USN										

CMRIT CMR INSTITUTE OF TECHNOLOGY, BENGALURU. ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test II –Nov. 2021

Sub:	Machine Learning					Sub Code:	21AI63		Branch:	AIML
Date:	06/06/2024	Duration:	90 mins	Max Marks:	50	Sem / Sec:	A	Time	12.15 – 1.45PM	OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1	Explain different types of Machine Learning Systems in brief	[10]	CO1	L2

Machine learning systems can be broadly classified into several categories based on different criteria such as the type of data they work with, the learning approach they use, and their output. Here's a brief overview of the main types of machine learning systems:

1. Supervised Learning

In supervised learning, the algorithm is trained on a labeled dataset, which means that each training example is paired with an output label. The goal is to learn a mapping from inputs to outputs, so the model can predict the output for new, unseen inputs.

- **Classification**: The output variable is a category. For example, classifying emails as spam or not spam.
- **Regression**: The output variable is a continuous value. For example, predicting house prices.

2. Unsupervised Learning

In unsupervised learning, the algorithm is given data without explicit instructions on what to do with it. The goal is to find hidden patterns or intrinsic structures in the input data.

- **Clustering**: The task is to group similar data points together. For example, customer segmentation in marketing.
- **Association**: The task is to find rules that describe large portions of the data. For example, market basket analysis to find product purchase correlations.

3. Semi-Supervised Learning

This approach is a mix of supervised and unsupervised learning. It uses a small amount of labeled data and a large amount of unlabeled data to train the model. This is useful when labeling data is expensive or time-consuming.

4. Reinforcement Learning

Reinforcement learning involves training an agent to make a sequence of decisions by rewarding it for good actions and penalizing it for bad ones. The agent learns to achieve a goal in an uncertain, potentially complex environment.

- **Markov Decision Processes (MDP)**: The decision-making is modeled with states, actions, and rewards.

5. Self-Supervised Learning

Self-supervised learning is a form of unsupervised learning where the data provides the supervision. This approach involves creating tasks from the data itself, allowing the model to learn representations by predicting parts of the data from other parts.

6. Transfer Learning

Transfer learning involves taking a pre-trained model developed for a task and adapting it to a different but related task. This is particularly useful when there is limited labeled data available for the new task.

7. Multi-Instance Learning

In multi-instance learning, the model is trained on bags of instances. The individual instances within a bag are not labeled, but the bag itself is labeled. The goal is to predict the label of new bags based on the instances they contain.

8. Online Learning

Online learning algorithms update the model incrementally as new data arrives. This is useful for scenarios where the data is too large to fit into memory or arrives in a stream.

9. Batch Learning

In batch learning, the model is trained on the entire dataset at once. This approach is suitable when the data is static and can fit into memory.

10. Active Learning

Active learning is a special case of supervised learning where the algorithm selectively queries the user to label new data points with the desired outputs. This is useful when labeling data is expensive and the model can identify the most informative examples to label.

11. Dimensionality Reduction

Dimensionality reduction methods are used to reduce the number of features in the dataset while retaining as much information as possible. This helps in improving computational efficiency and reducing overfitting.

- **Principal Component Analysis (PCA)**: A method to reduce dimensions by transforming the data into a new coordinate system.
- **t-Distributed Stochastic Neighbor Embedding (t-SNE)**: A technique for dimensionality reduction that is particularly good at visualizing high-dimensional data.

2 Write FIND-S algorithm and explain with example given below [10] CO1 L3

Color	Toughness	Fungus	Appearance	Poisonous
Green	Hard	No	Wrinkled	Yes
Green	Hard	Yes	Smooth	No
Brown	Soft	No	Wrinkled	No
Orange	Hard	No	Wrinkled	Yes
Green	Soft	Yes	Smooth	Yes
Green	Hard	Yes	Wrinkled	Yes
Orange	Hard	No	Wrinkled	Yes

The FIND-S algorithm is a simple machine learning algorithm used for learning a hypothesis from a set of positive examples. It finds the most specific hypothesis that fits all the positive examples. The algorithm works as follows:

1. Initialize the hypothesis hhh to the most specific hypothesis, h=⟨ϕ,ϕ,…,ϕ⟩h = \langle \phi, \phi, \ldots, \phi \rangleh=⟨ϕ,ϕ,…,ϕ⟩.
2. For each positive training instance xxx:
   - For each attribute aia_iai:
     - If the attribute aia_iai of xxx matches the attribute aia_iai of hhh, do nothing.
     - If the attribute aia_iai of xxx does not match the attribute aia_iai of hhh, replace aia_iai of hhh with '?'.
3. Output the hypothesis hhh.

Example Dataset

Given the dataset:

Color	**Toughness**	**Fungus**	**Appearance**	**Poisonous**
Green	Hard	No	Wrinkled	Yes
Green	Hard	Yes	Smooth	No
Brown	Soft	No	Wrinkled	No
Orange	Hard	No	Wrinkled	Yes
Green	Soft	Yes	Smooth	Yes
Green	Hard	Yes	Wrinkled	Yes
Orange	Hard	No	Wrinkled	Yes

Applying FIND-S Algorithm

1. **Initialization**: Start with the most specific hypothesis. h=⟨ϕ,ϕ,ϕ,ϕ⟩h = \langle \phi, \phi, \phi, \phi \rangleh=⟨ϕ,ϕ,ϕ,ϕ⟩
2. **First Positive Example** (Green, Hard, No, Wrinkled, Yes\text{Green, Hard, No, Wrinkled, Yes}Green, Hard, No, Wrinkled, Yes): h=⟨Green,Hard,No,Wrinkled⟩h = \langle \text{Green}, \text{Hard}, \text{No}, \text{Wrinkled} \rangleh=⟨Green,Hard,No,Wrinkled⟩
3. **Second Positive Example** (Orange, Hard, No, Wrinkled, Yes\text{Orange, Hard, No, Wrinkled, Yes}Orange, Hard, No, Wrinkled, Yes): h=⟨?,Hard,No,Wrinkled⟩h = \langle ?, \text{Hard}, \text{No}, \text{Wrinkled} \rangleh=⟨?,Hard,No,Wrinkled⟩
4. **Third Positive Example** (Green, Soft, Yes, Smooth, Yes\text{Green, Soft, Yes, Smooth, Yes}Green, Soft, Yes, Smooth, Yes): h=⟨?,?,?,?⟩h = \langle ?, ?, ?, ? \rangleh=⟨?,?,?,?⟩
5. **Fourth Positive Example** (Green, Hard, Yes, Wrinkled, Yes\text{Green, Hard, Yes, Wrinkled, Yes}Green, Hard, Yes, Wrinkled, Yes):
   - Color: ???
   - Toughness: ???
   - Fungus: ???
   - Appearance: ???
6. **Fifth Positive Example** (Orange, Hard, No, Wrinkled, Yes\text{Orange, Hard, No, Wrinkled, Yes}Orange, Hard, No, Wrinkled, Yes):
   - Color: ???
   - Toughness: HardHardHard
   - Fungus: NoNoNo
   - Appearance: WrinkledWrinkledWrinkled

Explanation

The final hypothesis h=⟨?,Hard,No,Wrinkled⟩h = \langle ?, Hard, No, Wrinkled \rangleh=⟨?,Hard,No,Wrinkled⟩ indicates that the specific attributes necessary for a mushroom to be poisonous, according to the positive examples, are:

- **Toughness**: Hard
- **Fungus**: No
- **Appearance**: Wrinkled

This hypothesis means that any mushroom that is hard, has no fungus, and has a wrinkled appearance is predicted to be poisonous. The color attribute is irrelevant according to the hypothesis because it was generalized to '?', indicating that the color does not affect whether a mushroom is poisonous or not.

