UNIT-1

Introduction to Machine Learning

Evolution of Machine Learning:

Machine learning (ML) is a powerful branch of artificial intelligence (AI) that allows computers to learn without explicit programming. Instead of following a set of rigid instructions, ML algorithms can analyze data, identify patterns, and make predictions

Teaching a computer to recognize your friends in photos without showing it every single picture is like training it to learn from examples. Imagine you have a bunch of photos where your friends are tagged with their names. The computer looks at these photos and tries to find patterns. For example, it might notice that one friend has blue eyes and curly hair, while another friend has brown eyes and wears glasses.

Using these patterns, the computer starts to guess who's who in new photos. If it sees someone with blue eyes and curly hair, it might say, "Hey, that looks like Arushi!" But the computer doesn't always get it right at first. It needs lots of practice and feedback to improve, just like how we learn from our mistakes.

This is where machine learning comes in. It's like giving the computer a super smart brain that learns from experience. As the computer sees more and more photos and gets feedback on its guesses, it gets better

at recognizing your friends. It's like **"teaching a robot"** to be a detective, but instead of clues, it uses pictures!

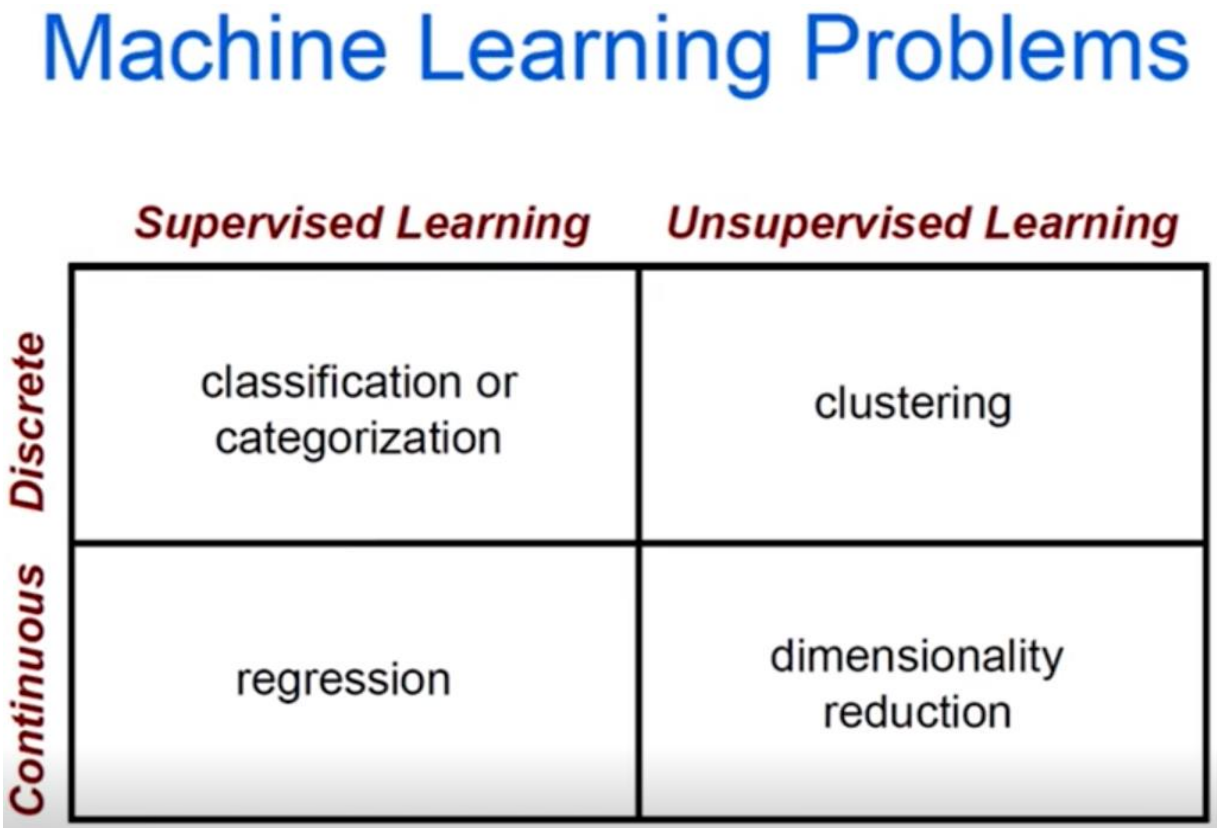
In the big world of artificial intelligence (AI), machine learning is like the secret sauce that helps AI systems become smarter over time. It's what makes AI capable of doing amazing things like understanding speech, playing games, and yes, even recognizing your friends in photos. It's pretty cool how technology can learn and improve, just like we do!

The evolution of machine learning is a captivating story. Early ideas emerged in the 1943s, but limitations in computing power hampered significant progress. With the recent explosion of data and computing muscle, machine learning has taken center stage. Today, it's used everywhere from spam filters in your email to recommending movies you might enjoy. Machine learning is constantly evolving, shaping the future of technology and impacting our lives in profound ways.

Paradigms for ml:

Machine learning (ML) is a dynamic field dedicated to developing methods that enable machines to learn from extensive datasets to enable machines to learn and make predictions. The learning paradigms in ML are categorized based on their resemblance to human interventions, each serving specific purposes and applications. This

dynamic field encompasses various learning paradigms, each with its unique approach to handling data.

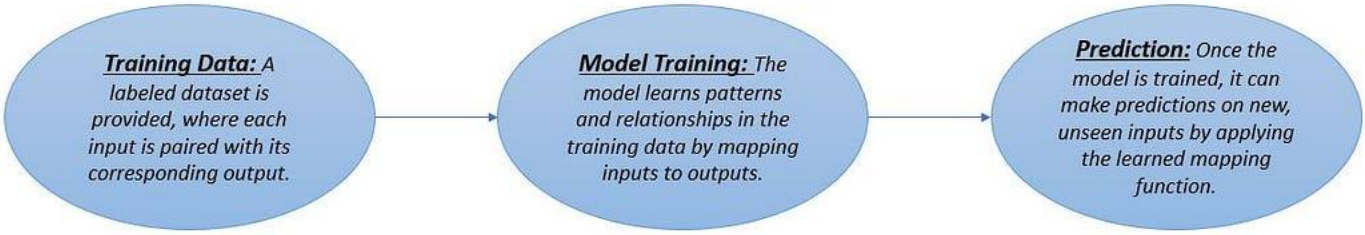


Supervised and Unsupervised learning

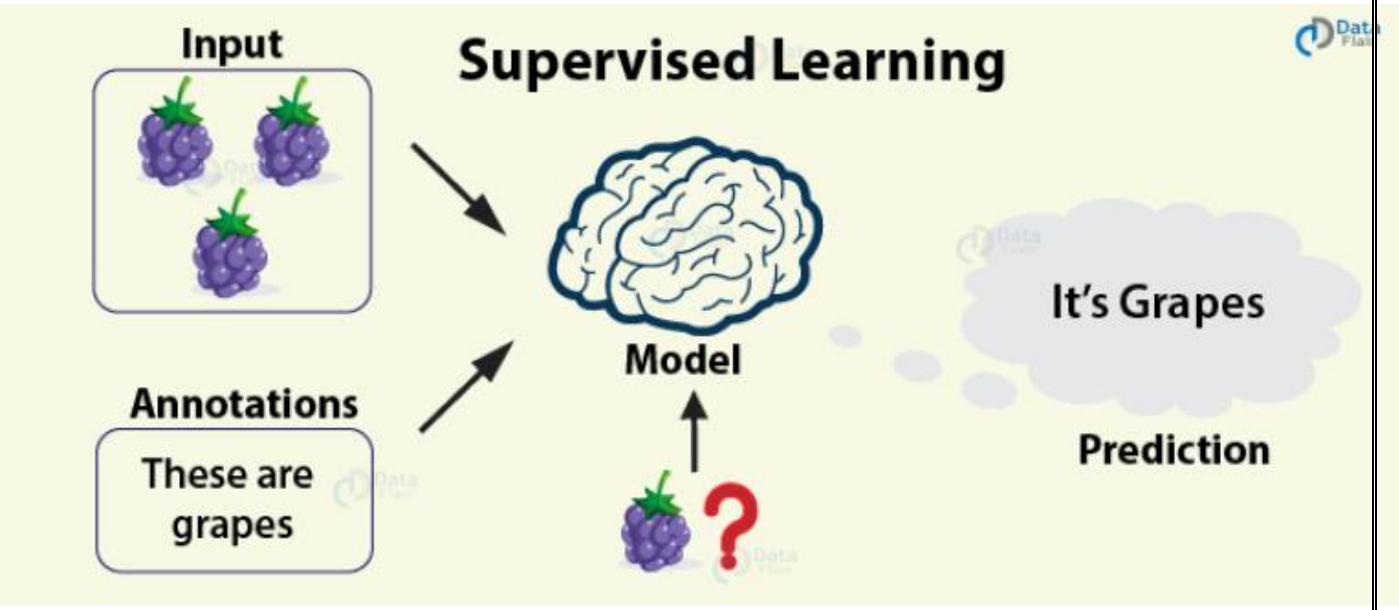
Supervised Learning (SL)

Supervised learning involves labelled datasets, where each data observation is paired with a corresponding class label. Algorithms in supervised learning aim to build a mathematical function that maps

input features to desired output values based on these labeled examples. Common applications include classification and regression.



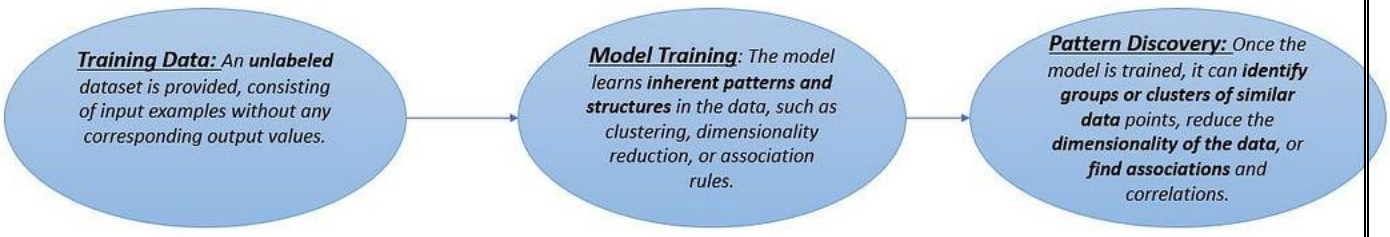
Stages in Supervised Learning



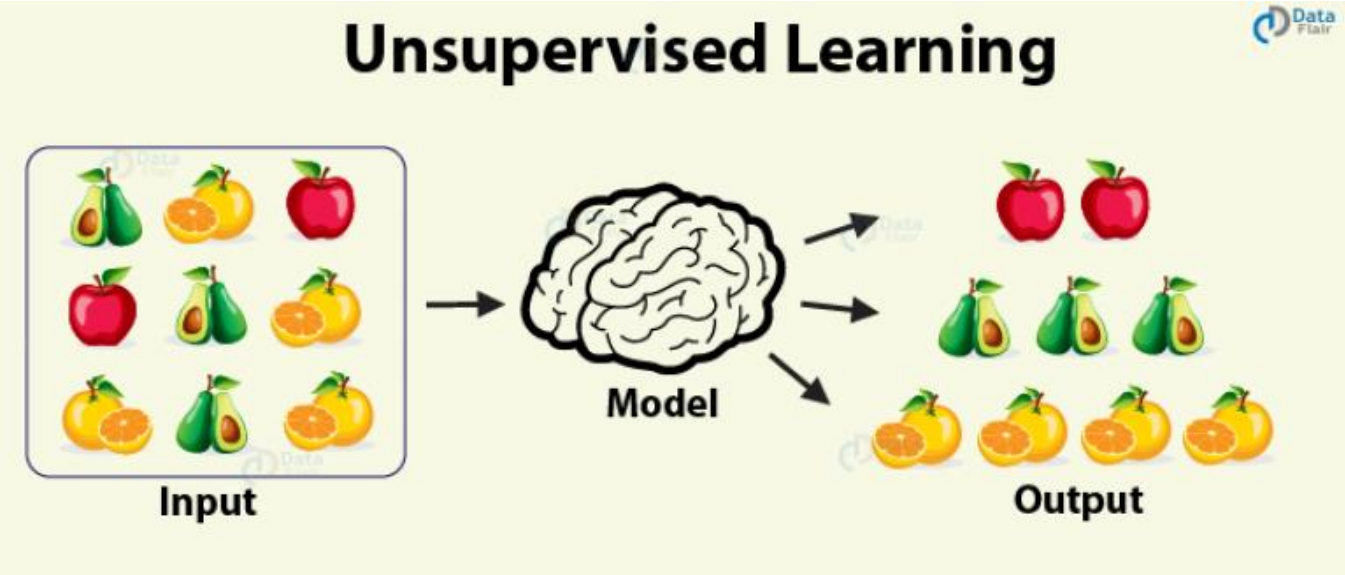
Understanding Supervised Learning pictorially

Unsupervised Learning

In unsupervised learning, algorithms work with unlabeled data to identify patterns and relationships. These methods uncover commonalities within the data without predefined categories. Techniques such as clustering and association rules fall under unsupervised learning.



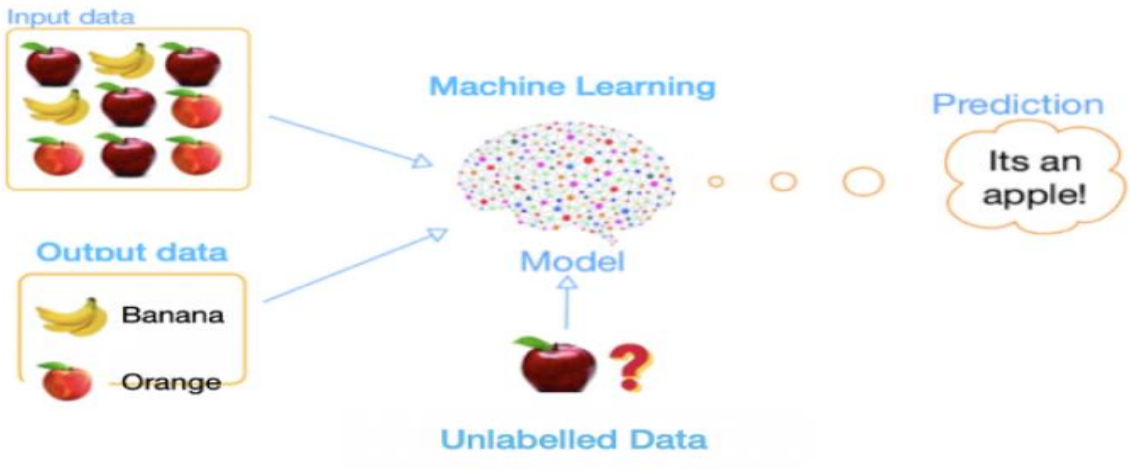
Stages in Unsupervised Learning



Understanding Unsupervised Learning pictorially

Semi-supervised Learning

Semi-supervised learning strikes a balance by combining a small amount of labelled data with a larger pool of unlabeled data. This approach leverages the benefits of both supervised and unsupervised learning paradigms, making it a cost-effective and efficient method for training models when the labeled data is limited.



Understanding Semi-supervised Learning pictorially

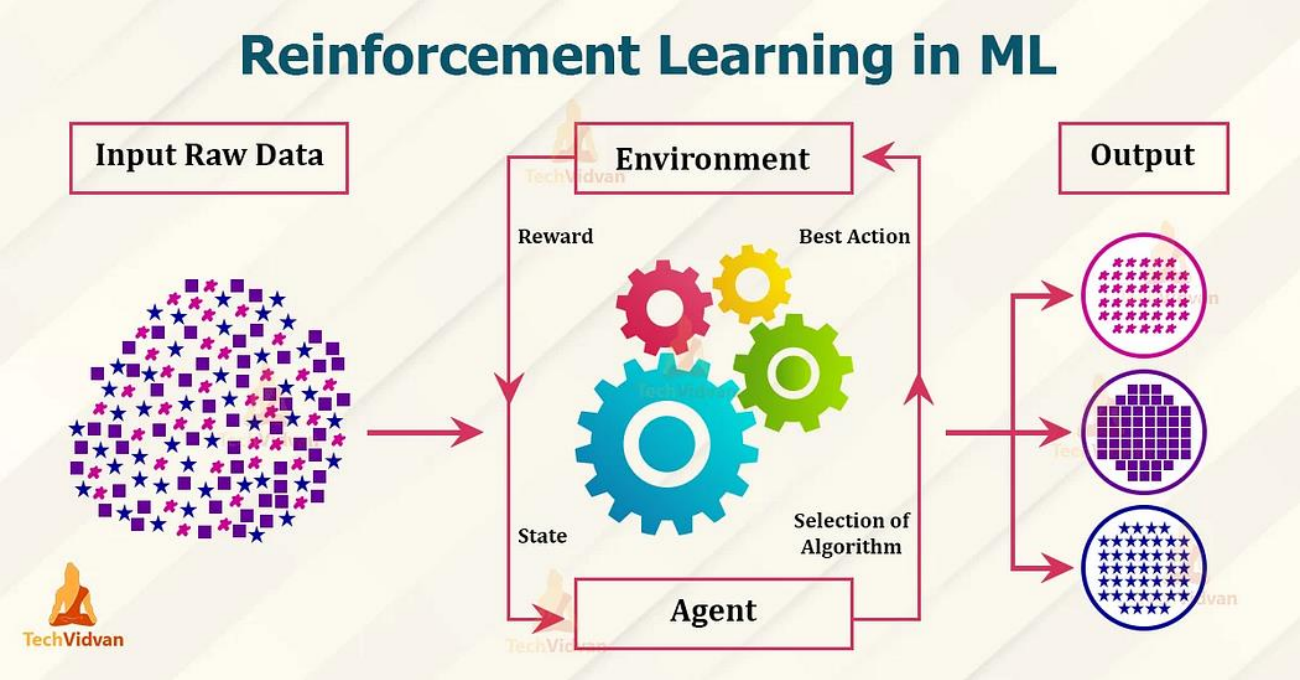
Self-supervised Learning (SSL)

In scenarios where obtaining high-quality labeled data is challenging, self-supervised learning emerges as a solution. In this paradigm, models are pre-trained using unlabeled data, and data labels are generated automatically during subsequent iterations. SSL transforms unsupervised ML problems into supervised ones, enhancing learning efficiency. This paradigm is particularly relevant with the rise of *large language models*.

Reinforcement Learning

Reinforcement learning focuses on enabling intelligent agents to learn tasks through trial-and-error interactions with dynamic environments. Without the need for labelled datasets, agents make decisions to

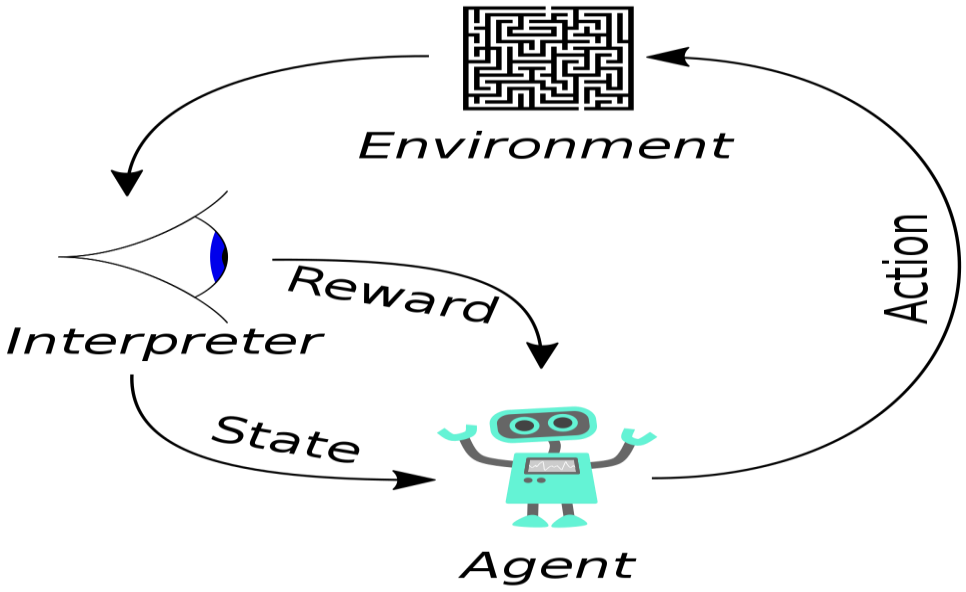
maximize a reward function. This autonomous exploration and learning approach is crucial for tasks where explicit programming is challenging.



Action-Reward feedback loop: an agent takes actions in an environment, which is interpreted into a reward and a representation of the state, which are fed back into the agent.

Action-Reward Feedback Loop:

Reinforcement learning operates on an action-reward feedback loop, where agents take actions, receive rewards, and interpret the environment's state. This iterative process allows the agent to autonomously learn optimal actions to maximize positive feedback.



Understanding these ML paradigms provides valuable insights into the diverse approaches used to address different types of problems. Each paradigm comes with its strengths and applications, contributing to the versatility of machine learning in various domains.

Learning by Rote:

The meaning of “rote” it self means learning by repetition. The process of repeating something over and over engages the short-term memory and allows us to quickly remember basic things like facts, dates, names, multiplication tables ,etc. It differs from other forms of learning in that it doesn’t require the learner to carefully think about something, and is rather dependent on the act of repetition itself

Even though complete and holistic learning is not dependent on rote learning techniques alone, they do allow students to quickly recall basic facts and laws and master foundational knowledge of a topic in students. Some examples of rote learning in schools can be found in the following:

- Repeating words to instil them in your vocabulary.
- Learning scales in music.
- Memorizing the periodic table.
- Learning the basic laws and formulae in physics and sundry sciences.

Having to memorize the basic facts and principles of a field is an import prerequisite to later analyse and study them. This is where rote learning techniques come in handy and allow you to remember the building blocks of concepts without having to dive deep into them

Rote learning techniques:

Rote learning techniques are aplenty, and they all require time and effort in repetition. The more you repeat for longer periods, the easier it will be to recall. Even if you only have a few hours to memorize something, the following rote learning techniques will help you remember quickly:

Read it aloud-Read the text out loud with understanding. You can even try it before a mirror, ask a friend to listen to you, or read it out just under your breath. You can experiment with how slow or fast you want to read, how expressive you want to be, and internalize the rhythm of the text. Auditory learners will greatly benefit from this rote learning technique.

Write it down- Writing down the text information after reading is one of the best rote learning techniques. Doing so will help identity difficult passengers and areas that need more practice. If you're preparing for a written exam, this kinesthetic rote technique will serve as a rehearsal and commit the information for easy retrieval

Visualize - Humans are visual creatures and our brains are wired to remember things better with images. For every line and connected phrase, come up with ways to visualize it and remember it. The memory palace can be a useful trick for such rote learning techniques.

Free association-Free association is one of the more intresting rote learning techniques, and a very useful way of remembering things quickly, especially if they are too messy for the traditional rote learning techniques. The main idea of this method is to combine new information with what you already know in a fun and personal way. For instance, if you're learning the "Circle of Fifths" in music ,you can associate each node to the numbers on the clock, one for each of the 12

notes in music. you are free to form your own associations as you see fit, as long as it helps you to recall the information

Advantages of Rote Learning Techniques:

Rote learning is considered as useful for a variety of reasons. Here are a few:

- Rote learning requires very little analysis.
- With rote, one can remember just about anything over little analysis
- Rote learning allows one to recall information wholly, and even to retain it for life.
- Rote learning makes it easier for people to score who find it difficult to understand or master reading and maths concepts.
- Rote learning can help improve short-term memory.

Disadvantages of rote learning techniques:

On the other hand, there are a few drawbacks of rote learning that you need to be aware of as well

- The repetitive nature of rote learning can become dull.
- One can easily lose focus while rote learning.
- Rote learning is not holistic.

- There is no connection between new and old information with rote learning.
- Rote learning doesn't lead to a deeper understanding of the information.

Learning by Induction:

What is Inductive Learning Algorithm?

Inductive Learning Algorithm (ILA) is an iterative and inductive machine learning algorithm that is used for generating a set of classification rules, which produces rules of the form "IF-THEN", for a set of examples, producing rules at each iteration and appending to the set of rules.

There are basically two methods for knowledge extraction firstly from domain experts and then with machine learning. For a very large amount of data, the domain experts are not very useful and reliable. So we move towards the machine learning approach for this work. To use machine learning One method is to replicate the expert's logic in the form of algorithms but this work is very tedious, time taking, and expensive. So we move towards the inductive algorithms which generate the strategy for performing a task and need not instruct separately at each step.

Why you should use Inductive Learning?

The ILA is a new algorithm that was needed even when other reinforcement learnings like ID3 and AQ were available.

- The need was due to the pitfalls which were present in the previous algorithms, one of the major pitfalls was the lack of generalization of rules.
- The ID3 and AQ used the decision tree production method which was too specific which were difficult to analyze and very slow to perform for basic short classification problems.
- The decision tree-based algorithm was unable to work for a new problem if some attributes are missing.
- The ILA uses the method of production of a general set of rules instead of decision trees, which overcomes the above problems

Basic Requirements to Apply Inductive Learning Algorithm

1. List the examples in the form of a table 'T' where each row corresponds to an example and each column contains an attribute value.
2. Create a set of m training examples, each example composed of k attributes and a class attribute with n possible decisions.
3. Create a rule set, R, having the initial value false.
4. Initially, all rows in the table are unmarked.

Necessary Steps for Implementation

- **Step 1:** divide the table 'T' containing m examples into n sub-tables (t1, t2,.....tn). One table for each possible value of the class attribute. (repeat steps 2-8 for each sub-table)

- **Step 2:** Initialize the attribute combination count ‘ j ‘ = 1.
- **Step 3:** For the sub-table on which work is going on, divide the attribute list into distinct combinations, each combination with ‘j ‘ distinct attributes.
- **Step 4:** For each combination of attributes, count the number of occurrences of attribute values that appear under the same combination of attributes in unmarked rows of the sub-table under consideration, and at the same time, not appears under the same combination of attributes of other sub-tables. Call the first combination with the maximum number of occurrences the max-combination ‘ MAX’.
- **Step 5:** If ‘MAX’ == null, increase ‘ j ‘ by 1 and go to Step 3.
- **Step 6:** Mark all rows of the sub-table where working, in which the values of ‘MAX’ appear, as classified.
- **Step 7:** Add a rule (IF attribute = “XYZ” –> THEN decision is YES/ NO) to R whose left-hand side will have attribute names of the ‘MAX’ with their values separated by AND, and its right-hand side contains the decision attribute value associated with the sub-table.
- **Step 8:** If all rows are marked as classified, then move on to process another sub-table and go to Step 2. Else, go to Step 4. If no sub-tables are available, exit with the set of rules obtained till then.

An example showing the use of ILA suppose an example set having attributes Place type, weather, location, decision, and seven

examples, our task is to generate a set of rules that under what condition is the decision.

Example no.	Place type	weather	location	decision
1.	hilly	winter	kullu	Yes
2.	mountain	windy	Mumbai	No
3.	mountain	windy	Shimla	Yes
4.	beach	windy	Mumbai	No
5.	beach	warm	goa	Yes
6.	beach	windy	goa	No
7.	beach	warm	Shimla	Yes

Subset – 1

s.no	place type	weather	location	decision
1.	hilly	winter	kullu	Yes

s.no	place type	weather	location	decision
2.	mountain	windy	Shimla	Yes
3.	beach	warm	goa	Yes
4.	beach	warm	Shimla	Yes

Subset – 2

s.no	place type	weather	location	decision
5.	mountain	windy	Mumbai	No
6.	beach	windy	Mumbai	No
7.	beach	windy	goa	No

- **At iteration 1** rows 3 & 4 column weather is selected and rows 3 & 4 are marked. the rule is added to R IF the weather is warm then a decision is yes.
- **At iteration 2** row 1 column place type is selected and row 1 is marked. the rule is added to R IF the place type is hilly then the decision is yes.

- **At iteration 3** row 2 column location is selected and row 2 is marked. the rule is added to R IF the location is Shimla then the decision is yes.
- **At iteration 4** row 5&6 column location is selected and row 5&6 are marked. the rule is added to R IF the location is Mumbai then a decision is no.
- **At iteration 5** row 7 column place type & the weather is selected and row 7 is marked. the rule is added to R IF the place type is beach AND the weather is windy then the decision is no.

Finally, we get the rule set:- Rule Set

- **Rule 1:** IF the weather is warm THEN the decision is yes.
- **Rule 2:** IF the place type is hilly THEN the decision is yes.
- **Rule 3:** IF the location is Shimla THEN the decision is yes.
- **Rule 4:** IF the location is Mumbai THEN the decision is no.
- **Rule 5:** IF the place type is beach AND the weather is windy THEN the decision is no.

Reinforcement Learning:

• It tries to improve its performance of doing the task.

• When a sub-task is accomplished successfully, a reward is given.

• When a sub-task is not executed correctly, obviously no reward is given.

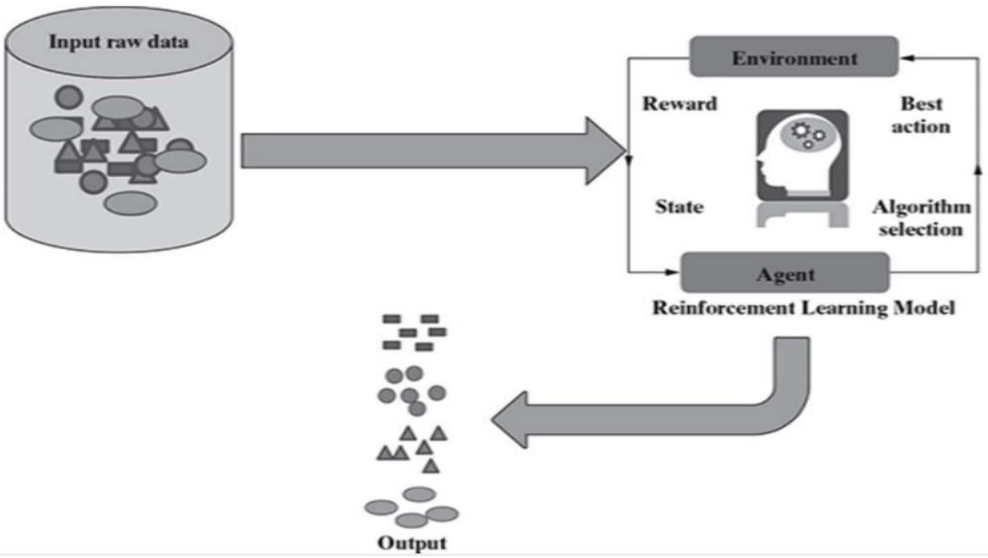
• This continues till the machine is able to complete execution of the whole task.

• This process of learning is known as reinforcement learning.

• One contemporary example of reinforcement learning is self-driving cars.

• The critical information which it needs to take care of are speed and speed limit in different road segments, traffic conditions, road conditions, weather conditions, etc.

• The tasks that have to be taken care of are start/stop, accelerate/decelerate, turn to left / right, etc.



- This type of learning is used when there is no idea about the class or label of a particular data. The model has to do the classification it will get rewarded if the classification is correct, else get punished.
- The model learns and updates itself through reward/punishment
- Model is evaluated by means of the reward function ater it had some time to learn.

- Most complex to understand and apply
- Standard algorithms are
    - Q-Learning
    - Sarsa
- Practical applications are
    - Self-driving cars
    - Intelligent robots
    - AlphaGo Zero [The latest version of DeepMind's AI system playing Go]

Types of Data Matching:

Software tools have been developed to automate the process of data matching.

Large enterprises have a lot of data. And with all that data comes the challenge of keeping it organized and accurate – or more specifically, making sure that their data is being leveraged to its full potential.

One key way to do this is through record or data matching, which is the process of connecting data records that correspond to the same canonical (master) entity.

Most enterprise databases, due to their breadth and depth of data, will have some degree of duplicates or inaccuracies (e.g. in a database of locations, "San Francisco" may be written as "SF", "San Fran", or "SFO"). Data matching tools help to standardize data and improve its

quality by identifying these duplicates and linking them to a single, accurate record.

Naturally, software tools have been developed to automate this process. Below, we'll take a look at the various types of data matching tools available, how they work, and some use cases for each.

Types of Data Matching Tools

There are two main types of data matching tools:

1) probabilistic and

2) deterministic.

Probabilistic Data Linkage Tools:

Probabilistic data linkage tools use statistical methods to determine the likelihood that two records refer to the same entity. They work by comparing different fields in the records and assigning a similarity score for each field; the overall similarity score for the two records is then used to make a probabilistic determination of whether they should be linked.

Today, most probabilistic tools employ machine learning algorithms to provide even more accurate results. Regression and natural language processing techniques are often used to automatically identify and extract important features from records, which are then used in the similarity scoring process.

Advantages

- Probabilistic record matching tools can be used on data of any type, **including unstructured data**.

- Real world data tends to be unstructured, and often poorly maintained due to manual data entry.
- They are able to handle many spelling/coding variants and exceptions that would not be easy to cover with a predefined rigid rule set or even a robust dictionary.
- They are generally more accurate than their counterpart, deterministic data linkage tools (more on this below).

Disadvantages:

- They can be more difficult to configure and tune, since there are more parameters that need to be set.
- They require a good amount of data in order to train the machine learning models used for feature extraction and similarity scoring.
- Some of their results are difficult to interpret; since they are based on probabilistic methods, much of the process occurs behind-the-scenes.

Deterministic Data Matching Tools:

Deterministic data matching tools, on the other hand, use rule-based methods to connect records. That is, they compare different fields in the records using a system like RegEx and look for exact matches; if two fields match, then the records are linked.

Since deterministic tools use a predetermined set of rules, they are generally much easier to configure than their probabilistic

counterparts. However, this also means that they can be less accurate, since they may miss some relationships that don't fit the rules.

Deterministic data matching tools are often used in cases where data is highly structured and well-defined; for example, when linking records from two different databases that use the same schema. By contrast, probabilistic data linkage tools are better suited for data that is unstructured or has many different schemas.

Advantages

- They are much easier to configure and tune than probabilistic data linkage tools.
- They can be used on data of any type, including unstructured data.
- You don't need much data in order to use them, since they don't require training data for machine learning models.

Disadvantages

- They can be less accurate than probabilistic data matching tools, since they may miss some relationships that don't fit the rules.
- They are often less flexible than probabilistic data matching tools, since they can only compare fields in a predetermined way.
- Some of their results are difficult to interpret; since they are based on deterministic methods, much of the process occurs behind-the-scenes.

Uses of Data Matching Tools

Record matching tools can be used for a variety of tasks, including:

- **Data deduplication** – identifying and removing duplicate records from your database as a specific form of data quality assurance.
- **Data enrichment** – combining your data with other data sources in order to enrich it.
- **Data integration** – connecting different data sources that use different schemas.
- **Fraud detection** – identifying records that are likely to be fraudulent, based on their similarity to other records in your database or specific features of known examples.
- **Building customer 360 profiles:** connecting customer data from different sources (e.g. social media, website interactions, customer service interactions) in order to get a complete view of the customer.

In addition to direct benefits to revenue, there are also longer-term benefits to be gained from using record matching tools.

For example, by linking customer data across different channels, you can develop a deeper understanding of customer behavior. This, in turn, can lead to better targeted marketing campaigns and improved customer retention rates. Perhaps the company is gearing up to sell to a large acquirer. In this case, having accurate and up-to-date data is critical in order to get the best price for the business.

Machine Learning in Data Matching Tools:

Probabilitic reocrd linkage tools make use of machine learning trained for specific tasks. ML algorithms are what handles confidence scoring of records. In this case we are dealing with machine learning rather than the current wave of AI tools

Machine learning employed for data matching does its task very well, making it a very reliable way of reducing manual work and improving the quality of your data.

Stages in Machine Learning:

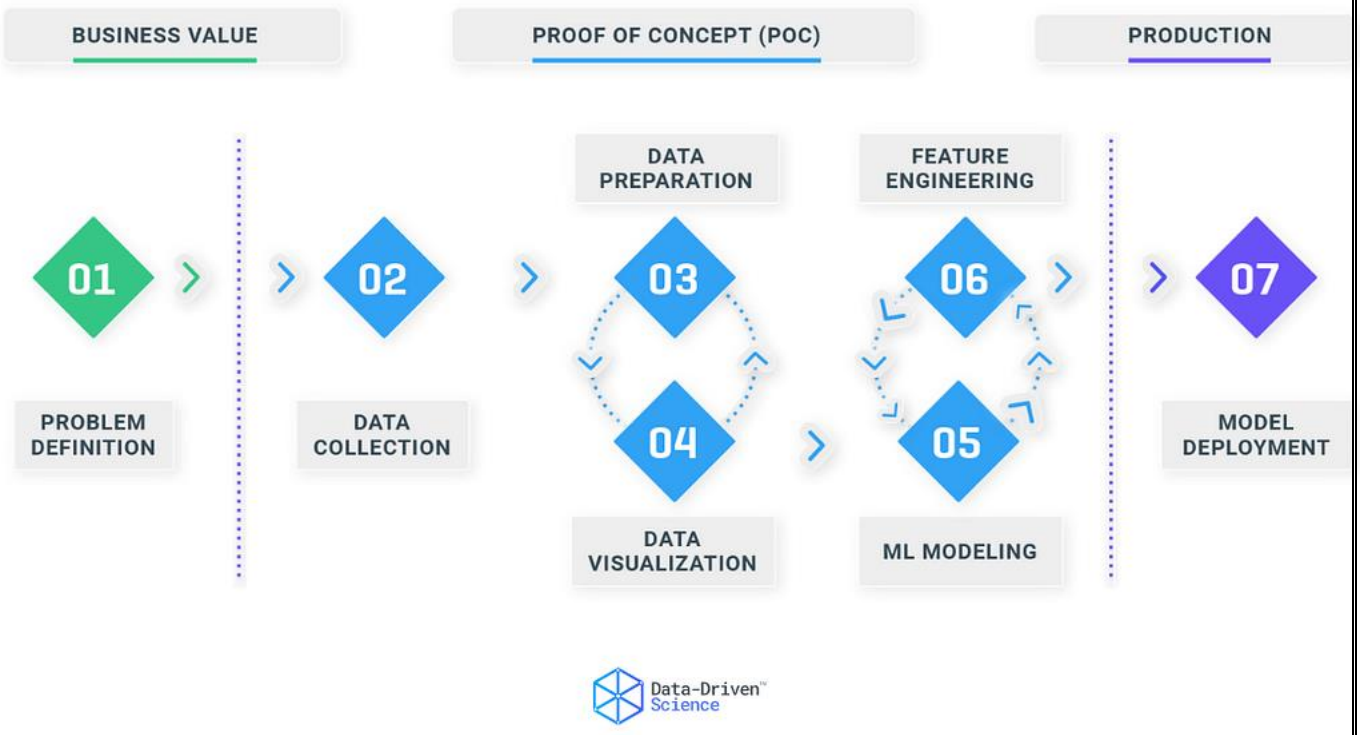
The goal of the Seven Stages framework is to break down all necessary tasks in Machine Learning and organize them in a logical way. At the end, the framework acts as a general process that can be universally applied to any project independently of industry and type of business.

Data-Driven Science (DDS) will also use that framework for its upcoming comprehensive Machine Learning online course published on Udemy — stay tuned. We will go deep into each stage and give you everything that is needed to complete Data Science projects successfully.

The 7 Stages of Machine Learning are:

1. Problem Definition

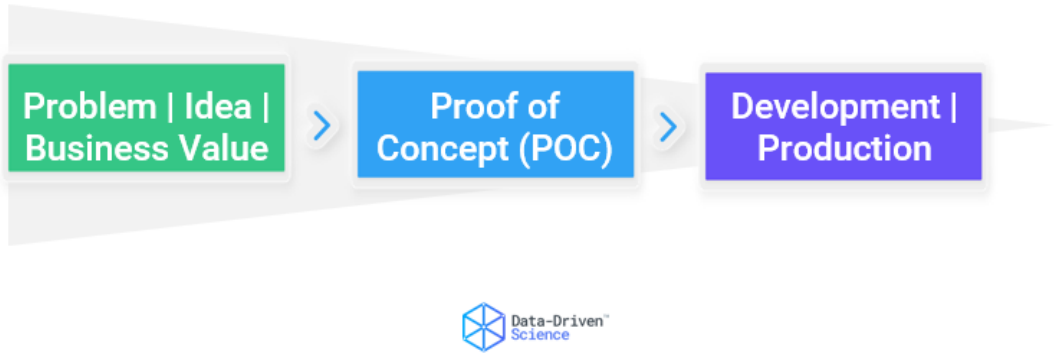
2. Data Collection
3. Data Preparation
4. Data Visualization
5. ML Modeling
6. Feature Engineering
7. Model Deployment



These 7 stages are the key steps in our framework. We have categorized them additionally into groups to get a better understanding of the larger picture.

The stages are grouped into 3 phases:

1. Business Value
2. Proof of Concept (POC)
3. Production



Phase 1 — Business Value

It is absolutely crucial to adopt a business mindset when thinking sabout a problem that should be solved with Machine Learning — defining customer benefits and creating business impact is top priority. Domain expertise and knowledge is also essential as the true power of data can only be harnessed if the domain is well known and understood.

Phase 2 — Proof of Concept (POC)

Proof of Concept (POC) is the most comprehensive part of our framework. From Data Collection to Feature Engineering, 5 stages of

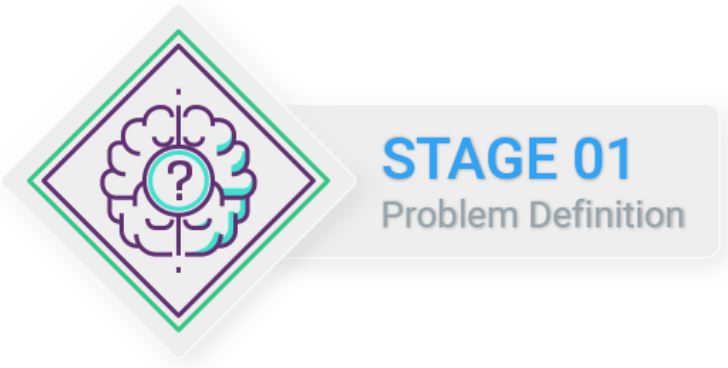
our ML framework are included here. Core of any POC to test an idea in terms of its feasibility and value to the business. Also, questions around performance and evaluation metrics are answered in that phase. Only a strong POC that delivers business value and is feasible allows one putting the ML Model into production.

Phase 3 — Production

In the third phase, one is taking the ML model and scaling it. The goal is to integrate Machine Learning into a business process solving a problem with a superior solution compared to, for example, traditional programming. The process of taking a trained ML model and making its predictions available to users or other systems is known as model deployment. Lastly, it is also essential to iterate on the ML model over time to improve it.

7 Stages of Machine Learning

1. Problem Definition



The first stage in the DDS Machine Learning Framework is to define and understand the problem that someone is going to solve. Start by analyzing the goals and the *why* behind a particular problem statement. Understand the power of data and how one can use it to make a change and drive results. And asking the right questions is always a great start.

Few possible questions:

- What is the business?
- Why does the problem need to be solved?
- Is a traditional solution available to solve the problem?
- If probabilistic in nature, then does available data allow to model it?
- What is a measurable business goal?

2. Data Collection

Once the goal is clearly defined, one has to start getting the data that is needed from various available data sources.

At this stage, some of the questions worth considering are:

- What data do I need for my project?
- Where is that data available?
- How can I obtain it?
- What is the most efficient way to store and access all of it?

There are many different ways to collect data that is used for Machine Learning. For example, focus groups, interviews, surveys, and internal usage & user data. Also, public data can be another source and is usually free. These include research and trade associations such as banks, publicly-traded corporations, and others. If data isn't publicly available, one could also use web scraping to get it (however, there are some legal restrictions).

3. Data Preparation

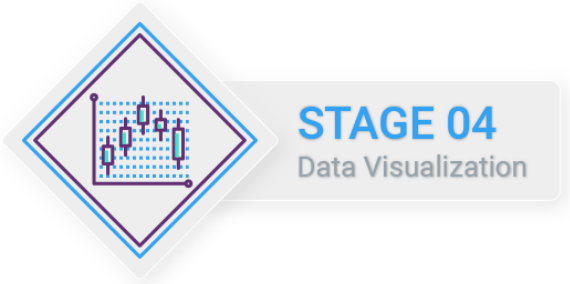
The third stage is the most time-consuming and labor-intensive. Data Preparation can take up to 70% and sometimes even 90% of the overall project time. But what is the purpose of this stage?

Well, the type and quality of data that is used in a Machine Learning model affects the output considerably. In Data Preparation one explores, pre-processes, conditions, and transforms data prior to modeling and analysis. It is absolutely essential to understand the data, learn about it, and become familiar before moving on to the next stage.

Some of the steps involved in this stage are:

- Data Filtering
- Data Validation & Cleansing
- Data Formatting
- Data Aggregation & Reconciliation

4. Data Visualization

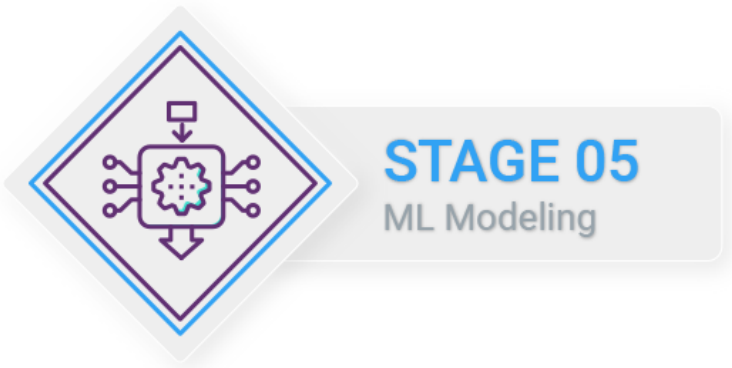


Data Visualization is used to perform Exploratory Data Analysis (EDA). When one is dealing with large volumes of data, building graphs is the best way to explore and communicate findings. Visualization is an incredibly helpful tool to identify patterns and trends in data, which leads to clearer understanding and reveals important insights. Data Visualization also helps for faster decision making through the graphical illustration.

Here are some common ways of visualization:

- Area Chart
- Bar Chart
- Box-and-whisker Plots
- Bubble Cloud
- Dot Distribution Map
- Heat Map
- Histogram
- Network Diagram
- Word Cloud

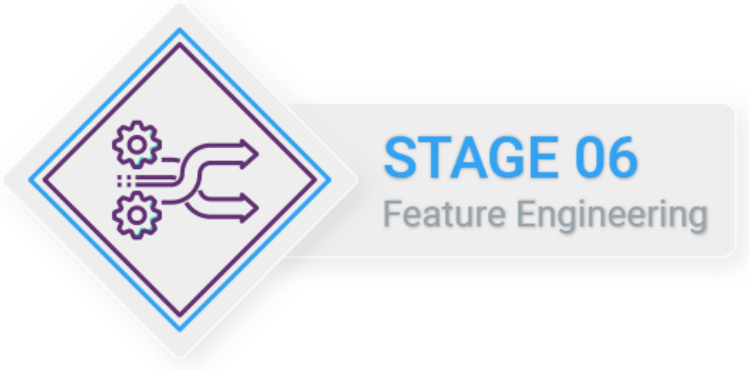
5. ML Modeling



Finally, this is where *'the magic happens'*. Machine Learning is finding patterns in data, and one can perform either supervised or unsupervised learning. ML tasks include regression, classification, forecasting, and clustering.

In this stage of the process one has to apply mathematical, computer science, and business knowledge to train a Machine Learning algorithm that will make predictions based on the provided data. It is a crucial step that will determine the quality and accuracy of future predictions in new situations. Additionally, ML algorithms help to identify key features with high predictive value.

6. Feature Engineering



Machine Learning algorithms learn recurring patterns from data. Carefully engineered features are a robust representation of those patterns.

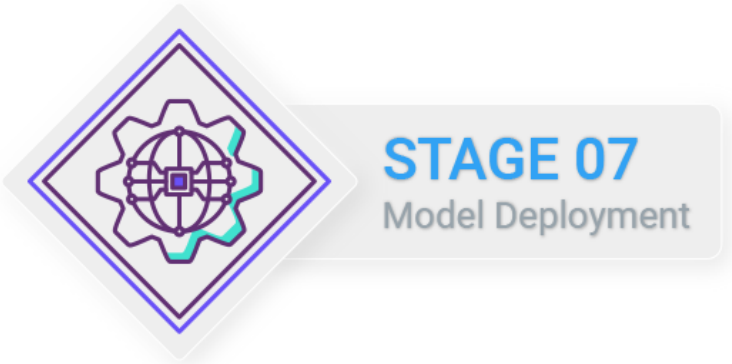
Feature Engineering is a process to achieve a set of features by performing **mathematical, statistical,** and heuristic procedures. It is a collection of methods for identifying an optimal set of inputs to the Machine Learning algorithm. Feature Engineering is extremely important because well-engineered features make learning possible with simple models.

Following are the characteristics of good features:

- Represents data in an unambiguous way
- Ability to captures linear and non-linear relationships among data points

- Capable of capturing the precise meaning of input data
- Capturing contextual details

7. Model Deployment



The last stage is about putting a Machine Learning model into a production environment to make data-driven decisions in a more automated way. Robustness, compatibility, and scaleability are important factors that should be tested and evaluated before deploying a model. There are various ways such as Platform as a Service (PaaS) or Infrastructure as a Service (IaaS). For containerized applications, one can use container orchestration platform such as Kubernetes to rapidly scale the number of containers as demand shifts.

Another important part of the last stage is iteration and interpretation. It is critical to constantly optimize the model and pressure test the results. At the end, Machine Learning has to provide value to the

business and make a positive impact. Therefore, monitoring the model in production is key.

Conclusion

This was an overview about '**The 7 Stages of Machine Learning**' — a framework that helps to structure the typical process of a ML project. The idea is to equip practitioners with a template that can be universally applied and simplifies the process from idea to implementation.

Data acquisition:

Data acquisition, or DAQ, is the cornerstone of machine learning. It is essential for obtaining high-quality data for model training and optimizing performance. Data-centric techniques are becoming more and more important across a wide range of industries, and DAQ is now a vital tool for improving productivity, preserving quality, and stimulating innovation.

What is Data Acquisition?

"The process of collecting and storing data for machine learning from a variety of sources is known as data acquisition (DAQ)."

The procedure entails gathering, examining, and using crucial data to guarantee precise measurements, instantaneous observation, and knowledgeable decision-making. Sensors, measuring devices, and a

computer work together in DAQ systems to transform physical parameters into electrical signals, condition and amplify those signals, and then store them for analysis.

What is Data Acquisition in Machine Learning?

In machine learning, "data acquisition" refers to the procedure of obtaining and compiling data from diverse sources in order to test and train machine learning models.

In order to enable computers and software to manipulate and modify signals from real-world occurrences, this technique entails digitizing such signals. Data Acquisition aims to get a complete and representative dataset that successfully captures the patterns and changes in the data that are crucial for productive machine learning results.

The process of acquiring data also include taking the variable into account that affect its quality and utility, such as volume, velocity and diversity

Successful machine learning begins with data collecting, which supplies the raw information required to train models and make defensible conclusions. The gathering of high-quality data is essential for providing machine learning algorithms with the necessary input to enable them to learn and perform better.

What Does a DAQ System Measure?

A Data Acquisition (DAQ) system is capable of measuring several physical parameters, such as:

- **Temperature:** Temperature can be measured using **R**esistance **T**emperature **D**etector, thermistors, or thermocouples in DAQ systems.
- **Pressure:** In a variety of settings, including industrial operations and medical equipment, pressure is measured using pressure sensors.
- **Voltage:** Power systems, electronics, and electrical engineering all depend on the ability of DAQ devices to monitor the voltage levels in electrical circuits.
- **Current:** DAQ systems can measure current flow using current sensors or shunts. Current measurement is essential in electrical systems.
- **Strain and Pressure:** Deformation and pressure in materials are measured using strain gauges and pressure sensors, which is crucial for material science and structural health monitoring.
- **Shock and Vibration:** In a variety of fields, including mechanical, aeronautical, and civil engineering, accelerometers and vibration sensors are used to monitor shock, vibration, and acceleration.
- **RPM, Angle, and Discrete Events**: DAQ systems are crucial for robotics, automation, and mechanical systems because they can measure rotational speed, angle, and discrete events.
- **Distance and Displacement:** Ultrasonic, laser, and encoder sensors are among the sensors that DAQ systems can use to detect distance and displacement.

- **Weight**: Measuring weight is crucial for a number of applications, including quality control, logistics, and industrial automation.

Components of Data Acquisition System

To understand how data is selected and processed, a data acquisition system consists of below key basic components: sensors, measuring instruments, and a computer.

1**. Sensors:** Sensors are devices that quantify and translate physical parameters like voltage, pressure, or temperature into electrical impulses. Later, these signals are sent to the measuring devices for additional analysis.

2.Signal Conditioner: Signal conditioning is the process of improving raw sensor signals so they can be reliably understood. To make sure that the signals are dependable, clear, and compatible with the rest of the system, signal conditioning procedures include isolation, amplification, and filtering.

- **Amplification:** It helps in improving accuracy by maximizing the signal strength
- **Filtering:** Filters extra and unwanted noise from the signal
- **Isolation:** Helps in separating sensor from DAQ system.

3. Analog-to-digital Converter: After the signals are conditioned, they must be translated into a digital format that computers can comprehend using an analog-to-digital converter (ADC). The continuous analog signals are transformed into discrete digital values so that the system can process and store them.

4. Data Logger: The data logger serves as the operation's central nervous system. A device or software program known as a data logger is responsible for managing incoming data, controlling the acquisition process, and storing it for subsequently analysis.

5. Data Processing Unit: After receiving data from ADC, the system has dedicated card to process the signals like sampling, buffering and Data Transfer.

6. Data Storage: Acquired data is stored in the computer's memory for real-time monitoring.

The physical parameters are measured using sensors, which convert the physical signals into electrical signals. The signals are then conditioned, amplified, and converted into digital data using analog-to-digital converters (ADCs). The digital data is then processed, analyzed, and stored using computers and software.

What are the Major Purposes of Data Acquisition?

Although there are many different and important reasons, some of the most important ones are as follows:

- **Long-term analysis and trend detection:** Long-term analysis are made possible by data acquisition systems, which make it possible to log, capture, and store measurement of data over an extended period of time.
- **Measurement that is accurate and dependable**: DAQ systems and equipment provide measurement that is accurate and dependable, enabling uses like optical analysis and light intensity monitoring.

- **Industry Leading devices:** DAQ systems and devices are widely used, connecting to a variety of sensors and collaborating with contemporary computers, which makes them an excellent option for scientists and researchers looking for accurate data.
- **Enhanced productivity and dependability of machines:** Data capture gives an organization more control over its operations and enables quicker reaction to potential breakdowns, maximizing procedure optimization.
- **Faster problem analysis and resolution**: Real-time data acquisition systems allow measurements to be produced and shown instantly, which allows personnel to respond to issues more quickly and get the machine operating at peak efficiency in less time.
- **Reduction of data redundancy**: DAQ systems let businesses operate without interference from extraneous data by making it easier to analyze the information they have collected.

Types of Data Acquisition Sources

- **Sensors**: Convert physical parameters to electrical signals.
- **IoT devices**: Collect data from remote sources using secure communication channels and encryption.
- **Network devices**: Collect data from network devices using secure communication channels and encryption.

- **Manual data entry**: Implement robust access control mechanisms, authentication, and authorization processes to increase the security of manual data entry.
- **Experiments**: Collect primary data through experiments, such as wet lab experiments like gene sequencing.
- **Observations**: Collect primary data through observations, such as surveys, sensors, or in situ collection.
- **Simulations**: Collect primary data through simulations, such as theoretical models like climate models.
- **Scraping or compiling**: Collect primary data through web scraping, text mining, or compiling data from various sources.
- **Institutionalized data banks**: Collect secondary data from institutionalized data banks, such as census or gene sequences.
- **Published datasets**: Collect secondary data from published datasets, such as those found on Kaggle, GitHub, or UCI Machine Learning Repository.
- **APIs**: Collect secondary data through application programming interfaces (APIs), which allow clients to request data from a website's server.
- **Surveys**: Collect primary data through surveys, which can be online or offline.

Data Acquisition Tools

Tools for gathering, analyzing, and recording data from a variety of sensors, instruments, or devices are software and hardware systems

known as data acquisition tools. **Data Acquisition Tools** are useful in scientific research, industrial automation, engineering, and other domains where data gathering and processing are critical. Few Tools for Acquiring Data are:

- **DriveSpy:** A data collection tool for Windows operating systems created by Digital Intelligence Forensic Solutions.
- **DewesoftX**: A software suite for acquiring and analyzing data that provides strong tools for these tasks.
- **LabVIEW:** A popular software program used in many different industries that offers tools for data collection, processing, and visualization.
- **Catman**: A data acquisition software package that offers tools for data acquisition, analysis, and visualization, and is commonly used in industrial automation and engineering.
- **Matlab**: A software package that provides tools for data acquisition, analysis, and visualization, and is widely used in various industries.
- **FlexPro**: A data acquisition software package that offers tools for data acquisition, analysis, and visualization, and is commonly used in industrial automation and engineering.

Conclusion

In conclusion, Data Acquisition (DAQ) is the crucial first step in building successful machine learning models. It involves gathering high-quality, relevant data to train your models and achieve optimal performance. By following the best practices outlined above, you can

ensure your DAQ process is efficient and effective, laying a strong foundation for your machine learning project

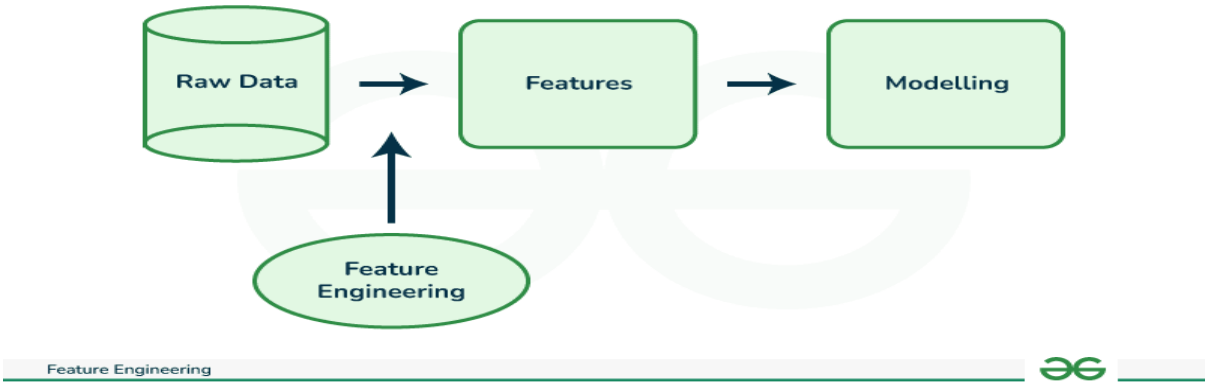
Feature Engineering:

Feature Engineering is the process of creating new features or transforming existing features to improve the performance of a machine-learning model. It involves selecting relevant information from raw data and transforming it into a format that can be easily understood by a model. The goal is to improve model accuracy by providing more meaningful and relevant information.

What is Feature Engineering?

Feature engineering is the process of **transforming raw data into features that are suitable for machine learning models**. In other words, it is the process of selecting, extracting, and transforming the most relevant features from the available data to build more accurate and efficient machine learning models.

The success of machine learning models heavily depends on the quality of the features used to train them. Feature engineering involves a set of techniques that enable us to create new features by combining or transforming the existing ones. These techniques help to highlight the most important patterns and relationships in the data, which in turn helps the machine learning model to learn from the data more effectively.



What is a Feature?

In the context of machine learning, a feature (also known as a variable or attribute) is an individual measurable property or characteristic of a data point that is used as input for a machine learning algorithm.

Features can be numerical, categorical, or text-based, and they represent different aspects of the data that are relevant to the problem at hand.

- For example, in a dataset of housing prices, features could include the number of bedrooms, the square footage, the location, and the age of the property. In a dataset of customer demographics, features could include age, gender, income level, and occupation.
- The choice and quality of features are critical in machine learning, as they can greatly impact the accuracy and performance of the model.

Need for Feature Engineering in Machine Learning?

We engineer features for various reasons, and some of the main reasons include:

- **Improve User Experience:** The primary reason we engineer features is to enhance the user experience of a product or service. By adding new features, we can make the product more intuitive, efficient, and user-friendly, which can increase user satisfaction and engagement.
- **Competitive Advantage:** Another reason we engineer features is to gain a competitive advantage in the marketplace. By offering unique and innovative features, we can differentiate our product from competitors and attract more customers.
- **Meet Customer Needs:** We engineer features to meet the evolving needs of customers. By analyzing user feedback, market trends, and customer behavior, we can identify areas where new features could enhance the product's value and meet customer needs.
- **Increase Revenue:** Features can also be engineered to generate more revenue. For example, a new feature that streamlines the checkout process can increase sales, or a feature that provides additional functionality could lead to more upsells or cross-sells.
- **Future-Proofing:** Engineering features can also be done to future-proof a product or service. By anticipating future trends and potential customer needs, we can develop features that ensure the product remains relevant and useful in the long term.

Processes Involved in Feature Engineering

Feature engineering in Machine learning consists of mainly 5 processes: Feature Creation, Feature Transformation, Feature

Extraction, Feature Selection, and Feature Scaling. It is an iterative process that requires experimentation and testing to find the best combination of features for a given problem. The success of a machine learning model largely depends on the quality of the features used in the model.

1. Feature Creation

Feature Creation is the process of generating new features based on domain knowledge or by observing patterns in the data. It is a form of feature engineering that can significantly improve the performance of a machine-learning model.

Types of Feature Creation:

1. **Domain-Specific:** Creating new features based on domain knowledge, such as creating features based on business rules or industry standards.
2. **Data-Driven:** Creating new features by observing patterns in the data, such as calculating aggregations or creating interaction features.
3. **Synthetic:** Generating new features by combining existing features or synthesizing new data points.

Why Feature Creation?

1. **Improves Model Performance:** By providing additional and more relevant information to the model, feature creation can increase the accuracy and precision of the model.

2. **Increases Model Robustness:** By adding additional features, the model can become more robust to outliers and other anomalies.
3. **Improves Model Interpretability:** By creating new features, it can be easier to understand the model's predictions.
4. **Increases Model Flexibility:** By adding new features, the model can be made more flexible to handle different types of data.

2. Feature Transformation

Feature Transformation is the process of transforming the features into a more suitable representation for the machine learning model. This is done to ensure that the model can effectively learn from the data.

Types of Feature Transformation:

1. **Normalization:** Rescaling the features to have a similar range, such as between 0 and 1, to prevent some features from dominating others.
2. **Scaling:** Scaling is a technique used to transform numerical variables to have a similar scale, so that they can be compared more easily. Rescaling the features to have a similar scale, such as having a standard deviation of 1, to make sure the model considers all features equally.
3. **Encoding:** Transforming categorical features into a numerical representation. Examples are one-hot encoding and label encoding.

4. **Transformation:** Transforming the features using mathematical operations to change the distribution or scale of the features. Examples are logarithmic, square root, and reciprocal transformations.

Why Feature Transformation?

1. **Improves Model Performance:** By transforming the features into a more suitable representation, the model can learn more meaningful patterns in the data.
2. **Increases Model Robustness:** Transforming the features can make the model more robust to outliers and other anomalies.
3. **Improves Computational Efficiency:** The transformed features often require fewer computational resources.
4. **Improves Model Interpretability:** By transforming the features, it can be easier to understand the model's predictions.

3. Feature Extraction

Feature Extraction is the process of creating new features from existing ones to provide more relevant information to the machine learning model. This is done by transforming, combining, or aggregating existing features.

Types of Feature Extraction:

1. **Dimensionality Reduction:** Reducing the number of features by transforming the data into a lower-dimensional space while retaining important information. Examples are PCA and t-SNE.

2. **Feature Combination:** Combining two or more existing features to create a new one. For example, the interaction between two features.
3. **Feature Aggregation:** Aggregating features to create a new one. For example, calculating the mean, sum, or count of a set of features.
4. **Feature Transformation:** Transforming existing features into a new representation. For example, log transformation of a feature with a skewed distribution.

Why Feature Extraction?

1. **Improves Model Performance:** By creating new and more relevant features, the model can learn more meaningful patterns in the data.
2. **Reduces Overfitting:** By reducing the dimensionality of the data, the model is less likely to overfit the training data.
3. **Improves Computational Efficiency:** The transformed features often require fewer computational resources.
4. **Improves Model Interpretability:** By creating new features, it can be easier to understand the model's predictions.

4. Feature Selection

Feature Selection is the process of selecting a subset of relevant features from the dataset to be used in a machine-learning model. It is an important step in the feature engineering process as it can have a significant impact on the model's performance.

Types of Feature Selection:

1. **Filter Method:** Based on the statistical measure of the relationship between the feature and the target variable. Features with a high correlation are selected.
2. **Wrapper Method:** Based on the evaluation of the feature subset using a specific machine learning algorithm. The feature subset that results in the best performance is selected.
3. **Embedded Method:** Based on the feature selection as part of the training process of the machine learning algorithm.

Why Feature Selection?

1. **Reduces Overfitting:** By using only the most relevant features, the model can generalize better to new data.
2. **Improves Model Performance:** Selecting the right features can improve the accuracy, precision, and recall of the model.
3. **Decreases Computational Costs:** A smaller number of features requires less computation and storage resources.
4. **Improves Interpretability:** By reducing the number of features, it is easier to understand and interpret the results of the model.

5. Feature Scaling

Feature Scaling is the process of transforming the features so that they have a similar scale. This is important in machine learning because the scale of the features can affect the performance of the model.

Types of Feature Scaling:

1. **Min-Max Scaling:** Rescaling the features to a specific range, such as between 0 and 1, by subtracting the minimum value and dividing by the range.
2. **Standard Scaling:** Rescaling the features to have a mean of 0 and a standard deviation of 1 by subtracting the mean and dividing by the standard deviation.
3. **Robust Scaling:** Rescaling the features to be robust to outliers by dividing them by the interquartile range.

Why Feature Scaling?

1. **Improves Model Performance:** By transforming the features to have a similar scale, the model can learn from all features equally and avoid being dominated by a few large features.
2. **Increases Model Robustness:** By transforming the features to be robust to outliers, the model can become more robust to anomalies.
3. **Improves Computational Efficiency:** Many machine learning algorithms, such as k-nearest neighbors, are sensitive to the scale of the features and perform better with scaled features.
4. **Improves Model Interpretability:** By transforming the features to have a similar scale, it can be easier to understand the model's predictions.

What are the Steps in Feature Engineering?

The steps for feature engineering vary per different Ml engineers and data scientists. Some of the common steps that are involved in most machine-learning algorithms are:

1. **Data Cleansing**
    - Data cleansing (also known as data cleaning or data scrubbing) involves identifying and removing or correcting any errors or inconsistencies in the dataset. This step is important to ensure that the data is accurate and reliable.
2. **Data Transformation**
3. **Feature Extraction**
4. **Feature Selection**
    - Feature selection involves selecting the most relevant features from the dataset for use in machine learning. This can include techniques like correlation analysis, mutual information, and stepwise regression.
5. **Feature Iteration**
    - Feature iteration involves refining and improving the features based on the performance of the machine learning model. This can include techniques like adding new features, removing redundant features and transforming features in different ways.

Overall, the goal of feature engineering is to create a set of informative and relevant features that can be used to train a machine learning model and improve its accuracy and performance. The specific steps involved in the process may vary

depending on the type of data and the specific machine-learning problem at hand.

Techniques Used in Feature Engineering

Feature engineering is the process of transforming raw data into features that are suitable for machine learning models. There are various techniques that can be used in feature engineering to create new features by combining or transforming the existing ones. The following are some of the commonly used feature engineering techniques:

One-Hot Encoding

One-hot encoding is a technique used to transform categorical variables into numerical values that can be used by machine learning models. In this technique, each category is transformed into a binary value indicating its presence or absence. For example, consider a categorical variable "Colour" with three categories: Red, Green, and Blue. One-hot encoding would transform this variable into three binary variables: Colour_Red, Colour_Green, and Colour_Blue, where the value of each variable would be 1 if the corresponding category is present and 0 otherwise.

Binning

Binning is a technique used to transform continuous variables into categorical variables. In this technique, the range of values of the continuous variable is divided into several bins, and each bin is assigned a categorical value. For example, consider a continuous variable "Age" with values ranging from 18 to 80. Binning would

divide this variable into several age groups such as 18-25, 26-35, 36-50, and 51-80, and assign a categorical value to each age group.

Scaling

The most common scaling techniques are standardization and normalization. Standardization scales the variable so that it has zero mean and unit variance. Normalization scales the variable so that it has a range of values between 0 and 1.

FeatureSplit

Feature splitting is a powerful technique used in feature engineering to improve the performance of machine learning models. It involves dividing single features into multiple sub-features or groups based on specific criteria. This process unlocks valuable insights and enhances the model's ability to capture complex relationships and patterns within the data.

TextDataPreprocessing

Text data requires special preprocessing techniques before it can be used by machine learning models. Text preprocessing involves removing stop words, stemming, lemmatization, and vectorization. Stop words are common words that do not add much meaning to the text, such as "the" and "and". Stemming involves reducing words to their root form, such as converting "running" to "run". Lemmatization is similar to stemming, but it reduces words to their base form, such as converting "running" to "run". Vectorization involves transforming text data into numerical vectors that can be used by machine learning models.

Feature Engineering Tools

There are several tools available for feature engineering. Here are some popular ones:

1. Featuretools

Featuretools is a Python library that enables automatic feature engineering for structured data. It can extract features from multiple tables, including relational databases and CSV files, and generate new features based on user-defined primitives. Some of its features include:

- Automated feature engineering using machine learning algorithms.
- Support for handling time-dependent data.
- Integration with popular Python libraries, such as pandas and scikit-learn.
- Visualization tools for exploring and analyzing the generated features.
- Extensive documentation and tutorials for getting started.

2. TPOT

TPOT (Tree-based Pipeline Optimization Tool) is an automated machine learning tool that includes feature engineering as one of its components. It uses genetic programming to search for the best combination of features and machine learning algorithms for a given dataset. Some of its features include:

- Automatic feature selection and transformation.

- Support for multiple types of machine learning models, including regression, classification, and clustering.
- Ability to handle missing data and categorical variables.
- Integration with popular Python libraries, such as scikit-learn and pandas.
- Interactive visualization of the generated pipelines.

3. DataRobot

DataRobot is a machine learning automation platform that includes feature engineering as one of its capabilities. It uses automated machine learning techniques to generate new features and select the best combination of features and models for a given dataset. Some of its features include:

- Automatic feature engineering using machine learning algorithms.
- Support for handling time-dependent and text data.
- Integration with popular Python libraries, such as pandas and scikit-learn.
- Interactive visualization of the generated models and features.
- Collaboration tools for teams working on machine learning projects.

4. Alteryx

Alteryx is a data preparation and automation tool that includes feature engineering as one of its features. It provides a visual interface for creating data pipelines that can extract, transform, and generate features from multiple data sources. Some of its features include:

- Support for handling structured and unstructured data.
- Integration with popular data sources, such as Excel and databases.
- Pre-built tools for feature extraction and transformation.
- Support for custom scripting and code integration.
- Collaboration and sharing tools for teams working on data projects.

5. H2O.ai

H2O.ai is an open-source machine learning platform that includes feature engineering as one of its capabilities. It provides a range of automated feature engineering techniques, such as feature scaling, imputation, and encoding, as well as manual feature engineering capabilities for more advanced users. Some of its features include:

- Automatic and manual feature engineering options.
- Support for structured and unstructured data, including text and image data.
- Integration with popular data sources, such as CSV files and databases.
- Interactive visualization of the generated features and models.
- Collaboration and sharing tools for teams working on machine learning projects.

Overall, these tools can help streamline and automate the feature engineering process, making it easier and faster to create informative and relevant features for machine learning models

Data Representation

Introduction:

In the realms of signal processing and machine learning, the representation of data in a numerical format is crucial for analysis, processing, and modelling. Depending on the dimensionality and complexity of the data, it can be represented in different numerical forms. The most common forms of data representation include scalar values, vectors, matrices, and tensors.

Definitions:

- **Scalar:** A scalar is a single numerical value. It has magnitude but no direction. For instance, in mathematical terms, the number 5 or -3.2 are scalar values.

- **Vector:** A vector is an ordered list of numbers. It can be visualized as a line segment in space that has both magnitude and direction. Mathematically, a vector is typically represented as a column (or sometimes row) of numbers (aka 1-D data)

- **Matrix:** A matrix is a **two-dimensional array of numbers**. It can be visualized as a rectangular grid of numbers. A matrix has rows and columns, and its shape is often described by the number of rows by the number of columns, e.g., a 3x2 matrix has 3 rows and 2 columns.

- **Tensor:** A tensor is a **multi-dimensional array of numbers**. While a scalar is 0-dimensional, a vector is 1-dimensional, and a matrix is 2-dimensional, a tensor can be 3-dimensional or more. For instance, a 3-dimensional tensor can be visualized as a cube of numbers.

Examples:

- **Scalar:** An example of a scalar is the **current temperature**. If it's 22°C right now, that's a single numerical value.

- **Vector:** An example of a vector could be the **average high temperatures forecasted for the next week**. For instance, if the forecasted high temperatures for the next seven days are 22°C, 23°C, 24°C, 25°C, 26°C, 27°C, and 28°C, then the vector representation might be: [22, 23, 24, 25, 26, 27, 28]

- **Matrix:** An example of a matrix is a **grayscale image.** The pixels of the image are represented as values between 0 (black) and 255 (white). The image's resolution, say 100x100 pixels, will determine the size of the matrix. Each entry in the matrix corresponds to the grayscale value of a pixel.

- **Tensor:** A tensor example is a **colored image**. In the most common format, an image has three color channels: Red, Green, and Blue (RGB). Each channel can be thought of as a matrix (like the grayscale image), and the three matrices combined form a 3-dimensional tensor. If the image is of resolution 100x100 pixels, the tensor's shape would be 100x100x3, with each slice of size 100x100 representing one of the RGB channels.

Summary & Conclusion

- **Scalar:** A single numerical value.

- **Vector:** A 1D array of values, often representing a series or sequence of numbers.

- **Matrix:** A 2D array of values, typically visualized as a grid or table of numbers.

- **Tensor:** An array of values with 3 or more dimensions, commonly used in applications requiring multi-channel or multi-modal data.

Data representation is crucial in domains like signal processing and machine learning. Gaining a practical understanding of how these data structures manifest in real-world engineering scenarios enables one to look beyond the terminology. Rather than being daunted by the

jargon, professionals can leverage these data formats as powerful tools for both analysis and synthesis in their respective fields.

Model Selection

Introduction

Model selection is an essential phase in the development of powerful and precise predictive models in the field of machine learning. Model selection is the process of deciding which algorithm and model architecture is best suited for a particular task or dataset. It entails contrasting various models, assessing their efficacy, and choosing the one that most effectively addresses the issue at hand.

The choice of an appropriate machine learning model is crucial since there are various levels of complexity, underlying assumptions, and capabilities among them. A model's ability to generalize to new, untested data may not be as strong as its ability to perform effectively on a single dataset or problem. Finding a perfect balance between the complexity of models & generalization is therefore key to model selection.

Choosing a model often entails a number of processes. The first step in this process is to define a suitable evaluation metric that matches the objectives of the particular situation. According to the nature of the issue, this statistic may refer to precision, recall, accuracy, F1-score, or any other relevant measure.

The selection of numerous candidate models is then made in accordance with the problem at hand and the data that are accessible. These models might be as straightforward as decision trees or linear regression or as sophisticated as deep neural networks, random forests, or support vector machines. During the selection process, it is important to take into account the assumptions, constraints, and hyperparameters that are unique to each model.

Using a suitable methodology, such as cross-validation, the candidate models are trained and evaluated after being selected. To do this, the available data must be divided into validation and training sets, with each model fitting on the training set before being evaluated on the

validation set. The models are compared using their performance metrics, then the model with the highest performance is chosen.
Model selection is a continuous process, though. In order to make wise selections, it frequently calls for an iterative process that involves testing several models and hyperparameters. The models are improved through this iterative process, which also aids in choosing the ideal mix of algorithms & hyperparameters.

Model Selection

In machine learning, the process of selecting the top model or algorithm from a list of potential models to address a certain issue is referred to as model selection. It entails assessing and contrasting various models according to how well they function and choosing the one that reaches the highest level of accuracy or prediction power.
Because different models have varied levels of complexity, underlying assumptions, and capabilities, model selection is a crucial stage in the machine-learning pipeline. Finding a model that fits the training set of data well and generalizes well to new data is the objective. While a model that is too complex may overfit the data and be unable to generalize, a model that is too simple could underfit the data and do poorly in terms of prediction.
The following steps are frequently included in the model selection process:

- **Problem formulation:** Clearly express the issue at hand, including the kind of predictions or task that you'd like the model to carry out (for example, classification, regression, or clustering).
- **Candidate model selection:** Pick a group of models that are appropriate for the issue at hand. These models can include straightforward methods like decision trees or linear regression as well as more sophisticated ones like deep neural networks, random forests, or support vector machines.
- **Performance evaluation:** Establish measures for measuring how well each model performs. Common measurements include area under the receiver's operating characteristic curve (AUC-

ROC), recall, F1-score, mean squared error, and accuracy, precision, and recall. The type of problem and the particular requirements will determine which metrics are used.

- **Training and evaluation:** Each candidate model should be trained using a subset of the available data (the training set), and its performance should be assessed using a different subset (the validation set or via cross-validation). The established evaluation measures are used to gauge the model's effectiveness.
- **Model comparison:** Evaluate the performance of various models and determine which one performs best on the validation set. Take into account elements like data handling capabilities, interpretability, computational difficulty, and accuracy.
- **Hyperparameter tuning:** Before training, many models require that certain hyperparameters, such as the learning rate, regularisation strength, or the number of layers that are hidden in a neural network, be configured. Use methods like grid search, random search, and Bayesian optimization to identify these hyperparameters' ideal values.
- **Final model selection:** After the models have been analyzed and fine-tuned, pick the model that performs the best. Then, this model can be used to make predictions based on fresh, unforeseen data.

Model Selection in machine learning:

Model selection in machine learning is the process of selecting the best algorithm and model architecture for a specific job or dataset. It entails assessing and contrasting various models to identify the one that best fits the data & produces the best results. Model complexity, data handling capabilities, and generalizability to new examples are all taken into account while choosing a model. Models are evaluated and contrasted using methods like cross-validation, and grid search, as well as indicators like accuracy and mean squared error. Finding a model that balances complexity and performance to produce reliable predictions and strong generalization abilities is the aim of model selection.

There are numerous important considerations to bear in mind while selecting a model for machine learning. These factors assist in

ensuring that the chosen model is effective in solving the issue at its core and has an opportunity for outstanding performance. Here are some crucial things to remember:

- **The complexity of the issue:** Determine how complex the issue you're trying to resolve is. Simple models might effectively solve some issues, but more complicated models can be necessary to fully represent complex relationships in the data. Take into account the size of the dataset, the complexity of the input features, and any potential for non-linear connections.
- **Data Availability & Quality:** Consider the accessibility and caliber of the data you already have. Using complicated models with a lot of parameters on a limited dataset may result in overfitting. Such situations may call for simpler models with fewer parameters. Take into account missing data, outliers, and noise as well as how various models respond to these difficulties.
- **Interpretability:** Consider whether the model's interpretability is crucial in your particular setting. Some models, like decision trees or linear regression, offer interpretability by giving precise insights into the correlations between the input data and the desired outcome. Complex models, such as neural networks, may perform better but offer less interpretability.
- **Model Assumptions:** Recognise the presumptions that various models make. For instance, although decision trees assume piecewise constant relationships, linear regression assumes a linear relationship between the input characteristics and the target variable. Make sure the model you choose is consistent with the fundamental presumptions underpinning the data and the issue.
- **Scalability and Efficiency:** If you're working with massive datasets or real-time applications, take the model's scalability and computing efficiency into consideration. Deep neural networks and support vector machines are two examples of models that could need a lot of time and computing power to train.
- **Regularisation and Generalisation:** Assess the model's capacity to apply to fresh, untested data. By including penalty terms to the objective function of the model, regularisation

approaches like L1 or L2 regularisation can help prevent overfitting. When the training data is sparse, regularised models may perform better in terms of generalization.

- **Domain Expertise:** Consider your expertise and domain knowledge. On the basis of previous knowledge of the data or particular features of the domain, consider if particular models are appropriate for the task. Models that are more likely to capture important patterns can be found by using domain expertise to direct the selection process.
- **Resource Constraints:** Take into account any resource limitations you may have, such as constrained memory space, processing speed, or time. Make that the chosen model can be successfully implemented using the resources at hand. Some models require significant resources during training or inference.
- **Ensemble Methods:** Examine the potential advantages of ensemble methods, which integrate the results of various models in order to perform more effectively. By utilizing the diversity of several models' predictions, ensemble approaches, such as bagging, boosting, and stacking, frequently outperform individual models.
- **Evaluation and Experimentation:** experimentation and assessment of several models should be done thoroughly. Utilize the right evaluation criteria and statistical tests to compare their performance. To evaluate the models' performance on unknown data and reduce the danger of overfitting, use hold-out or cross-validation.

Model Selection Techniques

Model selection in machine learning can be done using a variety of methods and tactics. These methods assist in comparing and assessing many models to determine which is best suited to solve a certain issue. Here are some methods for selecting models that are frequently used:

- **Train-Test Split:** With this strategy, the available data is divided into two sets: a training set & a separate test set. The models are evaluated using a predetermined evaluation metric on the test set after being trained on the training set. This

method offers a quick and easy way to evaluate a model's performance using hypothetical data.

- **Cross-Validation:** A resampling approach called cross-validation **divides the data into various groups or folds**. Several folds are used as the test set & the rest folds as the training set, and the models undergo training and evaluation on each fold separately. Lowering the variance in the evaluation makes it easier to generate an accurate assessment of the model's performance. Cross-validation techniques that are frequently used include leave-one-out, stratified, and k-fold cross-validation.
- **Grid Search:** Hyperparameter tuning is done using the grid search technique. In order to do this, a grid containing hyperparameter values must be defined, and all potential hyperparameter combinations must be thoroughly searched. For each combination, the models are trained, assessed, and their performances are contrasted. Finding the ideal hyperparameter settings to optimize the model's performance is made easier by grid search.
- **Random Search:** A set distribution for hyperparameter values is sampled at random as part of the random search hyperparameter tuning technique. In contrast to grid search, which considers every potential combination, random search only investigates a portion of the hyperparameter field. When a thorough search is not possible due to the size of the search space, this strategy can be helpful.
- **Bayesian optimization:** A more sophisticated method of hyperparameter tweaking, Bayesian optimization. It models the relationship between the performance of the model and the hyperparameters using a probabilistic model. It intelligently chooses which set of hyperparameters to investigate next by updating the probabilistic model and iteratively assessing the model's performance. When the search space is big and expensive to examine, Bayesian optimization is especially effective.
- **Model averaging:** This technique combines forecasts from various models to get a single prediction. For regression issues, this can be accomplished by averaging the predictions, while for

classification problems, voting or weighted voting systems can be used. Model averaging can increase overall prediction accuracy by lowering the bias and variation of individual models.

- **Information Criteria:** Information criteria offer a numerical assessment of the trade-off between model complexity and goodness of fit. Examples include the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). These criteria discourage the use of too complicated models and encourage the adoption of simpler models that adequately explain the data.
- **Domain Expertise & Prior Knowledge:** Prior understanding of the problem and the data, as well as domain expertise, can have a significant impact on model choice. The models that are more suitable given the specifics of the problem and the details of the data may be known by subject matter experts.
- **Model Performance Comparison:** Using the right assessment measures, it is vital to evaluate the performance of various models. Depending on the issue at hand, these measurements could include F1-score, mean squared error, accuracy, precision, recall, or the area beneath the receiver's operating characteristic curve (AUC-ROC). The best-performing model can be found by comparing many models.

Summary

The important machine learning stage of model selection entails selecting the best model and algorithm for a certain task. To make precise predictions on unknown data, it is crucial to find a balance between model complexity & generalization. Model selection involves selecting potential candidates, assessing each model's performance, and selecting the model with the best results.

Assessing the problem's complexity, data quality and availability, interpretability, model assumptions, scalability, efficiency, regularisation, domain knowledge, resource restrictions, and the possible advantages of ensemble approaches are all factors that should be taken into account when choosing a model. These factors aid in

ensuring that the chosen model complies with the limits and needs of the issue.
There are many methods for choosing a model, such as train-test split, cross-validation, grid searches, random search, Bayesian optimization, model averaging, information criteria, expertise in the domain, and model performance comparison. These methods make it possible to thoroughly assess, tune hyperparameters, and compare various models to get the best fit.

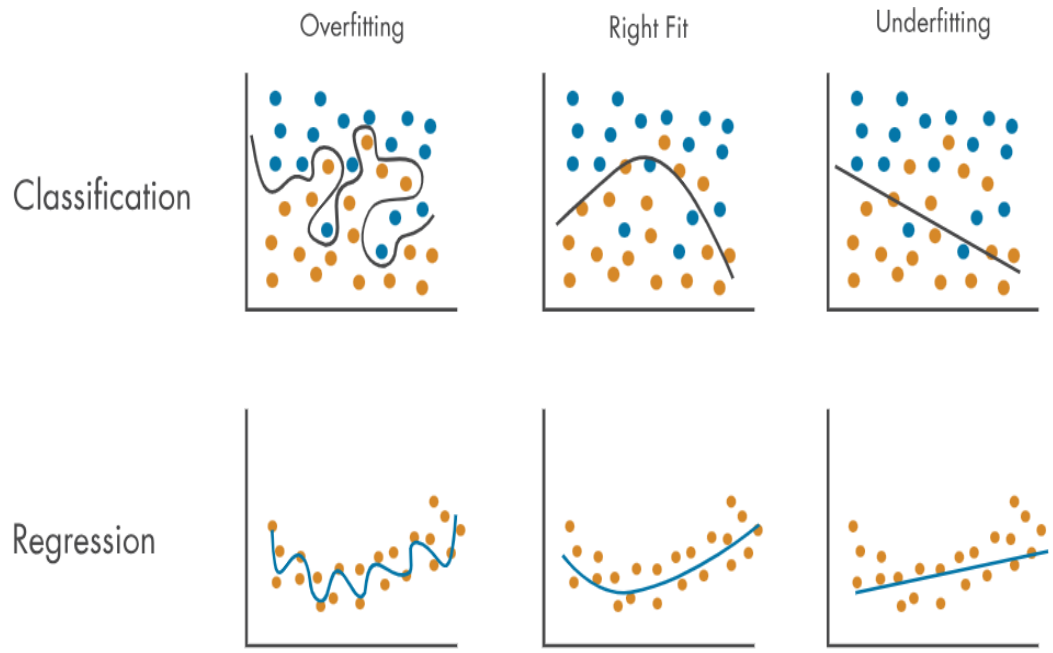
Model Evaluation

What is Model Evaluation?

Model evaluation is the process of using different evaluation metrics to understand a machine learning model's performance, as well as its strengths and weaknesses.

Why is Evaluation necessary for a successful model?

Evaluation is necessary for ensuring that machine learning models are reliable, generalizable, and capable of making accurate predictions on new, unseen data, which is crucial for their successful deployment in real-world applications. Overfitting and underfitting are the two biggest causes of poor performance of machine learning algorithms.



Overfitting: *Occurs when the model is* ***so closely*** *aligned to the training data that it does not know how to respond to new data.*

Underfitting*: Occurs when the model* ***cannot adequately capture*** *the underlying structure of the data.*

Right Fit: *Occurs when both the training data error and the test data are minimal*

Error	Overfitting	Right Fit	Underfitting
Training	Low	Low	High
Test	High	Low	High

Error Risks in the models

Evaluation Metrics

There are different metrics for the tasks of classification, regression, ranking, clustering, topic modeling, etc. Some of the metrics are as follows:

1. *Classification Metrics (accuracy, precision, recall, F1-score, ROC, AUC, ...)*
2. *Regression Metrics (MSE, MAE, R2)*
3. *Ranking Metrics (MRR, DCG, NDCG)*
4. *Statistical Metrics (Correlation)*
5. *Computer Vision Metrics (PSNR, SSIM, IoU)*
6. *NLP Metrics (Perplexity, BLEU score)*
7. *Deep Learning Related Metrics (Inception score, Frechet Inception distance)*

→ *Today, we will talk about Classification Metrics.*

1. Classification Metrics

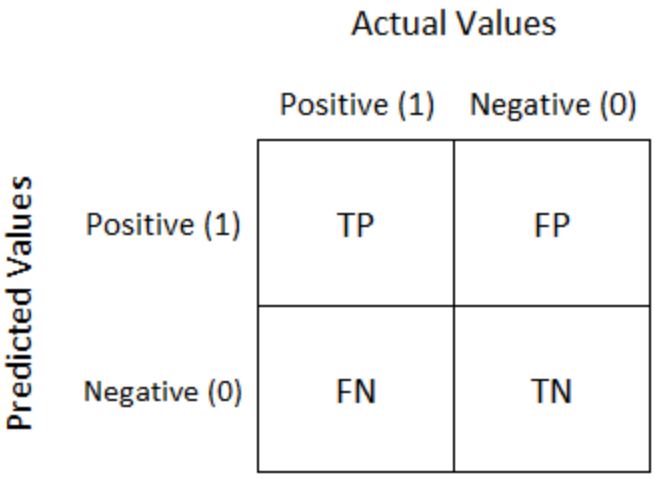
When our target is categorical, we are dealing with a classification problem. The choice of the most appropriate metrics depends on different aspects, such as the characteristics of the dataset, whether it's imbalanced or not, and the goals of the analysis.

Confusion Matrix

A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known.

I can summarize as before and after happenings. How?

As you see we have 2 main situations. Predicted (Before), Actual Values (After).

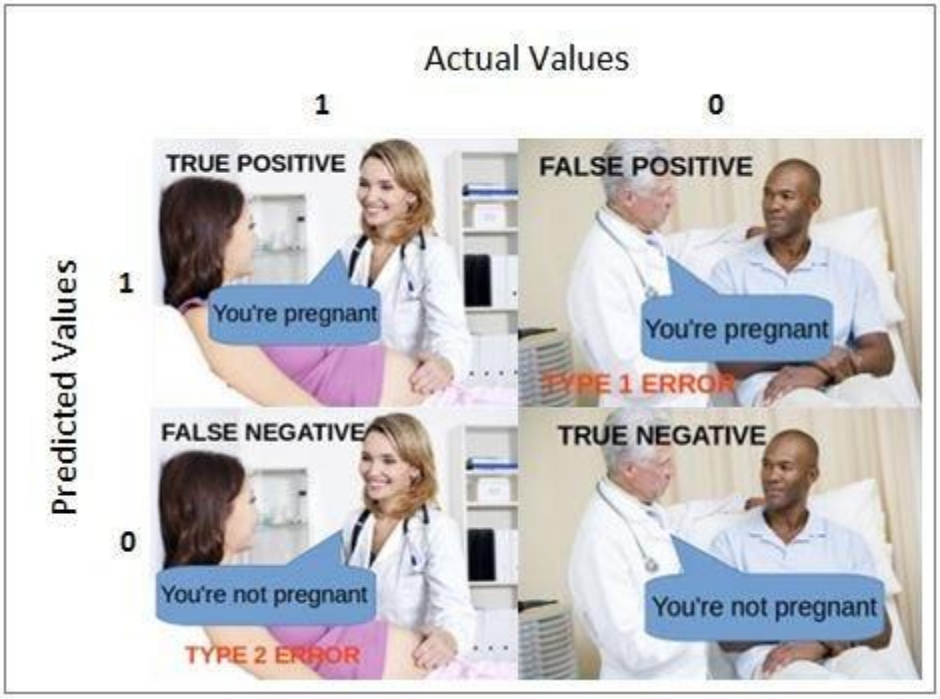


Predicted and Actual Values

1. ***Predicted*:** Negative & ***Actual Value***: Positive → Your predicted False (FN)
2. ***Predicted*:** Negative & ***Actual Value***: Negative → Your predicted True(TN)
3. ***Predicted*:** Pozitive & ***Actual Value***: Positive → Your predicted True (TP)

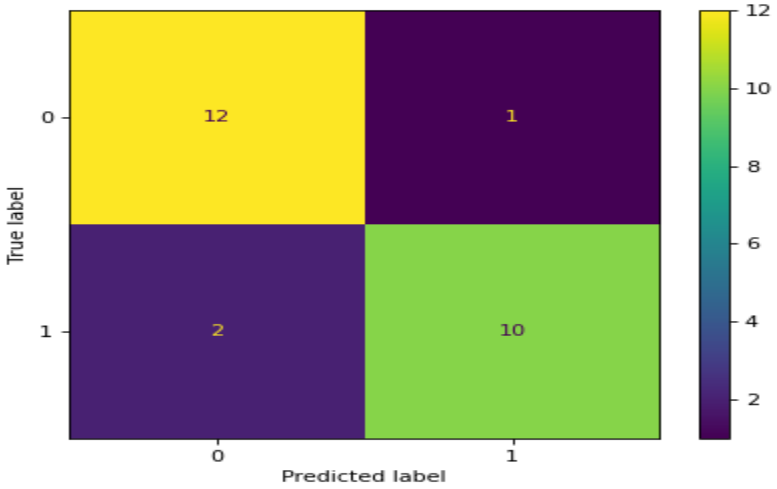
4. ***Predicted*:** Pozitive & ***Actual Value***: Negative→ Your predicted False (FP)

These four scenarios are illustrated in the following figure.



Four possible combinations of reality and our binary pregnancy test results

Example Label for Accuracy, Precision, and Recall



True Positive (TP) =10

True Negative (TN)=12

False Positive (FP)=1

False Negative (FN)=2

Accuracy

Accuracy is one metric for evaluating classification models. Formally accuracy could be defined as the number of correct predictions to a total number of predictions.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Accuracy = \frac{10 + 12}{10 + 12 + 1 + 2} = 88\%$$

Precision

Precision is a measure of the accuracy.

$$\textbf{Precision} = \frac{\textbf{TP}}{\textbf{TP + FP}}$$

$$\text{Precision} = \frac{10}{10+1} = 91\%$$

Recall

Recall is the true positive rate

$$\text{Recall} = \frac{\text{TP}}{\text{TP + FN}}$$

$$Recall = \frac{10}{10+2} = 83\%$$

F1 Score

F1 score is a machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model.

The accuracy metric computes how many times a model made a correct prediction across the entire dataset.

$$\text{F1} = \frac{\text{2 * Precision * Recall}}{\text{Precision + Recall}}$$

$$F1 = \frac{2 * 0.91 * 0.83}{0.91 + 0.83} = 0.87$$

In some scenarios, precision and recall may have varying levels of importance depending on the specific requirements of the application. The F1 score, which balances both precision and recall, may not perfectly capture the relative importance of these metrics for a given task. F1 score or seeing the PR or ROC curve can help.

ROC

ROC curve provides a comprehensive view of a model's ability to discriminate between classes, especially in binary classification tasks. It helps in understanding the trade-offs between sensitivity and specificity at different decision thresholds, and the AUC offers a single metric for summarizing the overall performance of the model.

- True Positive Rate (Recall)
- False Positive Rate (FPR)

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

$$FPR = \frac{1}{1 + 12} = 0.077$$

Model prediction

What Is Predictive Modeling?

In short, predictive modeling is a statistical technique using machine learning and data mining to predict and forecast likely future outcomes with the aid of historical and existing data. It works by analyzing current and historical data and projecting what it learns on a model generated to forecast likely outcomes. Predictive modeling can be used to predict just about anything, from TV ratings and a customer's next purchase to credit risks and corporate earnings.

A predictive model is not fixed; it is validated or revised regularly to incorporate changes in the underlying data. In other words, it's not a one-and-done prediction. Predictive models make assumptions based on what has happened in the past and what is happening now. If incoming, new data shows changes in what is happening now, the impact on the likely future outcome must be recalculated, too. For example, a software company could model historical sales data against marketing expenditures across multiple regions to create a model for future revenue based on the impact of the marketing spend.

Most predictive models work fast and often complete their calculations in real time. That's why banks and retailers can, for example, calculate the risk of an online mortgage or credit card application and accept or decline the request almost instantly based on that prediction.

Some predictive models are more complex, such as those used in computational biology and quantum computing; the resulting outputs take longer to compute than a credit card application but are done much more quickly than was possible in the past thanks to advances in technological capabilities, including computing power.

Top 5 Types of Predictive Models

Fortunately, predictive models don't have to be created from scratch for every application. Predictive analytics tools use a variety of vetted models and algorithms that can be applied to a wide spread of use cases.

Predictive modeling techniques have been perfected over time. As we add more data, more muscular computing, AI and machine learning and see overall advancements in analytics, we're able to do more with these models.

The top five predictive analytics models are:

1. **Classification model:** Considered the simplest model, it categorizes data for simple and direct query response. An example use case would be to answer the question "Is this a fraudulent transaction?"
2. **Clustering model:** This model nests data together by common attributes. It works by grouping things or people with shared characteristics or behaviors and plans strategies for each group at a larger scale. An example is in determining credit risk for a loan applicant based on what other people in the same or a similar situation did in the past.
3. **Forecast model:** This is a very popular model, and it works on anything with a numerical value based on learning from historical data. For example, in answering how much lettuce a restaurant should order next week or how many calls a customer support agent should be able to handle per day or week, the system looks back to historical data.
4. **Outliers model:** This model works by analyzing abnormal or outlying data points. For example, a bank might use an outlier model to identify fraud by asking whether a transaction is outside of the customer's normal buying habits or whether an expense in a given category is normal or not. For example, a $1,000 credit card charge for a washer and dryer in the cardholder's preferred big box store would not be alarming, but $1,000 spent on designer clothing in a

location where the customer has never charged other items might be indicative of a breached account.

