyx, A,. *Embedded Ethics in Practice: A Toolbox for Integrating the Analysis of Ethical and Social Issues into Healthcare AI Research*. Science and Engineering Ethics, 31(1), 1-22, 2025.

List of Practicals

1. AI Ethics Case Study Debate (https://www.aiethicist.org/ethics-cases-registries)

Choose an AI ethics case study from the above mentioned URL (e.g., facial recognition in policing, autonomous vehicle accidents, ChatGPT and misinformation). Study the selected case in depth and work in a team of six students.

Each team will role-play the following three stakeholder groups: Developers, Victims of bias and Regulators (Two students per stakeholder group). Prepare 5-minute arguments both in favour of and against the deployment of the AI system under discussion. Each team must present a balanced and well-researched perspective, considering ethical, technical, and societal implications. Next, prepare a one-page individual write-up that should reflect on one of the debates conducted in class, in addition to the one in which they participated. **(14 hours)**

2. Select and download a classification dataset of your choice. Train a complex model, such as a Random Forest or a Neural Network, on the dataset. Use LIME to explain the model's predictions for three correctly classified and three misclassified test instances.

Submit the notebook containing the complete code for data preprocessing, model training, evaluation, and LIME-based explanation. Also, submit a report summarising your findings, including visualisations from LIME and your interpretation of the explanations. **(8 hours)**

3. Choose one of the following datasets.

- MBIC – A Media Bias Annotation Dataset**:** https://arxiv.org/pdf/2105.11910
- COMPAS - Recidivism Racial Bias: https://www.kaggle.com/datasets/danofer/compass
- Adult Income dataset: https://archive.ics.uci.edu/dataset/2/adult
- Dataset provided in the class

Train a classification model (e.g., Decision Tree, Logistic Regression, or XGBoost) on a suitable dataset. Use SHAP (SHapley Additive exPlanations) to interpret the model's predictions. Select five correct and five incorrect predictions and analyse them using SHAP values to understand the key features contributing to each decision. Interpret feature importance and identify patterns that influence the model's decisions. Detect any potential biases or discriminatory behaviour in the model's predictions. Summarise your findings, and reflect on how the identified biases can be mitigated to make the model fairer and more robust. **(8 hours)**

List of DSEs for Semester II

DSE201: Social Networks Analysis [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSE201	**4**	**3**	**0**	**1**	**Basic knowledge of graphs**

Course Objectives:

The course aims to equip students with various SNA approaches to data collection, cleaning, and pre-processing of network data.

Course Learning Outcomes:

On completing this course, students will be able to:

1. explain the basic concepts and principles of social network.
2. identify different types of social networks and their characteristics.
3. implement and apply various social network analysis techniques, such as, influence maximization, community detection, link prediction, and information diffusion.
4. apply network models to understand phenomena such as social influence, diffusion of innovations, and community formation.

Syllabus:

Unit-I **(9 hours)**

Introduction to Social Network Analysis: Introduction to Social Network Analysis, Types of Networks, Nodes Edges, Node Centrality, betweenness, closeness, eigenvector centrality, network centralization, Assortativity, Transitivity, Reciprocity, Similarity, Degeneracy and Network Measure, Networks Structures, Network Visualization, Tie Strength, Trust, Understanding Structure Through User Attributes and Behavior.

Unit-II **(12 hours)**

Link Analysis and Link Prediction: Applications of Link Analysis, Signed Networks, Strong and Weak Ties, Link Analysis and Algorithms, Page Rank, Personalized PageRank, DivRank, SimRank, PathSim. Temporal Changes in a Network, Evaluation Link Prediction Algorithms, Heuristic Models, Probabilistic Models, Applications of Link Prediction.

Unit-III **(12 hours)**

Community Detection: Applications of Community Detection, Types of Communities, Community Detection Algorithms, Disjoint Community Detection, Overlapping Community Detection, Local Community Detection, Evaluation of Community Detection Algorithms.

Unit-IV **(12 hours)**

Influence Maximization: Applications of Influence Maximization, Diffusion Models, Independent Cascade Model, Linear Threshold Model, Triggering Model, Time-Aware Diffusion Model, Non-Progressive Diffusion Model. Influence Maximization Algorithms, Simulation-Based Algorithms, Proxy-Based Algorithms, Sketch-Based Algorithms, Community-Based Influence Maximization, and Context-Aware Influence Maximization.

