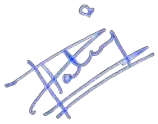
Department of CSE

Details of Lesson Plan

S.No.	Particulars	Details
1.	Course Name	Applied Data Sciences
2.	Course Code	COM – 702 (C)
3.	Academic Year	2024- 2025
4.	Semester	7th
5.	Number of Lesson plans	51
6.	Faculty Assigned	Dr. Palvi Sharma



Faculty Signature

Lesson Plan No. 1	Course Name: Applied Data Science Topic: Introduction to Data Science	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the fundamental concept of Data Science. b. Identify different applications of Data Science across industries. c. Recognize the role of data in decision-making. d. Differentiate between Data Science, AI, and Machine Learning.
Teaching Aids (if any)	a. Infographic on the data science workflow. b. Videos of real-world Data Science applications. c. Use of interactive polls via Nearpod.
Teaching Development	1. **Introduction** (5 minutes) - Ask questions to engage students: - Have you heard of Data Science? - Can you think of areas where data is critical (businesses, healthcare, etc.)? - Introduce the basic concept of Data Science. - Show a simplified data science workflow diagram (from collection to insights). 2. **Development** (30 minutes) a. **What is Data Science?** Discuss the definition and scope of Data Science. Explain the importance of data-driven decisions. b. **Applications of Data Science** Present examples of Data Science applications across various domains: Healthcare: Predictive analytics for disease diagnosis. Retail: Customer behavior analysis for personalized recommendations. Finance: Fraud detection and risk analysis. c. **Key Components of Data Science** Discuss the role of statistics, data analysis, machine learning, and programming in Data Science. Briefly introduce common tools like Python, R, and SQL. d. **Difference between Data Science, AI, and ML** Clarify the relationships and differences between Data Science, Artificial Intelligence, and Machine Learning. 3. **Exercise (5 minutes)** Give different real-life examples (e.g., Netflix recommendations, credit scoring) and ask students to identify whether Data Science, AI, or ML is being used.
Closure	1. Summarize the learning outcomes and affirm understanding. 2. Suggested Reading:

	- "Data Science for Business" by Provost and Fawcett (Introductory chapters). 3. Homework: - Write a short blog post on a real-world application of Data Science and submit it on Google Classroom.
Evaluation	1. Reflective Questions: What is Data Science? How does Data Science benefit organizations? 2. Conduct a short quiz on Nearpod about the applications and basic concepts of Data Science. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 2	Course Name: Applied Data Science Topic: Overview of Data Science and its Applications	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Describe the primary fields where Data Science is applied. b. Understand how Data Science drives decision-making in different industries. c. Explore case studies from different domains.
Teaching Aids (if any)	a. Case study videos from healthcare, finance, and marketing sectors. b. Infographics showing data growth and usage.
Teaching Development	1. **Introduction** (5 minutes) - Ask students about industries where they think data plays an important role. - Briefly introduce how Data Science is revolutionizing industries. 2. **Development** (30 minutes) **a. Discuss major fields such as:** - Healthcare: Predictive models for disease diagnosis, personalized medicine. - Finance: Credit scoring, fraud detection, portfolio optimization. - Retail: Customer segmentation, product recommendation systems. **b. Case Study Discussion:** - Healthcare: Predicting patient readmission rates using data science techniques. - Finance: How financial institutions use predictive analytics to mitigate risks. **c. Impact on Decision Making** - Discuss how Data Science is helping industries make better, faster, and data-driven decisions. 3. **Exercise (5 minutes)** - Provide students with case study scenarios and ask them to describe how Data Science could be applied.
Closure	1. Summarize applications discussed in the lesson. 2. Suggested Reading: - "Practical Data Science with R" by Zumel and Mount (Chapter 1). 3. Homework: - Write a short report on a recent innovation in Data Science that impacted a specific industry.
Evaluation	1. Reflective Questions: - How is Data Science used in healthcare or finance?

	- How does Data Science benefit organizations? 2. Conduct a short quiz on Nearpod about the applications and basic concepts of Data Science. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 3	Course Name: Applied Data Science Topic: Introduction to Data Analysis Using Python and Jupyter Notebooks	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Set up Python and Jupyter Notebooks for data analysis. b. Execute basic Python commands for data manipulation. c. Load datasets in Jupyter and perform basic operations.
Teaching Aids (if any)	a. Python installation guide. b. Sample datasets for practice. c. Live demo using Jupyter Notebook.
Teaching Development	1. **Introduction** (5 minutes) - Ask students if they have used Python before. - Introduce Python as the primary tool for data analysis. 2. **Development** (30 minutes) **a. Python for Data Analysis:** - Install and set up Python and Jupyter Notebook. - Explain the importance of Python in Data Science due to its extensive libraries (Pandas, NumPy). **b. Jupyter Notebooks** - Demonstrate how Jupyter Notebooks work, their advantages (e.g., markdown and code in the same document). - Walk through basic Python operations: variables, loops, and basic data types. **c. Loading Data in Python** - Load a sample dataset (CSV format) using Pandas. - Perform basic operations like reading, describing, and cleaning the data. 3. **Exercise (5 minutes)** - Ask students to load their datasets and perform basic operations.
Closure	1. Summarize the process of using Python and Jupyter for data analysis. 2. Suggested Reading: - "Python for Data Analysis" by Wes McKinney (Chapters 1-2). 3. Homework: - Practice loading and analyzing a dataset in Python and upload the results to Google Classroom.
Evaluation	1. Reflective Questions: - Why is Python useful for data analysis? - Conduct a short quiz using Jupyter cells to test Python skills. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 4	Course Name: Applied Data Science Topic: Data Preprocessing and Cleaning Techniques	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Set up Python and Jupyter Notebooks for data analysis. b. Execute basic Python commands for data manipulation. c. Load datasets in Jupyter and perform basic operations.
Teaching Aids (if any)	a. Python installation guide. b. Sample datasets for practice. c. Live demo using Jupyter Notebook. d. PowerPoint slides to illustrate key concepts. e. Whiteboard for interactive discussion.
Teaching Development	1. **Introduction** (5 minutes) - Ask students: What happens when we feed bad data into machine learning models? - Highlight the significance of data quality in data science. - Discuss the concept of "Garbage in, Garbage out." - Introduce the need for data preprocessing and cleaning techniques. - Briefly overview common issues in real-world datasets, including missing data, noisy data, and outliers. - Show a flowchart of the data science process and emphasize the role of preprocessing. 2. **Development** (30 minutes) **a. Introduction to Data Preprocessing:** - Define data preprocessing and its significance in the data science pipeline - Show real-world examples where improper data cleaning led to poor model performance. - Explain common preprocessing techniques: handling missing data, detecting outliers, and removing noisy data. **b. Handling missing data** - Demonstrate using a dataset with missing values in Jupyter Notebook. - Introduce methods to handle missing data - Provide real-world use cases and discuss the advantages and disadvantages of each method. **c. Handling Outliers** - Explain what outliers are and how they can skew analysis. - Demonstrate outlier detection using visualization tools such as box plots and histograms - Show how to handle outliers using techniques like trimming, capping, or transforming data **d. Dealing with noisy data** - Define noisy data and its impact on model performance.

	- Introduce techniques for smoothing noisy data, such as binning and moving averages. - Show students a demonstration of noise filtering in a dataset using Pandas and numpy functions. 3. **Exercise (5 minutes)** - Ask students to load their datasets and perform basic operations.
Closure	**1. Summarize the Lesson Learning Outcomes:** - Ask students to recap the main techniques for handling missing data, outliers, and noisy data. - Emphasize the role of preprocessing in improving model accuracy and performance. **2. Suggested Reading:** - Documentation on Pandas: https://pandas.pydata.org/ **3. Homework** - Download a real-world dataset from Kaggle or UCI Machine Learning Repository. - Apply data preprocessing techniques learned in class and submit a Jupyter Notebook showing the steps.
Evaluation	**1. Reflective Questions:** - What are the different techniques to handle missing data, and when would you choose each one? - How do outliers affect analysis, and what methods can be used to address them? - What tools did you find most helpful in Pandas for cleaning datasets? **2. Nearpod Quiz on Data Preprocessing:** - Multiple-choice questions to assess understanding of key concepts. - Include a scenario-based question asking students to select the appropriate method for handling noisy data or outliers.

