EasyChair Preprint

№ 11396

Metaverse in InterPlanet Internet: the Artificial Intelligence Based Pyramid Principle Applied to Space Robots in Automated Learning and Execution

Poondru Prithvinath Reddy

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

November 27, 2023

Metaverse in InterPlanet Internet: The Artificial Intelligence based Pyramid Principle Applied to Space Robots in Automated Learning and Execution

Poondru Prithvinath Reddy

ABSTRACT

The interplanet internet is a conceived computer network in space, consisting of a set of network nodes that can communicate with each other. These nodes are the planet's orbiters (satellites) and landers (e.g. robots, autonomous machines, etc.) and the earth ground stations, and the data can be routed through Earth's internal internet. As resource depletion on Earth becomes real, the idea of extracting valuable elements from asteroids or using space-based resources to build space habitats becomes more attractive, one of the key technologies for harvesting resources is robotic space mining (minerals, metals, etc.,) or robotic building of space settlement. The metaverse is essentially a simulated digital environment mimicking the real world. The metaverse would be something very similar to real world planetary activities where users (space colonies or internet users on Earth) interact with overlaying objects represented by robots, drones, etc. for real-world planetary activities like space mining, building space settlements, etc. in a completely virtual manner. Here we show how pyramid principle with hierarchical structure is applied to AI algorithms and to give a structural knowledge to space robots in operation. In a pyramid system, the lowest level of the hierarchy is data acquisition i.e. data generated from robotic sensors/ terminal devices that consistently collect data representing transaction processing systems for robots. On top, a rich variety of dense data sources to create a value in a AI pyramid of needs with AI observability resides at the top with trained AI models, fulfilling the role of a watchful deployer. Here , the AI models are fragile pieces of software to classify any new data generated from robots/endpoints and AI models are adaptive makes them possible to grasp what is happening on the ground fundamentally. Therefore, this will be an important loop in the data intelligence lifecycle, the one of application understanding. Here we show an implementation of two ways of pyramidal classifying transactions from robots into categories are:

- Linear Classification (SVM)

- Nearest Neighbhour (by analysing K nearest observations)

Since we have learning models in the form of observability residing at the top of pyramid of robotic shapes along with transactions from robots we show an implementation of combining shape patterns and transactions classification by means of a small model in obscene of real-world model of Metaverse for autonomous space operations. In this way, the desired response was measured, and new operational conditions and robotic classification predictions were synergistically combined for diverse outcomes. The results of the study show that the real individual behaviour on a distant planet of undertaking of space related activities with pyramid principle using deep learning models could be of reality even in interplanet environment provided the interplanet internet is available as pathway communication.

INTRODUCTION

Inter-planetary exploration, be it Lunar habitation, asteroid mining, Mars colonization or planetary science/mapping missions of the solar system, will increase demands for inter-planetary communications. The movement of people and material throughout the solar system will create the economic necessity for an information highway to move data throughout the solar system in support of inter-planetary exploration and exploitation. The communication capabilities of this interplanet information highway need to be designed to offer; 1) continuous data, 2) reliable communications, 3) high bandwidth and 4) accommodate data, voice and video.

The interplanetary Internet is a conceived computer network in space, consisting of a set of network nodes that can communicate with each other. These nodes are the planet's orbiters (satellites) and landers (e.g., robots), and the earth ground stations. For example, the orbiters collect the scientific data from the Landers on Mars through near-Mars communication links, transmit the data to Earth through direct links from the Mars orbiters to the Earth ground stations, and finally the data can be routed through Earth's internal internet. Interplanetary communication is greatly delayed by interplanetary distances, so a new set of protocols and technology that are tolerant to large delays and errors are required. The interplanetary Internet is a store and forward network of internets that is often disconnected, has a wireless backbone fraught with error-prone links and delays ranging from tens of minutes to even hours, even

when there is a connection. In the core implementation of Interplanetary Internet, satellites orbiting a planet communicate to other planet's satellites. Simultaneously, these planets revolve around the Sun with long distances, and thus many challenges face the communications. The reasons and the resultant challenges are: The interplanetary communication is greatly delayed due to the interplanet distances and the motion of the planets. The interplanetary communication also suspends due to the solar conjunction, when the sun's radiation hinders the direct communication between the planets.

