Date __/__/____

Saathi

S. Bhanupriya
ISE-A
220401040

POAI OBSERVATION

Page No.

Saathi

Date __/__/____

S NO	DATE	TITLE	MARKS	T.S
1.	01-08-24	Basic python programs	10	
2.	07-08-24	Domain sentiment analysis	10	
3.	09-08-24	N Queens	10	
4.	16-08-24	Depth First search	10	
5	06-09-24	A* algorithm	10	
6.	-09-24	AO* algorithm	10	
7.	25-10-24	Decision Tree	10	
8.	25-10-24	K-means	10	
9.	25-10-24	Artificial Neural Network	10	
10.	30-08-24	Min max	10	
11.	18-09-24	Introduction to Prolog.	10	
12.	18-09-24	Prolog family Tree.	10	
13.	23-08-24	Depth First - Watering	10	

completed

Page No.

Saathi

Date __/__/____

Python Programs

1. Simple calculator

```
def add(x,y):
    return x+y
def subtract(x,y):
    return x-y
def multiply(x,y):
    return x*y
def division(x,y):
    if y==0:
        return "Error: Division by zero!"
    else:
        return x/y
print("select operation:")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")
choice = input("Enter choice (1/2/3/4): ")
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
if choice == '1':
    print(num1, "+", num2, "=", add(num1,num2))
elif choice == '2':
    print(num1, "-", num2, "=", subtract
                                (num1, num2))
elif choice == '3':
    print(num1, "*", num2, "=", Multiply
                                (num1, num2))
elif choice == '4':
    print(num1, "÷", num2, "=", divide
                                (num1, num2))
```

Page No.

Date __/__/____

Saathi

```
else:
    print("Invalid Input")
```

Output:

Enter choice (1|2|3|4): 1
Enter first number: 1
Enter second number: 3
1.0 + 3.0 = 4.0

2. Counting the no. of occurences of an element in the list.

```
def count(lst, x):
    count = 0
    for ele in lst:
        if (ele == x):
            count = count + 1
    return count
lst = [8, 6, 8, 10, 8, 20, 10, 8, 8]
x = 10
print('{} has occured {} times'.format
                (x, count x (lst, x)))
```

Output:

10 has occured 2 times.

3. Multiply 2 list.

```
def multiplyList(myList):
    result = 1
    for x in myList:
        result = result * x
    return result
list1 = [9, 2, 12]
```

Page No.

Date __/__/____

Saathi

```
list 2 = [3,7,4]
print (multiplyList (list1))
print (multiplyList (list2))
```

Output:

216

84

4. Deleting the item:

```
lst = ['Manish', 'xyz', 'Rose', 'Priya', 'Liy',
       'canva']
print ("Original list is :", lst)
lst.remove ('canva')
print ("After deleting the item :", lst)
```

Output:

Original list is : ['Manish', 'xyz', 'Rose', 'Priya', 'Liy', 'canva']

After deleting the item : ['Manish', 'xyz', 'Rose', 'Priya', 'Liy']

5. Matrix Multiplication:

```
def matrix_multiply (A,B):
  n, m = len (A), len (B[0])
  p = len (B)
  result = [[0] * m for _ in range (n)]
  for i in range (n):
    for j in range (m):
      for k in range (p):
        result [i] [j] += A [i] [k] * B [k] [j]
  return result
A = [
```

Page No.

```
    [5,2],
    [8,4]
]
B = [
    [6,6],
    [7,9]
]
result = matrix_multiply(A,B)
print("Matrix A:")
for row in A:
    print(row)
print("\n Matrix B:")
for row in B:
    print(row)
```

Output:

Matrix A: [5,2]
[8,4]
Matrix B: [6,6]
[7,9]

6. Concatenating list:

```
test_list3 = [1,4,5,6,5]
test_list4 = [3,5,7,2,5]
test_list3 = test_list3 + test_list4
print("concatenated list using +:" + str
                (test_list3))
```

Output:

concatenated list using +:
[1,4,5,6,5,3,5,7,2,5]

Saathi

Date __/__/____

Saathi

7. Second largest Element:

```
list1 = [10, 20, 87, 4, 45, 45, 22, 99, 9]
list2 = list(set(list1))
list2.sort()
print("Second largest element :", list2[-2])
```

