A

INDUSTRIAL TRAINING REPORT

ON

"PYTHON AND ARTIFICIAL INTELLIGENCE"

AT DEVTOWN

SUBMITTED BY

MR. JADHAV GURUDATTA D. ROLL NO. 24

BHARATI VIDYAPEETH'S COLLEGE OF ENGINEERING,

KOLHAPUR

BACHELOR OF TECHNOLOGY

DEPARTMENT OF

ELECTRONICS AND TELECOMMUNICATION ENGINEERING

YEAR 2024-25

CERTIFICATE

This is to certify that, Industrial training report entitled, **"Python and Artificial Intelligence** "at **DevTown** submitted by **"Mr. Jadhav Gurudatta D."** training carried out by them under the guidance of **Mr Shaurya Sinha,** Founder of Devtown and it is approved for partial fulfilment of the requirement of the Shivaji University, Kolhapur during the academic year 2024-2025.

Mr. V.S. Mandlik	**Dr. K. R. Desai**	**Dr. V. R. Ghorpade**
(Guide)	(H.O.D.)	(Principal)

Date: 25/09/2024

Place: Kolhapur

ACKNOWLEDGEMENT

First of all, I would like to express my gratitude to **"DEVTOWN"** enabling me to complete this report.

Successfully completion of any type of project requires help from a number of persons. I have also taken help from different people for preparations of this report. Now, there is a little effort to show my deep gratitude to that helpful person.

I convey my sincere gratitude to my Academics Supervisor **Dr. K. R. DESAI SIR, Head of department of ETC, BHARATI VIDYAPEETH's COLLEGE OF ENGINEERING, KOLHAPUR.** Without his kind direction and proper guidance this study would have been a little success. In every phase of the project his supervision and guidance shaped this report to be completed perfectly.

I would also like to thanks my colleagues to give a clear idea during the training Programme.

Mr. Jadhav Gurudatta D.

B.E. B. Tech.
(Electronics & Telecommunication Engineering)

INDEX

1. ABSTRACT

This report outlines the industrial training undertaken on Python and Artificial Intelligence (AI) at Devtown. The training focused on gaining hands-on experience with Python, one of the most widely used programming languages in AI development, and understanding its practical applications. Over the course of the training, we explored various Python libraries, such as NumPy, Pandas, TensorFlow, and Scikit-learn, which are critical tools for AI and machine learning projects.

The training included a comprehensive study of Python's role in AI development, covering topics like data manipulation, machine learning algorithms, and neural networks. We also developed a project on implementing an AI-powered Tic-Tac-Toe game using the Minimax algorithm, which demonstrated AI's decision-making capabilities in competitive games.

This report encapsulates the training schedule, the projects covered, and the skills gained during the course. It highlights how the knowledge acquired during the training aligns with the theoretical concepts learned in our academic curriculum, preparing us for practical applications in the field of artificial intelligence.

2. CHAPTER 1: INTRODUCTION OF COURSE

1.1 History of Platform:

Owners of platform:

Mr. Shaurya Sinha & Mr. Ashish Modi

Founders of platform:

Mr. Shaurya Sinha & Mr. Ashish Modi

Year of Establishment of the platform:

2020

Vision/Mission Statements of the platform:

Vision -

Be the technical platform to be known for excellence in innovation and development.

Mission -

Devtown is driven by a singular mission: to democratize education and create a world where learning knows no boundaries.

1.2 Objectives of course:

Getting to Know Python

Getting to Know Artificial Intelligence.

Creating a Project using AI

1.3 Training Schedule:

Week	Start Date-End date	Points to be Studied	Completion status
Week 1	6 May 2024 To 12 May 2024	Introduction to Course, Features And Objectives of Course, Python Data types, variables, libraries, use, loops, arguments and AI.	Completed
Week 2	13 May 2024 To 19 May 2024	Hands on Project Completion and Submission.	Completed

1.4 Projects Covered:

Tic-Tac-Toe AI Using Minimax Algorithm.

