Syllabus

Machine Learning Techniques (BCS055)		
Course Outcome (CO)		**Bloom's Knowledge Level (KL)**
At the end of course , the student will be able:		
CO 1	To understand the need for machine learning for various problem solving	K_1 , K_2
CO 2	To understand a wide variety of learning algorithms and how to evaluate models generated from data	K_1 , K_3
CO 3	To understand the latest trends in machine learning	K_2 , K_3
CO 4	To design appropriate machine learning algorithms and apply the algorithms to a real-world problems	K_4 , K_6
CO 5	To optimize the models learned and report on the expected accuracy that can be achieved by applying the models	K_4, K_5
DETAILED SYLLABUS		**3-0-0**
Unit	**Topic**	**Proposed Lecture**
I	**INTRODUCTION** – Learning, Types of Learning, Well defined learning problems, Designing a Learning System, History of ML, Introduction of Machine Learning Approaches – (Artificial Neural Network, Clustering, Reinforcement Learning, Decision Tree Learning, Bayesian networks, Support Vector Machine, Genetic Algorithm), Issues in Machine Learning and Data Science Vs Machine Learning;	**08**
II	**REGRESSION:** Linear Regression and Logistic Regression **BAYESIAN LEARNING** - Bayes theorem, Concept learning, Bayes Optimal Classifier, Naïve Bayes classifier, Bayesian belief networks, EM algorithm. **SUPPORT VECTOR MACHINE:** Introduction, Types of support vector kernel – (Linear kernel, polynomial kernel,and Gaussiankernel), Hyperplane – (Decision surface), Properties of SVM, and Issues in SVM.	**08**
III	**DECISION TREE LEARNING** - Decision tree learning algorithm, Inductive bias, Inductive inference with decision trees, Entropy and information theory, Information gain, ID-3 Algorithm, Issues in Decision tree learning. **INSTANCE-BASED LEARNING** – k-Nearest Neighbour Learning, Locally Weighted Regression, Radial basis function networks, Case-based learning.	**08**
IV	**ARTIFICIAL NEURAL NETWORKS** – Perceptron's, Multilayer perceptron, Gradient descent and the Delta rule, Multilayer networks, Derivation of Backpropagation Algorithm, Generalization, Unsupervised Learning – SOM Algorithm and its variant; **DEEP LEARNING** - Introduction,concept of convolutional neural network , Types of layers – (Convolutional Layers , Activation function , pooling , fully connected) , Concept of Convolution (1D and 2D) layers, Training of network, Case study of CNN for eg on Diabetic Retinopathy, Building a smart speaker, Self-deriving car etc.	**08**
V	**REINFORCEMENT LEARNING**–Introduction to Reinforcement Learning , Learning Task,Example of Reinforcement Learning in Practice, Learning Models for Reinforcement – (Markov Decision process , Q Learning - Q Learning function, Q Learning Algorithm), Application of Reinforcement Learning,Introduction to Deep Q Learning. **GENETIC ALGORITHMS:** Introduction, Components, GA cycle of reproduction, Crossover, Mutation, Genetic Programming, Models of Evolution and Learning, Applications.	**08**
Text books: 1. Tom M. Mitchell, —Machine Learning, McGraw-Hill Education (India) Private Limited, 2013.		

Unit 3

SUBJECT: MACHINE LEARNING TECHNIQUES

Decision Tree Learning Algorithm

Decision Tree Learning is a supervised machine learning algorithm used for classification and regression tasks. It operates on a treelike structure, where each internal node represents an attribute or feature, each branch represents a decision rule, and each leaf node represents an outcome or class label.

Steps of the Decision Tree Learning Algorithm

The process involves recursively splitting the dataset into subsets based on the attribute that provides the most significant separation of classes. Here's a breakdown of the key steps:

1. Select the Best Attribute: Choose the attribute that best divides the dataset. Typically, metrics like information gain or Gini index are used to measure how well a particular attribute splits the data.

2. Split the Data: Using the best attribute, split the data into smaller subsets, where each subset corresponds to one of the values or ranges of the chosen attribute.

3. Repeat Recursively: For each subset, repeat the process of selecting the best attribute and splitting the data until one of the stopping criteria is met (e.g., all data points in a subset belong to the same class, or a maximum depth is reached).

4. Assign Leaf Nodes: Once splitting stops, assign each leaf node with a class label based on the majority class within that subset.

