NAME:- KRUNAL RANK

ADM. NO:- U18C0081

B TECH 3RD YEAR

U18CO081

9/9/20

AI/ML Tutorial 4

Ans 1: There are many examples where Machine Learning is implemented in our day to day lives. Some of the examples are as follows:-

- Virtual Personal Assistants:-

Siri, Alexa, Google Now are some of the popular examples of Virtual Personal Assistants. As the name suggests, they assist in finding information when asked over voice.
They collect and refine the information on the basis of previous interactions with them.
Later this set of ~~info~~ data is utilized to render results that are tailored to the user's experience.
They can be improved further in a way that they can automatically detect health issues for the user via their voice, sleeping patterns, frequency of usage through which they can then inform the user of their degrading health and necessary steps to be taken.

- Traffic Predictions:-

Google Maps is a well known service that provides GPS as well as guidance services for people in transit. They collect data from their users, particularly users who are moving, mainly their positions and map this data in accordance with other previously collected data to generate estimated traffic reports wherein the user is guided to the path with least resistance.

However, recently a user created a large virtual traffic on google maps by collecting a massive number of

cellphones in a trolley cart and switched their location to ON condition. This made Google Maps assume that the road where the cart was going through had a load of traffic. Such types of incidents need to be prevented. It can be done using Precise Location feature and using density of devices rather than number for a particular area.

Ans 2)

- Social Media Services

In today's world, Social Media has become the epitome of communication and managing one's personality in Social Media profiles has also become important for such people.

Ans 3)

Machine Learning allows users to connect to different strangers with similar interests and locations, based on the user's interaction with the application, mainly their likes and dislikes, their favourite type of content, their uptime, etc..

Sometimes, it is observed that the model suggests unknown people quite frequently which may make the user paranoid. This can be prevented by increasing the bias factor for such predictions to make the user feel more comfortable while using the application.

Ans 2: The parameters through which a Search algorithm can be evaluated are as follows:

- Time Complexity: It is a theoretical measure of how long will it take to reach the goal state in terms of several factors such as branching factor, cost of solutions, etc...
- Space Complexity: It is the amount of memory required in proportional to the factors associated with the algorithm, while running the algorithm.
- Completeness: It is a property which determines whether an algorithm is able to reach the goal state.
- Optimality: It is a property which determines the number of steps in reaching the solution.

Ans 3: The Bredth First Search can be evaluated as follows:
Let s = depth of shallowest solution, n' = no. of nodes at level i

- Time Complexity:
Equivalent to the number nodes traversed in BFS until the shallowest solution.

$$T(n) = 1 + n + n^2 + \cdots + n^s = O(n^s)$$

- Space Complexity:
Equivalent to how large can a fringe get $S(n) = O(n^s)$.

- Completeness: BFS is complete.
- Optimality: BFS is optimal as long as the cost of all edges are equal.

The Iterative Deeping Depth First Search can be evalu... as follows:

- Time Complexity:
Suppose we have a tree with branching factor 'b' and its depth 'd', then, there are b^d nodes.
In IDDFS, the nodes at the bottom are expanded once, their parents twice, their grand parents thrice and so on.
Thus,
$$db + (d-1)b^2 + (d-2)b^3 + \dots + b^d = O(b^d)$$

- Space Complexity:
Suppose in IDDFS only one path is stored to depth d.
Hence, $O(d)$ is the space requirement.

- Completeness:
The search algorithm is completed complete.

- Optimality:
The algorithm is highly optimal due to less memory requirement.

Ans 4; Uniform Cost Search is a variant of Dijkstra's algorithm. Here, instead of inserting all vertices into a priority ~~queue~~ queue, we insert only source, then one by one insert the other nodes if and when needed.
In every step we check if the item is already in priority queue. If yes, we perform decrease key, ~~else~~ else we insert it.
It is important for big, infinite graphs where memory becomes an issue.

The time complexity for the uniform cost search is $O(m^{1+e})$
Here m = maximum number of ~~nodes~~ neighbours a node has
l = length of shortest path to goal state
e = least cost of an edge.