Output:
Second largest element is :87

8. Odd frequency characters:

```
test_str = 'Hello world, how'
x = set(test_str)
res = []
for i in x:
    if(test_str.count(i)%2!=0):
        res.append(i)
print("The odd frequency characters are
        : " + str(res))
```

Output:
The odd frequency characters are :
['', ',', 'r', 'W', 'd', 'i', 'e', 'l']

9. Harshad No:

```
def checkHarshad(n):
    sum=0
    temp=n
    while temp>0:
        sum = sum + temp%10
        temp = temp//10
    return n%sum==0
```

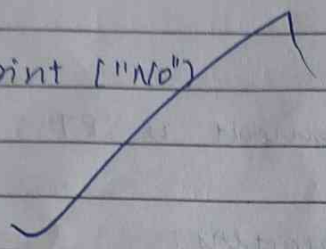
str

Page No.

Saathi

Date __/__/____

```
if [checkMarshad (12)]:
    print ("Yes")
else:
    print ("No")
if (checkMarshad (15)):
    print ("Yes")
else:
    print ["No"]
```



Page No.

Date 09/08/2024

Saathi

PROJECT NAME: TOURISM RECOMMENDATION SYSTEM

DOMAIN: TOURISM AND TRAVEL

PROBLEM STATEMENT: Travelers struggle to find personalized travel recommendations that match their interests, leading to inefficient planning and less satisfying experiences.

SOLUTION: Create a machine learning system that provides personalized travel recommend-ations by analyzing user preferences and past behaviour, enhancing travel planning and satisfaction.

TARGET AUDIENCE: 1. Individual travelers
2. Travel Agencies
3. Tourism Boards
4. Travel Enthusiasts

1. Individual travelers: Seeking tailored travel ideas.
2. Travel Agencies: Enhancing client recommendations.
3. Tourism Boards: Promoting destinations personally.
4. Travel Enthusiasts: Wanting customized trip suggestions.

OBJECTIVES: 1. Personalization: Provide customized travel recommendations based on user preferences.
2. Efficiency: Improve travel planning with accurate and relevant

Page No.

Date __/__/____

Saathi

suggestions.

3. User Experience: Enhance overall satisfaction with targeted travel options.

ALGORITHM USED: Singular Value Decomposition (SVD). It effectively decomposes user-item interaction data into latent factors to provide personalized recommendations based on user preferences and item characteristics.

ABSTRACT:

Travelers often struggle to find personalized travel recommendations, resulting in inefficient planning and suboptimal experiences. To address this, we propose a machine learning system using Singular Value Decomposition (SVD) to offer tailored travel suggestions based on user preferences and past behaviours. Our system aims to improve travel planning efficiency and user satisfaction. SVD simplifies complex data, handles missing information, identifies hidden patterns, and scales efficiently, providing accurate and relevant recommendations for individual travelers, travel agencies, tourism boards and travel enthusiasts.

DATASET:

→ Yelp Dataset Challenge

Page No.

06-09-24

1. N- Queens Problem

AIM:

To execute N- Queens Problem using Python.

PROGRAM:

```
def print_board(board):
    for row in board:
        print(" ".join("Q" if col else "." for col in row))
    print()
def is_safe(board, row, col, N):
    for i in range(row):
        if board[i][col]:
            return False
    for i, j in zip(range(row, -1, -1),
                    range(col, -1, -1)):
        if board[i][j]:
            return False
    for i, j in zip(range(row, -1, -1),
                    range(col, N)):
        if board[i][j]:
            return False
    return True
def solve_n_queens_util(board, row, N):
    if row >= N:
        return True
    for col in range(N):
        if is_safe(board, row, col, N):
            board[row][col] = True
            if solve_n_queens_util(board, row+1, N):
                return True
            board[row][col] = False
```

```
return False
def solve_n_queens(N):
    board = [[False] * N for _ in range(N)]
    if not solve_n_queens_util(board, 0, N):
        print("No solution exists")
    else:
        print_board(board)
def main():
    try:
        N = int(input("Enter the number of
                queens (N): "))
        if N <= 0:
            print("The number of queens must
                be a positive integer.")
        else:
            solve_n_queens(N)
    except ValueError:
        print("Invalid input. Please enter a
                positive integer.")

if __name__ == "__main__":
    main()
```

Output:

1. Enter the number of queens (N): 4

```
. Q . .
. . . Q
Q . . .
. . Q .
```

2. Enter the number of queens (N): 2
   No Solution exists.

3. Enter the number of queens (N): 8

Q . . . . . .
. . . . Q . . .
. . . . Q
. . . . . Q . .
. . Q . . . . .
. . . . . . Q .
. Q . . . . . .
. . . Q . . . .

2: DEPTH FIRST SEARCH

AIM:

To implement Depth First Search using Python.

CODE:

```
class Node:
  def __init__(self, value):
    self.value = value
    self.neighbors = []
  def add_neighbor(self, neighbor):
    self.neighbors.append(neighbor)
class Graph:
  def __init__(self):
      self.nodes = {}
def add_node(self, value):
    if value not in self.nodes:
      self.nodes[value] = Node[value]
def add_edge(self, from_value, to_value):
    if from_value in self.nodes and to_value
                                  in self.nodes:
      self.nodes[from_value].add_neighbor
                    (self.nodes[to_value])
```

```
        self.nodes[to_value].add_neighbor
                    (self.nodes[from_value])
    def dfs(self, start_value):
        visited = set()
        stack = [self.nodes.get(start_value)]
        while stack:
            node = stack.pop()
            if node and node.value not in visited
                print(node.value)
                visited.add(node.value)
            for neighbor in reversed(node.neighbors
                if neighbor.value not in visited:
                    stack.append(neighbor)

graph = Graph()
graph.add_node('x')
graph.add_node('y')
graph.add_node('z')
graph.add_node('u')
graph.add_node('v')
graph.add_node('w')

graph.add_edge('x', 'y')
graph.add_edge('x', 'z')
graph.add_edge('y', 'u')
graph.add_edge('y', 'v')
graph.add_edge('z', 'w')
graph.add_edge('v', 'w')

print("DFS starting from 'x':")
result = graph.dfs('x')
print(result)
```

Date __/__/____

Saathi

Output:

DFS starting from 'x':

x
y
v
w
z

3. A* SEARCH ALGORITHM

AIM:

To implement A* Search Algorithm using Python.

CODE:

```
def astaralgorithm(start_node, stop_node):
  open_set = set(start_node)
  closed_set = set()
  g={}
  parents = {}
  g[start_node] = 0
  parents [start_node] = start_node
  while len(open_set) > 0:
    n= None
    for v in open_set:
      if n == None or g[v] + heuristic(v) < g[n]
                    +heuristic(n):
        n=v
    if n == stop_node or graph_nodes[n]==None:
      pass
    else:
      for (m, weight) in get_neighbors(n):
        if m not in open_set and m not in
                                closed_set:
```

```
            open_set.add(m)
            parents[m]=n
            g[m] = g[n] + weight
        else:
            if g[m] > g[n] + weight:
                g[m] = g[n] + weight
                parents[m]=n
                if m in closed_set:
                    closed_set.remove(m)
                    open_set.add(m)
    if n==None:
        print("path does not exist!")
        return None
    if n==stop_node:
        path=[]
        while parents[n]!=n:
            path.append(n)
            n=parents[n]
        path.append(start_node)
        path.reverse()
        print("Path found: {}".format(path))
        return
    open_set.remove(n)
    closed_set.add(n)
print("Path does not exist!")
return None
def get_neighbors(v):
    if v in Graph_nodes:
        return graph_nodes[v]
    else:
        return None
def heuristic(n):
    H_dist ={
        'A' : 11,
```

```
'B' : 6
'C' : 99,
'D' : 1
'E' : 7
'G' : 0,
}
return H-dist[n]
graph-nodes = {
    'A' : [('B', 2), ('E', 3)],
    'B' : [('C', 1), ('G', 9)],
    'C' : None
    'E' : [('D', 6)],
    'D' : [('G', 1)],
}
astaralgorithm ('A', 'G')
```

Output:

Path found ['A', 'E', 'D', 'G']

4. WATERJUG PROBLEM USING DFS

AIM:

To implement water jug problem using Python.