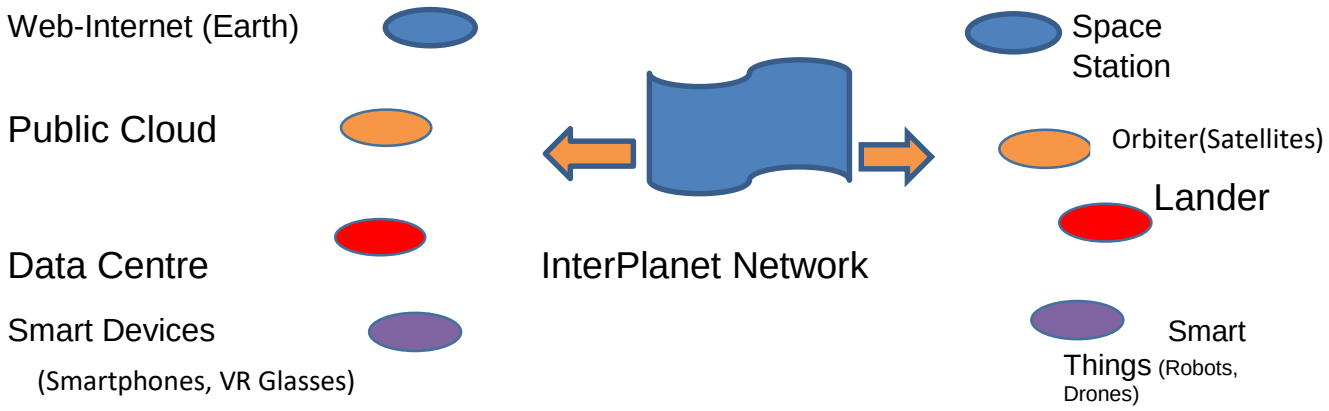
NETWORK ARCHITECTURE

A **Computer Network Architecture** is a design in which all computers in a computer network are organized. An architecture defines how the computers should get connected to get the maximum advantages of a computer network such as better response time, security, scalability, etc.

Network architecture refers to the way network devices and services are structured to serve the connectivity needs of client devices.

- Network devices typically include switches and routers.
- Types of services include DHCP and DNS.
- Client devices comprise end-user devices, servers, and smart things.

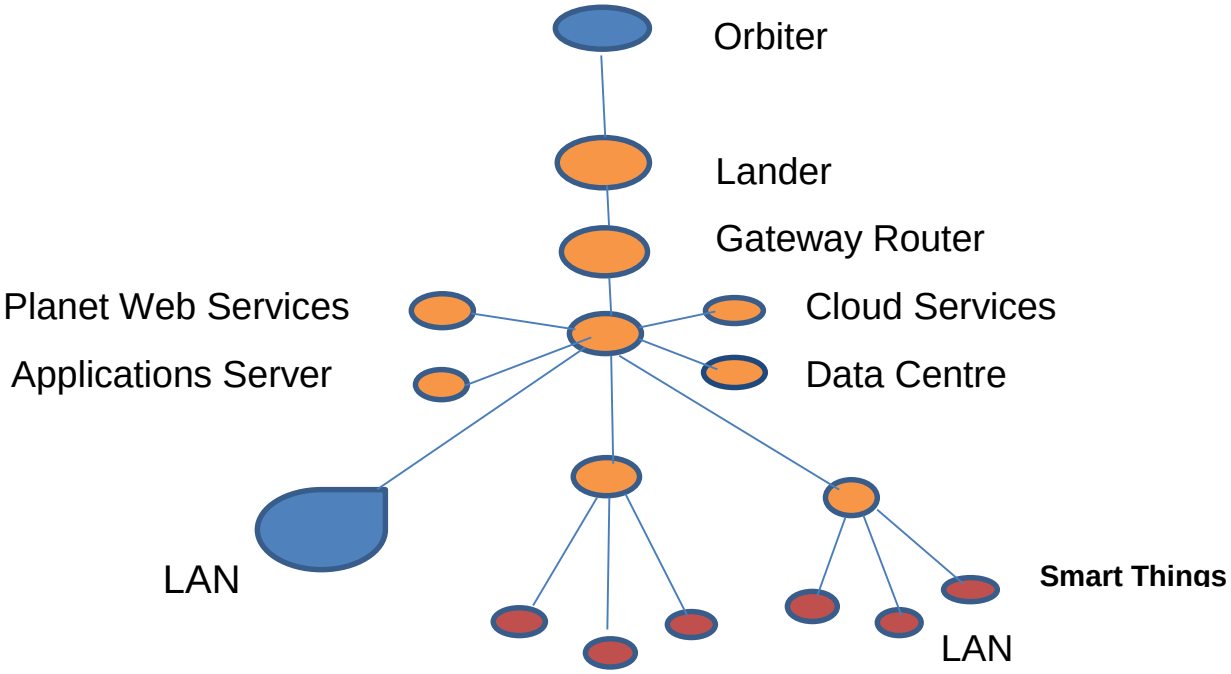
The network architecture for the planet Mars or the Moon is as shown in below figure:-