3. CHAPTER 2: STUDY OF PYTHON AND AI

2.1 Introduction to Python and Artificial Intelligence (AI)

Python has rapidly become one of the most popular programming languages for artificial intelligence (AI) due to its simplicity, flexibility, and vast ecosystem of libraries and frameworks. AI, a branch of computer science, focuses on creating systems capable of performing tasks that typically require human intelligence, such as decision-making, speech recognition, visual perception, and language translation.

2.2 Why Python for AI?

Python's growth in AI is largely attributed to its easy-to-read syntax, which allows developers to focus more on solving complex AI problems than on the intricacies of the language itself. Python also boasts a wide range of powerful libraries and tools designed specifically for AI development, such as:

- **NumPy** and **Pandas** for efficient data manipulation.
- **TensorFlow** and **PyTorch** for building machine learning and deep learning models.
- **Scikit-learn** for implementing traditional machine learning algorithms.

Additionally, Python's community support is extensive, providing a wealth of documentation, tutorials, and code examples that make AI development more accessible to beginners and experts alike.

2.3 Applications of AI

AI, powered by Python, is widely used across various fields:

- **Machine Learning (ML)**: Python facilitates the training and deployment of algorithms that allow machines to learn from data. Applications include recommendation systems, fraud detection, and predictive analysis.
- **Natural Language Processing (NLP)**: AI systems like chatbots and language translation tools utilize Python to process and understand human language.

- **Computer Vision**: AI models trained using Python can interpret and process visual information, used in facial recognition systems, self-driving cars, and healthcare diagnostics.
- **Game Development**: Python is often employed in AI game development, allowing systems to predict and simulate optimal strategies using algorithms like Minimax, widely implemented in games such as Tic-Tac-Toe and Chess.

2.4 Python and AI Algorithms

- AI algorithms range from simple decision-making rules to advanced neural networks. In this report, we focus on the **Minimax algorithm**, a classical AI technique used in decision-making and game theory. It forms the basis for building AI systems that can simulate possible future moves in competitive games, making Python an ideal choice for implementing such algorithms due to its built-in data handling and computation capabilities.

2.5 Minmax Algorithm

The **Minimax Algorithm** is a decision-making algorithm used in artificial intelligence, game theory, and decision-making scenarios, especially in two-player games like chess, tic-tac-toe, and checkers. The goal of the Minimax Algorithm is to minimize the possible loss for a worst-case scenario while maximizing the player's potential gain. In simple terms, one player tries to maximize their score (called the **"maximiser"**), while the other tries to minimize it (called the **"minimizer"**).

Key Concepts of Minimax Algorithm:

1. **Game Tree**: The Minimax Algorithm explores a game tree where nodes represent the possible states of the game, and branches represent the possible moves.
2. **Maximiser vs. Minimizer**:
    - **Maximiser**: The player who tries to get the highest possible score (e.g., AI).
    - **Minimizer**: The opponent who tries to minimize the maximiser's score.
3. **Evaluation Function**: The algorithm assigns a score to the end states of the game tree. For example:
    - A positive score indicates a favourable outcome for the maximiser.
    - A negative score indicates a favourable outcome for the minimizer.
    - A score of zero means a draw.

4. **Recursion**: The Minimax Algorithm recursively evaluates the game tree. It simulates each possible move by both players, eventually reaching a terminal state (win, lose, or draw).
5. **Backtracking**: Once it reaches a terminal state, the algorithm backtracks and assigns scores to previous nodes based on the best possible outcome for each player. The maximiser picks the move that gives the highest score, while the minimizer picks the move that gives the lowest score.

4. CHAPTER 3: PROJECT

Project Title: Tic-Tac-Toe AI Using Minimax Algorithm

3.1 Objective

To develop a Tic-Tac-Toe game in Python where the computer uses the Minimax algorithm to always make the optimal move, either resulting in a win or a draw for the computer.

3.2 Project Description

This project involves creating a Tic-Tac-Toe game where the user plays against an AI-powered computer. The computer is programmed using the Minimax algorithm, which evaluates all possible moves and selects the best one to maximize its chances of winning. The project will help demonstrate how artificial intelligence can be applied to solve game-based problems by simulating all possible future states and choosing the optimal strategy.