Readings:

1. Chakraborty, T. *Social Network Analysis*, Wiley India, 2021.
2. Knoke, D. and Yang, S. *Social network analysis*. SAGE publications, 2019.
3. Golbeck, J. *Analyzing the social web*, Morgan Kaufmann, 2013.
4. Wasserman, S and Faust, K. *Social network analysis: Methods and applications*, Cambridge University Press, 2012.
5. Newman, M.E.J. *Networks: An introduction*. Oxford University Press, 2010.
6. Chen, W. Castillo, C and Lakshmanan, L.V.S. *Information and influence propagation in social networks,* Springer Nature, 2014
7. Srinivas, V and Mitra, P. *Link prediction in social networks: role of power law distribution,* New York: Springer International Publishing, 2016

List of Practical's:

1. Implement heuristic link prediction methods such as Common Neighbors, Adamic-Adar, and Jaccard Coefficient to identify potential future links in a network. **(4 hours)**
2. Apply SimRank and PathSim to compute similarity scores for link prediction in graph-based data. **(2 hours)**
4. Compare heuristic and probabilistic link prediction models using evaluation metrics like precision and recall. **(2 hours)**
5. Use the Girvan–Newman algorithm to detect disjoint communities by iteratively removing high-betweenness edges. **(2 hours)**
6. Apply the Louvain or Leiden algorithm to uncover modular communities in large-scale networks. **(2 hours)**
7. Evaluate community detection results using modularity and normalized mutual information (NMI). **(2 hours)**
8. Simulate the Independent Cascade Model (ICM) to observe the spread of influence from a given set of seed nodes. **(2 hours)**
9. Simulate the Linear Threshold Model (LTM) to model influence propagation based on node activation thresholds. **(2 hours)**
10. Implement Greedy and CELF algorithms to efficiently select the top-k influential nodes for maximizing spread. **(4 hours)**
11. Benchmark different influence maximisation techniques, including simulation-based, sketch-based, and community-based methods. **(8 hours)**

DSE202: Combinatorial Optimization [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSE202	**4**	**3**	**0**	**1**	**knowledge of Linear Algebra**

Course Objectives:

The course aims to equip students with the technique of linear and integer programs to solve optimization problems via LP-based solutions to shortest path problems, minimum spanning tree problems, max-flow problems and maximum matching problems.

Course Learning Outcomes:

On completion of this course, students will be able to:

1. model problems using linear and integer programs.

2. differentiate between the computational complexities of LP and IP.
3. understand polyhedral analysis and apply it to develop algorithms.
4. understand the concept of duality and use it to design exact and approximate algorithms.
5. understand and explain the mathematical theory forming the basis of many algorithms for combinatorial optimization (particularly graph theoretic).

Syllabus:

Unit-I **(15 hours)**

Introduction: Optimization problems, neighborhoods, local and global optima, convex sets and functions, simplex method, degeneracy; duality and dual simplex algorithm, computational considerations for the simplex and dual simplex algorithms-Dantzig-Wolfe algorithms.

Unit-II **(10 hours)**

Integer Linear Programming: Cutting plane algorithms, branch and bound technique and approximation algorithms for travelling salesman problem.

Unit-III **(15 hours)**

Graph Algorithms: Primal-Dual algorithm and its application to shortest path (Dijkstra's algorithm, Floyd-Warshall algorithms), max-flow problem (Ford and Fulkerson labeling algorithms), matching problems (bipartite matching algorithm, non-bipartite matching algorithms, bipartite weighted matching-hungarian method for the assignment problem, non-bipartite weighted matching problem), efficient spanning tree algorithms.

Unit-IV **(5 hours)**

Matroids: Independence Systems and Matroids, Duality, Matroid Intersection.

Readings:

1. Korte, Bernhard H., Jens Vygen, B. Korte, and J. Vygen. *Combinatorial optimization*. Springer, 2018.
2. Matoušek, Jiří, and Bernd Gärtner. *Understanding and using linear programming*. Springer, 2007.
3. Papadimitriou, Christos H., and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover Publication, 1998.
4. Bazaraa, Mokhtar S., John J. Jarvis, and Hanif D. Sherali. *Linear programming and network flows*. John Wiley & Sons, 2011.
5. Taha, Hamdy A. *Operations research: an introduction*. Pearson Education India, 2014.

List of Practical's:

1. WAP to implement Simplex Method **(4 hours)**
2. WAP to implement Dual Simplex Method **(6 hours)**

3. WAP to implement Primal-Dual algorithm for shortest path problem **(6 hours)**
4. WAP to implement Floyd-Warshall algorithm for shortest path problem **(6 hours)**
5. Implement algorithms for matching problems. **(8hours)**

DSE203: Cyber Security [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
DSE203	4	3	0	1		NILL

Course Objectives:

This course will be responsible for laying the foundation for creating a comprehensive understanding and expertise in the field of cyber security. This paper will set the level field for all the students to be able to come at par and move together as they must go deeper into hard-core cyber security topics during the course duration.

Course Learning Outcomes:

On completion of this course, students will be able to

1. state the need and scope for cyber laws.
2. enumerate various network attacks, describe their sources, and mechanisms of prevention.
3. describe the genesis of SCADA policies and their implementation framework.
4. carry out malware analysis.

Syllabus:

Unit-I **(7 hours)**

Introduction: Cyberspace, Internet, Internet of things, Cyber Crimes, cybercriminals, Cyber Security, Cyber Security Threats, Cyber laws and legislation, Law Enforcement Roles and Responses.