Computer networks are built to serve the needs of certain functionality and also their clients. Described below are three types of planetary networks:

- Access networks, for campuses and local areas, are built to bring machines and things onboard, such as connecting robots, drones, etc. within a location.
- Networks for data center connect servers that host data and applications and make them available to smart devices.
- Wide-area networks (WANs) connect robots and others to applications, sometimes over long distances, such as connecting robots to cloud applications related to space mining operations.

We give below the architecture of network on the planet Mars or the Earth's Moon is as shown in below figure:-



Local Area Network

An Internet is a "network of networks" in which routers move data among a multiplicity of networks with multiple admin. domains.

The main aim of networks is to connect remote endpoints with end-to-end principle and network should provide only those services that cannot be provided effectively by endpoints.

Since the networks are predominantly wireless, the fundamental impact of distance due to speed-of-light delays and impact on interactive applications – for both data and control is to be considered. Also power consumption of wireless links as a function of distance is to be examined.

The interplanetary internet is a conceived networks of nodes and these nodes are space station, planet's orbiters (satellites), planet's landers, robots (drones, autonomous machines, etc.), earth ground stations and earth's internal internet.

METHODOLOGY

1. Augmented Reality

The word 'augmented' means to add. Augmented reality uses different tools to make the real and existing environment better and provides an improved version of reality.

As Augmented Reality (AR) technologies improve, we are starting to see use cases and these include product visualization. There are AR apps that allow a customer to place virtual furniture in their house before buying and it is also a powerful tool for marketing as it allows users to try products before buying.

At its core, AR is driven by advanced computer vision algorithms that compares visual features between camera frames in order to map and track the environment. But we can do more. By layering machine learning systems on top of the core AR tech, the range of possible use cases can be expanded greatly.

Augmented Reality (AR) can be defined as a system that incorporates three basic features: a combination of real and virtual worlds, real-time interaction, and accurate 3D registration of virtual and real objects

2. Camera Representation

A camera is a device that converts the 3D world into a 2D image. A camera plays a very important role in capturing three-dimensional images and storing them in two-dimensional images. And the following equation can represent the camera.

$$x=PX$$

Here x denotes 2-D image point, P denotes camera matrix and X denotes 3-D world point.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

The above is vector representation of x=PX [1].

The camera representation method is frequently used in image processing and is intended to identify the geometric characteristics of the image creation process. This is a vital step to perform in many computer vision applications, especially when metric information on the scene is needed.