3.3 Key Features

- A 3x3 Tic-Tac-Toe board for user input and computer moves.
- The computer will use the Minimax algorithm to calculate the optimal move at each turn.
- The user will always play as "X", and the computer will play as "O".
- The game ends when either the user or the computer wins, or if the game ends in a draw.

3.4 Technologies and Libraries

- **Python**: Core programming language for logic, decision-making, and game flow.
- **Minimax Algorithm**: For decision-making in game moves.
- **NumPy (optional)**: For board representation, though lists can also be used.

3.5 Step-by-Step Implementation

1. **Initialize the Game Board** The game board is a 3x3 grid, represented as a list of 9 elements. Each spot is either empty, taken by an "X" (user), or an "O" (computer).

```
board = [' ' for _ in range (9)] # A 3x3 empty board
```

- **Print the Board Function** A simple function to display the current state of the game board.

```
def print_board(board):
        for row in [board [i * 3:(i + 1) * 3] for i in range(3)]:
        print ("| " + " | ".join(row) + " |")
```

- **Check for Winner** A function to check if there's a winner or if the game ends in a draw.

```
def check_winner(board, player):
        win_conditions = [(0, 1, 2), (3, 4, 5), (6, 7, 8), (0, 3, 6), (1, 4, 7), (2, 5, 8), (0, 4, 8), (2, 4, 6)]
  return any(board[a] == board[b] == board[c] == player for a, b, c in win_conditions)
```

- **Minimax Algorithm Implementation** This function evaluates all possible moves for the computer and returns the optimal move based on the Minimax algorithm.

```
def minimax (board, depth, is_maximizing):
 if check_winner(board, 'O'):
    return 1
 elif check_winner(board, 'X'):
    return -1
 elif ' ' not in board:
    return 0

 if is_maximizing:
    best_score = -float('inf')
    for i in range(9):
       if board[i] == ' ':
          board[i] = 'O'
          score = minimax(board, depth + 1, False)
          board[i] = ' '
          best_score = max(score, best_score)
    return best_score
```

```
    else:
        best_score = float('inf')
        for i in range(9):
            if board[i] == ' ':
                board[i] = 'X'
                score = minimax(board, depth + 1, True)
                board[i] = ' '
                best_score = min(score, best_score)
        return best_score
```

- **Computer Move Using Minimax** The computer evaluates all available moves and chooses the best one using the Minimax algorithm.

```
def computer_move(board):
    best_score = -float('inf')
    best_move = 0
    for i in range(9):
        if board[i] == ' ':
            board[i] = 'O'
            score = minimax(board, 0, False)
            board[i] = ' '
            if score > best_score:
                best_score = score
                best_move = i
    board[best_move] = 'O'
```

- **Main Game Loop** The main loop where the game alternates between the player and the computer until there's a winner or a draw.

```
def tic_tac_toe():
    while True:
        print_board(board)
        user_move = int(input("Enter a position (1-9): ")) - 1
        if board[user_move] == ' ':
```

```
            board[user_move] = 'X'
            if check_winner(board, 'X'):
                print_board(board)
                print("You win!")
    break
if ' ' not in board:
    print_board(board)
    print("It's a draw!")
    break

computer_move(board)
if check_winner(board, 'O'):
    print_board(board)
    print("You lose!")
    break
if ' ' not in board:
    print_board(board)
    print("It's a draw!")
    break
```

- **Run the Game**

```
tic_tac_toe()
```

5. CONCLUSION

In this project, we implemented a Tic-Tac-Toe game with a computer opponent that uses the Minimax algorithm to always make optimal moves. The AI either wins or forces a draw, making it a robust implementation of decision-making in AI. This project not only showcases the use of Python for game development but also demonstrates the application of fundamental AI algorithms.

6. FUTURE ENHANCEMENTS

- Adding a graphical user interface (GUI) using libraries like **Tkinter** or **Pygame**.
- Expanding the AI to other board games like chess.
- Optimizing the algorithm for larger, more complex games using techniques like **Alpha-Beta pruning**.

7. REFERENCES

- Campus Ambassador : Gurudatta Jadhav.
- Youtube : Devtown

8. E- CERTIFICATE: