INDEX

NAME Dhanyaa Vanisha A. (2207071062) SUBJECT _____

STD. _____ DIV. _____ ROLL NO. _____ SCHOOL _____

SR. NO.	~~PAGE NO.~~ Date	TITLE	~~DATE~~ marks	TEACHER'S SIGN / REMARKS
~~X~~.	.	Python Programs	9	
~~X~~		Project : Details.	9	
2		Code for DFS Search	9	
~~3~~ 1		N- Queens Problem	10	
3.		A* Algorithm	10	
4		Water Jug Problem.	10	
5.		Implementation of decition tree classification techniques	10	
6		Implementation of K-Means clustering technique	10	
7		Implementation of Artifical Neural Networks for An Application in Regression	10	
8		Introduction to Prolog -	10	
9		Prolog family tree	10	
10		Minimax Algorithm.	10	

completed

Date : _____

1

```
def factorial (n):
    if n< 0:
        return 0
    elif n==0 or n == 1:
        return 1
    else:
        fact = 1
        while (n>1):
            fact *= n
            n-=1
        return fact

num = int (input("enter a number: "))
print(" Factorial of", num, "is", factorial (num))
```

Output : enter a number : 4
Factorial of 4 is 24.

2.

```
def largest (arr, n):
    max = arr[0]
    for i in range (1, n):
        if arr[i] > max:
            max = arr[i]
    return max
if __name__ == '__main__':
    arr = [10, 324, 45, 90, 9808]
    n = len(arr)
    ans = largest(arr, n)
    print (" Largest in given array is ", ans)
```

output : Largest in gvn array is 9808 .

Date :

3
```
def palindrome(s):
    return s == s[::-1]
s = "malayalam"
ans = palindrome(s)
if ans:
    print("Yes")
else
    print("No")
```

Output : yes

4
```
def armstrong(n):
    sum = 0
    temp = n
    while temp > 0:
        digit = temp % 10
        sum += digit ** 3
        temp = temp // 10
    if n == sum:
        print(n, "is a armstrong number")
    else
        print(n, "is not an armstrong number")
n = 153
armstrong(n)
```

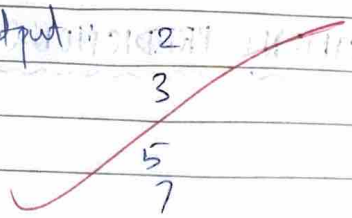
output : 153 is an armstrong number.

5)
```
def prime(a,b):
    for num in range(a, b+1):
        if num > 1:
            for i in range(2, num):
                if (num % i) == 0:
                    break
            else:
                print(num)
a = 1
b = 10
```

Date : _____

prime (a, b)

output: 2
3
5
7



Date :

Project

FUTURE INVESTMENT PREDICTION (FIP)

Problem Statement

Most investors find it hard to make intelligent decisions due to the difficulty in handling financial markets: cryptocurrency, stocks, and valuable metals. They lack the tools that aid in easy analysis of past and present data of the market, including the capacity of predicting future trends that help in tailoring investment strategies to maximize gains within a shorter period of time. Hence, the need for this service project is to work out a platform that can handle these challenges by its service of data-driven insights, predictive analysis and personalized investment recommendations based on individual goals with respect to financial and market dynamics.

Problem Description

In the high-speed financial markets of today, it becomes difficult for investors to make the right decisions with respect to various assets, such as cryptocurrencies, stocks and precious metals like gold and silver. Huge volumes of data, along with the volatile nature of markets, make it really hard for an individual investor to discover profitable opportunities and optimize his investment strategy. Besides, more than ever, timely and accurate predictions along with personalized recommendations are required. The above-discussed platform aims to empower users by processing their investment preferences and financial goals to give out strategies that maximize profit in the shortest possible time, thereby bridging the gap between complicated & actionable insights.