3	Illustrate some of the basic design issues and approaches to machine learning considering designing a program to learn to play checkers. Designing a machine learning program to play checkers involves addressing several fundamental design issues and choosing appropriate approaches. Below are some of the key issues and potential solutions:	[10]	CO1	L2

1. Representation of the Game State

Issue: How to represent the board and game state in a way that is useful for the learning algorithm. **Approach**: Use a matrix or a list to represent the 8x8 board, where each element represents a square that can be empty, occupied by a black piece, or occupied by a white piece. Additional information such as whose turn it is can be included as well.

2. Representation of the Policy or Value Function

Issue: How to represent the policy (strategy) that the program will use to choose its moves, or the value function that evaluates the desirability of board positions. **Approach**: Use a neural network, decision tree, or other function approximator to represent the policy or value function. For example, a neural network can take the board state as input and output the predicted value or the probability of choosing each possible move.

3. Choice of Learning Algorithm

Issue: Which learning algorithm to use to train the policy or value function. **Approach**: Possible algorithms include:

- **Reinforcement Learning (RL)**: Algorithms like Q-learning, SARSA, or more advanced methods like Deep Q-Networks (DQN) and Policy Gradient methods can be used to learn the value of states or the best actions to take.
- **Supervised Learning**: If a dataset of expert moves is available, supervised learning can be used to train the policy network to imitate expert play.

4. Exploration vs. Exploitation

Issue: How to balance exploring new moves (which might be better) with exploiting known good moves. **Approach**: Use techniques like epsilon-greedy (with a decay schedule), where the program chooses a random move with probability ε and the best-known move with probability 1-ε, or more advanced methods like Upper Confidence Bound (UCB) for better exploration.

5. Feature Extraction

Issue: How to extract useful features from the raw board state that can help the learning algorithm. **Approach**: Use domain knowledge to hand-craft features (e.g., number of pieces, number of kings, piece positions) or use deep learning to automatically learn features from the raw board state.

6. Training and Evaluation

Issue: How to train the model effectively and how to evaluate its performance. **Approach**:

	• **Training**: Use self-play, where the program plays against itself, to generate training data. Techniques like Experience Replay can be used to store and reuse past game experiences. • **Evaluation**: Play games against other programs or human players to evaluate the performance. Use metrics such as win/loss ratio and the quality of moves to measure improvement. **7. Handling the Complexity of the Game** **Issue**: Checkers has a large state space and a long game horizon, making learning and planning difficult. **Approach**: Use techniques like: • **Search Algorithms**: Incorporate search algorithms like Minimax with Alpha-Beta pruning to look ahead several moves and evaluate the outcomes. • **Hierarchical Learning**: Break the problem into smaller sub-problems, such as learning separate policies for different phases of the game (e.g., opening, mid-game, endgame). **8. Dealing with Opponent Strategies** **Issue**: The program must be robust to a variety of opponent strategies. **Approach**: Train the program against a diverse set of opponents, including different versions of itself with varied strategies, to ensure it can handle different playing styles.			
4	Apply the Candidate Elimination algorithm to a set of training examples to demonstrate how it identifies the boundary hypotheses in the version space.	[6]	CO1	L3

Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

The Candidate Elimination algorithm is used in machine learning to find the set of all hypotheses that are consistent with the given training examples. It maintains two sets of hypotheses: the most specific hypotheses (S) and the most general hypotheses (G). The algorithm iteratively refines these sets as it processes each training example.

Training Examples

Given the dataset:

Sky	AirTem p	Humidity	Wind	Wate r	Forecas t	EnjoySport
Sunny	Warm	Normal	Strong	Warm	Same	Yes

Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Steps of the Candidate Elimination Algorithm

1. **Initialization**:
   - SSS (most specific hypothesis): ⟨ϕ,ϕ,ϕ,ϕ,ϕ,ϕ⟩\langle \phi, \phi, \phi, \phi, \phi, \phi \rangle⟨ϕ,ϕ,ϕ,ϕ,ϕ,ϕ⟩
   - GGG (most general hypothesis): ⟨?,?,?,?,?,?⟩\langle ?, ?, ?, ?, ?, ? \rangle⟨?,?,?,?,?,?⟩
2. **Processing each example**:
   - For a positive example, update SSS:
     - Generalize SSS minimally to cover the example.
     - Remove any hypotheses from GGG that do not cover the example.
   - For a negative example, update GGG:
     - Specialize GGG minimally to exclude the example.
     - Remove any hypotheses from SSS that cover the example.

