(226)

(9)

Scoring Indicators

COURSE NAME: FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

COURSE CODE: 5133C (21)

QID: 2109230305A

Q. No.		Scoring Indicators	Split Score	Sub Total	Total Score
PART A					**9**
I	1	Perception, reasoning, and action	1	1	
I	2	Keras, Tensorflow	1	1	
I	3	0.75,3,0,81	1	1	
I	4	4	1	1	
I	5	Dictionary is a collection of key-value pairs, with unique key enclosed in ''. Example: {'ME101' : 'Strength of materials', 'EE101' : 'Electronics'}	1	1	
I	6	Machine Learning is one of the most popular fields of AI. The basic concept of this filed is to make the machine learning from data as the human beings can learn from his/her experience. It contains learning models on the basis of which the predictions can be made on unknown data.	1	1	
I	7	Supervised learning	1	1	
I	8	The goal of search algorithms is to find the optimal set of moves so that they can reach at the final destination and win. These algorithms use the winning set of conditions, different for every game, to find the best moves.	1	1	
I	9	Minimax algorithm	1	1	
PART B					**24**
II	1	- Supervised Learning: In this type of learning, the AI system is trained on labeled data, where inputs are paired with corresponding outputs. For example, in training an email spam filter, the AI algorithm is fed a dataset of labeled "spam" and "non-spam" emails. It learns to classify new emails as either spam or non-spam based on the patterns it identified during training. - Unsupervised Learning: Here, the AI system learns from unlabeled data to identify patterns or clusters. For example, in customer segmentation, an AI algorithm can analyze purchasing behavior from a dataset without predefined categories. It identifies natural groupings of customers based on similar purchasing habits. - Reinforcement Learning: The AI agent learns by interacting with an environment, receiving feedback in the form of rewards or penalties. For example, in training a game-playing AI, the agent explores different moves and receives rewards for successful actions and penalties for poor ones. It learns to maximize rewards over time, resulting in improved game-playing performance. (1.5 marks for explanation and 1.5 marks for the examples)	1 1 1	3	
II	2	- Natural Language Processing (NLP): NLP focuses on enabling computers to understand, interpret, and generate human language. AI techniques in NLP include language translation, sentiment analysis, and chatbots. For instance, virtual assistants like Siri and Alexa use NLP to respond to voice commands and engage in conversations.	1	3	

		- Computer Vision: Computer vision involves teaching computers to interpret and understand visual information from images or videos. AI in computer vision enables object detection, facial recognition, and image classification. Applications range from surveillance and security systems to self-driving cars. - Robotics: Robotics involves the design and construction of intelligent machines that can perform tasks with minimal human intervention. AI-powered robots are used in manufacturing, healthcare, and space exploration for tasks that require precision and adaptability. Examples include robotic surgery and exploration rovers. (one mark, for each field)	1 1		
II	3	Some key features of Python and its role in machine learning: 1. Free: - Python is free to use and distribute and is supported by community. - Python interpreter is available for every major platform. 2. Software quality: - Better than traditional and scripting languages. - Readable code, hence reusable and maintainable. - Support for advance reuse mechanisms. 3. Developer productivity: - Much better than statically typed languages. - Much smaller code. - Less to type, debug and maintain. - No lengthy compile and link steps. 4. Program portability: - Python programs run unchanged on most platforms. - Python runs on every major platform currently in use. - Porting program to a new platform usually need only cut and paste. This is true even for GUI, DB access, Web programming interfacing, Directory access, etc 5. Support libraries: - Strong library support from Text pattern matching to networking. - Vast collection of third-party libraries. - Libraries for Web site construction, Numeric programming, Game development, Machine Learning etc. 6. Enjoyment: - Ease of use. - Built-in toolset. - Programming becomes pleasure than work. Python's features, extensive libraries, and the active support of the data science community make it a preferred choice for machine learning and data analysis tasks. Its simplicity, versatility, and rich ecosystem enable developers to work efficiently and effectively on a wide range of machine learning projects, from data preprocessing and model development to deployment and production.	1.5 1.5	3	