Date : _____

Abstract:

It propose an all-new investment platform mastering cutting-edge data analytics and machine learning to analyze historical and real-time data across cryptocurrency, stock and precious metal-gold and silver- markets. this investment platform will be capable of integrating time series analysis, predictive modeling, and sentiment analysis in a manner that derives forecasts related to prospective price movements and market trends. Users can input the investment amount, which, coupled with market data, the system processes to come up with relevent personally important investment strategies. This tries to maximize the users profit within a time constraint by using data-driven actionable insights and optimized portfolio allocation across multiple asset classes.

Target audience:

Investors' needs range from those of retail investors, needing basic or sophisticated tools on investing in cryptocurrencies, to enthusiasts looking for more data-driven insights. Precious metals and collectibles investors zero in on market predictions and intrinsic value. The risk-averse, including retirees, look forward to safety, which DIY investors would like the proper tools that will help them in making decision. HNWIs pursue tailored solutions, while educational institutions implement learning platforms. Technology and AI lovers discover another frontier, innovative finance solutions.

Objective:

The objective is to create a platform that provides an accurate price prediction and personalized investment recommendation system in line with user inputs, all targeted towards securing the best possible investment strategy to be adopted in order to yield maximum profitability within a short period. Integrate data-driven insight with predictive modeling in a manner that

Date :

allows users to make informed decisions towards better financial outcomes.

Domain:

Our fintech project focuses on leveraging innovative financial technologies and data analytics to predict market trends and optimize investments, key areas include applying machine learning, deep learning and NLP for price prediction and sentiment analysis, using quantitative finance for asset pricing and strategy development, and investing in software development to ensure robust user interfaces and data integration.

Date : _____

1. N-Queens Problem

Aim: To write the code for N-Queens Problem.

Code:

```
N= int(input("Enter N:"))

def solveNQueens(board, col):
    if col == N:
        print(board)
        return True
    for i in range(N):
        if isSafe(board, i, col):
            board[i][col] = 1
            if solveNQueens(board, col+1):
                return True
            board[i][col] = 0
    return False

def isSafe(board, row, col):
    for x in range(col):
        if board[row][x] == 1:
            return False
    for x,y in zip(range(row,-1,-1), range(col,-1,-1)):
        if board[x][y] == 1:
            return False
    for x,y in zip(range(row,N,1), range(col,-1,-1)):
        if board[x][y] == 1:
            return False
    return True

board = [[0 for x in range(N)] for y in range(N)]
if not solveNQueens(board, 0):
    print("No Solution found").
```

Date :

Output

for N=4 :

```
Enter N=4
[[0, 0, 1, 0],
 [1, 0, 0, 0],
 [0, 0, 0, 1],
 [0, 1, 0, 0]]
```

for N=8 :

```
Enter N=8
[[1, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 1, 0],
 [0, 0, 0, 0, 1, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 1],
 [0, 1, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 1, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 1, 0, 0],
 [0, 0, 1, 0, 0, 0, 0, 0]]
```

Result :

Thus, the program for N-Queen's problem have been executed & verified successfully.

Date : _____

2) Depth first Search

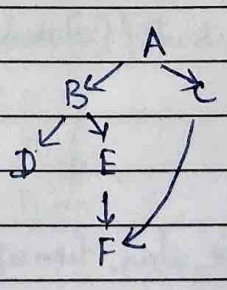
Aim: To write a program for depth first search algorithm.

Code:

```
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)
    print(start, end=" ")
    for neighbour in graph[start]:
        if neighbour not in visited:
            dfs(graph, neighbour, visited)

graph = {
'A' = ['B', 'C'],
'B' = ['D', 'E'],
'C' = ['F'],
'D' = [],
'E' = ['F'],
'F' = []
}
dfs(graph, 'A')
```

Output:

A B D E F C



Result:

Thus the program for depth first search has been executed & verified successfully.

Date :

3) A* Search algorithm:

Aim:

To write a program to execute A* search algorithm.