Step-by-Step Processing

1. Processing the first example: ⟨Sunny, Warm, Normal, Strong, Warm, Same, Yes⟩\langle \text{Sunny, Warm, Normal, Strong, Warm, Same, Yes} \rangle⟨Sunny, Warm, Normal, Strong, Warm, Same, Yes⟩

- SSS (generalize to cover the example): S=⟨Sunny, Warm, Normal, Strong, Warm, Same⟩S = \langle \text{Sunny, Warm, Normal, Strong, Warm, Same} \rangleS=⟨Sunny, Warm, Normal, Strong, Warm, Same⟩
- GGG (remove inconsistent hypotheses, but no change needed as GGG covers the example): G=⟨?,?,?,?,?,?⟩G = \langle ?, ?, ?, ?, ?, ? \rangleG=⟨?,?,?,?,?,?⟩

2. Processing the second example: ⟨Sunny, Warm, High, Strong, Warm, Same, Yes⟩\langle \text{Sunny, Warm, High, Strong, Warm, Same, Yes} \rangle⟨Sunny, Warm, High, Strong, Warm, Same, Yes⟩

- SSS (generalize to cover both positive examples): S=⟨Sunny, Warm, ? , Strong, Warm, Same⟩S = \langle \text{Sunny, Warm, ? , Strong, Warm, Same} \rangleS=⟨Sunny, Warm, ? , Strong, Warm, Same⟩
- GGG (no change): G=⟨?,?,?,?,?,?⟩G = \langle ?, ?, ?, ?, ?, ? \rangleG=⟨?,?,?,?,?,?⟩

3. Processing the third example: ⟨Rainy, Cold, High, Strong, Warm, Change, No⟩\langle \text{Rainy, Cold, High, Strong, Warm, Change, No} \rangle⟨Rainy, Cold, High, Strong, Warm, Change, No⟩

- SSS (no change as it is a negative example): S=⟨Sunny, Warm, ? , Strong, Warm, Same⟩S = \langle \text{Sunny, Warm, ? , Strong, Warm, Same} \rangleS=⟨Sunny, Warm, ? , Strong, Warm, Same⟩
- GGG (specialize GGG to exclude the negative example): G={⟨Sunny, ?, ?, ?, ?, ?⟩,⟨?,Warm, ?, ?, ?, ?⟩,⟨?,?,?,?,?,Same⟩,⟨?,?,?,?,Warm, ?⟩}G = \{ \langle \text{Sunny, ?, ?, ?, ?, ?} \rangle, \langle ?, \text{Warm, ?, ?, ?, ?} \rangle, \langle ?, ?, ?, ?, ?, \text{Same} \rangle, \langle ?, ?, ?, ?, \text{Warm, ?} \rangle \}G={⟨Sunny, ?, ?, ?, ?, ?⟩,⟨?,Warm, ?, ?, ?, ?⟩,⟨?,?,?,?,?,Same⟩,⟨?,?,?,?,Warm, ?⟩}

Final Version Space

The final version space contains all hypotheses that lie between SSS and GGG.

- SSS (most specific hypothesis): S=⟨Sunny, Warm, ? , Strong, ? , ?⟩S = \langle \text{Sunny, Warm, ? , Strong, ? , ?} \rangleS=⟨Sunny, Warm, ? , Strong, ? , ?⟩
- GGG (most general hypotheses): G={⟨Sunny, ?, ?, ?, ?, ?⟩,⟨?,Warm, ?, ?, ?, ?⟩}G = \{ \langle \text{Sunny, ?, ?, ?, ?, ?} \rangle, \langle ?, \text{Warm, ?, ?, ?, ?} \rangle \}G={⟨Sunny, ?, ?, ?, ?, ?⟩,⟨?,Warm, ?, ?, ?, ?⟩}

5	Explain the steps involved in classification using MNIST Dataset.	[4]	CO2	L2

The MNIST (Modified National Institute of Standards and Technology) dataset is a large database of handwritten digits commonly used for training various image processing systems. Here's a step-by-step guide to performing classification using the MNIST dataset:

Step 1: Import Required Libraries

First, import the necessary libraries. You will typically need libraries such as numpy for numerical operations, matplotlib for plotting, and tensorflow or scikit-learn for machine learning tasks.

python
Copy code
```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical
```

Step 2: Load the MNIST Dataset

The MNIST dataset can be directly loaded from TensorFlow or Keras datasets.

python
Copy code
```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Step 3: Data Exploration and Visualization

Explore the dataset to understand its structure and visualize some samples.

python
Copy code
```
print("Training data shape:", X_train.shape)
print("Testing data shape:", X_test.shape)

# Plot the first image in the training dataset
plt.imshow(X_train[0], cmap='gray')
plt.title('Digit: {}'.format(y_train[0]))
plt.show()
```

Step 4: Preprocess the Data

Normalize the images to have pixel values between 0 and 1 and convert the labels to categorical format.

python
Copy code
```
# Normalize the images
X_train = X_train.astype('float32') / 255
X_test = X_test.astype('float32') / 255

# Convert labels to categorical
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

Step 5: Build the Model

Create a neural network model using Keras. Here, a simple feedforward neural network is used.