CODE:

```
def water-jug-problem-dfs (jug1_cap, jug2-cap,
                           target-amount):
    j1 = 0
    j2 = 0
    actions = [("fill", 1), ("fill", 2), ("empty", 1),
    ("empty", 2), ("pour", 1, 2), ("pour", 2, 1)]
    visited = set()
    stack = [(j1, j2, [])]
    while stack:
```

```
j1, j2, seq = stack.pop()
if (j1, j2) not in visited:
    visited.add((j1, j2))
if j1 == target_amount or j2 == target_amount:
    return seq
for action in actions:
    if action[0] == "fill":
        if action[1] == 1:
            next_state = (jug1_cap, j2)
        else:
            next_state = (j1, jug2_cap)
    elif action[0] == "empty":
        if action[1] == 1:
            next_state = (0, j2)
        else:
            next_state = (j1, 0)
    else:
        if action[1] == 1:
            amount = min(j1, jug2_cap - j2)
            next_state = (j1 - amount,
                          j2 + amount)
        else:
            amount = min(j2, jug1_cap - j1)
            next_state = (j1 + amount, j2 - amount)

    if next_state not in visited:
        next_seq = seq + [action]
        stack.append((next_state[0],
                      next_state[1], next_seq))
return None
result = water_jug_problem_dfs(4, 3, 2)
print(result)
```

Saathi

Date __/__/____

Output:

[('fill', 2), ('pour', 2,1), ('fill', 2),
('pour', 2,1)].

18/10/24. Tourism Recommendation System

J.A.Adlin Layola
Assistant Professor
CSE
adlinlayola.ja@rajalakshmi.edu.in.

A)

B) cite link

* Add Authors,
* Reduce content
* Add more literature.
* In Ref.
  - cite link for all papers,
  - Add formula links.

Page No.

Saathi

Date __/__/____

5. MINMAX ALGORITHM

AIM:

To implement Minmax problem using Python

CODE:

```
def minimax(depth, nodeIndex, isMaximumPlayer,
                                scores, height):
    if depth == height:
        return scores[nodeIndex]
    if isMaximizingPlayer:
        best = float('-inf')
        for i in range(2):
            val = minimax(depth+1, nodeIndex * 2,
                          False, scores, height)
            best = max(best, val)
        return best
    else:
        best = float('Inf')
        for i in range(2):
            val = minimax(depth+1, nodeIndex
                  * 2 + i, True, scores, height)
            best = min(best, val)
        return best
num_leaf_nodes = 8
scores = []
print("Enter the terminal values for the
                    leaf nodes one by one:")
leaf_nodes = ['H', 'I', 'J', 'K', 'L', 'M', 'N', 'O']
for node in leaf_nodes:
    value = int(input(f"Enter the value for
                leaf node {node}: "))
    scores.append(value)
```

Page No.

Date __/__/____

Saathi

height = 3

optimal_value = minimax(0, 0, True, scores, height)

print(f"The optimal value for the Maximizer is {optimal_value}")

Output:

Enter the terminal values for the leaf nodes:

H: 2

I: -1

J: 3

K: 5

L: 8

M: 0

N: 6

O: 9

The optimal value of the Maximizer is 8

Result:

Thus the Minmax problem is done and executed successfully.

Page No.

Date ___/___/_____

Saathi

6. IMPLEMENTATION OF DECISION TREE CLASSIFICATION TECHNIQUES

AIM:

To implement a decision tree classification technique for gender classification using python.

EXPLANATION:

- Import tree from sklearn
- Call the function DecisionTreeClassifier from tree.
- Assign values for x and y
- Call the function predict for Predicting on the basis of given random values for each given feature.
- Display the output.

CODE:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
data = {
    'Height': [150, 160, 170, 180, 165, 175, 155, 185, 162, 172],
    'Weight': [50, 60, 70, 80, 65, 75, 55, 85, 68, 78],
    'Gender': ['Female', 'Female', 'Male', 'Male',
               'Female', 'Male', 'Female', 'Male', 'Female', 'Male']
}
df = pd.DataFrame(data)
x = df [['Height', 'Weight']]
Y = df ['Gender']
classifier = DecisionTreeClassifier()
classifier.fit(X, Y)
height = float(input("Enter height (in cm) for
                     prediction : "))
```

Page No.

Date __/__/____

Saathi

```
weight = float(input("Enter weight (in kg)
                    for prediction: "))
random_values = pd.DataFrame([[height,
                weight]], columns=['Height', 'Weight'])
predicted_gender = classifier.predict(random_
                                      values)
print(f"Predicted gender for height {height}
      cm and weight {weight} kg: {predicted_
                                 gender[0]}")
```

Input & Output:

Enter height (in cm) : 150
Enter weight (in kg) : 50
Predicted gender for height 150.0 cm
and weight 50.0 kg : Female.

Result:

Thus the implementation of Decision Tree classification technique is done and executed successfully

Page No.

Saathi

Date __/__/____

7 IMPLEMENTATION OF CLUSTERING TECHNIQUES
K-MEANS

AIM:
To implement a K-Means clustering technique using python language

EXPLANATION:
- Import K-Means from sklearn.cluster
- Assign x and y
- Call the function KMeans()
- Perform scatter operation and display the output

CODE:

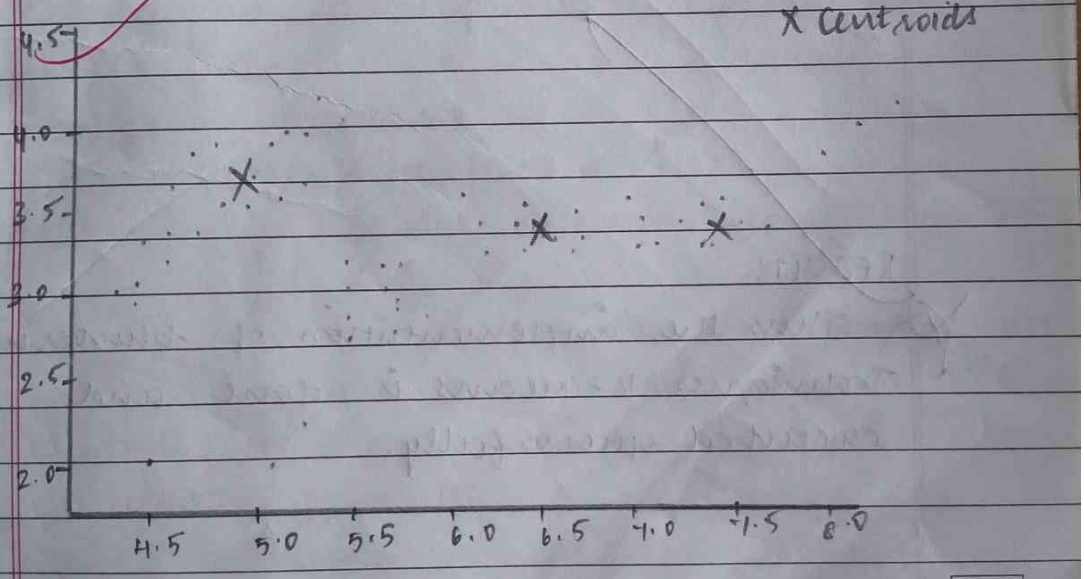
```
import numpy as np
import matplotlib.pyplot as plt
num_points = int(input("Enter the number of
                        data points :"))
X = np.zeros((num_points, 2))
for i in range(num_points):
    x= float(input(f"Enter x-coordinate for
                    data point {i+1} :"))
    y= float(input(f"Enter y-coordinate for
                    data point {i+1} :"))
    X[i] = [x,y]
num_clusters = int(input("Enter the number of
                          clusters :"))
def kmeans(X, num_clusters, max_iters=100):
    centroids = X[np.random.choice(X.shape[0]
                num_clusters, replace=False)]
for _ in range(max_iters):
    distances = np.linalg.norm(X[:,np.newaxis]
                -centroids, axis=2)
```

Page No.

Saathi

Date __/__/____

```
    labels = np.argmin(distances, axis=1)
    new_centroids = np.array([X[labels==k].
        mean(axis=0) for k in range
                    (num_clusters)])
    if np.all(centroids == new_centroids):
        break
    centroids = new_centroids
return labels, centroids
labels, centroids = kmeans(X, num_clusters)
print("cluster labels:\n", labels)
print("centroids:\n", centroids)
plt.figure(figsize=(8,6))
plt.scatter(X[:,0], X[:,1], c=labels, cmap='viridis',
            marker='o', label='Data points')
plt.scatter(centroids[:,0], centroids[:,1] c='red',
            s=200, marker='x', label='centroids')
plt.title('K-means clustering (from scratch)')
plt.xlabel('Feature1')
plt.ylabel('Feature2')
plt.legend()
plt.grid(True)
plt.show()
```



Page No.

Date __/__/____

Saathi

RESULT:

Thus the implementation of Clustering Techniques K-means is done and executed successfully.

Page No.

Date __/__/____

Saathi

7. AIM:

To implementing artificial neural networks for an application in Regression using python.

Algorithm:

→ Generate Data: Create sample data with a linear relationship.
→ Prepare data: Scale the input and output values.
→ Split data: Divide the data into training and testing sets.
→ Build model: Create an ANN with input, hidden and output layers.
→ Compile model: Choose an optimizer and a loss function.
→ Train model: Fit the model to the training data over several epochs.
→ Evaluate Model: Test the model using the testing test.
→ Make Predictions: Use the trained model to predict outcomes.
→ Visual Results: Plot actual vs. predicted values to assess performance.

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test
                                    _split
from sklearn.preprocessing import StandardScaler
from tensorflow import keras
from tensorflow.keras import layers
np.random.seed(0)
X = 2 * np.random.rand(1000, 1)
```

Page No.

Saathi

Date __/__/____

```
y = 4 + 3 * X + np.random.randn(1000,1)
scaler_X = StandardScaler()
scaler_y = StandardScaler()
X_scaled = scaler_X.fit_transform(X)
y_scaled = scaler_y.fit_transform(y)
X_train, X_test, y_train, y_test = train_test_
        split(X_scaled, y_scaled, test_size=0.2,
                random_state=42)
model = keras.Sequential([
  layers.Dense(64, activation='relu', input_shape
                = (X_train.shape[1],)),
  layers.Dense(64, activation='relu'),
  layers.Dense(1)
])
model.fit(X_train, y_train, epochs=100,
          batch_size=32, verbose=0)
loss = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss}')
y_pred_scaled = model.predict(X_test)
y_pred = scaler_y.inverse_transform
                (y_pred_scaled)
plt.figure(figsize=(10,6))
plt.scatter(scaler_X.inverse_transform
  (X_test), scaler_y.inverse_transform(y_test),
  label='Actual Data', color='blue', alpha=0.5)
plt.scatter(scaler_X.inverse_transform
  (X_test), y_pred, label='Predictions',
          color='red', alpha=0.5)
plt.title('Actual vs Predicted')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()
```

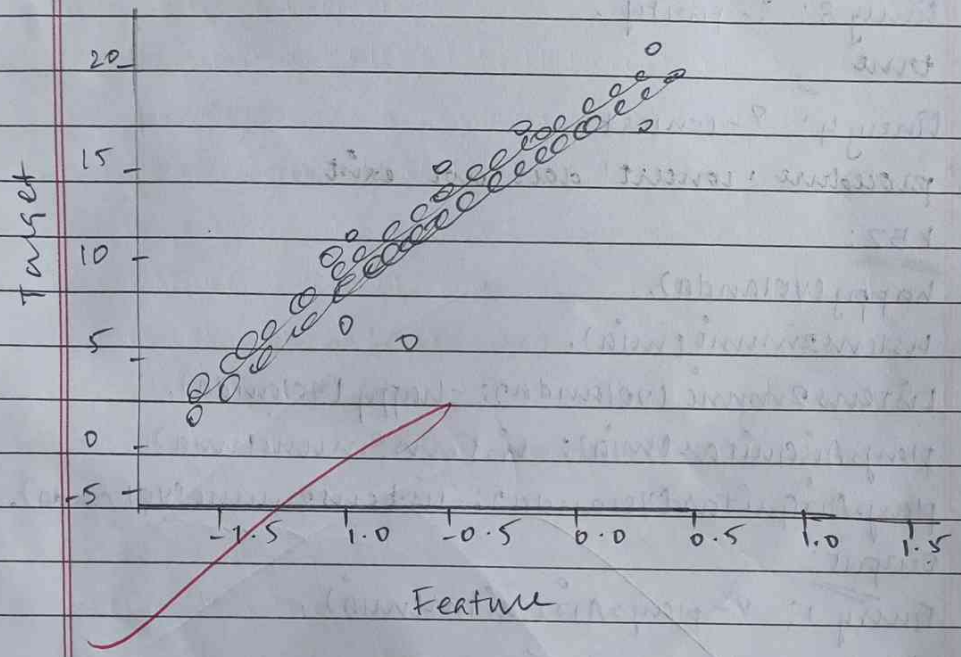
Page No.