Code:

```
import heapq

class Node:
    def __init__(self, parent=None, position=None):
        self.parent = parent
        self.position = position

        self.g = 0
        self.h = 0
        self.f = 0

    def __eq__(self, other):
        return self.position == other.position

def astar(maze, start, end):
    start_node = Node(None, start)
    end_node = Node(None, end)

    open_list = []
    closed_list = []

    open_list.append(start_node)

    while len(open_list) > 0:
        current_node = open_list[0]
        current_index = 0
        for index, item in enumerate(open_list):
            if item.f < current_node.f:
                current_node = item
                current_index = index
```

Date : _____

```
open_list.pop(current_index)
closed_list.append(current_node)

if current_node == end_node:
    path = []
    current = current_node
    while current is not None:
        path.append(current.position)
        current = current.parent
    return path[::-1]

children = []
for new_position in [(0,-1),(0,1),(-1,0),(1,0),
                     (-1,-1),(-1,1),(1,-1),(1,1)]:
    node_position = (current_node.position[0] +
        new_position[0], current_node.position[
            + new_position[1])

    if node_position[0] > (len(maze)-1) or
    node_position[0] < 0 or node_position[1] >
    (len(maze[len(maze)-1])-1) or node_position
                                    [1] < 0:
        continue

    if maze[node_position[0]][node_position[1]]
                                    != 0:
        continue

    new_node = Node(current_node, node_position)
    children.append(new_node)

for child in children:
    for closed_child in closed_list:
        if child == closed_child:
            continue
```

Date :

```
        child.g = current_node.g + 1
        child.h = ((child.position[0] - end_node.position[0]) ** 2)
                  + ((child.position[1] - end_node.position[1]) ** 2)
        child.f = child.g + child.h

        for open_node in open_list:
            if child == open_node and child.g > open_node.g:
                continue

        open_list.append(child)

def main():
    maze = [[0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

    start = (0, 0)
    end = (7, 6)

    path = astar(maze, start, end)
    print(path)

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

Output:

[(0,0), (1,1), (2,2), (3,3), (4,3), (5,4), (6,5)]

Result:

Thus, the python program for A* algorithm is verified & executed successfully.

Date: _____

4) Water jug problem:

Aim:

To write a program to execute WaterJug problem.

program code:

```
def fill_4_gallon(x,y,x_max,y_max):
    return (x_max,y)
def fill_3_gallon(x,y,x_max,y_max):
    return (x,y_max)
def empty_4_gallon(x,y,x_max,y_max):
    return (0,y)
def empty_3_gallon(x,y,x_max,y_max):
    return (x,0)
def pour_4_to_3(x,y,x_max,y_max):
    transfer = min(x,y_max-y)
    return (x-transfer,y+transfer)
def pour_3_to_4(x,y,x_max,y_max):
    transfer = min(y,x_max-x)
    return (x+transfer,y-transfer)
def dfs_water_jug(x_max,y_max,goal_x,
                  visited=None,start=(0,0)):
    if visited is none:
        visited = set()
    stack = [start]
    while stack:
        state = stack.pop()
        x,y = state
        if state in visited:
            continue
        visited.add(state)
        print(f"visiting state:{state}")
        if x==goal_x:
            print(f"Goal reached:{state}")
            return state
```

Date :

```
next_states = [fill_4_gallon(x, y, x_max, y_max),
               fill_3_gallon(x, y, x_max, y_max),
               empty_4_gallon(x, y, x_max, y_max),
               empty_3_gallon(x, y, x_max, y_max),
               pour_2to3(x, y, x_max, y_max),
               pour_3to4(x, y, x_max, y_max)]

for new_states in next_states:
    if new_state not in visited:
        stack.append(new_state)

return none.

x_max = 4
y_max = 3
goal_x = 2

dfs_water_jug(x_max, y_max, goal_x)
```

Output.

```
Visiting state : (0,0)
Visiting state : (0,3)
Visiting state : (3,0)
Visiting state : (3,3)
Visiting state : (4,2)
Visiting state : (4,0)
Visiting state : (1,3)
Visiting state : (1,0)
Visiting state : (0,1)
Visiting State = (4,1)
Visiting state : (2,3)
Goal reached : (2,3)
(2,3)
```

Date : _____

Result:

Thus, the python program to solve water jug problem is executed and verified successfully.

Date :

5. 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 fn. DecisionTreeClassifier() from tree
- Assign values for x and y
- call the function predict for predicting on the basis of given random value for each given feature
- Display the output.

program code:

```
from sklearn.tree import DecisionTreeClassifier
X = [[170, 65, 40], [160, 55, 38], [180, 80, 42],
     [175, 75, 41], [158, 52, 34]]

Y = [1, 0, 1, 1, 0]

clf = clf.fit(X, Y)
prediction = clf.predict([[165, 60, 39]])
print("Predicted Gender:", "Male", if prediction[0]
                                == 1 else "female")
```

Output:

Predicted Gender: Female.

Result:

Thus the python program to implement a decision tree classification technique for gender classification is executed successfully

Date : _____

6. Implementation of k-Means clustering technique.

Aim:

To implement a k-Means clustering technique using python language.

Explanation:

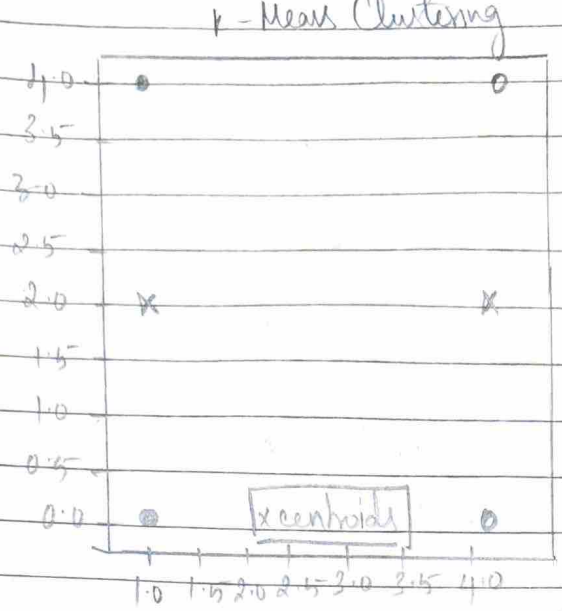
- Import kMeans from sklearn.cluster
- Assign x and y
- Call the function kMeans()
- Perform scatter operation and display the output.

Program code:

```
from sklearn.cluster import kMeans
import matplotlib.pyplot as plt
x = [[1,2],[1,4],[1,0],[4,2],[4,4],[4,0]]
kmeans = kMeans(n_clusters = 2)
kmeans.fit(x)
y = kmeans.labels_
centers = kmeans.cluster_centers_
for i, label in enumerate(y):
    plt.scatter(x[i][0], x[i][1], color='blue'
        if label==0 else 'green')
plt.scatter(centers[:,0], centers[:,1],
    color = 'red', marker='x', s=100,
    label = 'centroids')
plt.xlabel('Feature1')
plt.ylabel('Feature2')
plt.title('k-Means Clustering')
plt.legend()
plt.show()
```

Date :

Output :



Result :

Thus, the python program to implement k-means clustering technique is executed successfully.

Date : _____

4. Implementation of Artificial Neural Networks for An Application in Regression

Aim:

To implement artificial neural networks for an application in regression using python.

Program code:

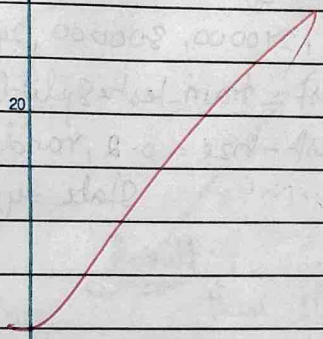
```
import numpy as np.
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split.
from sklearn.preparing import standardScaler

X = np.array([[1200,3,20],[1500,4,15],[1800,3,30],
    [2000, 5,10],[1600,4,25]])
Y = np.array([200000, 250000, 270000, 300000, 240000])
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                    test_size = 0.2, random_
                    State = 42)

Scaler = StandardScaler()
X_train = Scaler.fit_transform(X_train)
X_test = Scaler.transform(X_test)
model = Sequential()
model.add(Dense(32, input_dim = X.shape[1],
                    activation = 'relu'))
model.add(Dense(1))
Model.compile(optimizer = 'adam', loss = 'mean_
                    Squared_error')
Model.fit(X_train, Y_train, epochs = 10, batch_size
                    = 5, verbose = 1)
predictions = model.predict(X_test)
```

Date :

```
for i in range(min(5, len(predictions))):
    print(f"Predicted: {predictions[i][0]:.2f}, Actual: {y_test[i]}")
```

Output:

predicted : 0.06 , Actual = 250000



Result:

Thus, the python program to implement artificial neural networks for a application in regression is executed successfully.

Date : 2nd 4

8. Introduction to Prolog

Aim:

To learn PROLOG terminologies and write basic programs

Terminologies:

1. Atomic terms:-

Atomic terms are usually strings made up of lower-and uppercase letters, digits and the underscore, starting with a lowercase letter.

eg: dog
ab-c-321

2. Variables:

Variables are string of letters, digits and the underscore, starting with a capital letter or an underscore.

eg: Dog
apple-420

3. Compound terms:-

Compound terms are made up of a PROLOG atom and a number of arguments (PROLOG terms, ie:, atoms, number, variables, or other compound terms) enclosed in parenthesis and seperated by commas

eg.
is_bigger(elephant, X)
f(g(X,), 4)

4. Facts:-

A fact is a predicate followed by a dot.

Ex. bigger_animal(whale).
life_is_beautiful.

5. Rules:-

A rule consists of a head (a predicate) and a body (a sequence of predicates seperated by commas).

Ex: smaller(X,Y) :- is_bigger(Y,X)
aunt(Aunt, Child) :- sister(Aunt, Parent), parent(Parent, Child)

Date :

Source code:

KB1 :

```
woman(mia).
woman(jody).
woman(yolanda).
playsAirGuitar(jody).
party.

query 1 :- ?- woman(mia).
query 2 :- ?- playsAirGuitar(mia).
query 3 :- ?- party.
query 4 :- ?- concert.
```

Output :

```
?- woman(mia).
true.
?- playsAirGuitar(mia).
false
?- party.
true.
?- concert.
Error : Unknown procedure : concert/0 (DWIM could not correct goal)
```

KB2 :

```
happy(yolanda).
listens2music(mia).
listens2music(yolanda) :- happy(yolanda).
playsAirGuitar(mia) :- listens2music(mia).
playsAirGuitar(yolanda) :- listens2music(yolanda).
```

Output :

```
?- playsAirGuitar(mia).
true
?- playsAirGuitar(yolanda).
true
```

Date : _____

KB 3:

```
likes(dan, Sally)
likes(Sally, dan)
likes(john, britney)
married(X,Y) :- likes(X,Y), likes(Y,X)
friends(X,Y) :- likes(X,Y); likes(Y,X).
```

Output:

```
?- likes(dan, X).
X = Sally
?- married(dan, sally).
true.
?- married(john, britney).
false.
```

KB 4

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

Output:

```
?- food(pizza).
true
?- meal(X), lunch(Y).
X = Sandwich
?- dinner(Sandwich).
false
```

Date :

KBS :

```
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)).
```

Output :

```
? - owns (john, x).
x = car (chevy).
? - owns (john, x).
true.
? - owns (who, car (chevy)).
who = john.
? - own (jane, x), sedan (x).
false
? - owns (jane, x), truck (x).
x = car (chevy)
```

Result :

Thus, the basic programs on PROLOG terminologies are successfully run and executed.

Date :

9. Prolog family tree

Aim:

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

Knowledge Base:

```
/* facts */
male(peter)
male(john)
male(chris)
male(kelvin)
female(betty)
female(jeny)
female(lisa)
female(helen)
parentOf(chris, peter)
parentOf(chris, betty)
parentOf(helen, peter)
parentOf(helen, betty)
parentOf(kelvin, chris)
parentOf(kelvin, lisa)
parentOf(jeny, john)
parentOf(jeny, kelvin)
/* Rules */
```

a) father(X,Y) :- male(Y), parentOf(X,Y)

X	Y
peter	Chris
peter	Helen
john	jeny
Chris	kevin.

Date :

KBS:

```
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)).
```

Output:

```
? - owns(john, X).
X = car(chevy).
? - owns(john, _).
true.
? - owns(who, car(chevy)).
who = john.
? - own(jane, X), sedan(X).
false
? - owns(jane, X), truck(X).
X = car(chevy)
```

Result:

Thus, the basic programs on PROLOG terminologies are successfully run and executed.

Date :

9. Prolog family tree

Aim:

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

Knowledge Base:

```
/* facts */
male(peter)
male(john)
male(chris)
male(kelvin)
female(betty)
female(jeny)
female(lisa)
female(helen)
parentOf(chris, peter)
parentOf(chris, betty)
parentOf(helen, peter)
parentOf(helen, betty)
parentOf(kelvin, chris)
parentOf(kelvin, lisa)
parentOf(jeny, john)
parentOf(jeny, kelvin)
/* Rules */
```

a) father(X,Y) :- male(Y), parentOf(X,Y)

X	Y
peter	chris
peter	Helen
john	jeny
Chris	kevin.

Date :

b) female(Y), parentOf(X,Y)

Y	Y
betty	Chris
betty	Helen
lisa	Kelvin
Helen	Jeny.

c) Male(Y), parentOf(X,Z), parentOf(Z,Y)

Y	X	Z
peter	kelvin	Chris
peter	jeny	Helen

d) female(Y), parentOf(X,Z), parentOf(Z,Y)

Y	X	Z
betty	kelvin	Chris
betty	Jeny	Helen

e) Male(Y), father(X,Z), father(Y,W), Z==W

procedure father('A','B') doesn't exist

f) female(Y), father(X,Z), father(Y,W), Z==W

procedure father('A','B') does not exist.

Result:

Thus, the prolog family tree is executed successfully.

Date : _____

10. MINIMAX Algorithm.

Aim:

↳ A simple example can be used to explain how the minimax algorithm works we have included in this example.

↳ There are 2 players in this scenario one named Maximize and other named Minimize.

↳ Maximizer will strive for higher possible score's and minimizers will strive for lowest score

Program:

```
import Math
def minimax (depth, node_index, is_maximize, scores, height)
    if depth == height:
        return scores[node_index]
    if is_maximizes:
        return max(minimax(depth+1, nodeindex*2, False, scores, height), minimax (depth+1, node index*2+1, False, score, height))

else:
    return min (minimax (depth+1, node_index*2, False, score, height)),
        minimax (depth+1, node_index*2+1, False, score, height)).

def calculate_tree_height(num_leaves):
    return math.ceil(math.log2(num_leaves))
scores = list(map(int(input("Enter the score seperated by spaces: ").split())))
tree_height = calculate_tree_height(0.0, True, scores, tree_height),
optimal_score = minimax(0.0, True, scores, tree_height).
```

Date :

print("The optimal score is: ", optimal_score);

Output:

Enter the scores separated by spaces
4 2 6 -3 -5 0 7
The optimal score is 4.

Result:

Thus, the minimax algorithm successfully determinate the optimal score for player.