3. *Metaverse Algorithm*

1. **Physical Reality Modeling -** required information
   - The goal of the agent/robot
   - What the robot sees, Materials & location
   - Real Simulation for Task Execution
2. **Task Execution (Simulation)**
   - Generating actual materials (how materials arrive at the site)
   - Robots arrive in the environment (speed and goal)
   - Task Execution (Simulation Steps), is updated as the work process progresses in line with the simulation
   - Task execution performance, as we have fully functional simulator and to make a realistic system, we would like to see how well it performs and mirrors real world execution(Artificial Intelligence)
   - Implementation of Graphical Version of the Task Execution

Models for Metaverse & Algorithm

Minimum amount of required information

- The current state of the robot/agent and its environment
- The goal of the agent/robot
- What the agent sees, materials & it's location

Agents – Attributes

We opt for the agents and they have the attributes: the sight and the

goal. While the goal is chosen randomly when an agent arrives on the location, the sight is always fixed to the some value. We define the autonomous robots as entities whose primary concern is to avoid failure; they should consequently not exhibit any preference for a certain speed as long as they are working safely. Furthermore, we add an attribute to these learning agents; this is their probability of choosing a random action at each time step.

Agents as workmen

Given that we define learning agents the same way as the type of workers, we can seamlessly add them at the location. The only difference is how they will choose an action: by using their learning model, a neural network. We can therefore adapt the site's time step's algorithm to take the learning agent into account for the observation step. To decide what action it should take, the learning agent uses a neural network to approximate the Q-function. Thus, at every time step t, the agent c observes its state $s_{c,t}$; this state is then processed in some way so that it can be passed to a neural network.

Neural Network Models

Presently different neural network models are available that we will use to train our autonomous robots. These models define what information the learning agents use and how they are encoded as inputs to the neural networks. Before we start with our model, we need to define the building structure; how these neural networks are used by the learning agents. We use a feedforward neural network whose outputs correspond to the possible actions. Our models define different ways of using information about the agent's current state.

Required Information

We start by defining the minimum amount of information that an autonomous robot should have. Consequently, the model that we design will possess these pieces of information. They are:

- The goal of the agent/robot
- What the robot sees, Materials & location
- The current location that the agent is in
- The current speed of the agent
- Real Simulation for Task Execution

Task Execution (Simulation)

- Generating actual materials (how materials arrive at the site)
- Robots arrive in the environment (speed and goal)

- Task Execution (Simulation Steps), is updated as the work process progresses in line with the simulation
- Task execution performance and to make a realistic system, we would like to see how well it performs and mirrors real world execution (Artificial Intelligence with learning algorithm)

Robotic Design and Placement

Robots are **collections of task executors** and have no brain system of their own. But they can be programmed to work autonomously and collaborate with other robots, or eventually to do other things tackling everything from space mining to deep space exploration.

Robots can be programmed as specific executor of an assigned task for a number of situations and also using artificial intelligence to figure out the best shape for the Robots to perform in group on a more consistent basis to have better control over performance of assigned work.

Using a computational model that simulates the nature of work and everything of the Robot Capability, the process yields the robotic shape best suited to ensure the shape of the actual Robots into more efficient form suitable to a particular situation/task and accordingly enables robots to gather together in their environment forming them into groups with the same capability.

The revolution of modern computing has been largely enabled by remarkable advances in robotics however, majority of today's robots designed are not suitable for high-end space exploration, resulting in the need to speculate about how to optimize the next generation of robots for the machine learning (ML) models. Further, dramatically shortening the robot design/shape requirement would allow hardware to adapt to the rapidly advancing field of ML. The ML itself could provide the means to the robot design/shape requirement , creating a more integrated relationship between space exploration and ML.

Vast arrays of robots with different make are required for complex space applications thus, improving the selection of design patterns of these autonomous robots would be critical in improving the performance and efficiency of remote space applications and use of AI to achieve high-performance execution and robotic performance relevant to the work.

In order for the AI to design with an RL agent and the technique proved that AI can not only learn to design robotic patterns from scratch but that

those structural patterns are accurate and faster than designed using any of the latest validation tools. Here an AI agent could design neural graphs and such a graph is converted into a class of robots with connection (relevant shapes) using a link generator.