Lesson Plan No. 5	Course Name: Applied Data Science Topic: Exploratory Data Analysis (EDA)	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the concept and importance of Exploratory Data Analysis (EDA). b. Utilize various techniques to summarize and visualize data. c. Perform EDA using Python libraries like Pandas, Matplotlib, and Seaborn. d. Derive meaningful insights from datasets through statistical and graphical techniques.
Teaching Aids (if any)	a. Sample datasets for live EDA demonstration. b. Jupyter Notebook for practical examples. c. PowerPoint slides for key concepts and visual examples. d. Charts and graphs for visualization exercises.
Teaching Development	1. **Introduction** (5 minutes) - Ask students: How do you understand a dataset before applying a machine learning model? - Define Exploratory Data Analysis (EDA) as an approach to analyzing datasets to summarize their main characteristics. - Emphasize that EDA is crucial for understanding data distributions, relationships between variables, and uncovering patterns. - Highlight the goals of EDA: uncovering underlying patterns, spotting anomalies, testing hypotheses, and checking assumptions. - Introduce the tools commonly used in EDA, such as Pandas, Matplotlib, and Seaborn. 2. **Development** (30 minutes) **a. Overview of EDA Techniques:** - Summary Statistics (mean, median, mode, variance, standard deviation). - Data Distributions (histograms, density plots). - Correlation and Covariance (scatter plots, correlation heatmaps). - Boxplots (to detect outliers and understand data spread). **b. Descriptive Statistics:** - Use a sample dataset in Jupyter Notebook to demonstrate how to calculate summary statistics. - Show students how to use Pandas functions like describe (), mean (), and std () to obtain insights from data. - Discuss how descriptive statistics help in understanding the central tendency, dispersion, and overall distribution of the dataset.

	c. Data Visualization Techniques: - Introduce the importance of visualizing data to identify trends, patterns, and anomalies. - Demonstrate how to use Matplotlib and Seaborn for visualizing data distributions **d. Identifying Relationships and Patterns:** - Discuss how EDA can help in discovering relationships between variables (e.g., correlation between age and income). - Explain how these visualizations help inform feature selection for model building. **e. Uncovering Anomalies and Outliers:** - Explain the role of EDA in identifying outliers and anomalies in the data. - Show how boxplots and scatter plots can help in spotting anomalies. - Discuss the importance of outlier detection in improving data quality and subsequent analysis. 3. **Exercise (5 minutes)** - Provide a real-world dataset (e.g., from UCI Machine Learning Repository). - Ask students to calculate summary statistics, create visualizations (histograms, boxplots, and scatter plots), and interpret their findings. - Encourage students to use Pandas, Matplotlib, and Seaborn for this activity in Jupyter Notebook. - Discuss student findings in class, focusing on insights derived from the dataset.
Closure	**1. Summarize the Lesson Learning Outcomes:** - Ask students to recap key EDA techniques: summary statistics, visualizations, and identifying relationships. - Reiterate the importance of EDA in gaining insights from data and preparing it for further analysis. **2. Suggested Reading:** - Seaborn Documentation: https://seaborn.pydata.org/ **3. Homework:** - Select a dataset from Kaggle or UCI Repository and perform a complete EDA. - Include summary statistics, at least three types of visualizations, and a brief analysis of the relationships between variables. - Submit a Jupyter Notebook with your findings.
Evaluation	**1. Reflective Questions:** - How do summary statistics help in understanding the distribution of a dataset? - What are the most useful visualizations for identifying relationships between variables?

Lesson Plan No. 6	Course Name: Applied Data Science Topic: Introduction to Data Analysis using Python	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the basics of data analysis using Python. b. Explore essential Python libraries for data analysis, such as Pandas, NumPy, and Matplotlib. c. Perform basic data analysis tasks like data loading, cleaning, and visualization using Python.
Teaching Aids (if any)	a. Python code examples in Jupyter Notebook. b. Sample datasets for hands-on exercises. c. PowerPoint slides to introduce concepts.
Teaching Development	1. **Introduction** (5 minutes) - Ask students: Why is Python popular for data analysis? - Highlight the flexibility of Python for handling data through libraries like Pandas, NumPy, and Matplotlib. - Introduce Jupyter Notebook as an interactive environment for writing and running Python code, especially for data analysis. 2. **Development** (30 minutes) **a. Getting Started with Python for Data Analysis:** - Discuss the advantages of using Python for data analysis: simplicity, large community, and extensive library support. - Walk students through setting up Jupyter Notebook for data analysis. **b. Exploring Python Libraries:** - Introduce Pandas for data manipulation and analysis (dataframes, series, etc.). - Demonstrate NumPy for numerical computations and array manipulation. - Use Matplotlib for basic data visualizations. - Show examples in Jupyter Notebook to load, manipulate, and visualize a small dataset. **c. Performing Basic Data Analysis** - Load a dataset (e.g., CSV) using Pandas read_csv () function. - Demonstrate basic data manipulation techniques (e.g., selecting columns, filtering rows, handling missing values). - Perform basic statistical analysis (mean, median, count, etc.) using Pandas. **d. Visualizing Data** - Show how to create simple plots like line charts, bar plots, and histograms using Matplotlib. - Explain how visualizing data helps in gaining better insights.

	3. Exercise (5 minutes) - Provide students with a small dataset and ask them to: • Load the dataset using Pandas. • Perform basic statistical analysis. • Create a simple plot (line chart or histogram) using Matplotlib. - Discuss the findings and challenges faced by students.
Closure	**1. Summarize Lesson Outcomes** - Recap how Python is used for data analysis and the key libraries. - Highlight the importance of Jupyter Notebook in streamlining the data analysis workflow. **2. Suggested Reading:** - "Python for Data Analysis" by Wes McKinney (Chapters 1-2). **3. Homework:** - Choose a dataset from Kaggle, load it into a Jupyter Notebook, and perform basic data analysis tasks, including visualization.
Evaluation	**1. Reflective Questions:** - What are the benefits of using Pandas for data manipulation? - How do you visualize data using Matplotlib? - Test basic Python data analysis concepts, including loading datasets, using Pandas, and visualizing with Matplotlib. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 7	Course Name: Applied Data Science Topic: Data Preprocessing and Cleaning Techniques	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Apply advanced techniques for handling missing data. b. Identify and address data inconsistencies and errors. c. Use Python libraries to preprocess large datasets effectively.
Teaching Aids (if any)	a. Large and complex sample datasets. b. Jupyter Notebook for demonstration. c. Visual aids on data cleaning best practices.
Teaching Development	1. **Introduction** (5 minutes) - Recap previous lesson's basic data cleaning techniques. - Ask students: What challenges have you faced when cleaning real-world data? - Introduce the need for advanced data cleaning techniques when dealing with larger, more complex datasets.**Development** (30 minutes) **a. Development:** - Discuss advanced techniques for handling missing data: imputation (mean, median, mode), forward/backward filling, and dropping missing values. - Demonstrate each method with a sample dataset using Pandas (fillna(), dropna() methods). - Discuss common issues like duplicate entries, inconsistent data formats, and incorrect data types. - Show how to identify and handle duplicates using Pandas (drop_duplicates()). - Demonstrate data type conversion and fixing formatting issues (e.g., date/time data - **Jupyter Notebooks** - Demonstrate how Jupyter Notebooks work, their advantages (e.g., markdown and code in the same document). - Walk through basic Python operations: variables, loops, and basic data types. **b. Scaling and Normalization** - Explain the importance of data scaling and normalization for machine learning. - Demonstrate how to normalize data using Python's MinMaxScaler from Scikit-learn **c. Handling Large Datasets** - Introduce strategies for efficiently preprocessing large datasets (e.g., working with chunks of data, memory optimization). - Demonstrate how to load large datasets in chunks using Pandas read_csv() with the chunksize parameter.

	3. **Exercise (5 minutes)** - Give students a large dataset with missing data, inconsistencies, and duplicates. - Handle the missing data using imputation and other methods. - Remove duplicate entries and standardize data formats. - Discuss the challenges and solutions in class.
Closure	1. Summarize the process of using Python and Jupyter for data analysis. - Review advanced data cleaning techniques and their importance. - Highlight the efficient handling of large datasets. 2. Suggested Reading: - Scikit-learn documentation on data preprocessing: https://scikit-learn.org/stable/modules/preprocessing.html 3. Homework: - Apply advanced data cleaning techniques to a large dataset from Kaggle.
Evaluation	1. Reflective Questions: - What are the benefits of imputation over dropping missing data? - How do you handle large datasets that don't fit in memory? - Multiple-choice and scenario-based questions on data cleaning techniques. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 8	Course Name: Applied Data Science Topic: Data Wrangling and Transformation	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the importance of data wrangling for data analysis. b. Use Python tools like Pandas to transform and reshape datasets. c. Combine and merge datasets using Pandas functions.
Teaching Aids (if any)	a. Sample datasets with multiple sources. b. Jupyter Notebook for live demonstration.
Teaching Development	1. **Introduction** (5 minutes) - Ask students: What is data wrangling and why is it crucial for data analysis? - Define data wrangling as the process of cleaning, structuring, and enriching raw data into the desired format. 2. **Development** (30 minutes) **a. Reshaping Data:** - Introduce functions for reshaping data in Pandas (melt(), pivot(), stack(), and unstack()). - Demonstrate reshaping data with examples in Jupyter Notebook **b. Combining Datasets** - Discuss the need for combining data from multiple sources. - Demonstrate merging datasets using Pandas (merge(), concat() functions). - Show how to handle different types of joins (inner, outer, left, right). **c. Data Transformation** - Explain how to transform data by applying functions to columns and rows. - Show examples using Pandas apply() and transform() methods. **d. Handling Time Series Data** - Introduce time series data and show how to handle it in Pandas (date parsing, resampling, etc.). 3. **Exercise (5 minutes)** - Provide a sample dataset that requires reshaping and merging. - Ask students to transform, combine, and reshape the datasets to create a clean and structured dataset.
Closure	1. Summarize Lesson Outcomes - Recap data wrangling techniques and how to reshape, merge, and transform datasets. 2. Suggested Reading:

	- Pandas Documentation on Merging and Reshaping Data: https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html 3. Homework: - Take two different datasets from Kaggle, merge them, and apply transformations to derive insights.
Evaluation	1. Reflective Questions: - Why is Python useful for data analysis? - Conduct a short quiz using Jupyter cells to test Python skills. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 9	Course Name: Applied Data Science Topic: Final Review of Data Science Concepts	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the importance of data preprocessing and cleaning. b. Handle missing data, outliers, and noisy data. c. Use Pandas to preprocess datasets.
Teaching Aids (if any)	a. Sample dirty datasets. b. Live demonstration in Jupyter Notebook.
Teaching Development	1. **Introduction** (5 minutes) - Ask students if they have used Python before. - Introduce Python as the primary tool for data analysis. 2. **Development** (30 minutes) **a. Python for Data Analysis:** - Install and set up Python and Jupyter Notebook. - Explain the importance of Python in Data Science due to its extensive libraries (Pandas, NumPy). **b. Jupyter Notebooks** - Demonstrate how Jupyter Notebooks work, their advantages (e.g., markdown and code in the same document). - Walk through basic Python operations: variables, loops, and basic data types. **c. Loading Data in Python** - Load a sample dataset (CSV format) using Pandas. - Perform basic operations like reading, describing, and cleaning the data. 3. **Exercise (5 minutes)** - Ask students to load their datasets and perform basic operations.
Closure	1. Summarize the process of using Python and Jupyter for data analysis. 2. Suggested Reading: - "Python for Data Analysis" by Wes McKinney (Chapters 1-2). 3. Homework: - Practice loading and analyzing a dataset in Python and upload the results to Google Classroom.
Evaluation	1. Reflective Questions: - Why is Python useful for data analysis? - Conduct a short quiz using Jupyter cells to test Python skills. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 10	Course Name: Applied Data Science Topic: Data Validation Techniques	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand why data validation is crucial. b. Perform data validation using tools like regular expressions and data type checks. c. Validate numerical and categorical data.
Teaching Aids (if any)	a. Python code demonstration for data validation b. Example datasets for validation
Teaching Development	1. **Introduction** (5 minutes) - Ask: "What happens if the data you use doesn't meet your expectations? How can we ensure data correctness?" - Introduce data validation as a critical step before analysis to ensure data accuracy and reliability. 2. **Development** (30 minutes) **a. Introduction to Data Visualization:** - Define data validation as ensuring data correctness, completeness, and consistency. - Discuss why validation is necessary in both input data and during pre-processing. **b. Techniques for Data Visualization:** - Numerical validation (range checks, type checks). - Categorical validation (using regex to match patterns, validating against a set of known categories). **c. Demonstration in Python :** - Use pandas for validating data types and ranges. - Demonstrate validation using regular expressions to check data formats (e.g., email validation). 3. **Exercise (5 minutes)** - Validate numerical and categorical data in a sample dataset.
Closure	1. Recap the importance of validating data and the techniques used. 2. "Python Data Science Handbook" (Data Validation section). 3. Homework: - Validate a provided dataset, ensuring all values meet expected types and formats.
Evaluation	1. Students will be given a dataset with potential validation errors (e.g., incorrect data types, invalid ranges). They will correct the errors using Python and submit their solutions, which will be graded on accuracy and thoroughness. 2. Short-answer questions covering different validation techniques, such as numerical validation, regex-based validation, and type checks. 3. Students will write a reflection on their experience with data validation in this lesson, particularly any challenges they faced and how they overcame them.

	Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 2	Course Name: Applied Data Science Topic: Introduction to Data Acquisition	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the importance of data acquisition. b. Identify different sources of data (APIs, web scraping, databases, files). c. Perform basic data collection using tools like Python and SQL.
Teaching Aids (if any)	a. API demo video b. Use of Google Colab for hands-on coding c. Sample dataset for exercise
Teaching Development	1. **Introduction** (5 minutes) - Ask students where they think the data for a Netflix recommendation comes from. - Introduce the concept of data acquisition. - Discuss real-world sources of data like APIs, databases, and web scraping. 2. **Development** (30 minutes) **a. Data Sources:** - Different types: structured, semi-structured, unstructured. - Introduce APIs, databases (MySQL), and web scraping tools like BeautifulSoup. - Demonstrate basic API call in Python. **b. Connecting to databases:** - Using Python libraries (pandas, sqlalchemy) to fetch data from SQL databases. - Example: Pulling data from a MySQL database. **c. Web Scraping:** - Introduce tools like BeautifulSoup and Selenium for scraping unstructured data. - Discuss ethical considerations and legality. 3. **Exercise (5 minutes)** - Pull data from an open-source API (like weather data). - Demonstrate how to load data into pandas for further processing.
Closure	1. Summarize the lesson by recapping API usage, database connections, and web scraping. - Recap the key steps in the data wrangling process. 2. Suggested reading: Pandas documentation (Data Input/Output section). 3. Homework: - Pull weather data from an API and visualize it using Python.
Evaluation	1. Nearpod quiz: sources of data, API basics, data acquisition methods. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 3	Course Name: Applied Data Science Topic: Data Integration	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the importance of integrating data from multiple sources. b. Perform data merging and concatenation. c. Handle schema mismatches and overlapping keys.
Teaching Aids (if any)	a. Video tutorial on data merging b. Python (pandas) for hands-on exercises
Teaching Development	1. **Introduction** (5 minutes) - Ask students how different services like Uber integrate maps, payment systems, and ride data. - Define data integration and its challenges. 2. **Development** (30 minutes) **a. Merging Data:** - Introduce pandas merge and concat functions. - Illustrate merging with different join types (inner, outer, left, right). **b. Handling Schema Mismatches:** - Discuss key errors and column name mismatches when integrating. - Show how to rename columns and handle missing keys. **c. Real world example:** - Integrating sales and customer data for analysis. 3. **Exercise (5 minutes)** - Merge two datasets (sales and customer data) with mismatched keys.
Closure	1. Summarize key points: merging strategies, schema mismatch handling. 2. Suggested reading: Pandas documentation (Data Input/Output section). 3. Homework: - Merge two CSV files (provided) and find missing keys.
Evaluation	1. Nearpod quiz: types of joins, handling schema mismatches. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 4	Course Name: Applied Data Science Topic: Handling Missing Data	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Identify different types of missing data (MCAR, MAR, MNAR). b. Handle missing data using techniques like imputation or removal. c. Decide the best strategy for different scenarios.
Teaching Aids (if any)	a. Video: Missing Data in Real-world Datasets b. Use of pandas for handling missing data
Teaching Development	1. **Introduction** (5 minutes) - Ask: "Have you ever seen incomplete forms? What if data is missing in research?" - Introduce the concept of missing data and its types. 2. **Development** (30 minutes) **a. Detecting Missing data :** - Using pandas isnull() and dropna() to identify and drop missing data. **b. Imputation Techniques:** - Mean, median, mode imputation; forward fill, backward fill. - Demonstrate simple imputation techniques using pandas. **c. Real world example:** - Handling missing data in a healthcare dataset. 3. **Exercise (5 minutes)** - Identify and handle missing data in a dataset using pandas.
Closure	1. Summarize the lesson by discussing imputation and dropping strategies. 2. Suggested reading: Pandas documentation (Handling Missing Data). 3. Homework: - Choose an imputation method for a dataset with missing values and explain why.
Evaluation	1. Nearpod quiz on missing data strategies. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5	Course Name: Applied Data Science Topic: Handling Outliers	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Detect outliers in a dataset using statistical methods. b. Handle outliers by removal or transformation techniques. c. Understand when to keep or remove outliers in analysis.
Teaching Aids (if any)	a. Video tutorial on outlier detection b. Python for hands-on coding
Teaching Development	1. **Introduction** (5 minutes) - Ask: "What happens if an extreme value skews analysis?" - Introduce the concept of outliers and their impact. 2. **Development** (30 minutes) **a. Detecting Outliers :** - Use of IQR and Z-score to detect outliers. - Visualizing outliers using box plots. **b. Handling Outliers:** - Removing or transforming outliers. - Demonstrate techniques like log transformation or Winsorizing. 3. **Exercise (5 minutes)** - Detect and handle outliers in a given dataset.
Closure	1. Recap outlier detection and handling techniques. 2. Suggested reading: Python Data Science Handbook, Chapter on Outliers. 3. Homework: - Find and remove outliers in a dataset and explain your choices.
Evaluation	1. Nearpod quiz on outlier detection methods. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 6	Course Name: Applied Data Science Topic: Feature Engineering	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of feature engineering. b. Create new features using existing data. c. Apply transformations to improve model performance.
Teaching Aids (if any)	a. Video on feature creation b. Python for hands-on exercises
Teaching Development	1. **Introduction** (5 minutes) - Ask: "How can you make better predictions using existing data?" - Introduce feature engineering as a way to enhance model performance. 2. **Development** (30 minutes) **a. Creating Features:** - Examples: deriving new features (e.g., date → year, month). - Feature scaling and normalization techniques (min-max, z-score). **b. Tranformations:** - Using log, square root, and other transformations to handle skewed data. 3. **Exercise (5 minutes)** - Create new features in a given dataset and apply transformations.
Closure	1. Recap feature engineering techniques. 2. Feature Engineering and Selection (book). 3. Homework: - Create two new features from a given dataset and justify their use.
Evaluation	1. Nearpod quiz on feature engineering. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 7	Course Name: Applied Data Science Topic: Data Transformation	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand why data transformation is necessary. b. Apply transformation techniques to prepare data for analysis. c. Perform basic transformations like log, square root, and binning.
Teaching Aids (if any)	a. Python code demonstration for transformations b. Figures showing skewed vs. transformed data distributions
Teaching Development	1. **Introduction** (5 minutes) - Start by asking students, "Have you ever seen data that is not evenly distributed? How would you make it easier to analyze?" - Introduce the concept of data transformation and its necessity in data preparation. - Explain the impact of skewed data on statistical analysis and machine learning models. 2. **Development** (30 minutes) **a. Introduction to Data Transformation:** - Define data transformation as modifying data values to better fit an analysis framework. - Introduce log, square root, and cube root transformations for skewed data. **b. Binning:** - Explain how binning (categorizing continuous data into intervals) can simplify analysis. - Show an example of binning age data into different age groups (child, adult, senior). **c. Demonstration in Python:** - Walk students through code to apply log transformation and binning using pandas. - Show the effect of transformations on the skewness of data through visualizations (e.g., histograms). 3. **Exercise (5 minutes)** - Transform a skewed dataset by applying log and binning techniques.
Closure	1. Recap key points: data transformation, when to use it, and techniques such as log transformation and binning. 2. Data Science from Scratch" (Chapter on Data Wrangling). 3. Homework: - Apply transformation to a skewed dataset, visualize it, and explain how the transformation improved its distribution.
Evaluation	1. Students will be given a dataset with skewed data and asked to apply log transformation, square root transformation, and binning techniques. The exercise will be graded based on correctness and ability to explain the reasoning behind each transformation.