Certainly! Decision Tree Learning has several advantages and disadvantages, which are important to consider depending on the specific application.

Advantages of Decision Tree Learning

1. Easy to Understand and Interpret: Decision trees create a flowchart like structure that is easy to visualize and interpret, even for nonexperts.

2. Little Data Preprocessing Needed: They do not require data normalization or scaling and can handle both numerical and categorical data.

3. Handles Nonlinear Relationships: Decision trees capture nonlinear patterns effectively, making them versatile for various types of data.

4. Feature Importance: They provide insights into the importance of different features, helping with feature selection in complex datasets.

5. Can Handle Missing Values: Many decision tree algorithms handle missing data by choosing optimal splits or using surrogate splits.

Disadvantages of Decision Tree Learning

1. Prone to Overfitting: Decision trees often overfit the training data, especially when they are deep or have many branches. This reduces their generalizability to new data.

2. Unstable to Small Variations in Data: Small changes in the data can lead to a completely different tree structure, making the model unstable.

3. Less Effective with Imbalanced Data: Decision trees may perform poorly when classes are imbalanced, as they tend to favor the majority class.

4. High Computational Cost for Large Trees: Constructing a deep tree can be computationally expensive, especially with large datasets, as the algorithm explores multiple splits.

5. Biased towards Features with More Levels: If features have many unique values, the tree may become biased towards these, which could misrepresent the actual importance of such features.

Application of decision tree algorithm

Decision Trees are widely used across various fields due to their interpretability and flexibility. Here are some common applications:

1. Medical Diagnosis

Decision Trees help in diagnosing diseases by categorizing symptoms and patient data to predict the likelihood of certain diseases. For example, they are used to predict conditions like diabetes or heart disease based on medical history and test results.

2. Customer Segmentation in Marketing

Companies use Decision Trees to segment customers based on demographics, purchase history, or behavioral data. This helps in targeting marketing campaigns more effectively by identifying high potential customer groups.

3. Credit Risk Analysis in Finance

In the financial sector, Decision Trees assess creditworthiness by analyzing factors like income, employment history, and existing debts. This helps in making decisions about loan approvals and risk management.

4. Fraud Detection

Decision Trees are used to detect fraudulent activities by identifying unusual patterns in transaction data. This application is especially useful in banking, insurance, and ecommerce.

5. Manufacturing and Quality Control

Decision Trees assist in quality control by identifying defective products based on factors from the manufacturing process. This ensures that high quality products reach consumers and production issues are addressed.

6. Recommendation Systems

They are used to build recommendation systems for products, movies, books, etc. For example, a Decision Tree can be used to suggest movies based on a user's genre preferences and past ratings.

Inductive Bias

Inductive Bias is the set of prior assumptions or beliefs that a machine learning algorithm uses to make predictions on new data based on its training. This helps the algorithm apply what it has learned to new situations, making it more efficient and accurate. Without inductive bias, the algorithm would struggle to generalize and would need to start learning from scratch with each new set of data.

Types of Inductive Bias

1. **Preference Bias**: Prefers simpler or specific solutions over complex ones (e.g., Decision Trees favoring shorter trees).
2. **Restriction Bias**: Limits the hypothesis space by restricting possible solutions (e.g., linear regression assumes a linear relationship).

Benefits of Inductive Bias

- **Improves Generalization**: Helps the model apply learned knowledge to new data.
- **Enhances Efficiency**: Reduces the need for large amounts of training data by guiding learning.
- **Reduces Overfitting**: Guides the model toward simpler solutions that generalize better.

Inductive Inference with Decision Trees

Inductive Inference with Decision Trees involves using examples from training data to create a general rule (or "inference") that can classify new data. In Decision Trees, this is achieved by analyzing the training data and recursively splitting it based on attributes that best separate the classes.

Key Steps in Inductive Inference with Decision Trees:

1. Data Splitting: The algorithm splits the dataset at each node based on an attribute that maximizes information gain or minimizes impurity (e.g., Gini index).

2. Creating Decision Rules: Each split represents a decision rule. The tree continues to split until it reaches pure nodes or meets stopping criteria.

3. Generalization: The resulting decision tree structure (rules) can then be applied to new data to make predictions.

Information Theory

Information Theory provides a framework for measuring uncertainty and information in data. It quantifies how much information a feature or data point contributes to making predictions or decisions. One of the key concepts in information theory is entropy, which measures the amount of uncertainty or randomness in a message.

Entropy

Entropy is a measure of uncertainty or unpredictability in a dataset. In information theory, it indicates how much "disorder" or randomness exists in a set of outcomes. Higher entropy means greater unpredictability (more mixed classes), while lower entropy suggests more order and predictability (pure or similar classes).

Mathematically, entropy is calculated using the formula:

$$H(X) = -\sum p(x) \log_2 p(x)$$

where $p(x)$ is the probability of each possible outcome. The negative sign ensures that entropy is always positive, as probabilities are between 0 and 1, and the logarithm of values in this range is negative.

To calculate the entropy of the given dataset, we can follow these steps:

Step 1: Define the Classes and Their Frequencies

- Class A: 2 students
- Class B: 3 students
- Class C: 5 students

Step 2: Calculate the Total Number of Students

Total Students = Number of students in Class A + Number of students in Class B + Number of students in Class C

$$\text{Total} = 2 + 3 + 5 = 10$$

Step 3: Calculate the Probability of Each Class

- Probability of Class A (p_A):

$$p_A = \frac{\text{Number of students in Class A}}{\text{Total Students}} = \frac{2}{10} = 0.2$$

- Probability of Class B (p_B):

$$p_B = \frac{\text{Number of students in Class B}}{\text{Total Students}} = \frac{3}{10} = 0.3$$

- Probability of Class C (p_C):

$$p_C = \frac{\text{Number of students in Class C}}{\text{Total Students}} = \frac{5}{10} = 0.5$$

Step 4: Calculate the Entropy

The formula for entropy (H) is:

$$H = -\sum_{i} p_i \log_2(p_i)$$

In our case, we will calculate the entropy for each class and then sum them up.

Calculating each term:

- For Class A:

$$H_A = -p_A \log_2(p_A) = -0.2 \log_2(0.2) \approx -0.2 \times (-2.3219) \approx 0.4644$$

- For Class B:

$$H_B = -p_B \log_2(p_B) = -0.3 \log_2(0.3) \approx -0.3 \times (-1.7369) \approx 0.5211$$

- For Class C:

$$H_C = -p_C \log_2(p_C) = -0.5 \log_2(0.5) \approx -0.5 \times (-1) = 0.5$$

Step 5: Sum Up the Entropy Values

Now, we add the individual entropies:

$$H = H_A + H_B + H_C \approx 0.4644 + 0.5211 + 0.5 \approx 1.4855$$

Conclusion

The entropy of the dataset is approximately **1.4855**. This value indicates the level of uncertainty or disorder in the distribution of students across the classes.

Information Gain

Information Gain is a metric used in decision tree learning that quantifies the decrease in uncertainty or entropy achieved by splitting a dataset based on a specific attribute. It measures how much a feature contributes to distinguishing between class labels, with a higher Information Gain indicating that the attribute is more effective in making informed decisions.

The formula for calculating **Information Gain (IG)** is as follows:

$$IG(D, A) = H(D) - H(D|A)$$

Where:

- $IG(D, A)$ is the Information Gain for dataset D when split on attribute A.
- $H(D)$ is the entropy of the dataset before the split.
- $H(D|A)$ is the weighted entropy of the dataset after the split on attribute A.

ID3 (Iterative Dichotomiser 3) Algorithm

The ID3 (Iterative Dichotomiser 3) algorithm is a popular algorithm for creating decision trees in machine learning, primarily used for classification tasks. Developed by Ross Quinlan, ID3 builds the tree by making decisions based on the concept of information gain, which measures how well a feature separates the data into distinct classes.

Key Steps of the ID3 Algorithm

1. Select the Root Node: Start with the entire dataset. For each feature, calculate its information gain based on the target (class) attribute. The feature with the highest information gain becomes the root node.

2. Split Data: Divide the dataset according to the chosen feature's values, creating branches.

3. Repeat Recursively: For each subset created in the previous step, repeat the process. For each subset, calculate the information gain for remaining features, choosing the one with the highest gain to create the next node.

4. Stop Condition: The process continues until:

All data in a subset belong to a single class.

No remaining features can further split the data.

Key Concepts in ID3

Entropy: Measures the uncertainty or impurity in a dataset. Lower entropy means more homogeneous data.

Information Gain: Measures the reduction in entropy when a dataset is split based on a feature. The feature with the highest information gain is chosen for the split.

Example Use

ID3 is commonly used in scenarios like customer segmentation, diagnosis, and credit scoring, where classification is based on a series of decisions or attributes.

In summary, ID3 builds a decision tree by choosing features that provide the most informative splits, leading to an efficient and interpretable model for classification tasks.

Issues in Decision tree learning

Decision tree learning, while a powerful and intuitive method for classification and regression tasks, has several challenges and issues that can affect its performance. Here are some of the key issues:

1. Overfitting

Description: Decision trees can become overly complex by creating very deep trees that fit the training data perfectly, including noise and outliers.

Impact: This results in poor generalization to unseen data, leading to high variance and low predictive accuracy.

2. Underfitting

Description: Conversely, if a tree is too shallow or stops splitting too early, it may fail to capture important patterns in the data.

Impact: This leads to high bias, resulting in a model that is not complex enough to make accurate predictions.

3. Sensitivity to Noisy Data

Description: Decision trees can be sensitive to noisy or irrelevant features in the dataset.

Impact: Noisy data can lead to splits that do not generalize well, negatively affecting the model's performance.

4. Biased towards Dominant Classes

Description: If the dataset is imbalanced (i.e., one class significantly outweighs others), the tree may favor the majority class.

Impact: This results in poor predictive performance for the minority class, leading to misleading accuracy metrics.

5. Feature Selection Bias

Description: Decision trees prefer features with more levels or categories for splitting.

Impact: This can lead to bias towards certain features, which may not be the most informative.

6. Lack of Interpolation

Description: Decision trees make decisions based on feature values without considering the relationships between them.

Impact: They can struggle with interpolation, particularly in regression tasks, where they may miss trends or patterns in the data.

kNearest Neighbor (kNN) Learning

kNearest Neighbor (kNN) Learning is a simple, instance based learning algorithm used for classification and regression tasks. It works by identifying the closest data points (neighbors) to a given query instance from the training set. For classification, the algorithm assigns the most common label among these neighbors to the query instance. For regression, it calculates the average value of these neighbors as the prediction.

Key features of kNN include:

Instancebased: It does not build an explicit model; rather, it makes predictions based on stored instances.

Lazy learning: kNN does not perform any computations until it receives a query, making it a "lazy" learner.

Distance metric: The closeness of neighbors is typically measured using a distance metric, such as Euclidean distance for continuous data or Hamming distance for categorical data.

Since it relies on actual data points for predictions, kNN can handle complex, nonlinear data patterns but may be computationally expensive with large datasets.

Example Problem

Suppose we have a small dataset of points labeled either "Class A" or "Class B". Each point has two features (e.g., x and y coordinates on a 2D plane).

Here's our dataset:

Point	x	y	Class
1	1	2	A
2	2	3	A
3	3	3	B
4	6	5	B
5	7	8	B

Task

We want to predict the class of a new point, Point X with coordinates $x = 4$ and $y = 4$, using the **k-NN** algorithm with $k = 3$.

Solution Steps

Step 1: Calculate the Distance Between Point X and All Points in the Dataset

We use the **Euclidean distance formula**:

$$\text{Distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Let's calculate the distance between Point X (4, 4) and each point in our dataset:

1. **Distance to Point 1 (1, 2)**:

$$= \sqrt{(4-1)^2 + (4-2)^2} = \sqrt{3^2 + 2^2} = \sqrt{9+4} = \sqrt{13} \approx 3.61$$

2. **Distance to Point 2 (2, 3)**:

$$= \sqrt{(4-2)^2 + (4-3)^2} = \sqrt{2^2 + 1^2} = \sqrt{4+1} = \sqrt{5} \approx 2.24$$

3. **Distance to Point 3 (3, 3)**:

$$= \sqrt{(4-3)^2 + (4-3)^2} = \sqrt{1^2 + 1^2} = \sqrt{1+1} = \sqrt{2} \approx 1.41$$

4. **Distance to Point 4 (6, 5)**:

$$= \sqrt{(4-6)^2 + (4-5)^2} = \sqrt{(-2)^2 + (-1)^2} = \sqrt{4+1} = \sqrt{5} \approx 2.24$$

5. **Distance to Point 5 (7, 8)**:

$$= \sqrt{(4-7)^2 + (4-8)^2} = \sqrt{(-3)^2 + (-4)^2} = \sqrt{9+16} = \sqrt{25} = 5.0$$

Step 2: Identify the k Nearest Neighbors

With $k = 3$, we need the three points with the smallest distances to Point X. The distances, sorted from smallest to largest, are:

- **Point 3** (3, 3) — Distance: 1.41 — Class B
- **Point 2** (2, 3) — Distance: 2.24 — Class A
- **Point 4** (6, 5) — Distance: 2.24 — Class B

So, the **3 nearest neighbors** are:

1. Point 3 (Class B)
2. Point 2 (Class A)
3. Point 4 (Class B)

Step 3: Determine the Majority Class Among Neighbors

Out of the 3 nearest neighbors:

- 2 neighbors are **Class B**
- 1 neighbor is **Class A**

Since **Class B** is the majority class, we predict that **Point X belongs to Class B.**

Final Prediction

The predicted class for Point X (4, 4) is **Class B.**

Locally Weighted Regression (LWR)

Locally Weighted Regression (LWR) is a type of regression in which a model is built specifically for each query point, focusing on nearby data points rather than the entire dataset. This makes LWR a nonparametric and instance based learning technique, especially useful for capturing nonlinear relationships in data.

How Locally Weighted Regression Works

In LWR, when we want to predict the output for a specific input, the algorithm:

1. Assigns weights to nearby points: Points closer to the query point (the input where we want to make a prediction) are given higher weights, meaning they have more influence on the prediction. This is typically done using a kernel function (such as Gaussian), which decreases weight as distance increases.

2. Fits a local model: Instead of fitting a single model to all the data, LWR fits a linear regression model that is "localized" to the neighborhood of the query point. This model is trained on the weighted data points, where closer points impact the model more.

3. Predicts the output: The model uses this localized linear regression to predict the output for the query point.

Key Characteristics of LWR

Instancebased: LWR doesn't create a single global model. It makes predictions using local models specific to each query point.

Nonparametric: LWR doesn't assume a fixed form for the relationship between inputs and outputs. Instead, it adapts to the data locally.

Good for nonlinear data: LWR is particularly effective in capturing nonlinear patterns, as each region of the data can have its own localized model.

Example Use Case

Locally Weighted Regression is often used in robotics and control systems, where it's helpful to adapt to changes in realtime by using local data, as well as in any application where data may not fit a single global pattern.

Radial Basis Function (RBF) Networks

Radial Basis Function (RBF) Networks are a type of artificial neural network that use radial basis functions as activation functions. They are particularly wellsuited for applications involving pattern recognition, function approximation, and time series prediction.

Structure of RBF Networks

An RBF Network typically has three layers:

1. **Input Layer**: Passes the input features directly to the next layer without processing.
2. **Hidden Layer**: Contains neurons that use radial basis functions as activation functions. The most common radial basis function is the **Gaussian function**. Each hidden neuron represents a "center" point in the input space, and it measures the distance between this center and the input. The output of the neuron decreases with distance from its center, giving a localized response.
3. **Output Layer**: Produces the final prediction by taking a weighted sum of the hidden layer outputs.

How RBF Networks Work

1. **Compute Distance**: For a given input, the RBF neurons in the hidden layer compute the distance between the input and their respective centers.
2. **Apply Activation Function**: Each neuron in the hidden layer then applies the radial basis function to this distance, resulting in an output that is high if the input is close to the neuron's center and low if it is far away.
3. **Combine Outputs**: The outputs from the hidden layer are passed to the output layer, where they are combined (typically using a weighted sum) to produce the final result.

Applications of RBF Networks

RBF Networks are commonly used in:

- **Pattern recognition**: Such as image and speech recognition.
- **Function approximation**: For tasks that require smooth interpolation, like curve fitting.
- **Time series prediction**: Often used in forecasting applications.

Case Based Learning (CBL)

Case Based Learning (CBL) is a method of learning in which new problems are solved by referring to previously encountered cases (examples) that are stored in memory. Instead of learning general rules, CBL focuses on recalling and adapting solutions from similar past cases to address new situations.

Key Points of Case Based Learning

1. Memory Based: CBL relies on a database of past cases, each containing a problem and its solution.

2. Similarity Matching: When a new problem arises, the system searches for similar past cases, compares them, and selects the most relevant one(s).

3. Adaptation: The solution from the retrieved case(s) is then adapted to fit the new problem's specifics.

Applications

CBL is widely used in fields like medical diagnosis, customer support, and legal reasoning, where past cases provide valuable insights and guidance for current decisions.