python
Copy code
```
model = Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

	Step 6: Compile the Model
	
	Compile the model by specifying the loss function, optimizer, and metrics.
	
	python
	Copy code
	model.compile(loss='categorical_crossentropy',
	optimizer='adam',
	metrics=['accuracy'])
	
	Step 7: Train the Model
	
	Train the model on the training dataset.
	
	python
	Copy code
	model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))
	
	Step 8: Evaluate the Model
	
	Evaluate the model on the test dataset to see its performance.
	
	python
	Copy code
	test_loss, test_acc = model.evaluate(X_test, y_test)
	print('Test accuracy:', test_acc)
	
	Step 9: Make Predictions
	
	Use the trained model to make predictions on new data.
	
	python
	Copy code
	predictions = model.predict(X_test)
	
	# Example prediction
	print('Predicted label:', np.argmax(predictions[0]))
	print('True label:', np.argmax(y_test[0]))			

6(a)	Discuss limitations of Find-S over Candidate Elimination Algorithm The Find-S and Candidate Elimination algorithms are both used in the field of machine learning to find hypotheses that are consistent with given training examples. However, they have different approaches and limitations. Here are the key limitations of the Find-S algorithm compared to the Candidate Elimination algorithm: **1. Inability to Handle Inconsistent Training Data**	[05]	CO1	L2

- **Find-S**: Assumes that there are no contradictory examples in the training data. If there are conflicting examples (e.g., the same instance with different classifications), Find-S will not be able to handle them and will produce an incorrect hypothesis.
- **Candidate Elimination**: Can handle inconsistent training data by maintaining a version space of hypotheses. It eliminates inconsistent hypotheses as new training examples are processed.

2. Limited to the Most Specific Hypothesis

- **Find-S**: Only finds the most specific hypothesis that fits all the positive examples. It does not consider any general hypotheses that might also fit the data. This can lead to overfitting, where the hypothesis is too specific and does not generalize well to unseen data.
- **Candidate Elimination**: Maintains both the most specific (S) and the most general (G) hypotheses, allowing for a broader and more flexible search within the version space. This helps in balancing specificity and generality.

3. No Consideration of Negative Examples

- **Find-S**: Ignores negative examples entirely, which can result in a hypothesis that incorrectly classifies negative instances as positive. This is because it only focuses on generalizing positive examples.
- **Candidate Elimination**: Uses both positive and negative examples to refine the hypothesis space, ensuring that the resulting hypotheses correctly classify both positive and negative instances.

4. Sensitivity to Noise

- **Find-S**: Highly sensitive to noise in the training data. Since it focuses on the most specific hypothesis, any noisy or incorrect positive example can lead to a highly specific and incorrect hypothesis.
- **Candidate Elimination**: More robust to noise because it considers a range of hypotheses and refines the hypothesis space incrementally. However, it still can be affected by noise, but less severely than Find-S.

5. Output Hypothesis Quality

- **Find-S**: The final hypothesis produced by Find-S is highly dependent on the order of the training examples and the presence of noisy data. It may not always represent the best hypothesis that fits the training data.
- **Candidate Elimination**: Provides a more comprehensive set of hypotheses by considering all possible generalizations and specializations, leading to a more reliable final hypothesis that fits the training data well.

6. Complexity and Computation

- **Find-S**: Simple and computationally efficient since it only updates the hypothesis with each positive example and does not maintain multiple hypotheses.

	• **Candidate Elimination**: More computationally complex because it maintains and updates both the S and G sets with each training example. This can be more resource-intensive, especially with a large hypothesis space and many examples.			
6(b)	Define the following terms: (i) Concept Learning (ii)Version Space (iii) Hypothesis Space (iv)General Boundary (v) Specific Boundary **(i) Concept Learning** **Concept Learning** is the task of inferring a Boolean-valued function from training examples of its input and output. It involves finding a general rule that covers all the positive examples and none of the negative examples. In simpler terms, concept learning is about finding a hypothesis that correctly classifies given data points into positive and negative categories based on their attributes. **(ii) Version Space** **Version Space** is the subset of the hypothesis space that is consistent with all the training examples seen so far. It represents the set of all hypotheses that correctly classify the training examples. The version space is bounded by the most specific hypothesis (S) and the most general hypothesis (G). **(iii) Hypothesis Space** **Hypothesis Space** (H) is the set of all possible hypotheses that can be formulated using a given hypothesis language. It includes all the potential rules or functions that could explain the relationship between input features and output labels. In concept learning, the hypothesis space contains all the possible ways to classify the examples based on their attributes. **(iv) General Boundary** **General Boundary** (G) of the version space is the set of the most general hypotheses that are consistent with the training examples. These hypotheses are as general as possible without misclassifying any of the negative examples. The general boundary represents one limit of the version space, ensuring that no hypothesis more general than those in G is consistent with all the training examples. **(v) Specific Boundary** **Specific Boundary** (S) of the version space is the set of the most specific hypotheses that are consistent with the training examples. These hypotheses are as specific as possible while still correctly classifying all the positive examples. The specific boundary represents the other limit of the version space, ensuring that no hypothesis more specific than those in S is consistent with all the training examples.	[05]	CO1	L2

CO PO Mapping

CO-PO and CO-PSO Mapping																			
Course Outcomes		**Blooms Level**	**Modules**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**	**PSO4**
CO1	Understand the concept of Machine Learning and Concept Learning.	**L1**	**1,2,**	3	2	1	-	-	-	-	-	-	-	-	1	-	-	-	-
CO2	Apply the concept of ML and various classification methods in a project.	**L2, L3**	**1,2**	2	2	1	-	-	-	-	-	-	-	-	-	-	-	-	-
CO3	Analyse various training models in ML and the SVM algorithm to be implemented.	**L2**	**1,2**	3	2	1	-	-	-	-	-	-	-	-	1	-	-	-	-
CO4	Apply the ML concept in a decision tree structure and implementation of Ensemble learning and Random Forest	**L3**	**3,4**	3	2	1	-	-	-	-	-	-	-	-	2	-	-	-	-
CO5	Apply Bayes techniques and explore more about the classification in ML.	**L2, L3**	**4,5**	3	2	1	-	-	-	-	-	-	-	-	1	-	-	-	-

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low
PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		

PO6	The Engineer and society	PO12	Life-long learning	
PSO1	Develop applications using different stacks of web and programming technologies			
PSO2	Design and develop secure, parallel, distributed, networked, and digital systems			
PSO3	Apply software engineering methods to design, develop, test and manage software systems.			
PSO4	Develop intelligent applications for business and industry			