	2. Each group will analyze a case study where data transformation played a key role in improving model performance. They will present their findings to the class, focusing on why certain transformations were chosen. 3. In pairs, students will transform a real-world dataset (e.g., customer sales data) and write a report on how the transformation improved data quality for analysis. The report should include before and after visualizations and justifications for the transformations used. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 8	Course Name: Applied Data Science Topic: Data Normalization and Scaling	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Differentiate between normalization and scaling. b. Apply scaling techniques such as min-max and z-score normalization. c. Understand when to use normalization or scaling in data preparation.
Teaching Aids (if any)	a. Python code demonstration b. Figures illustrating the impact of normalization and scaling on data distributions
Teaching Development	1. **Introduction** (5 minutes) - Ask students, "What if different columns in a dataset are on different scales (e.g., salary in thousands, age in years)? How might this affect machine learning models?" - Introduce the importance of scaling and normalization for ensuring models handle features appropriately. 2. **Development** (30 minutes) **a. Normalization vs Scaling:** - Define normalization as adjusting values to a common range (e.g., [0, 1]) and scaling as adjusting values to a common statistical measure (e.g., mean, standard deviation). - Discuss when to use normalization (e.g., algorithms like k-NN) vs. scaling (e.g., linear regression). **b. Min Max Scaling:** - Explain how min-max scaling adjusts values to a [0, 1] range. - Demonstrate min-max scaling in Python using sklearn. **c. Z-score Normalization:** - Define z-score normalization as adjusting values based on mean and standard deviation. - Demonstrate z-score normalization in Python using sklearn. **d. Effect on Machine Learning Model:** - Show an example where data without scaling leads to poor model performance, while scaling improves results. 3. **Exercise (5 minutes)** - Normalize and scale features from a sample dataset using Python.
Closure	1. Recap the difference between normalization and scaling, and the common methods used for each. 2. “Python Data Science Handbook” (Scaling and Normalization section).

	3. Homework: - Apply transformation to a skewed dataset, visualize it, and explain how the transformation improved its distribution.
Evaluation	1. Short questions on the differences between normalization and scaling, and when each technique should be used. 2. Students will be tasked with normalizing and scaling a dataset using Python. They'll be graded on the accuracy of the implementation and the interpretation of the scaled data's impact on a machine learning model (e.g., logistic regression). Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 9	Course Name: Applied Data Science Topic: Data Quality Assessment	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the importance of data quality in analytics. b. Assess data quality using methods such as completeness, consistency, and accuracy. c. Use tools like pandas for checking data quality.
Teaching Aids (if any)	a. Python for hands-on data quality checks b. Sample datasets with quality issues
Teaching Development	1. **Introduction** (5 minutes) - Ask: "How important is the quality of the data you use for analysis? What if there are errors in the data?" - Introduce the concept of data quality and why poor data quality can result in misleading analyses. 2. **Development** (30 minutes) **a. Characteristics of Data Quality:** - Explain data quality dimensions: completeness, consistency, accuracy, timeliness. - Illustrate each characteristic with examples (e.g., missing values, inconsistent formats, inaccurate entries). **b. Data Quality Assessment in Python:** - Show how to assess data quality using pandas functions like isnull(), describe(), and unique(). - Discuss how to detect and handle duplicates and inconsistent data formats. **c. Real-worl Example:** - Examine a dataset with missing values and duplicates, and assess its quality. 3. **Exercise (5 minutes)** - Perform a data quality assessment on a sample dataset using pandas.
Closure	1. Recap the importance of assessing data quality and the key dimensions of quality. 2. Python for Data Analysis" (Chapter on Data Cleaning and Quality). 3. Homework: - Assess the quality of a provided dataset and describe the issues found.
Evaluation	1. Students will assess the quality of a new dataset provided in class, identifying issues like missing data, duplicates, and inconsistencies. They will submit a report detailing the issues found and suggest appropriate solutions. 2. Write a 500-word essay explaining why data quality is crucial in machine learning and analytics. Include examples from real-world cases.

	3. In groups, students will discuss the challenges of maintaining data quality in large datasets and present strategies for ensuring data remains high-quality in real-time applications. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 11	Course Name: Applied Data Science Topic: Model Selection & Hyperparameter Tuning	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. a. Understand the process of selecting the best model for a given problem. b. b. Use grid search and random search for hyperparameter tuning. c. c. Apply regularization techniques to improve model performance.
Teaching Aids (if any)	a. Slides on the importance of model selection and hyperparameter tuning. b. Jupyter Notebook for hands-on demonstration of tuning techniques. c. Video tutorial on grid search and regularization techniques.
Teaching Development	1. **Introduction** (5 minutes) - Discuss the need for selecting the best model among various algorithms (e.g., decision tree vs. random forest vs. SVM). - Introduce hyperparameter tuning and its role in improving model performance. - Discuss common hyperparameters to tune in machine learning models (e.g., learning rate, number of trees, max depth). 2. **Development** (30 minutes) **a. Model Selection:** - Discuss strategies for model selection based on the problem and dataset (e.g., evaluating multiple models with cross-validation). - Demonstrate how to compare different models on a single dataset using Scikit-learn **b. Hyperparameter Tuning:** - Introduce grid search and random search for hyperparameter tuning. - Walk through an example of using grid search to tune a decision tree or random forest in Python. **c. Regularization Techniques:** - Explain regularization (L1, L2) and how it helps prevent overfitting. - Demonstrate how to apply regularization techniques in Python. 3. **Exercise (5 minutes)** - Provide a dataset for students to build multiple models, tune hyperparameters, and select the best model.
Closure	1. Recap model selection strategies and the importance of hyperparameter tuning and regularization.