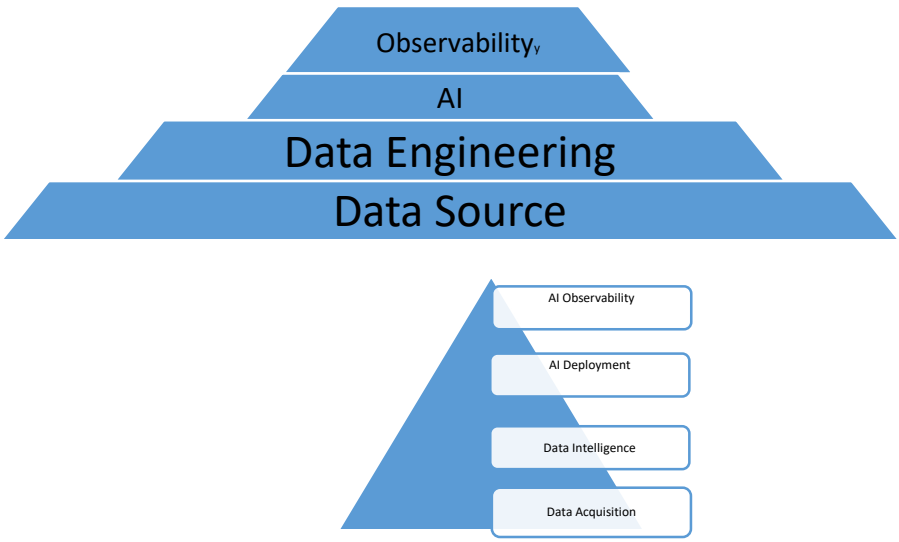
Pyramid of Knowledge in AI

The AI system consists of **training system and inference system**. The training system generates a trained model based on training dataset. The trained model is used at inference system to classify the new data from endpoints.

A pyramid is **a three-dimensional shape**. A pyramid has a polygonal base and flat triangular faces, which join at a common point called the apex. A pyramid is formed by connecting the bases to an apex. Each edge of the base is connected to the apex, and forms the triangular face, called the lateral face.

The learning pyramid **uses a triangle-shaped diagram to represent this information visually**. In most depictions, the diagram stacks each learning method in sequential order, from the least effective methods with low percentages at the top point of the triangle to the most dynamic approaches at the lower levels.

The DIKW Pyramid represents the relationships between data, information, knowledge and wisdom. Each building block is a step towards a higher level - first comes data, then is information, next is knowledge and finally comes wisdom. The first level represents transaction processing systems for workers.



As we all know, AI algorithms require examples to learn from. We would not be able to do much AI training without data gathered from the past. That brings us to the lowest level of the hierarchy, which is data acquisition. Be it customer data, open/external data, data generated from sensors or IoT devices, consistently collecting data is a necessity. On top, a rich variety of dense data sources to play around with.

In our new AI Pyramid, AI observability resides at the top, fulfilling the role of a watchful protector. There is a possibility of AI algorithms is going haywire due to the lack of oversight as AI models are fragile pieces of software and It is not uncommon to see them lose their value in the initial phase of deployment. While, if not monitored correctly, failing to be completely silent. A good observability stack detects and alerts data drift, manages machine learning health, and tracks business impact. Keeping these in check will help us prevent, expose, and properly handle algorithmic flaws in production.

However, it is not just about surveillance of performance but understanding how, why, and when data, business intelligence, and AI models are changing makes it possible to grasp what is happening on the ground fundamentally. This way, we close an important loop in the data science lifecycle: the one of business understanding. In other words, sophisticated AI monitoring allows us to extract business insights. However, AI deployment is very deserving of its place in the pyramid as there is literally no business value creation without operationalizing machine learning models in production.

AI Pyramid of Knowledge & Intelligence

In a pyramid of knowledge, each building block is a step towards a higher level of intelligence and experience – first comes data acquisition, then data engineering, then is data intelligence/ AI, and AI observability. Here, the primary role of AI observability is nothing but passing intelligence with experience with an ability of strong AI, and this ability is executed either through low orbit satellites or drones or through cloud services in a networked environment. Since pyramid is a triangle structure that ensures intelligence, knowledge and experience is passed at the top of the triangle to the most dynamic end points where robots are being in operation for different space applications. This essentially requires classification of robots by the AI based on their role in performing space operations and specific knowledge/experience will be passed down the line by the AI deployed at the top.

Pyramid Principle applied to Classification Algorithms

The Minto's Pyramid Principle is aimed to better organize and remember ideas with a pyramidal, and hierarchical structure. There are the different classification algorithms such as Random Forest, KNN, Naive Bayes, Logistic Regression, and so on. But the Minto's Pyramid Principle is applied in three basic ways however, we have chosen to classify data into following two categories only:

- **Linear classifier -** (Logistic Regression, **SVM** and Linear Discriminant Analysis)
- **Nearest Neighbors** by analyzing k nearest observations

ARCHITECTURE

Linear Classifier – SVM

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Support Vector Machine (SVM) is a powerful machine learning algorithm and SVM offers very high accuracy compared to other classifiers such as logistic regression, and decision trees. It is known for its kernel trick to handle linear or nonlinear input spaces. It is used in a variety of applications such as face detection, intrusion detection, classification of emails, classification of genes, and handwriting recognition.

SVM is an exciting algorithm and the concepts are relatively simple. The classifier separates data points using a hyperplane with the largest amount of margin. That's why an SVM classifier is also known as a discriminative classifier. SVM finds an optimal hyperplane which helps in classifying new data points.

SVM is **a powerful supervised algorithm that works best on smaller datasets but on complex ones**. Support Vector Machine work best in classification problems and SVM algorithms are very effective as the maximum separating hyperplane between the different classes available in the target feature.

Generally, Support Vector Machines is considered to be a classification approach, and It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes.

How does SVM algorithm work?

One reasonable choice as the best hyperplane is the one that represents the largest separation or margin between the two classes. So we choose the hyperplane whose distance from it to the nearest data point on each side is maximized.

The main objective is to segregate the given dataset in the best possible way. The distance between the either nearest points is known as the margin. The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum marginal hyperplane in the following steps:

1. Generate hyperplanes which segregates the classes in the best way.
2. Select the right hyperplane with the maximum segregation from the either nearest data points

Types of Support Vector Machine

Based on the nature of the decision boundary, Support Vector Machines (SVM) can be divided into two main parts:

- **Linear SVM:** Linear SVMs use a linear decision boundary to separate the data points of different classes.
- **Non-Linear SVM:** Non-Linear SVM can be used to classify data when it cannot be separated into two classes by a straight line.

SVM Kernels

The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form. SVM uses a technique called the kernel trick. Kernel trick helps us to build a more accurate classifier. Here, the kernel takes a low-dimensional input space and transforms it into a higher dimensional space. In other words, it converts nonseparable problem to separable problems by adding more dimension to it. It is most useful in non-linear separation problem.

Linear Kernel - A linear kernel can be used as normal dot product any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.

$K(x, xi) = sum(x * xi)$

Polynomial Kernel - A polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel can distinguish curved or nonlinear input space.

$K(x,xi) = 1 + sum(x * xi)^d$

Where d is the degree of the polynomial. d=1 is similar to the linear transformation. The degree needs to be manually specified in the learning algorithm.

SVM Implementation - Classifier Building

In the model building part, we have used the simulated population dataset, which is a multi-class classification problem. This dataset is computed from a digitized image of a human population study in a particular region. They describe characteristics of the population present in the image.

The dataset comprises 10 features (name, age, sex, education, profession, faith, income, physical status, health, marital status) and a target (type of population).

This data has two types of population classes: skilled (useful) and unskilled (not useful). Here, we can build a model to classify the type of population. The dataset is a simulated one and not part of a learning library available in the Machine Learning Library.

Linear Classifier: k-Nearest Neighbor

The K-Nearest Neighbor (kNN) algorithm is a popular machine learning technique used for classification and regression tasks. However, it is more widely used in classification problems in the industry. It relies on the idea that similar data points tend to have similar labels or values.

During the training phase, the kNN algorithm stores the entire training dataset as a reference. When making predictions, it calculates the distance between the input data point and all the training examples, using a chosen distance metric such as Euclidean distance.

Next, the algorithm identifies the k nearest neighbours to the input data point based on their distances. In the case of classification, the algorithm assigns the most common class label among the k neighbours as the predicted label for the input data point.