II	4	**Decision Control Instruction** • Simple if • if-else • If-elif-ladder **Repetition Control Instruction (Loop)** • While • For (One mark for listing and 2 marks for explanation)		3	

II	5	def factorial(n):
		fact = 1
		i = 1
		while i <= n:
		fact *= i
		i += 1
		return fact
		num = int(input("Enter a number: "))
		fact = factorial(num)
		print("The factorial is ",fact)
			3	3	

II	6	class Student:
		def __init__(self, name, reg_number, marks1, marks2, marks3):
		self.name = name
		self.reg_number = reg_number
		self.marks1 = marks1
		self.marks2 = marks2
		self.marks3 = marks3
		
		def calculate_grade(self):
		average_marks = (self.marks1 + self.marks2 + self.marks3) / 3
		if average_marks >= 90:
		return 'S'
		elif 80 <= average_marks < 90:
		return 'A'
		elif 70 <= average_marks < 80:
		return 'B'
		elif 60 <= average_marks < 70:
		return 'C'
		elif 50 <= average_marks < 60:
		return 'D'
		else:
		return 'F'
		def display_details(self):
		print("Name:", self.name)
		print("Register Number:", self.reg_number)
		print("Marks in Subject 1:", self.marks1)
		print("Marks in Subject 2:", self.marks2)
		print("Marks in Subject 3:", self.marks3)
		print("Grade:", self.calculate_grade())
		student1 = Student("Varun", "9876", 76,88,94)
		student1.display_details()
			3	3	

II	7	In supervised ML algorithm, the possible outcomes are already known and training data is also labeled with correct answers. Classification: A problem is called classification problem when we have the categorized output such as "black", "teaching", "non-teaching", etc. Regression: A problem is called regression problem when we have the real value output such as "distance", "kilogram", etc (1mark for supervised learning,1 for classification and 1 for regression)	1 1 1	3	
II	8	it is used to solve the clustering problems. It is basically a type of unsupervised learning. The main logic of K-Means clustering algorithm is to classify the data set through a number of clusters. Follow these steps to form clusters by K-means: • K-means picks k number of points for each cluster known as centroids. • Now each data point forms a cluster with the closest centroids, i.e., k clusters. • Now, it will find the centroids of each cluster based on the existing cluster members. • We need to repeat these steps until convergence occurs.	3	3	
II	9	**Decision Tree** Decision tree is a supervised learning algorithm that is mostly used for classification problems. Basically, it is a classifier expressed as recursive partition based on the independent variables. Decision tree has nodes which form the rooted tree. Rooted tree is a directed tree with a node called "root". Root does not have any incoming edges and all the other nodes have one incoming edge. These nodes are called leaves or decision nodes. **Random Forest** It is a supervised classification algorithm. The advantage of random forest algorithm is that it can be used for both classification and regression kind of problems. Basically, it is the collection of decision trees (i.e., forest) or you can say ensemble of the decision trees. The basic concept of random forest is that each tree gives a classification and the forest chooses the best classifications from them. Followings are the advantages of Random Forest algorithm: • Random forest classifier can be used for both classification and regression tasks. • They can handle the missing values. • It won't over fit the model even if we have a greater number of trees in the forest.	1.5 1.5	3	
II	10	For building bots to play two player games in AI, we need to install the easyAI library. **A Bot to Play Last Coin Standing** In this game, there would be a pile of coins. Each player has to take a number of coins from that pile. The goal of the game is to avoid taking the last coin in the pile. We will be using the class LastCoinStanding inherited from the TwoPlayersGame class of the easyAI library. **A Bot to Play Tic Tac Toe** Tic-Tac-Toe is very familiar and one of the most popular games. Can create this game by using the easyAI library in Python.	3	3	

PART C					42
III	1	Artificial Intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think and act like humans. **Necessities** • AI can teach itself • AI can respond in real time • AI achieves accuracy • AI can organize data to get most out of it (3 marks for AI, one mark for listing the necessities, one mark for explanation)	3 4	7	
IV	2	a) Healthcare: AI is used to analyze medical images for disease diagnosis and personalized treatment recommendations. Machine learning algorithms can analyze large volumes of medical data to detect patterns, leading to early diagnosis and more effective treatment plans. b) Transportation: AI powers self-driving cars, optimizing traffic flow, and predicting maintenance needs. Autonomous vehicles use AI algorithms to interpret data from sensors and make real-time decisions to navigate roads safely and efficiently. c) Finance: AI algorithms analyze market data to make trading decisions and detect fraudulent activities. AI-driven chatbots and virtual assistants in banking streamline customer interactions and improve customer service. (1mark for listing and 6 marks for the explanation)	1+3*2	7	
V	3	Python supports 3 categories of data types: 1. **Basic types** - int, float, complex, bool, string, bytes 2. **Container types** - list, tuple, set, dict 3. **User-defined types** – class (3 marks for basic types, 2marks for container types, 2 marks for user defined types) (List and explain each)	3+2+2	7	
VI	4	Python function is a block of code that performs a specific and well-defined task. Two main **advantages** of function are: • They help us divide our program into multiple tasks. For each task we can define a function. This makes the code modular. • Functions provide a reuse mechanism. The same function can be called any number of times. There are two types of Python functions: 1. Built-in functions - Ex. len( ), sorted( ), min( ), max( ), etc. 2. User-defined functions **SYNTAX:** # function definition def function_name(arguments separated by comma) : Body of the function **EXAMPLE:** def add(a,b): c=a+b (6 marks for function explanation, advantages and syntax 2 marks for example)	7	7	

VII	5	s=input("Enter the string:")
		for i in range(0,len(s)+1):
		if s[i]!=s[len(s)-i-1]:
		print("The string is not Palindrome")
		break
		else:
		print("The string is palindrome")
		break
			7	7	

VIII	6	numbers = []
		for i in range(5):
		num = int(input("Enter number: "))
		numbers.append(num)
		sum23 = 0
		for num in numbers:
		sum += num
		print ("The sum of the numbers is:",sum)
			7	7	

IX	7	In Unsupervised machine learning, there is no labelled data. Unsupervised learning there will be no correct answer and no teacher for the guidance. Algorithms help to discover interesting patterns in data. Unsupervised learning problems can be divided into the following two kinds of problem: **Clustering:** In clustering problems, we need to discover the inherent groupings in the data. For example, grouping customers by their purchasing behaviour. **Association:** A problem is called association problem because such kinds of problem require discovering the rules that describe large portions of our data. For example, finding the customers who buy both x and y. K-means for clustering, Apriori algorithm for association are the examples of unsupervised machine learning algorithms.	3 2 2	7	
X	8	**Steps in building a classifier** Step 1: Import Scikit-learn Import Sklearn Step 2: Import Scikit-learn's dataset Step 3: Organizing data into sets Step 4: Building the model Step 5: Evaluating the model and its accuracy	7	7	
XI	9	**Data preprocessing steps** Step 1: Importing the useful packages: Step 2: Defining sample data: Step3: Applying preprocessing technique: **Techniques for Data Preprocessing** **Binarization** data_binarized=preprocessing.Binarizer(threshold=0.5).transform(input_data) **Mean Removal** print("Mean =", input_data.mean(axis=0)) **Min max scaling** data_scaler_minmax=preprocessing.MinMaxScaler(feature_range=(0,1)) data_scaled_minmax = data_scaler_minmax.fit_transform(input_data) **Normalization** L1 Normalization L2 Normalization	7	7	
XII	10	**Linear Regression** Mainly **linear regression** is a linear model that assumes a linear relationship between the input variables say x and the single output	4	7	