	2. Suggested Reading: "Machine Learning Yearning" by Andrew Ng (Section on model tuning). 3. Homework: - Perform grid search or random search on a given model and dataset. Submit the results, including the best hyperparameters and model performance.
Evaluation	1. Lab Task: Submit the model selection and hyperparameter tuning results. 2. Quiz: Explain the difference between grid search and random search, and when to use regularization techniques. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 2	Course Name: Applied Data Science Topic: Linear Regression Algorithm (Supervised Learning)	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Explain the concept of regression and how linear regression works. b. Apply linear regression to a dataset and interpret the results.
Teaching Aids (if any)	a. Jupyter Notebook for live coding demonstration. b. Graphs to visualize regression.
Teaching Development	1. **Introduction** (5 minutes) - Ask students for examples where prediction (regression) is needed (e.g., house prices, stock market). 2. **Development** (30 minutes) **a. Regression Concept:** - Explain regression and the idea of predicting continuous values. **b. Linear Regression:** - Discuss the linear equation - $y = mx + c$ and how to fit a line through data points. - Demonstrate linear regression with a small dataset in Jupyter Notebook. 3. **Exercise (5 minutes)** - Provide a dataset for students to practice linear regression using Python.
Closure	1. Recap the concept of regression and linear regression. 2. Suggested Reading: "Python Data Science Handbook" (Chapter on Regression). 3. Homework: - Complete a linear regression exercise on a given dataset.
Evaluation	1. Practical Coding Exercise: Students will submit code performing linear regression. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 3	Course Name: Applied Data Science Topic: Logistic Regression Algorithm (Supervised Learning)	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the logistic regression algorithm. b. Apply logistic regression to a binary classification problem.
Teaching Aids (if any)	a. Logistic function visualization. b. Example use-cases (e.g., spam detection).
Teaching Development	1. **Introduction** (5 minutes) - Discuss classification vs. regression and introduce logistic regression for binary classification. 2. **Development** (30 minutes) **a. Logistic Regression Concept:** - Explain the sigmoid/logistic function and how it maps outputs between 0 and 1. **b. Application:** - Demonstrate logistic regression on a dataset (e.g., predicting pass/fail based on scores). 3. **Exercise (5 minutes)** - Students apply logistic regression to a dataset.
Closure	1. Recap the key concepts of logistic regression. 2. Suggested Reading: "Introduction to Machine Learning with Python." 3. Homework: - Use logistic regression to solve a classification problem and submit a report.
Evaluation	1. Multiple choice questions on logistic regression concepts. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 4	Course Name: Applied Data Science Topic: K-Nearest Neighbors (KNN) Algorithm (Supervised Learning)	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the KNN algorithm. b. Apply KNN to solve classification problems.
Teaching Aids (if any)	a. Slides with KNN visualization. b. Jupyter Notebook for demonstration.
Teaching Development	1. **Introduction** (5 minutes) - Ask: How do humans classify objects based on similarity? Introduce KNN as a proximity-based algorithm. 2. **Development** (30 minutes) **a. KNN Concept:** - Explain the KNN algorithm and how it classifies based on nearest neighbors. **b. Application:** - Demonstrate KNN using a simple dataset (e.g., iris dataset) in Python. 3. **Exercise (5 minutes)** - Students will classify a dataset using the KNN algorithm.
Closure	1. Recap KNN and its uses. 2. Suggested Reading: "Data Mining Concepts and Techniques." 3. Homework: - Write a short explanation on the advantages and limitations of KNN.
Evaluation	1. Practical Task: Submit the KNN classification solution for a given dataset. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5	Course Name: Applied Data Science Topic: Decision Trees (Supervised Learning)	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Explain the decision tree algorithm. b. Build and visualize a decision tree for classification.
Teaching Aids (if any)	a. Visual representation of decision trees. b. Jupyter Notebook for live coding.
Teaching Development	1. **Introduction** (5 minutes) - Discuss scenarios where decisions are based on rules (e.g., if-else conditions). Introduce decision trees. 2. **Development** (30 minutes) **a. Decision Tree Concept:** - Explain how decision trees work, using recursive binary splits to make decisions. **b. Application:** - Build a decision tree on a dataset and visualize the tree structure. 3. **Exercise (5 minutes)** - Provide a dataset for students to build their own decision trees.
Closure	1. Summarize decision trees and their applications. 2. Suggested Reading: "Machine Learning with Scikit-Learn and TensorFlow." 3. Homework: - Use a decision tree to solve a classification problem.
Evaluation	1. Coding Task: Build a decision tree and submit code and visualizations. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 6	Course Name: Applied Data Science Topic: Support Vector Machines (SVM)	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the basics of SVM. b. Apply SVM for binary classification.
Teaching Aids (if any)	a. SVM margin visualization slides. b. Jupyter Notebook for demo.
Teaching Development	1. **Introduction** (5 minutes) - Ask: How can we find a clear boundary between two classes of data? Introduce SVM. 2. **Development** (30 minutes) **a. SVM Concept:** - Implement SVM on a binary classification problem in Python. **b. Application:** - Build a decision tree on a dataset and visualize the tree structure. 3. **Exercise (5 minutes)** - Students practice applying SVM on a dataset.
Closure	1. Recap SVM concepts. 2. Suggested Reading: "Understanding Machine Learning: From Theory to Algorithms." 3. Homework: - Apply SVM to another binary classification dataset.
Evaluation	1. Quiz: Short answers on SVM concepts and its application. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 7	Course Name: Applied Data Science Topic: Clustering Techniques (Unsupervised Learning)	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Explain clustering and its use in unsupervised learning. b. Apply the K-Means algorithm for clustering.
Teaching Aids (if any)	a. Visualizations of clustering. b. Jupyter Notebook for demonstration.
Teaching Development	1. **Introduction** (5 minutes) - Discuss scenarios where we need to group data without labels (e.g., customer segmentation). 2. **Development** (30 minutes) **a. Clustering Concept:** - Explain clustering and introduce K-Means. **b. K-means Algorithm:** - Demonstrate the K-Means algorithm on a dataset (e.g., grouping customers based on purchase behavior 3. **Exercise (5 minutes)** - Provide a dataset for students to apply K-Means clustering.
Closure	1. Summarize clustering and its applications. 2. Suggested Reading: "Unsupervised Learning Algorithms." 3. Homework: - Apply clustering to a different dataset and submit results.
Evaluation	1. Coding Task: Submit the solution for K-Means clustering on a new dataset. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 8	Course Name: Applied Data Science Topic: Hierarchical Clustering (Unsupervised Learning)	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the hierarchical clustering algorithm and its two types: agglomerative and divisive. b. Perform hierarchical clustering and visualize the results using dendrograms. c. Compare the advantages and limitations of hierarchical clustering against other clustering algorithms like K-Means.
Teaching Aids (if any)	a. Hierarchical clustering tree and dendrogram visualizations. b. Jupyter Notebook for live coding. c. Dataset examples for hierarchical clustering (e.g., customer segmentation dataset). d. Video explaining hierarchical clustering intuitively.
Teaching Development	1. **Introduction** (5 minutes) - Review clustering concepts from the previous lesson on K-Means. - Introduce hierarchical clustering by discussing scenarios where hierarchical structure matters (e.g., taxonomy in biology). - Compare with flat clustering techniques like K-Means. 2. **Development** (30 minutes) **a. Hierarchical Clustering Concept:** - Agglomerative (bottom-up approach): Start with each point as a separate cluster and merge them. - Divisive (top-down approach): Start with all points in one cluster and divide them. **b. Dendograms:** - Introduce dendrograms as a tool to visualize the hierarchical structure of data and how to interpret them. - Demonstrate hierarchical clustering using Python (SciPy library) with dendrogram generation for a given dataset. 3. **Application** - Walk through a hierarchical clustering example on a dataset (e.g., using customer data for segmentation). 4. **Exercise (5 minutes)** - Allow students to perform hierarchical clustering on a dataset and generate their own dendrograms.
Closure	1. Summarize the key points: advantages of hierarchical clustering (e.g., no need to pre-define the number of clusters) and its limitations (e.g., complexity with large datasets). 2. Suggested Reading: "Cluster Analysis: Basic Concepts" and review articles on clustering methods.

	3. Homework: - Apply hierarchical clustering to a dataset of choice and generate a dendrogram. Submit results with explanations.
Evaluation	1. Lab Task: Submit hierarchical clustering results, including dendrograms and cluster interpretations. 2. Short Quiz: Explain the difference between agglomerative and divisive clustering. Compare with K-Means. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 9	Course Name: Applied Data Science Topic: Principal Component Analysis (PCA)	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the concept of dimensionality reduction and its importance in machine learning. b. Learn and apply the Principal Component Analysis (PCA) technique to reduce dimensions. c. Analyze the trade-off between dimensionality reduction and data loss.
Teaching Aids (if any)	a. Graphical illustrations of high-dimensional data projection. b. Jupyter Notebook for PCA implementation. c. Real-world dataset for demonstrating PCA (e.g., iris dataset, MNIST dataset). d. Video tutorial on how PCA works mathematically.
Teaching Development	1. **Introduction** (5 minutes) - Ask students: What happens when you have too many features in your data? Discuss issues such as the curse of dimensionality and overfitting. - Introduce dimensionality reduction as a technique to address these problems. - Provide real-world examples where dimensionality reduction is crucial (e.g., image processing, gene expression data analysis). 2. **Development** (30 minutes) **a. PCA Concept:** - Explain the goal of PCA: to transform high-dimensional data into fewer dimensions while preserving as much variance as possible. - Discuss the mathematics behind PCA (eigenvalues, eigenvectors) in a simplified manner for intuitive understanding. - Visualize the PCA transformation using a 2D scatter plot of high-dimensional data projected onto principal components. **b. Practical Demonstration:** - Perform PCA on a dataset (e.g., iris or MNIST) using Python. Show how the data is transformed into lower dimensions and explain the proportion of variance retained. 3. **Exercise (5 minutes)** - Provide students with a dataset to perform PCA, reduce dimensions, and visualize the transformed data.
Closure	1. Recap the key points about PCA and the trade-offs between reducing dimensions and potential data loss. 2. Suggested Reading: "Pattern Recognition and Machine Learning" (Chapter on PCA) and articles on dimensionality reduction.