Algorithm of kNN

We can implement a kNN model by following the below steps:

1. Loading the data of sample dataset
2. Initialise the value of k
3. For getting the predicted class, iterate from 1 to total number of training data points
    - Calculate the distance between test data and each row of training dataset. Here we will use Euclidean distance as our distance metric since it's the most popular method.
    - Sort the calculated distances in ascending order based on distance values
    - Get top k rows from the sorted array
    - Get the most frequent class of these rows
    - Return the predicted class

The Dataset

To implement the kNN algorithm, we work on a simulated case study one may find while working as a travelling salesman. Let's assume a travelling salesman at a bearings manufacturer, and he has been tasked with selling bearing parts at different locations/sites as a sales promotion transactions. The only features we have considered are:

- Dist_from_manufacturing site: The distance between the user's home location where the transaction was made and the manufacturing site
- Cost_of_travel: the transportation cost between the location of the item purchased to the manufactured location of that item.
- Mode_of_travel: Land (bus, truck, train, taxi or other mode), air, sea route or combination of them.

The simulated data has 20 observations which are individual travel scenarios. Let's start by visualizing our data using graph as we can plot our features in a mathematical diagram.

As we can see, there is a clear difference between the sales transactions, with travel mode of air & train cost of sales transactions being of much higher value, compared to the salesman's travel plans with alternative mode of transport or combinations of it

RESULTS

SVM

To understand model performance, we divided the dataset into a training set and a test set. Next we built support vector machine model by creating support vector classifier object as the linear kernel in SVC. Then, we fit the model on training set and performed prediction on the test set and estimated how accurately the classifier or model can predict the skill level of population in a region..

We got a classification rate of 70%, considered as good accuracy by comparing actual test set values and predicted values.

k – Nearest Neighbour

When training any machine learning model, it is important to split the data into training and test data. The training data is used to fit the model and as it tries to find a pattern in the training data that can be used to make predictions on new, unseen data. The test data is used to evaluate the performance of the model.

Fitting and Evaluating the Model

First, we used a fixed value of 4 for k, but we'll need to optimize this later on. We first created an instance of the kNN model, and then fit this to our training data. We passed the features and the target variable, so the model can learn.

The model is now trained and we can make predictions on the test dataset, which we can use to score the model. The simplest way to evaluate this model is by using accuracy. We check the predictions against the actual values in the test set and count up how many the model got right.

Unfortunately, there is no magic way to find the best value for k. We have to loop through many different values, as we train and assess our model performance using the best k value (tuning hyperparameter) and then evaluate for accuracy/ precision,

For this, we selected a range of values for k and create an empty list to store our results. We have observed from the chart that k = 10, 11, and 13 all have an accuracy score of just under 73%. As these values are tied for the best score, it is found to have better accuracy score when we use a smaller value for k. This is because when using higher values of k, the model will use more data points that are further away from the initial.

We can see that both the models predicted the class of alike in kind and the identical nearest neighbors . Hence we can conclude that our models run as expected.

CONCLUSION

The interplanetary computer network in space is a set of computer nodes that can communicate with each other. We proposed a network architecture with planet's orbiters, landers (robots, etc.), as well as the earth ground stations and linked through Earth's internal internet, and consisted of complex information routing through relay satellites. As we know, the metaverse will be very different from the internet of today due to massive parallelism, three-dimensional (3D) virtual space and multiple real-world spaces like space mining, building space habitats, etc. We presented a pyramid structure of robotic deployment equipped with AI-model driven learning that can effectively explore and execute complex applications in space environment and also designed a classifier of terminal devices in two ways to address transaction processing at the end. As a part of sharing intelligence with experience, we implemented classifiers such as SVM and Nearest k neighbour in a pyramidal structure to classify robots and for sharing knowledge/intelligence for successful execution. In this way, an AI - model assisted pyramidal system with linear classifiers for sharing intelligence that is part of Metaverse is feasible for automated execution of diverse space related outcomes depending on the applications and in that respect an implementation of SVM and Nearest k neighbour classifiers based on small model is presented. Although the platform model with AI learning model with pyramidal structure given us a method of optimizing space applications however, this need to be tested using natural allocation for real space applications.

REFERENCE

1. **Poondru Prithvinath Reddy**: **"Metaverse in InterPlanet Internet: Modeling, Validation, and Experimental Implementation", Google Scholar**