Unit-II **(15 hours)**

Cyberspace Attacks: Network Threat Vectors, MITM, OWASP, ARP Spoofing, IP & MAC Spoofing, DNS Attacks, SYN Flooding attacks, UDP ping-pong and fraggle attacks, TCP port scanning and reflection attacks, DoS, DDOS. Network Penetration Testing Threat assessment, Penetration testing tools, Penetration testing, Vulnerability Analysis, Threat matrices, Firewall and IDS/IPS, Wireless networks, Wireless Fidelity (Wi-Fi), Wireless network security protocols, Nmap, Network fingerprinting, BackTrack, Metasploit.

Unit-III **(15 hours)**

Introduction to SCADA (supervisory control and data acquisition): Understanding SCADA security policies, SCADA Physical and Logical Security, understanding differences between physical and logical security, define perimeter controls and terms, define various security zones, understand communication cyber threats, understand firewall, architectures.

Unit-IV **(8 hours)**

Introduction Malware Analysis: Static Analysis, Code Review, Dynamic Analysis, Behavioral analysis of malicious executable, Sandbox Technologies, Reverse-engineering malware, Defeat anti-reverse engineering technique, automated analysis, intercepting network connections, Network flow analysis, Malicious Code Analysis, Network analysis.

Readings:

1. Peter W. Singer and Allan Friedman, *Cybersecurity and Cyberwar*, Oxford University Press, 2014.
2. Michael Sikorski, Andrew Honig, *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software* 2012, No Starch Press, San Francisco.
3. Dejay, Murugan, *Cyber Forensics* Oxford university press India Edition, 2018.
4. R. Rajkumar, D. de. Niz and M. Klein, *Cyber Physical Systems*, Addision-Wesely, 2017
5. Jonathan Clough, *Principles of Cybercrime*, Cambridge University Press, 24-Sep-2015.

References:

1. CEH Official Certified Ethical Hacking Review Guide, Wiley India Edition, 2015.

List of practical's:

Disclaimer: These lab exercises are for educational purposes only. All activities should be performed within the boundaries of the law.

1. Identify your machine's IP configuration using *ipconfig (Windows)* or i*fconfig (Linux)*. Ping various local and remote hosts to verify connectivity, trace the route to a website using *tracert* or *traceroute,* perform DNS lookups using *nslookup*, and gather domain registration details using the *whois* tool. (2hours)
2. Use tools like John the Ripper and Hydra to crack passwords. The task will involve using wordlist-based and brute-force techniques to break into password-protected accounts or services. (4 hours)
3. SET tool is one of the tool available in Kali Linux. Explore the tool and list down all types of Social Engineering attacks available in the tool. Further, make a document mentioning the Name of the attack, how the attack can be launched, and how I can make my system safe from these attacks. (6 hours)
4. Use the Social Engineering Toolkit (SET) available in Kali Linux to clone a vulnerable website (e.g., http://www.itsecgames.com/). Compare the cloned website with the original, explore different social engineering attacks such as phishing or credential harvesting**. (4 hours)**
5. Use *theHarvester* tool to collect at least 25 email addresses from various publicly available sources on the internet. These addresses can later be used for phishing simulations or further social engineering attacks. (2 hours)
6. Utilize tools from the *iHunt intelligent framework* available on web to create a sock puppet account. Document the tools used, their functionality, and how this can be applied in a

simulated information-gathering exercise. **(2 hours)**

7. Using the email addresses collected in the lab 6, perform further information gathering. Use search engines and social media platforms to find publicly available information about the email owners. Document the results using tools from iHunt Framework for email investigation.**(2 hours)**

   Note: Don't actually contact the person of interest. Just gather information which available publicly. Don't break any law.

8. Conduct a vulnerability scan on target systems using *NMap*. Analyze the results to identify potential vulnerabilities such as open ports, weak configurations, and security loopholes. **(4 hours)**

9. Use the Metasploit framework to exploit vulnerabilities and gain access to a remote Linux machine. The exercise involves using Kali Linux with Metasploit to exploit a vulnerable system. **(4 hours)**

10. A lab exercise related to SCADA & Industrial Control Systems (ICS) Security **(4 hours)**

DSE204: Information Retrieval [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSE204	**4**	**3**	**0**	**1**	**Basic understanding of Statistics and Probability is required.**

Course Objectives:

This course introduces the basics of Information Retrieval (IR), focusing on how information is organized, searched, and ranked. Students will learn about different search models, document processing techniques, and ways to measure search accuracy. The course also covers web search methods, including link analysis and web crawling.

Course Learning Outcomes:

Upon successful completion of this course, students will be able to

1. understand the fundamental concepts of Information Retrieval (IR), including information need, relevance, and early developments in IR systems.
2. apply different retrieval models, such as Boolean retrieval and ranked retrieval, to effectively search and organize information.
3. evaluate search performance using precision-recall, ranking measures, and other standard evaluation metrics.
4. process and represent documents using techniques like vector space modelling, feature selection, stemming, and similarity measures.
5. explore web search techniques, including link analysis methods like PageRank and HITS, as well as web crawling strategies.