	3. Homework: - Perform PCA on a new dataset and visualize the results in lower dimensions. Submit a report on the retained variance and how it impacts the data.
Evaluation	1. Lab Task: Submit PCA results, including visualizations of transformed data and analysis of variance retained. 2. Short Quiz: Describe the purpose of PCA and when to use it. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 11	Course Name: Applied Data Science Topic: Model Selection & Hyperparameter Tuning	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. a. Understand the process of selecting the best model for a given problem. b. b. Use grid search and random search for hyperparameter tuning. c. c. Apply regularization techniques to improve model performance.
Teaching Aids (if any)	a. Slides on the importance of model selection and hyperparameter tuning. b. Jupyter Notebook for hands-on demonstration of tuning techniques. c. Video tutorial on grid search and regularization techniques.
Teaching Development	1. **Introduction** (5 minutes) - Discuss the need for selecting the best model among various algorithms (e.g., decision tree vs. random forest vs. SVM). - Introduce hyperparameter tuning and its role in improving model performance. - Discuss common hyperparameters to tune in machine learning models (e.g., learning rate, number of trees, max depth). 2. **Development** (30 minutes) **a. Model Selection:** - Discuss strategies for model selection based on the problem and dataset (e.g., evaluating multiple models with cross-validation). - Demonstrate how to compare different models on a single dataset using Scikit-learn **b. Hyperparameter Tuning:** - Introduce grid search and random search for hyperparameter tuning. - Walk through an example of using grid search to tune a decision tree or random forest in Python. **c. Regularization Techniques:** - Explain regularization (L1, L2) and how it helps prevent overfitting. - Demonstrate how to apply regularization techniques in Python. 3. **Exercise (5 minutes)** - Provide a dataset for students to build multiple models, tune hyperparameters, and select the best model.
Closure	1. Recap model selection strategies and the importance of hyperparameter tuning and regularization.

	2. Suggested Reading: "Machine Learning Yearning" by Andrew Ng (Section on model tuning). 3. Homework: - Perform grid search or random search on a given model and dataset. Submit the results, including the best hyperparameters and model performance.
Evaluation	1. Lab Task: Submit the model selection and hyperparameter tuning results. 2. Quiz: Explain the difference between grid search and random search, and when to use regularization techniques. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 10	Course Name: Applied Data Science Topic: Model Evaluation Techniques	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand various model evaluation techniques, including cross-validation and confusion matrices. b. Apply metrics like accuracy, precision, recall, F1-score, and AUC-ROC to assess model performance. c. Use cross-validation to evaluate the robustness of machine learning models.
Teaching Aids (if any)	a. Confusion matrix visualizations and metrics formulas. b. Jupyter Notebook for live coding and demonstration of evaluation metrics. c. Example datasets for classification (e.g., loan approval or medical diagnosis datasets). d. Video explaining AUC-ROC curve and interpretation.
Teaching Development	1. **Introduction** (5 minutes) - Start with a discussion on the importance of evaluating machine learning models before deploying them. - Introduce common evaluation techniques and metrics used for classification and regression problems. 2. **Development** (30 minutes) **a. Evaluation Metrics for Classification:** - Introduce accuracy, precision, recall, and F1-score, and explain how they are derived from a confusion matrix. - Explain when to use which metric (e.g., F1-score in imbalanced datasets). - Demonstrate the confusion matrix and metric calculations using a Python library like Scikit-learn on a classification dataset. **b. Cross-Validation:** - Explain the concept of cross-validation and why it is used to assess model robustness. - Demonstrate k-fold cross-validation using Python and interpret the results. **c. AUC-ROC Curve:** - Explain the AUC-ROC curve and how to interpret it for binary classification problems. - Demonstrate the generation of an AUC-ROC curve in Python and explain its significance. 3. **Exercise (5 minutes)** - Provide students with a dataset to build a model, calculate evaluation metrics, and generate a confusion matrix. Include cross-validation.

Closure	1. Summarize the importance of model evaluation and selecting appropriate metrics based on the problem. 2. Suggested Reading: "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" (Chapter on model evaluation). 3. Homework: - Apply cross-validation and compute evaluation metrics on a different dataset. Submit the results with an analysis.
Evaluation	1. Practical Task: Submit a report with the confusion matrix, evaluation metrics, and cross-validation results for a model. 2. Quiz: Questions on when to use precision vs. recall and how to interpret AUC-ROC curves. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 1	Course Name: Applied Data Science Topic: Introduction to Data Visualization	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the role and importance of data visualization in data analysis. b. Learn about the basic principles of effective data visualization. c. Identify common types of data visualizations (e.g., bar charts, line charts, scatter plots) and their uses.
Teaching Aids (if any)	a. Slides explaining data visualization principles. b. Example visualizations (good and bad) for discussion. c. Articles or videos on why data visualization is critical for decision-making.
Teaching Development	1. **Introduction** (5 minutes) - Ask students: "Why is data visualization important in data analysis?" - Provide real-world examples where visualizations helped communicate insights (e.g., COVID-19 dashboards). - Explain the key goals of data visualization: simplifying complex data, identifying patterns, and supporting decision-making. 2. **Development** (30 minutes) **a. Explain common chart types:** - Introduce the principles of effective data visualization, including clarity, simplicity, and avoiding distortion. - Bar charts for categorical data comparisons. - Line charts for trends over time. - Scatter plots for relationships between variables. - Pie charts for part-to-whole relationships (with caution about their misuse). **b. Activity:** - Show examples of good and bad visualizations, and lead a discussion on what makes a visualization effective or misleading. - Have students critique visualizations and suggest improvements. 3. **Exercise (5 minutes)** - Provide a dataset for students to build multiple models and perform data visualization
Closure	1. Recap the key principles of effective data visualization. 2. Suggested Reading: "The Visual Display of Quantitative Information" by Edward Tufte (Chapter on principles of graphical excellence). 3. Homework: - Find examples of visualizations in real-world reports (e.g., news, business reports) and critique their effectiveness.

Evaluation	1. Quiz: Identify the correct type of chart to use for specific data scenarios (e.g., trend data, categorical comparisons). Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 2	Course Name: Applied Data Science Topic: Introduction to Matplotlib	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the basics of the Matplotlib library. b. Create basic plots like line charts, bar charts, and scatter plots using Matplotlib. c. Customize visualizations by adding titles, labels, and legends.
Teaching Aids (if any)	a. Jupyter Notebook for live coding with Matplotlib. b. Example datasets (e.g., stock prices, population data). c. Video tutorials on the basics of Matplotlib.
Teaching Development	1. **Introduction** (5 minutes) - Review the importance of visualizing data and introduce Matplotlib as a fundamental library for creating visualizations in Python. - Explain the structure of a Matplotlib figure (Figure, Axes, and plotting elements). 2. **Development** (30 minutes) **a. Basic Plotting:** - Demonstrate how to create simple plots (line charts, bar charts, scatter plots) using Matplotlib. - Explain key functions like plot(), bar(), and scatter(). **b. Customization:** - Show how to customize plots by adding titles, x and y labels, gridlines, and legends. - Demonstrate how to change the appearance of plots (e.g., line styles, colors, markers). 3. **Exercise (5 minutes)** - Students create basic visualizations (line chart, bar chart, scatter plot) from a dataset provided in class (e.g., stock prices or temperature data).
Closure	1. Recap the basic plotting features of Matplotlib and how to customize visualizations. 2. Suggested Reading: "Python Data Science Handbook" (Chapter on Matplotlib basics). 3. Homework: - Create a bar chart, line chart, and scatter plot from a dataset of your choice and customize them with titles, labels, and colors.
Evaluation	1. Lab Task: Submit visualizations created with Matplotlib, including customizations. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 3	Course Name: Applied Data Science Topic: Advanced Plotting with Matplotlib	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Create subplots and multi-plot figures using Matplotlib. b. Generate advanced plots such as histograms and box plots. c. Use annotations and text to highlight specific data points in visualizations.
Teaching Aids (if any)	a. Jupyter Notebook for live coding. b. Example datasets (e.g., sales data, housing prices). c. Video tutorials on subplots and advanced visualizations.
Teaching Development	1. **Introduction** (5 minutes) - Review basic Matplotlib plots from the previous lesson. - Introduce the need for more complex visualizations, including multiple plots in a single figure (subplots). 2. **Development** (30 minutes) **a. Sub-plot and Multi-Plot Figures:** - Demonstrate how to create subplots using the subplot() function in Matplotlib. - Show examples of visualizing different aspects of data in a single figure. **b. Advanced Plotting Techniques:** - Explain and demonstrate how to create histograms to show data distributions. - Show how to create box plots for visualizing the spread and outliers in data. **c. Annotations:** - Demonstrate how to annotate important points on a plot using annotate () and add text labels to visualizations. 3. **Exercise (5 minutes)** - Students will create a multi-plot figure showing histograms and box plots for a dataset of their choice.
Closure	1. Summarize the process of creating subplots and advanced visualizations. 2. Suggested Reading: Matplotlib documentation on advanced plotting. 3. Homework: - Create a figure with multiple subplots, including a histogram and box plot, from a provided dataset.
Evaluation	1. Lab Task: Submit advanced visualizations with subplots and annotations. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 4	Course Name: Applied Data Science Topic: Introduction to Seaborn	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the basics of Seaborn and its advantages over Matplotlib for statistical plotting. b. Create statistical visualizations using Seaborn, such as bar plots, count plots, and violin plots. c. Customize Seaborn visualizations with themes and color palettes.
Teaching Aids (if any)	a. Jupyter Notebook for live coding with Seaborn. b. Example datasets (e.g., titanic, iris datasets). c. Video tutorials on Seaborn basics.
Teaching Development	1. **Introduction** (5 minutes) - Explain the limitations of Matplotlib and introduce Seaborn as a higher-level library for statistical data visualization. - Briefly review the types of plots available in Seaborn (e.g., categorical plots, relational plots). 2. **Development** (30 minutes) **a. Basic Plotting in Seaborn:** - Demonstrate how to create basic plots in Seaborn, such as barplot(), countplot(), and violinplot(). - Explain the advantages of Seaborn for handling data frames and statistical analysis. **b. Customization:** - Show how to customize Seaborn plots with themes (set_theme()) and color palettes (set_palette()). 3. **Exercise (5 minutes)** - Students will create Seaborn visualizations (bar plot, violin plot) from a dataset provided in class and customize the theme and colors.
Closure	1. Recap the key advantages of Seaborn over Matplotlib for statistical plotting. 2. Suggested Reading: Seaborn documentation and tutorials on basic plots. 3. Homework: - Create a Seaborn plot using the Titanic dataset and customize the color palette and theme.
Evaluation	1. Lab Task: Submit Seaborn visualizations, including customizations with themes and color palettes.. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5	Course Name: Applied Data Science Topic: Storytelling with Data	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the principles of storytelling with data. b. Construct a narrative around a dataset using visualization. c. Identify the key message to be communicated through visuals.
Teaching Aids (if any)	a. Sample datasets for storytelling. b. PowerPoint presentation on data storytelling.
Teaching Development	1. **Introduction** (5 minutes) - Ask: Why is storytelling important in data analysis? - Introduce storytelling as a way to convey complex insights simply. 2. **Development** (30 minutes) - Explain the structure of a data story (beginning, middle, end). - Provide an example: a sales data report highlighting trends, and create a story around the visualizations.. 3. **Exercise (5 minutes)** - Ask students to create a basic narrative around a small dataset.
Closure	1. Summarize learning outcomes. 2. Suggested reading: "Storytelling with Data" by Cole Nussbaumer Knaflic. 3. Homework: Create a data story with 3-4 visualizations. - Create a data story with 3-4 visualizations.
Evaluation	1. Students will create an interactive scatter plot using a dataset of their choice. 2. Evaluation will focus on their ability to add interactivity (hover tooltips, axis manipulation) and make basic customizations (e.g., colors, titles). 3. Students should be able to correctly save and display their interactive plot using Plotly. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 6	Course Name: Applied Data Science Topic: Storytelling with Data: Crafting a Narrative through Visualizations	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the principles of data storytelling and its importance in conveying insights. b. Identify and apply key elements for effective data storytelling (context, audience, narrative structure). c. Create compelling data visualizations to support a story using real datasets. d. Communicate data insights effectively to different audiences.
Teaching Aids (if any)	a. Sample storytelling video (e.g., TED Talks on data storytelling) b. Example data visualizations with narratives (e.g., Gapminder, Hans Rosling) c. Tools: PowerPoint slides, Jupyter Notebook (for live coding), Nearpod for quizzes
Teaching Development	1. **Introduction** (5 minutes) - Start with a short video on the importance of storytelling with data (e.g., Hans Rosling's famous data-driven presentation). 2. **Ask Questions** - Have you ever seen a chart or graph that changed your perspective? - What makes some visualizations stand out and convey information better than others? - Introduce the concept of data storytelling and its key components: the data, the visualization, and the narrative. 3. **Development** (30 minutes) **a) What is Data Storytelling?** - Define data storytelling and explain why it matters in today's data-driven world. - Highlight the importance of context, audience, and objective when crafting a data story. - Show real-world examples of impactful data stories (e.g., Gapminder, COVID-19 data dashboards). **b) The Three Elements of Data Storytelling** - Data: The foundation of the story. Explain how to select meaningful data to support a narrative. - Visualization: The tool for presenting the data. Discuss different types of visualizations (bar charts, line charts, infographics, etc.) and when to use each. - Narrative: The story that ties it all together. **c) Best Practices of Data Storytelling** 3. **Exercise (5 minutes)**

	- Ask students to create a basic narrative around a small dataset.
Closure	1. Summarize learning outcomes. 2. Suggested reading: "Storytelling with Data" by Cole Nussbaumer Knaflic. 3. Homework: Create a data story with 3-4 visualizations. - Create a data story with 3-4 visualizations.
Evaluation	1. Students will create an interactive scatter plot using a dataset of their choice. 2. Evaluation will focus on their ability to add interactivity (hover tooltips, axis manipulation) and make basic customizations (e.g., colors, titles). 3. Students should be able to correctly save and display their interactive plot using Plotly. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 1	Course Name: Applied Data Science Topic: Introduction to Text Mining and Natural Language Processing	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Define text mining and its applications. b. Understand the role of NLP in processing unstructured text. c. Explore pre-processing techniques such as tokenization, stop-word removal, and stemming.
Teaching Aids (if any)	a. Examples of text pre-processing (e.g., Jupyter Notebooks). b. Visual workflow of text mining using infographics.
Teaching Development	1. **Introduction** (5 minutes) - Brief discussion: What is unstructured data? - Show examples of text datasets. 2. **Development** (30 minutes) **a. What is Text Mining** - Define and explain its significance. - Discuss use cases in industries (e.g., sentiment analysis, chatbots). **b. NLP Basics:** - Introduce NLP components: tokenization, stemming, lemmatization. **c. Hands on Demonstration** - Pre-process text using Python (e.g., NLTK or spaCy). 3. **Exercise (5 minutes)** - Provide a small text dataset for students to tokenize and remove stop words.
Closure	1. Summarize text mining and NLP techniques. 2. Assign reading: "Speech and Language Processing" by Jurafsky and Martin. 3. Homework: Identify 3 real-life applications of text mining and explain their importance.
Evaluation	1. Conduct a short quiz on text preprocessing. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 2	Course Name: Applied Data Science Topic: Sentiment Analysis with NLP	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand what sentiment analysis is and why it is important. b. Explore real-world applications of sentiment analysis. c. Learn the two main approaches: lexicon-based and machine learning-based. d. Implement a basic sentiment analysis task using Python.
Teaching Aids (if any)	a. Sample datasets (e.g., Twitter sentiment data or product reviews). b. Pre-written Python code for hands-on tasks. c. Visual representation of sentiment polarity (positive, neutral, negative)
Teaching Development	1. **Introduction** (5 minutes) - Start by engaging students with a simple question: - "How does Google or Amazon know what customers think about their products?" - "Have you ever written a product review or commented on a social media post?" - Show examples of sentiment analysis in real life, such as: - Analyzing social media for brand reputation. - Identifying trends in political campaigns. - Briefly introduce sentiment analysis and its goal of classifying text based on emotion (positive, neutral, or negative). 2. **Development** (30 minutes) **a. What is Sentiment Analysis** - Define sentiment analysis: - "Sentiment analysis is the process of identifying and categorizing opinions expressed in a text to determine whether the sentiment is positive, negative, or neutral." - Discuss why it's important for businesses, politics, and healthcare. **b. ML based Approach:** - Explain that this method uses labeled data to train classifiers (e.g., Naive Bayes, SVM). - Briefly introduce how deep learning models like LSTMs are applied. - Pros: More accurate with enough data; Cons: Requires more resources and labeled datasets. **c. Hands on Demonstration** - Task: Conduct sentiment analysis on a small dataset using Python. - Use TextBlob or VADER for the lexicon-based approach.

	3. **Exercise (5 minutes)** - Provide the students with a few sentences and ask them to classify each as positive, negative, or neutral manually. - Ask students to verify their classifications using the Python code demonstrated in class.
Closure	1. Recap key points: 2. Definition and importance of sentiment analysis. 3. Two approaches (lexicon-based and ML-based). 4. Assign Homework: 5. Write a Python program to perform sentiment analysis on a dataset (e.g., product reviews from Kaggle). 6. Suggested Reading: 7. "Speech and Language Processing" by Jurafsky and Martin (Chapter on Sentiment Analysis).
Evaluation	1. Conduct a short quiz: 2. "What is sentiment analysis used for?" 3. "Name one lexicon-based tool and one ML-based algorithm for sentiment analysis." 4. Check Python code submissions for basic implementation and understanding. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 3	Course Name: Applied Data Science Topic: Time Series Analysis	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand what a time series is and its importance. b. Identify key components of a time series (trend, seasonality, and noise). c. Explore common time series analysis techniques. d. Visualize and interpret time series data using Python.
Teaching Aids (if any)	a. Sample time series datasets (e.g., stock prices, weather data, sales data). b. Python code templates for hands-on exercises. c. Visual aids to explain components of time series.
Teaching Development	1. **Introduction** (5 minutes) - Start by engaging students with real-world examples: - "What patterns can you observe in stock prices over time?" - "Can you predict seasonal sales (e.g., holiday shopping) based on past trends?" - Define a time series: - "A time series is a sequence of data points recorded at regular intervals over time." - Show visual examples of time series data (e.g., line chart of monthly sales). 2. **Development** (30 minutes) **a. Component of Time Series** - Explain the three main components of time series data: - Trend: The general direction of data over time (e.g., increasing sales). - Seasonality: Regular, repeating patterns (e.g., higher ice cream sales in summer). - Noise: Random fluctuations that cannot be explained. - Show visual examples of each component using line charts. **b. Common Techniques for Time Series Analysis:** - Exploratory Analysis - Decomposition - Stationarity and Differencing - Correlation and Autocorrelation **c. Hands on Demonstration** - Provide students with a small time series dataset (e.g., temperature data). - Ask them to: - Plot the data. - Decompose it into components using Python. - Identify the trend and seasonality.

