(1)

The subfield of CS that gives computers the ability to learn without explicitly programmed. Machine learns patterns from data and replicate the same in future. Example: Predictive typing. It predicts patterns from the data and replicates the same in future.

Machine Learning (ML) differs from traditional coding in **how solutions are created**. In traditional coding a programmer writes explicit rules and logic such as given an input the program follows these rules to produce an output. For example: to check if a number is even you write `if n % 2 == 0`. The computer doesn't "think". It just executes instructions.

On the other hand, ML doesn't rely on fixed rules. Instead you provide the computer with data and the model **learns patterns** from this data to make predictions on new, unseen inputs. This approach is especially useful when rules are complex or unknown such as detecting spam emails or recognizing images. In short, traditional coding is **rule-based**, while ML is **data-driven and adaptive**.

For example: You go to a browser and search something and then you'll see that based on that topic you searched there's millions of suggestions on that topic. This is machine learning. The machine learns the pattern of the data you gave and it gave you the related topics on that. The more you feed it the more accurately it can give you the answer with more accuracy. On the other hand, coding is like the core concept behind ML. Because without coding you can't make the machine do the task.

In the context of Machine Learning a **model** is essentially a mathematical representation or algorithm that **learns patterns from data** to make predictions or decisions. It's like a "digital brain" that generalizes from examples rather than following explicit rules.

1) During **training**, the model analyzes input data along with the correct outputs and adjusts its internal parameters to capture the relationships between them.

2) Once trained the model can take **new, unseen inputs** and produce predictions or classifications based on what it has learned.

For example: In an image recognition task the model might learn patterns of pixels that correspond to "cat" or "dog." In short a model is the **learned system that turns input data into meaningful output** in ML.

In order to create a model you can follow a pipeline:

1) When you have a problem statement you have to read the dataset.
2) Then you will do exploratory data analysis.
3) Preparation of data by data cleaning,feature engineering.
4) Divide the data into x and y variable.
5) Then train test split
6) Scale the data
7) Model train
8) And finally model evaluation

And this is how you can make a ML model.

(2)

(a)

There are 3 types of ML processes:

1. **Supervised Learning:**
   The model is trained using **labeled data**, meaning each input has a corresponding correct output. The goal is for the model to **learn the mapping** from input → output so it can predict correctly for new data. **Example:** Predicting whether an email is spam or not.

2. **Unsupervised Learning:**
   The model is trained using **unlabeled data**, so it has to **find patterns, similarities, or groups** on its own. **Example:** Grouping customers based on buying behavior (clustering) or reducing data dimensions.

3. **Reinforcement Learning (RL):**
   The model (agent) **learns by interacting** with an environment. It receives **rewards or penalties** based on actions and improves over time. **Example:** A robot learning to walk, a game AI learning to win.

(b)

For a rover that navigates and avoids obstacles we can use,

1. **Supervised Learning:**

   Could be used for **object recognition**: the rover learns to identify bottles, arrows, or other obstacles from labeled images. Example: Image → obstacle, Image of a left arrow→ go left.

2. **Reinforcement Learning:**
   Could be used for **navigation and decision-making**. The rover can learn optimal paths by exploring an environment and receiving rewards for reaching goals and penalties for collisions.

3. **Unsupervised Learning:**
   It is actually less directly applicable but could help in mapping unknown terrain or grouping similar obstacles without pre-labeled data.

(3)

(a)

Model Overfitting:

You train your rover on several datas to avoid obstacles or movements or read its movements. So you basically trained a ML model in your rover to make the tasks happen. Basically for autonomous tasks this ML model will be very useful for the rover.

So you trained your data and according to that a model is trained. As a result you see the accuracy is 95% (very good). But while analyzing the “Test Data” you see that by testing the

model on test datas you find the accuracy=65% (Low). Test data means unseen datas that are different predictions to the datas that may occur in future.

So this situation is called “OVERFITTING” where model performs well on train data but worst on test data.

Model Underfitting:

In this case, you trained your data on several datas for the rover and found your model accuracy=50% . But while analyzing the test (unseen datas) you found that the accuracy is 40%. That means the model didn’t learn anything from the data.

This situation is called “UNDERFITTING”.

(b)

Data Preprocessing:

Data preprocessing is the cleaning and preparation step before training a machine learning model. Real-world data often contains noise, missing values, or inconsistent formats, which can confuse the model.
Common preprocessing steps include removing duplicates or missing data, normalizing or scaling numerical values, encoding categorical data like converting text labels into numbers, resizing and standardizing images for computer vision tasks.

So data processing ensures the data is clean, consistent, and ready for the model to learn efficiently.

Data Augmentation:

Data augmentation is a technique used to **artificially increase the size and diversity** of a dataset by creating modified versions of existing data. This helps the model generalize better

and avoid overfitting such as rotating an image, cropping an image, zooming an image,flipping an image etc. It helps the model recognize objects in different orientations, lighting, and backgrounds.