Date __/__/____

	Feature	Target
0	3.745401	7.846204
1	9.507143	16.343597
2	7.319939	15.400275
3	5.986585	13.194341
4	1.560186	4.239954

Epoch 1/100
20/20 ———— 25 10ms/step loss: 143.6730
:
Epoch 100/100
20/20 ———— 0s 4ms/step - loss: 4.2248

Test loss (MSE): 3.5400872230529785



Result:

Thus the implementation AI neural networks for an application in Regression using python is done and executed successfully.

Page No.

Date __/__/____

Saathi

8. Introduction to Prolog

Source code:

KB1:

woman(mia).
woman(jody).
woman(yolanda).
plays Air Guitar(jody).
party.

Output:

Query 1: ?- woman(mia).
true

Query 2: ?- plays Air Guitar(mia).
false

Query 3: ?- party.
true

Query 4: ?- concert.
procedure 'concert' does not exist

KB2:

happy(yolanda).
listens2music(mia).
Listens2music(yolanda):- happy(yolanda).
plays Air guitar(mia):- listens2music(mia).
plays Air Guitar(yolanda):- listens2music(yolanda).

Output:

Query 1: ?- plays Air Guitar(mia).
true

Query 2: ?- plays Air guitar(yolanda).
true

KB3:

likes(dan, sally).
likes(sally, dan).
likes(jonn, brittney).

Page No.

Date __/__/____

Saathi

married(X,Y) :- likes(X,Y), likes(Y,X).
friends(X,Y) :- likes(X,Y); likes(Y,X).
Output:

Query 1: ?- likes(dan, X)
X = sally

Query 2: - married(dan, sally).
true

Query 3: ?- married(john, britney).
false

KB4:
food(burger).
food(sandwich).
food(pizza).
lunch(sandwich).
dinner(pizza).
meal(X) :- food(X).
Output:

Query 1: ?- food(pizza)
true

Query 2: ?- meal(X), lunch(X)
X = sandwich

Query 3: ?- dinner(sandwich)
false

KB5:
owns(jack, car(bmw)).
owns(john, car(chevy)).
owns(olivia, car(civic)).
owns(jane, car(chevy)).
sedan(car(bmw)).
sedan(car(civic)).
truck(car(chevy)).

Page No.

Saathi

Date / /

Output:

Query 1: ?- owns (John, X)
X = car (chevy)

Query 2: ?- owns (John, _)
true

Query 3: ?- owns (Who, car (chevy)).
Who = John

Query 4: ?- owns (Jane, X), sedan (X).
fail

Query 5: ?- owns (Jane, X), truck (X)
X = car (chevy)

Result:
Thus to learn PROLOG terminologies & write programs is done & executed successfully.

Page No.

Date __/__/____

Saathi

9. Prolog Family Tree

Aim:

To develop a family tree program using prolog with all possible facts, rules and queries.

Source code:

knowledge base:

```
/* FACTS. */
male (peter).
male (john).
male (chris).
male (kevin).
female (betty).
female (jessy).
female (lisa).
female (helen).
parentof (chris, peter).
parentof (chris, betty).
parent of (helen, peter).
parent of (helen, betty).
parent of (kevin, chris).
parentof (kevin, lisa).
parent of (jessy, john).

/* RULES ::*/
/* son, parent.
   son, grandparent */
father (x, y) :- male (y), parent of (x, y).
mother (x, y) :- female (y), parent of (x, y).
grandfather (x, y) :- male (y), parent of (x, z),
                      parent of (z, y)
grandmother (x, y) :- female (y), parent of (x, z),
```

Page No.

Saathi

Date __/__/____

parent of (X,Y).
brother(X,Y):- male(Y), father(X,Z),
father(Y,W), Z==W.
sister(X,Y):- female(Y), father(X,Z),
father(Y,W), Z==W.

Output:

?-male(Y), parent of (X,Y)

Y	X
peter	chris
peter	helen
john	jeny
chris	kevin

?- female(Y), parent of (X,Y)

Y	X
betty	chris
betty	helen
lisa	kevin
helen	jeny

?- male(Y), parent of (X,Z), parent of (Z,Y)

Y	X	Z
peter	kevin	chris
peter	jeny	helen

?- female(Y), parent(X,Z), parent of (Z,Y)

Y	X	Z
betty	kevin	chris
betty	jeny	helen

Page No.

Date __/__/____

Saathi

?- male(Y), father(X, Z), father(Y, W),
Z==W

procedure father(A,B) does not exist.

Result:

Thus, we have developed a family tree prolog program with all possible facts, rules and queries.

Page No.