Syllabus:

Unit-I **(15 hours)**

Introduction to Information Retrieval (IR), information, information need, and relevance, the IR system and its components, early developments in IR, user interfaces in IR, retrieval and IR models, Boolean retrieval, term vocabulary and postings list, ranked retrieval, inverted index, index construction, index compression.

Unit-II **(10 hours)**

Document processing, document representation techniques, vector space model, feature selection for IR, stop words, stemming, concept of document similarity, evaluation of information retrieval systems, notion of precision and recall, precision-recall curve.

Unit-III **(10 hours)**

Standard performance measures, MAP, reciprocal ranks, F-measure, NDCG, rank correlation, standard datasets for IR evaluation, web search and link analysis, web crawling techniques, link analysis methods, PageRank algorithm, HITS algorithm.

Unit-IV **(10 hours)**

Classification and Clustering: Notion of supervised and unsupervised algorithms, Naive Bayes, nearest neighbour and Rochio's algorithms for text classification, text clustering methods such as K-Means.

Readings:

1. Baeza-Yates, Ricardo, and Berthier Ribeiro-Neto. *Modern information retrieval*. Vol. 463, no. 1999. New York: ACM press, 1999.
2. Schütze, Hinrich, Christopher D. Manning, and Prabhakar Raghavan. *Introduction to information retrieval*. Vol. 39. Cambridge: Cambridge University Press, 2008.
3. Grossman, David A., and Ophir Frieder. *Information retrieval: Algorithms and heuristics*. Vol. 15. Springer Science & Business Media, 2004.

4. Buttcher, Stefan, Charles LA Clarke, and Gordon V. Cormack. *Information retrieval: Implementing and evaluating search engines*. Mit Press, 2016.
5. Croft, W. Bruce, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*. Vol. 520. Reading: Addison-Wesley, 2010.

List of Practical's

1. Design and implement a Boolean retrieval system that can process queries using AND, OR, and NOT operators on a given set of text documents. **(2 hours)**
2. Develop a text preprocessing pipeline that performs tokenization, case folding, stop-word removal, and stemming on a document corpus. **(2 hours)**
3. Write a program that constructs an inverted index for a collection of documents and supports retrieval of documents based on single-word queries, then implement index compression using variable-byte encoding and analyse the reduction in index size and its effect on retrieval performance. **(4 hours)**
4. Compute the Term Frequency-Inverse Document Frequency (TF-IDF) values for each term in a document collection and store them in a structured format. **(2 hours)**
5. Implement a ranked retrieval system using the vector space model and cosine similarity to score and rank documents based on a user query. **(2 hours)**
6. Use a publicly available IR dataset to implement and compare different retrieval models (e.g., Boolean, Vector Space), then calculate evaluation metrics including precision, recall, F1-score, MAP, and NDCG using provided relevance judgments, and finally, generate precision-recall curves for multiple queries to interpret the performance of each model. **(4 hours)**
7. Implement both the PageRank and HITS algorithms on a manually created web graph, simulate multiple iterations to observe rank convergence, and compute hub and authority scores for each node. **(4 hours)**
8. Train and evaluate a Naive Bayes classifier to categorize documents into predefined classes such as spam vs. ham or news categories. **(2 hours)**
9. Apply the K-Means clustering algorithm to a set of TF-IDF document vectors and visualize the resulting clusters using a 2D projection. **(4 hours)**
10. Develop a basic web crawler that extracts and stores titles and content from publicly accessible websites while respecting robots.txt. **(4 hours)**

DSE205: Digital Image Processing [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical /Practice	
DSE 205	**4**	**3**	**0**	**1**	**NIL**

Course Objectives:

The course will thoroughly introduce image enhancement in the spatial and frequency domain, followed by the image morphological operations such as dilation, erosion, hit-or-miss transformations, image segmentation and image compression.

Course Learning Outcomes:

Upon successful completion of this course, students will be able to

1. enhance the quality of an image using various transformations.
2. analyze the transform of an image in spatial domain to frequency domain.
3. apply required morphological operations to an image.
4. adequate to segment an image using various approaches.

Syllabus:

Unit-I **(7 Hours)**

Introduction: Applications of digital image processing, steps in digital image processing: image acquisition, image sampling and quantization, basic relationships between pixel.

Unit-II **(14 Hours)**

Image enhancement in the spatial domain and frequency domain: gray level transformations, histogram processing, local enhancement, image subtraction, image averaging, spatial filtering: smoothing and sharpening filters, Discrete Fourier transformation, filtering in the frequency domain: smoothening and sharpening filters, image restoration in spatial and frequency domains.

Unit-III **(12 Hours)**

Morphological image processing: erosion and dilation, opening and closing, hit-or-miss transformation, and some basic morphological algorithms.

Introduction to Image Compression: Image compression models, error free compression techniques, lossy compression techniques, JPEG, MPEG.

Unit-IV **(12 Hours)**

Image segmentation: Point, line and edge detection, gradient operator, edge linking and boundary detection, thresholding, region-based segmentation, representation schemes like chain codes, polygonal approximations, boundary segments, skeleton of a region, boundary descriptor, regional descriptor.

Readings:

1. Gonzalez, Rafael C. and Woods, Richard E., *Digital Image Processing*, 4th edition, Pearson Education, 2018.
2. Annadurai, S. and Shanmugalakshmi, R., *Fundamentals of Digital Image Processing*, 1st edition, Pearson, 2006.

3. Joshi, M. A., *Digital Image Processing: An Algorithmic Approach*, 2nd edition, PHI Learning, 2020.

References:

1. Jahne, Bernd, *Digital Image Processing*, 6th edition, Springer, 2005.
2. Chandra, B. and Majumder, D.D., *Digital Image Processing and Analysis*, 2nd edition, Prentice Hall India Learning Private Limited, 2011.

List of practicals:

You may choose a suitable programming language and all necessary/relevant inputs/ problems.

1. Implement a program for displaying grey-scale and RGB colour images, with those for accessing pixel locations, and investigate adding and subtracting a scalar value from an individual location. **(2 Hours)**

2. Implement a program to perform the image sampling and quantization technique and display the resulting images. **(4 Hours)**

3. Implement a program to apply histogram equalization to a colour image, and apply contrast stretching to the colour example image. Experiment with different parameter values to find an optimum for the visualization of this image. **(2 Hours)**

4. Implement a program to add different levels of salt and pepper and Gaussian noise to images, both in colour and grey-scale. Investigate the usefulness of all filtering for removing different levels of image noise. **(2 Hours)**

5. Write a program for all image restoration techniques. **(4 Hours)**

6. Write a program for the Fourier transform of an image. **(2 Hours)**

7. Write a program for image spatial and sharpening filtering. **(4 Hours)**

8. Implement a program to find the result of erosion, dilation, opening and closing with the below structuring element and choose any image. **(2 Hours)**

a) 1 0 0 **b)** 1 1 1

0 1 0 1 1 1

0 0 1 1 1 1

9. Write a program for all the compression techniques of an image. **(4 Hours)**

10. Write a program for all image segmentation techniques. **(4 Hours)**

DSE206: Advance Classification Methods [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSE207	**4**	**3**	**0**	**1**	**Basic understanding of machine learning concepts, and familiarity with Python programming.**

Course Objectives

By the end of this course, students will be able to apply various classification techniques and evaluate model performance using appropriate metrics. They will also gain expertise in advanced topics such as ensemble learning, semi-supervised and self-supervised methods, zero-shot and few-shot learning, and improving model robustness against adversarial attacks.

Course Outcomes

On completion of this course, students will be able to:

1. explain the fundamental and advanced classification techniques.
2. apply various binary, multi-class, and multi-label classification methods.
3. analyze and compare classification models using appropriate evaluation metrics.
4. design and optimize classification models for real-world applications.

Syllabus

Unit-I **(16 Hours)**

Review of classification problems and algorithms, Evaluation metrics, Handling imbalanced data, Cross-validation, and Model selection. Ensemble learning for classification: Weak learners and Strong learners, Model creation, Model combination; Bagging and Boosting; Random Forest; Combination Strategies, Meta-learning; Diversity in ensemble; Ensemble pruning.

Unit-II **(12 hours)**

Multi-class classification problem; Aggregation-based approach: One-versus-all, One-versus-one, Directed Acyclic Graph approach, Tree-based approach, Error-correcting output codes; multi-class SVMs; Class Imbalance in Multi-class Settings.

Unit-III **(12 hours)**

Multi-label Classification, Problem Transformation approach: Binary Relevance, Classifier Chains, Label Powerset, Algorithm Adaptation Methods; Neural Network Approaches for Multi-label Classification, Evaluation metrics for multi-label classification.

Unit-IV **(5 hours)**

Semi-supervised and Self-supervised Classification Methods, Zero-shot and Few-shot Learning, Adversarial Attacks and Robustness in Classification,

Readings

1. Zhou, Zhi-Hua. *Machine learning*. Springer nature, 2021.
2. Mohri, Mehryar, A. Rostamizadeh, and A. Talwalkar. Foundations of machine learning. The MIT Press. 2012
3. Charte, Francisco, Antonio J. Rivera, and María J. Del Jesus. *Multilabel classification: problem analysis, metrics and techniques*. Springer International Publishing, 2016.
4. Ian Goodfellow, Bengio, Yoshua, and Aaron Courville. *Deep learning*. MIT press, 2017.
5. Research papers on emerging classification techniques.

List of Practical's:

1. Implement basic classification models such as Logistic Regression, Decision Trees and SVM on benchmark datasets. Evaluate the performance of these classifiers using metrics such as accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix. Perform k-fold cross-validation and interpret the results. **(2 hours)**
2. Use a dataset with imbalanced classes and then apply techniques like SMOTE, random oversampling/under sampling, and evaluate their impact on model performance using appropriate metrics like precision-recall and balanced accuracy. **(4 hours)**
3. Implement ensemble models, including Random Forest (bagging) and AdaBoost/Gradient Boosting (boosting), using scikit-learn or XGBoost. Compare performance, analyse feature importance, and visualise ensemble effects**. (4 hours)**
4. Create a diverse ensemble of traditional machine learning classifiers such as Logistic Regression, SVM and Decision Tree using stacking or voting. Analyze diversity among models and perform ensemble pruning based on accuracy or diversity criteria. **(4 hours)**
5. Use multi-class datasets and implement one-vs-all and one-vs-one strategies. Train binary classifiers accordingly and compare both strategies in terms of performance and scalability. **(2 hours)**
6. Implement multi-class classification using tree-based methods and error-correcting output codes (ECOC). Evaluate performance and interpret how each approach handles class boundaries. **(4 hours)**
7. Train and evaluate multi-class SVMs using one-vs-all or one-vs-one schemes. Use imbalanced versions of multi-class datasets to analyze the effect of class imbalance and apply weighted loss or resampling. (**2 hours)**
8. Implement multi-label classification approaches such as Binary Relevance, Classifier Chains, and Label Powerset using scikit-multilearn or similar libraries. Compare performance using hamming loss, precision, recall, and subset accuracy**. (4 hours)**
9. Build a neural network using PyTorch or similarly libraries for multi-label classification with appropriate activation function. Train the model on a multi-label dataset and evaluate using multi-label metrics. (**2 hours)**
10. Explore few-shot classification using pre-trained models and implement semi-supervised learning on a partially labelled dataset to improve performance with limited annotations. **(2 hours)**

DSE207: Computer Vision [3-0-1]

Course title & Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical /Practice		
DSE 206	4	3	0	1		NIL

Course Objectives:

The primary objective of this course is to introduce the fundamentals of image formation & representation and find the solution for the real-time challenges using deep learning.

Course Learning Outcomes:

Upon successful completion of this course, the student will be able to

1. recognize and describe both the theoretical and practical aspects of computing with images.
2. understand image transformations, segmentation, feature detection, motion, matching, recognition, extraction, and categorization from images.
3. understand the geometric relationships between 2D images and the 3D world.
4. analyze the performance of deep networks within an image.

Syllabus:

Unit I: **(11 Hours)**

Overview of Computer Vision and Image Processing: Introduction of computer vision, definition, applications, and importance; historical context and evolution; image acquisition, image formats and representations, image enhancement, image filtering and convolution, RGB, HSV, and other color spaces, color manipulation in images.

Unit II: **(12 Hours)**

Early and Mid-level Vision: translation, rotation, scaling; affine and perspective transformations, thresholding, region-based segmentation, segmentation by clustering, segmentation by fitting a model, segmentation and fitting using probabilistic methods, feature detection and matching: points and patches, edges, lines, edge detection, corner detection, histogram equalization, adaptive histogram equalization, template matching, scale-invariant feature transform (SIFT).

Unit III: **(10 Hours)**

High Level Vision: Geometric methods: model-based vision, smooth surfaces and their outlines, aspect graphs, range data; probabilistic and inferential methods: recognition by relations between templates, geometric templates from spatial relations, application, image-based rendering.

Unit IV: **(12 Hours)**

Dense Motion Estimation: Translational alignment, parametric motion, spline-based motion, optical flow, layered motion.

Deep learning-based Computer Vision Techniques: Deep learning for computer vision, basics of neural networks, convolutional neural networks (CNNs), architecture of CNNs, training and fine-tuning CNNs, LeNet, AlexNet, GooleNet, VGG-Net, ResNet, comparative analysis of different architecture.

Readings:

1. Gonzalez, Rafael C. and Woods, Richard E., *Digital Image Processing*, 4th edition, Pearson Education, 2018.
2. Forsyth and Ponce, *Computer Vision: A Modern Approach*, 2nd edition, Pearson, 2015.
3. Szeliski, Richard, *Computer Vision: Algorithms and Applications*, Springer-Verlag, 2010.
4. Prince, Simon J. D., *Computer Vision: Models, Learning, and Inference*, 1st edition, Cambridge University Press, 2012.
5. Goodfellow, Ian, *Deep Learning (Adaptive Computation and Machine Learning series)*, The MIT Press, 2016

References:

1. Hartley, Richard and Zisserman, Andrew, *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press, 2004.

List of practical:

You may choose a suitable programming language and the necessary/relevant inputs/ problems.

1. Implement a program for filtering and convolution techniques on the image. **(2 Hours)**

2. Implement a program to perform all colour space conversion and colour manipulation techniques in images. **(2 Hours)**

3. Write a program for all image translation, rotation, and scaling techniques. **(3 Hours)**

4. Write a program for all image segmentation techniques. **(4 Hours)**

5. Implement a program to perform all feature detection and matching techniques in images. **(3 Hours)**

6. Implement a program to apply histogram equalization to a colour image, and apply contrast stretching to the colour example image. Experiment with different parameter values to find an optimum for the visualization of this image. **(2 Hours)**

7. Write a program for the scale-invariant feature transform of an image. **(3 Hours)**

8. Implement a program for geometric methods of the image. **(3 Hours)**

9. Implement a program for dense motion estimation techniques on the image. **(2 Hours)**

10. Implement a program to apply different architectures of CNN to images to find the result of the image and investigate the usefulness of different architectures of CNN. **(6 Hours)**

List of GEs for Semester II

GE201: DATA MINING [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
GE201	**4**	**3**	**0**	**1**		**NIL**

Course Objectives:

In this course, the objective is to introduce the KDD process. The course should enable students to translate real-world problems into predictive and descriptive tasks. The course also covers data cleaning and visualization and supervised and unsupervised mining techniques.

Course Learning Outcomes:

At the end of the course, students will be able to

1. distinguish between the process of knowledge discovery and Data Mining.
2. play with basic data exploration methods to develop understanding of given data
3. identify suitable pre-processing method for give problem.
4. describe different data mining tasks and algorithms.
5. use programming tools (e.g. Weka/Python/R etc) for solving data mining tasks.
6. follow formal notations and understand the mathematical concepts underlying data mining algorithms

Syllabus:

Unit-I **(9 hours)**

Overview: The process of knowledge discovery in databases, predictive and descriptive data mining techniques, and unsupervised learning techniques. Data preprocessing: Data cleaning, Data transformation, Data reduction, Discretization

Unit II **(12 hours)**

Classification: Supervised learning/mining tasks, Decision trees, Decision rules, Statistical (Bayesian) classification, Instance-based methods (nearest neighbor), Evaluation and Validation methods.

Unit III **(12 hours)**

Clustering: Basic issues in clustering, Partitioning methods (k-means, expectation maximization), Hierarchical methods for clustering, Density-based methods, Cluster Validation methods and metrics

Unit IV **(12 hours)**

Association Rule Mining: Frequent item set, Maximal and Closed item sets, Apriori property, Apriori algorithm.

Readings:

1. Han, Jiawei, Jian Pei, and Hanghang Tong. *Data Mining: Concepts and Techniques* 3rd edition. Morgan Kaufmann, 2022.
2. Zaki, Mohammed J., WM Jr, *Data Mining and Analysis: Fundamental Concepts and Algorithms,* Cambridge University Press, 2014.
3. Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*, Addison Wesley, 2006.
4. Charu, C. *Data Mining: The Textbook*, Springer, 2015

List of Practicals:

1. Data Preprocessing & Visualization (4 Hours)

- Load and explore real-world datasets using Python/Weka/R
- Perform data cleaning: handle missing values, detect/treat outliers
- Apply data transformation: normalization, encoding categorical variables
- Use data reduction techniques: PCA, sampling
- Apply discretization/binning on continuous features
- Visualize data using histograms, boxplots, scatter plots, and heatmaps

2. Supervised Learning – Classification (4 Hours)

- Implement a Decision Tree classifier and visualise the tree
- Apply Naive Bayes classifier and evaluate accuracy
- Implement k-Nearest Neighbor (k-NN) classifier with different values of *k*
- Evaluate classifiers using confusion matrix, precision, recall, and F1-score
- Perform k-fold cross-validation and plot ROC-AUC curves

3. Unsupervised Learning – Clustering (8 Hours)

- Apply k-Means clustering and determine optimal k using Elbow/Silhouette method
- Perform Agglomerative Hierarchical clustering and plot dendrogram
- Apply DBSCAN for density-based clustering and tune parameters (eps, minPts)
- Visualize and interpret clustering results

4. Association Rule Mining (4 Hours)

- Use Apriori algorithm to find frequent itemsets from transaction data
- Generate association rules using confidence and lift thresholds
- Analyze discovered rules for business decision insights

5. Mini-Project (10 Hours)

- Select a real-world dataset (e.g., healthcare, e-commerce, finance)
- Apply full KDD process: preprocessing, modelling, (classification/clustering/association), and evaluation

- Summarize findings with visualizations and model insights

GE202: Data Science using Python [3-0-1]

Course Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
GE202	**4**	**3**	**0**	**1**		**NIL**

Course Description

This course provides an in-depth introduction to data science using Python, covering foundational libraries, statistics, machine learning, deep learning, and recommendation systems. The course blends theoretical concepts with hands-on projects using real-world datasets.

Course Learning Outcomes

By the end of the course, students will be able to

1. apply Python libraries for data manipulation and visualization.
2. apply statistical and probability concepts to data science problems.
3. build and evaluate machine learning models for regression and classification.
4. understand deep learning architectures like CNNs and Transformers.