		variable say y. In other words, we can say that y can be calculated from a linear combination of the input variables x. The relationship between variables can be established by fitting a best line. Linear regression is of the following two types: **Simple linear regression**: A linear regression algorithm is called simple linear regression if it is having only one independent variable. **Multiple linear regression**: A linear regression algorithm is called multiple linear regression if it is having more than one independent variable. **K-Nearest Neighbors (KNN)** It is used for both classification and regression of the problems. It is widely used to solve classification problems. The main concept of this algorithm is that it used to store all the available cases and classifies new cases by majority votes of its k neighbors. The case being then assigned to the class which is the most common amongst its K-nearest neighbors, measured by a distance function. The distance function can be Euclidean, Minkowski and Hamming distance. Consider the following to use KNN: • Computationally KNN are expensive than other algorithms used for classification problems. • The normalization of variables needed otherwise higher range variables can bias it. • In KNN, we need to work on pre-processing stage like noise removal.	3		
XIII	11	Minimax algorithm is the strategy used by combinational search that uses heuristic to speed up the search strategy. The concept of Minimax strategy can be understood with the example of two player games, in which each player tries to predict the next move of the opponent and tries to minimize that function. Also, in order to win, the player always try to maximize its own function based on the current situation. Heuristic plays an important role in such kind of strategies like Minimax. Every node of the tree would have a heuristic function associated with it. Based on that heuristic, it will take the decision to make a move towards the node that would benefit them the most.	7	7	

XIV	12	board = [' '] * 10
		computer = 'X'
		human = 'O'
		
		def display_board(board):
		print(f'{board[1]} | {board[2]} |{board[3]}')
		print(f'{board[4]} | {board[5]} |{board[6]}')
		print(f'{board[7]} | {board[8]} |{board[9]}')
		print('-' * 10)
		
		def check_win():
		if board[1] == board[2] == board[3] and board[1] != ' ':
		return True
		elif board[4] == board[5] == board[6] and board[4] != ' ':
		return True
		elif board[7] == board[8] == board[9] and board[7] != ' ':
		return True
		elif board[1] == board[4] == board[7] and board[1] != ' ':
		return True
		elif board[2] == board[5] == board[8] and board[2] != ' ':
		return True
			7	7	

```
    elif board[3] == board[6] == board[9] and board[3] != ' ':
        return True
    elif board[1] == board[5] == board[9] and board[1] != ' ':
        return True
    elif board[7] == board[5] == board[3] and board[7] != ' ':
        return True
    else:
        return False

def is_win(letter):
    if board[1] == board[2] == board[3] and board[1] == letter:
        return True
    elif board[4] == board[5] == board[6] and board[4] == letter:
        return True
    elif board[7] == board[8] == board[9] and board[7] == letter:
        return True
    elif board[1] == board[4] == board[7] and board[1] == letter:
        return True
    elif board[2] == board[5] == board[8] and board[2] ==letter:
        return True
    elif board[3] == board[6] == board[9] and board[3] == letter:
        return True
    elif board[1] == board[5] == board[9] and board[1] == letter:
        return True
    elif board[7] == board[5] == board[3] and board[7] == letter:
        return True
    else:
        return False


def check_draw():
    if board.count(' ') < 2:
        return True
    else:
        return False


def is_available(pos):
    return True if board[pos] == ' ' else False


def insert(letter, pos):
    if is_available(pos):
        board[pos] = letter
        display_board(board)
        if check_win():
            if letter == 'X':
                print("Computer Wins")
                exit()
            else:
                print("Human wins")
                exit()
        if check_draw():
            print("Draw")
            exit()
    else:
        pos = int(input("Not Free! Please re-enter a position"))
        insert(letter, pos)
```

```
def human_move(letter):
    pos = int(input("Enter the position to insert:"))
    insert(letter, pos)


def computer_move(letter):
    best_score = -100
    best_pos = 0
    for index in range(1, len(board)):
        if is_available(index):
            board[index] = letter
            score = minimax(board, False)
            board[index] = " "
            if score > best_score:
                best_score = score
                best_pos = index
    insert(letter, best_pos)
    return


def minimax(board, is_maximizing):
    if is_win(computer):
        return 10
    elif is_win(human):
        return -10
    elif check_draw():
        return 0
    if is_maximizing: # computer turn   x
        best_score = -100
        for index in range(1, len(board)):
            if is_available(index):
                board[index] = computer
                score = minimax(board, False)
                board[index] = " "
                best_score = max(best_score,score)
        return best_score
    else: # human turn   o
        best_score = 100
        for index in range(1, len(board)):
            if is_available(index):
                board[index] = human
                score = minimax(board, True)
                board[index] = " "
                best_score = min(best_score,score)
        return best_score

# main loop
while not check_win():
    display_board(board)
    computer_move(computer)
    human_move(human)
```