Data Splitting:

Data splitting means dividing the dataset into separate parts so that the model can learn, validate, and be tested fairly.

1) Training set: Used to train the model (usually 70–80% of data).

2) Validation set: Used to tune model parameters and check performance during training (10–15%).

3) Test set: Used only after training to evaluate final performance (10–15%).

This ensures that the model learns patterns properly and is tested on test(unseen) data to check real world performance.

(4)

A good image dataset is one that helps a machine learning model learn effectively and generalize well to new, unseen data. In the context of image based tasks here are the key criteria and characteristics that make a dataset “good”:

1. High-Quality Images:

- Images should be **clear, sharp, and properly lit**.

- Avoid blurry, noisy, or distorted images.

- Consistent **resolution and aspect ratio** help the model focus on features instead of size differences.

2. Sufficient Quantity:

- The dataset should have enough examples for each class (e.g., bottles, arrows, empty road).
- A small dataset can cause overfitting.
- More data = more patterns learned = better accuracy.

3. Class Balance:

- Each class or category should have a similar number of images.
- If one class dominates the model becomes biased toward it.
- Example: If 80% of images are “road” and only 20% are “obstacle,” it may fail to detect obstacles reliably.

4. Diversity and Variation:

- Include images from different angles, lighting conditions, backgrounds, and distances.
- Helps the model recognize the same object in different real-world scenarios (like day vs night, shadow vs bright light).
- For a rover variation in terrain and obstacle types is crucial.

5. Proper Labeling:

- Labels must be accurate and consistent.

- Even small labeling mistakes can confuse the model.

- Example: mixing up “arrow left” and “arrow right.”

6. Relevance:

- The images must match the **real-world environment** where the model will be used.

- For a rover training with outdoor, ground-level, real terrain images is more useful than random indoor pictures.

7. Minimal Redundancy:

- Avoid too many **duplicate or near-identical images** as they don’t add new information.

- Diverse images improve learning efficiency.

(5)

(a)

Model Inferencing and Performance Measurement:

Model inferencing is the act where a trained machine learning model predicts on new unseen instances. After the model has been trained with training data, inferencing allows it to apply the acquired knowledge in real-life scenarios. For instance, for a rover, inferencing is when the rover's camera captures a live image and the model infers whether it has an obstacle, a bottle, or an arrow pointing direction.

In order to measure a model's accuracy and performance, we compare its predictions with actual correct labels of a test set. Various measures such as accuracy, precision, recall, and mAP (Mean Average Precision) are used to measure how accurately a model performs. These measures tell us about the efficiency of the model in detecting, classifying, and identifying objects so that it performs well under any circumstances.

(b)

Model Performance Metrics:

The mAP@50 or Mean Average Precision at IoU 0.5 is one of the performance measures for object detection model such as YOLO. It measures model precision to detect and classify objects based on overlapping bounding boxes. The "@50" means the predicted bounding box must have at least an overlap with ground truth 50% (IoU ≥ 0.5) to qualify as a correct detection. The greater mAP@50 score indicates better detection performance.

The mAP@50–95 is a stricter and more specific version of mAP. Instead of evaluating the model at one 50% overlap, it averages precision scores across multiple IoU thresholds, ranging from 0.5 to 0.95. This is a more realistic and representative measure of how well the model detects easy objects where the objects are partially visible or closely overlapping.

Recall is defined as the number of true positive instances (say, actual obstructions) that were identified correctly by the model. The model never misses numerous real objects if recall is high, and it is crucial for a rover to ensure that it captures all the obstacles in front of it.

Precision, on the other hand, shows the number of the detected instances that actually took place correctly. High precision means that the model generates fewer false detection. For example: for the rover high precision does not classify shadows or random shapes as an obstacle. Precision and recall together measure the model's reliability and safety in making decisions.

©

Determining if a Model is "Good Enough":

A model is good enough if it achieves high and well-balanced performance on all the key metrics and performs stably when tested in real-world scenarios. For object detection models, good performance generally involves achieving high figures for mAP (e.g., mAP@50 > 0.8) and both precision and recall > 0.85. For the rover, this would translate into the model correctly identifying all the objects of interest such as arrows and obstacles with minimal false alarms. It should also work reliably under varying lighting conditions, backgrounds, and camera orientations to represent real-world variations.

(d)

Confusion Matrix and Its Significance:

Confusion matrix is a table used to plot and measure the accuracy of a classification model based on separating predicted and actual results. It indicates four main outcomes: True Positives (correctly predicted positive cases), True Negatives (correctly predicted negative cases), False Positives (wrongly predicted positives), and False Negatives (false negatives).

In a rover scenario, True Positives can represent obstacles identified correctly, and False Positives can represent the case where the model identifies an obstacle that does not exist. False Negatives are more severe because they mean the rover did not identify an actual obstacle, which could lead to a collision. The confusion matrix can indicate the very type of errors a model makes, and that is crucial for fine-tuning and improving its performance.