	3. **Exercise (5 minutes)** - Ask students to think of other examples of time series data in real life (e.g., electricity usage, website traffic). - Discuss how businesses might use time series analysis in decision-making.Ask students to verify their classifications using the Python code demonstrated in class.
Closure	1. Recap key points: - Definition and importance of time series. - Key components (trend, seasonality, noise). - Basic techniques like plotting and decomposition. 2. Assign Homework: - Use a public dataset (e.g., temperature or sales data) to identify the trend and seasonality. Submit plots with explanations.
Evaluation	1. Conduct a short quiz: - "What are the key components of time series data?" - "Why is stationarity important for time series forecasting?" 2. Check Python code submissions for decomposition and trend identification. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 4	Course Name: Applied Data Science Topic: Recommendation Systems	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the purpose and importance of recommendation systems. b. Differentiate between content-based and collaborative filtering approaches. c. Explore real-world applications of recommendation systems. d. Build a simple recommendation system using Python.
Teaching Aids (if any)	a. Visual aids explaining types of recommendation systems. b. Pre-written Python code for implementing recommendation systems. c. Sample datasets (e.g., movie ratings, product reviews).
Teaching Development	1. **Introduction** (5 minutes) - Start with a question to engage students: - "How does Netflix recommend movies or shows you might like?" - "Why do e-commerce platforms like Amazon suggest items you didn't search for?" - Define recommendation systems: - "Recommendation systems are algorithms that suggest products, services, or content to users based on their preferences or behavior." - Briefly explain their importance in industries like entertainment, e-commerce, and education. 2. **Development** (30 minutes) **a. Content Based Filtering** - Explain that it uses item features to recommend similar items. - Example: If a user likes an action movie, the system recommends other action movies. - Discuss the pros (personalized recommendations) and cons (lack of diversity in suggestions). **b. Collaborative Filtering:** - Explain that it uses user-item interactions to find similar users or items. - Example: If user A and user B have similar preferences, items liked by user A may be recommended to user B. - Discuss two methods: - User-based filtering (finding users with similar tastes). - Item-based filtering (finding items liked by similar users). - Highlight the pros (diverse suggestions) and cons (cold start problem).

	c. Hybrid System - Briefly introduce how combining content-based and collaborative filtering can improve recommendations. 3. **Exercise (5 minutes)** - Provide a small dataset of movie ratings. - Ask students to identify whether content-based or collaborative filtering would work better for certain recommendations. - Students can modify the provided Python code to recommend items from the dataset.
Closure	1. Recap key points: - Definition and purpose of recommendation systems. - Types: Content-based, collaborative filtering, and hybrid systems. - Real-world applications and Python implementation. 2. Assign Homework: - Build a recommendation system using a public dataset (e.g., MovieLens) and submit code with results.
Evaluation	1. Conduct a short quiz: - “What are the two main types of recommendation systems?" - "What is the cold start problem, and how can hybrid systems address it?" 2. Check Python code submissions for correct implementation and logic.Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 5	Course Name: Applied Data Science Topic: Deep Learning and Neural Networks	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand the basics of deep learning and how it differs from traditional machine learning. b. Explore the concept of neural networks and their building blocks (e.g., neurons, layers, activation functions). c. Learn how deep learning is applied in real-world scenarios. d. Implement a simple neural network using Python and TensorFlow/Keras.
Teaching Aids (if any)	a. Visual representation of a neural network (e.g., input, hidden, and output layers). b. Sample datasets (e.g., MNIST or Iris dataset). c. Pre-written Python scripts for neural network implementation.
Teaching Development	1. **Introduction** (5 minutes) - Engage students with a question: - "Have you ever wondered how Siri understands your voice or how Google Translate works?" - Define deep learning: - "Deep learning is a subset of machine learning that uses neural networks to mimic the way humans think and learn." - Briefly explain its distinction from traditional machine learning: - ML relies on feature engineering, while DL automatically extracts features using neural networks. - Show a real-world application video, such as autonomous driving or facial recognition. 2. **Development** (30 minutes) **a. Neural Network Basic** - Explain the structure of a neural network: - Neuron: Receives inputs, applies weights and biases, and passes output through an activation function. - Layers: - Input layer (accepts features). - Hidden layers (process features). - Output layer (gives predictions). - Show a diagram of a simple neural network. - Explain the forward propagation process (inputs to outputs). - Introduce activation functions like ReLU, sigmoid, and softmax with simple examples. **b. Deep Learning Basics:** - Discuss how deep learning works in practice: - - Data collection and preprocessing.

	- Model building (choosing the number of layers and neurons). - Training (forward and backward propagation). - Evaluation (accuracy, loss). - Briefly introduce concepts like: - Overfitting and how regularization (e.g., dropout) addresses it. - Optimization algorithms (e.g., gradient descent). **c. Real World Applications** - Discuss examples where deep learning is widely used: - Image recognition (e.g., facial recognition on smartphones). - Natural language processing (e.g., chatbots, language translation). - Healthcare (e.g., cancer detection from scans). - Gaming (e.g., AlphaGo). 3. **Exercise (5 minutes)** - Provide students with a simple dataset (e.g., Iris dataset). - Ask them to build a neural network with one hidden layer and evaluate its performance.
Closure	1. Summarize key points: - What are neural networks and how do they work? - Key components (neurons, layers, activation functions). - Applications of deep learning in real life. 2. Assign Homework: - Train a neural network on the MNIST dataset and submit the results. Suggested Reading: - "Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville (Introductory chapters).
Evaluation	1. Reflective questions: "What is the role of activation functions in a neural network?" "How does deep learning differ from traditional machine learning?" 2. Check Python code submissions for correctness and basic implementation of neural networks. Spend 5 minutes to evaluate student assimilation of the lesson contents

Lesson Plan No. 6	Course Name: Applied Data Science Topic: Neural Network Training	Course No.: COM-702 (C)

Objectives	At the end of the lesson the student shall be able to: a. Understand Backpropagation b. Understand Gradient Descent: c. Apply Optimization Techniques:
Teaching Aids (if any)	a. Visual representation of a neural network (e.g., input, hidden, and output layers). b. Sample datasets (e.g., MNIST or Iris dataset). c. Pre-written Python scripts for neural network implementation.
Teaching Development	1. **Introduction** (5 minutes) - Visualize the Training - Forward Pass: - Loss Calculation: - Introduce the concept of a loss function (e.g., Mean Squared Error for regression or Cross-Entropy for classification). - Explain how the loss quantifies the difference between the predicted and actual values. - Backward Pass (Backpropagation): - Demonstrate how the network calculates gradients for each weight using the chain rule. - Explain how gradients indicate the direction to update weights to reduce loss. 2. **Development** (30 minutes) **a. Explain Gradient Descent** - Explain how weights are updated using the formula: - Learning Rate - Discuss how the learning rate affects convergence **b. Deep Optimization Techniques:** - Discuss how deep learning works in practice: - Data collection and preprocessing. - Model building (choosing the number of layers and neurons). - Training (forward and backward propagation). - Evaluation (accuracy, loss). - Briefly introduce concepts like: - Overfitting and how regularization (e.g., dropout) addresses it. - Optimization algorithms (e.g., gradient descent). **c. Real World Applications** - Discuss examples where deep learning is widely used: - Image recognition (e.g., facial recognition on smartphones).

	- Natural language processing (e.g., chatbots, language translation). - Healthcare (e.g., cancer detection from scans). - Gaming (e.g., AlphaGo). 3. **Exercise (5 minutes)** - Provide students with a simple dataset (e.g., Iris dataset). - Ask them to build a neural network with one hidden layer and evaluate its performance.
Closure	1. Recap: - Summarize key concepts: forward pass, loss calculation, backpropagation, and optimization. 2. Homework: - Assign a task to build a simple neural network for binary classification (e.g., classify odd vs. even numbers). 3. Evaluation: - Conduct a short quiz or ask reflective questions: - What is the purpose of the loss function? - Why is the learning rate important in gradient descent
Evaluation	1. Reflective questions: "What is the role of activation functions in a neural network?" "How does deep learning differ from traditional machine learning?" 2. Check Python code submissions for correctness and basic implementation of neural networks. Spend 5 minutes to evaluate student assimilation of the lesson contents