Syllabus:

Unit-I **(10 hours)**

Python & Statistical Foundations for Data Science: *Python Foundations:* Introduction to Python for Data Science, Python libraries: NumPy, Pandas, Matplotlib and Seaborn, *Exploratory Data Analysis (EDA):* Summary Statistics & Data Inspection, Detecting Outliers (Z-Score, IQR), Correlation & Covariance; *Introduction to Statistics*: Descriptive vs. Inferential Statistics, Probability Distributions: Binomial, Poisson, Normal, Exponential, Bayes' Theorem & Conditional Probability, Confidence Intervals & Hypothesis Testing

Unit-II **(10 hours)**

Machine Learning – Unsupervised: Introduction to Clustering, K-Means Clustering: Elbow Method, Silhouette Score, Hierarchical Clustering: Agglomerative & Divisive, DBSCAN: Density-Based Clustering, Gaussian Mixture Models (GMMs), Evaluation metrics.

Unit-III **(11 hours)**

Machine Learning – Supervised Learning: Regression, Linear Regression & Polynomial Regression, Multiple Linear Regression, Bias and Variance, Regularisation Techniques, Classification, Logistic Regression, k-Nearest Neighbours (k-NN), Naïve Bayes Classifier, Decision Trees; Overfitting, Underfitting, Cross-Validation, Bootstrapping & Resampling.

Unit-IV: **(14 hours)**

Deep Learning & Recommendation Systems: Introduction to Neural Networks, Perceptron & Multi-Layer Perceptron (MLP), Activation Functions, Backpropagation & Optimization Techniques; Convolutional Neural Networks (CNNs), Convolution & Pooling Layers, Image Classification using CNNs, Transfer Learning with Pre-trained Models; *Transformers & NLP:* Introduction to Transformers, Self-Attention & Positional Encoding, Applications: Sentiment

Analysis, Named Entity Recognition; *Recommendation Systems:* Collaborative Filtering, Content-Based Filtering, Matrix Factorization: Singular Value Decomposition (SVD)

Readings:

1. McKinney, Wes. *Python for Data Analysis: Data Wrangling with Pandas, NumPy and iPython*, 2nd Ed., O'Reilly, 2017.
2. Tan, P., Steinbach, M., Karpatne, A., and Kumar,V. *Introduction to Data Mining*, 2nd Edition, Pearson Education, 2018.
3. Grolemund,G., Wickham, H., *R for Data Science*, 1st Ed., O'Reilly, 2017.
4. Goodfellow, Ian. *Deep Learning*, MIT Press, 2016.

List of Practicals:

A. ***Exploratory Data Analysis and Statistical Inference on Real-World Health Data:*** Download dataset from the World Health Organisation (WHO) Life Expectancy dataset or any open-source COVID-19 dataset and perform the following tasks: **(5 hours)**

1. Load and clean data using Pandas and use Matplotlib/Seaborn for visualisation of data.
2. Compute and save summary statistics and detect outliers using Z-score and IQR.
3. Analyse the correlation between life expectancy and socio-economic indicators.
4. Apply Bayes' Theorem to assess conditional probabilities (e.g., likelihood of high life expectancy given high GDP).
5. Perform hypothesis testing (e.g., test if life expectancy is significantly different between developed and developing countries).

B. ***Customer Segmentation using Unsupervised Learning:*** Apply clustering techniques to segment customers based on their purchasing behaviour. Download the E-commerce dataset: E-commerce customer data and perform the following tasks: **(5 hours)**

1. Normalize data using standard scaling.
2. Apply K-Means Clustering and find optimal k using the Elbow Method and Silhouette Score. Also, visualize clusters using PCA.
3. Perform Hierarchical Clustering and plot dendrograms.
4. Use DBSCAN to identify clusters, noise/outliers in the data.
5. Evaluate clustering performance using internal metrics (Silhouette, Davies-Bouldin).

C. ***Predicting House Prices using Regression Techniques:*** Build regression models to predict house prices. Download Boston Housing dataset or Ames Housing dataset. Perform the following tasks: **(8 hours)**

1. Perform EDA and feature selection.
2. Apply Linear Regression, Polynomial Regression, and Multiple Linear Regression.

3. Evaluate the bias-variance tradeoff.
4. Use Ridge and Lasso Regression to reduce overfitting.
5. Apply *k*-NN, Logistic Regression, Naïve Bayes, and Decision Trees for binary classification (e.g., whether house price is above average).
6. Evaluate models using Cross-Validation and metrics like RMSE, R^2, accuracy, precision, recall.

D. ***Deep Leaning for classification and regression:*** Train a CNN model to classify digits and to predict house process. **(8 hours)**

1. Design and train a 2D CNN model for classification of MNIST or ImageNet dataset.
2. Design and train a DL model for house price prediction using Boston Housing dataset or Ames Housing dataset.
3. Use Transfer Learning with CNN model to boost image classification for a small-sample sized dataset.

E. **Sentiment Analysis on Product Reviews using Transformers:** Analyze customer sentiments from reviews. Download Amazon or IMDB Reviews. Perform the following tasks: **(4 hours)**

1. Use a pre-trained Transformer model (like BERT) to perform **Sentiment Analysis** on reviews.