5. **Time series model:** This model evaluates a sequence of data points based on time. For example, the number of stroke patients admitted to the hospital in the last four months is used to predict how many patients the hospital might expect to admit next week, next month or the rest of the year. A single metric measured and compared over time is thus more meaningful than a simple average.

Common Predictive Algorithms

Predictive algorithms use one of two things: machine learning or deep learning. Both are subsets of artificial intelligence (AI). Machine learning (ML) involves structured data, such as spreadsheet or machine data. Deep learning (DL) deals with unstructured data such as video, audio, text, social media posts and images—essentially the stuff that humans communicate with that are not numbers or metric reads.

Some of the more common predictive algorithms are:

1. **Random Forest:** This algorithm is derived from a combination of decision trees, none of which are related, and can use both classification and regression to classify vast amounts of data.
2. **Generalized Linear Model (GLM) for Two Values:** This algorithm narrows down the list of variables to find "best fit." It can work out tipping points and change data capture and other influences, such as categorical predictors, to determine the "best fit" outcome, thereby overcoming drawbacks in other models, such as a regular linear regression.
3. **Gradient Boosted Model:** This algorithm also uses several combined decision trees, but unlike Random Forest, the trees are related. It builds out one tree at a time, thus enabling the next tree to correct flaws in the previous tree. It's often used in rankings, such as on search engine outputs.
4. **K-Means:** A popular and fast algorithm, K-Means groups data points by similarities and so is often used for the clustering model. It

can quickly render things like personalized retail offers to individuals within a huge group, such as a million or more customers with a similar liking of lined red wool coats.

5. **Prophet:** This algorithm is used in time-series or forecast models for capacity planning, such as for inventory needs, sales quotas and resource allocations. It is highly flexible and can easily accommodate heuristics and an array of useful assumptions.

Search and learning:

Whenever someone performs a search, they expect to be greeted with results relevant to their requirements. However, traditional search techniques like "BM25 retrieval" return results based on how many times the phrase searched appears in a document.

While such techniques perform well to an extent, they fail to take into account the users' unique preferences. This is where machine learning (ML) techniques come into play for helping deliver personalized results, particularly within the realm of federated search.

Maximizing Federated Search Relevance with ML Techniques

Unlike conventional methods, ML-driven federated search delves deeper into the intricacies of user interactions, considering many factors beyond keyword frequency.

This enables the system to discern patterns, user intent, and contextual relevance, therefore delivering a more personalized and tailored search experience. The following ML techniques help facilitate this:

Vector and Semantic Search

Vector search is a technique that leverages mathematical representations to understand and organize complex relationships between words and

concepts, enhancing search accuracy. While semantic search focuses on interpreting the meaning of words and phrases within the context, providing a nuanced understanding of the users' context.

These approaches excel in handling synonyms, misspellings, and variations in language, contributing to a higher level of search accuracy. They also boost personalization based on the users' history and preferences, something which we could see lacking in the traditional approaches.

Query Understanding

Query Understanding involves analyzing user queries to grasp their intent, context, and semantics. It employs natural language processing (NLP) to interpret user input, discerning synonyms, and user-specific language, thus enhancing search engines' ability to deliver more accurate and relevant results.

Using ML algorithms also enables the search engines to "Query rephrasing," which essentially refers to suggesting alternate queries that would retrieve better results. Another technique, known as "Query expansion" helps expand the query to add related terms to the query and broadens its scope.

And voila! The search results powered by query understanding are much more thorough.

Neural Reranking

"Ranking" is often done based on how many times the search keyword appears in a document. However, neural reranking allows you to tune the search results so the top result is the most relevant one instead.

This is done by leveraging algorithms like k-nearest-neighbor (k-NN) for exact matches and approximate nearest-neighbor (ANN) for faster but slightly less accurate matches. Another benefit of neural reranking is that it can yield great results in zero-shot informational retrieval (IR) models ,i.e., models without much training.

Now if you were to implement these ML-based techniques for better search results, how would you determine if they're performing as expected? By finding out their impact on recall and precision. Let's dig deeper.

How do ML Techniques Enhance Recall & Precision in Search Results?

Precision refers to how accurate the search results are, while recall refers to the number of results returned. ML algorithms can ensure that the no relevant results are skipped over, and simultaneously rank the most relevant matches higher to reduce irrelevancy.

The following ML techniques help boost precision and recall:

- **Word Embeddings**: Embeddings are dense vector representations that capture semantic relationships and similarities between data points. They boost recall by capturing nuanced similarities in the data, allowing models to return better results.
- **Cross-Encoders:** These are neural networks that analyze pairs of things, such as questions and answers to determine how similar they are. The model gives a score to show how much the two things are connected or similar. This helps boost precision.
- **User History:** Taking the users' history into account helps search engines narrow what they might be looking for. This historical data can be used to fine-tune the search results and provide more personalized and relevant answers.

Therefore, integrating ML techniques into your federated search solutions is a great way to amp up search relevancy. Search Unify has been at it for quite some time! Keep reading to know more.

Delivering Search Excellence with Search Unify's ML Techniques

Search Unify uses three types of ML-powered search techniques to boost **recall** as explained below.

Lexical Search

Lexical refers to the vocabulary or words used in a language, such as their structure and meaning. Lexical search involves searching for specific words or terms within a dataset or a corpus of text. It focuses on finding documents or information that contain the exact words or phrases specified in the search query.

Neural Search

Neural search leverages ML models known as neural networks to improve the efficiency and relevance of search results. It leverages advanced NLP techniques to understand the context, semantics, and relationships between words. This helps provide more accurate and contextually relevant search results by understanding the meaning behind the queries.

Hybrid Search

Hybrid search combines multiple search approaches or technologies to enhance the overall search experience. It often involves integrating traditional search methods with newer technologies like ML or artificial intelligence (AI).

For example, it might use lexical search for precise keyword matching and neural search for understanding context and providing more nuanced results. Search Unify uses this combination to provide a more comprehensive and accurate search experience.

To improve **precision**, Search Unify federated search leverages the following techniques.

Auto Boosting

This helps optimize search precision by adjusting the importance of different features, ensuring relevant information is prioritized based on user interactions and feedback.

Persona-based Results

ML algorithms can track and take into account the user personas and tailor the results to user-specific profiles, considering individual preferences and behavior to deliver more accurate and personalized information.

Cross Encoders

Cross encoders utilize neural networks to analyze relationships between different pieces of content, facilitating a deeper understanding of context and relevance for more accurate search results.

Data Set:

Mahine Learning is at the peak of its popularity today. Despite this, a lot of decision-makers are in the dark about what exactly is needed to design, train, and successfully deploy a machine learning algorithm. The details about collecting the data, building a dataset, and annotation specifics are neglected as supportive tasks.

However, reality shows that working with datasets is the most time-consuming and laborious part of any AI project, sometimes taking up to 70% of the time overall. Moreover, building up a high-quality machine learning dataset requires experienced, trained professionals who know what to do with the actual data that can be collected.

Let's start from the beginning by defining what a dataset for machine learning is and why you need to pay more attention to it.

What Is a Dataset in Machine Learning and Why Is It Essential for Your AI Model?

According to the Oxford Dictionary, a dataset definition in machine learning is “a collection of data that is treated as a single unit by a computer”. This means that a dataset contains a lot of separate pieces of data, but can be used to teach the machine learning algorithm to find predictable patterns inside the whole dataset.

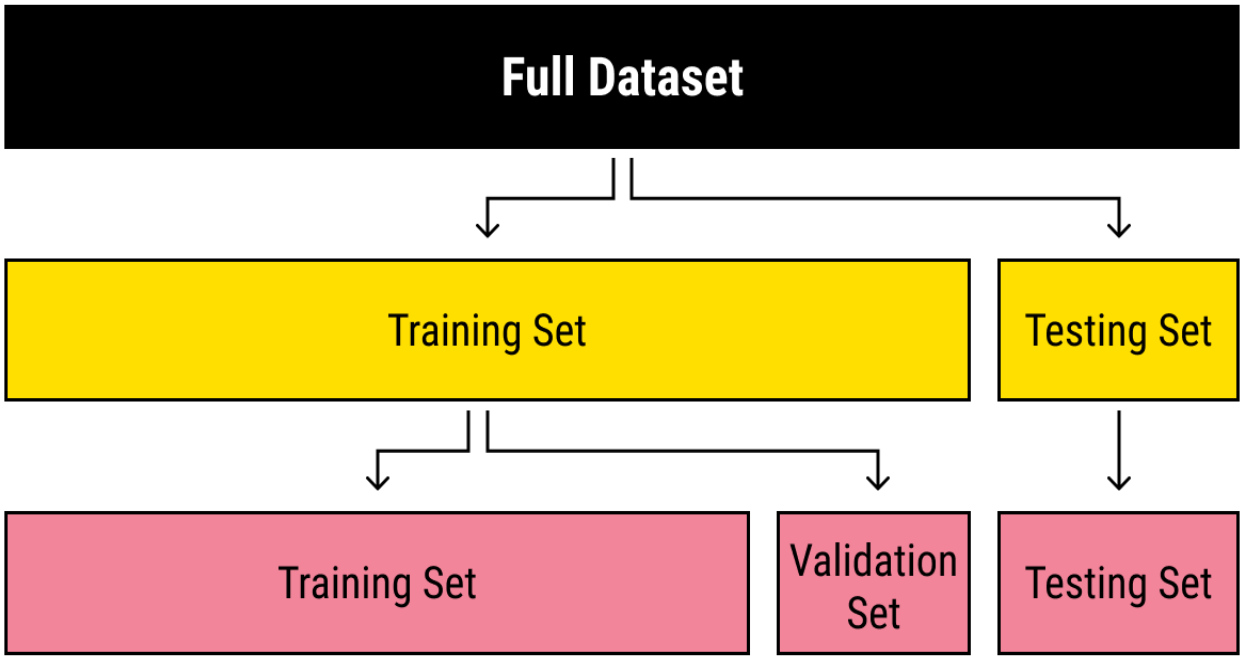
Data is an essential component of any AI model and, basically, the sole reason for the spike in popularity of machine learning that we witness today. Due to the availability of data, scalable ML algorithms became viable as actual products that can bring value to a business, rather than being a by-product of its main processes.

Your business has always been based on data. Factors such as what the customer bought, the popularity of the products, seasonality of the customer flow have always been important in business making. However, with the advent of machine learning, now it’s important to collect this data into datasets.

Sufficient volumes of data allow you to analyze the trends and hidden patterns and make decisions based on the dataset you’ve built. However, while it may look rather simple, working with data is more complicated. It requires proper treatment of the data you have, from the purposes of using a dataset to the preparation of the raw data for it to be actually usable.

Splitting Your Data: *Training, Testing, and Validation Datasets in Machine Learning*

Usually, a dataset is used not only for training purposes. A single training set that has already been processed is usually split into several types of datasets in machine learning, which is needed to check how well the training of the model went. For this purpose, a testing dataset is typically separated from the data. Next, a validation dataset, while not strictly crucial, is quite helpful to avoid training your algorithm on the same type of data and making biased predictions.

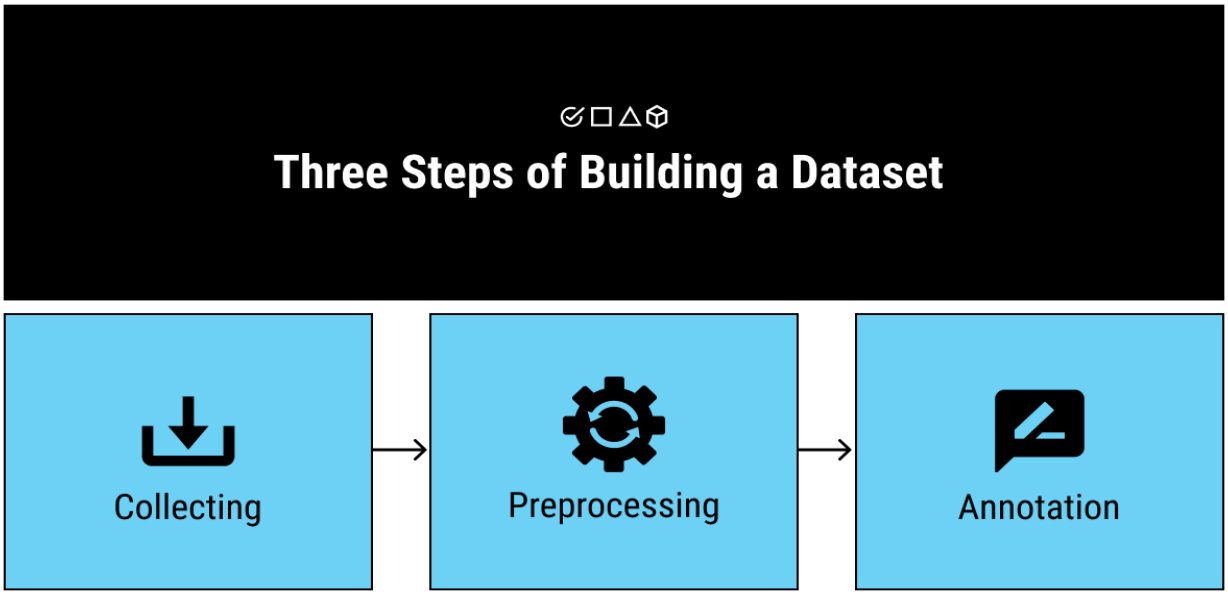


Splitting of a dataset into training, testing, and validation datasets

If you want to know more about how to split a dataset, we've covered this topic in detail in our article on training data.

Features of the Data: How to Build Yourself a Proper Dataset for a Machine Learning Project?

Raw data is a good place to start, but you obviously cannot just shove it into a machine learning algorithm and hope it offers you valuable insights into your customers' behaviors. There are quite a few steps you need to take before your dataset becomes usable.



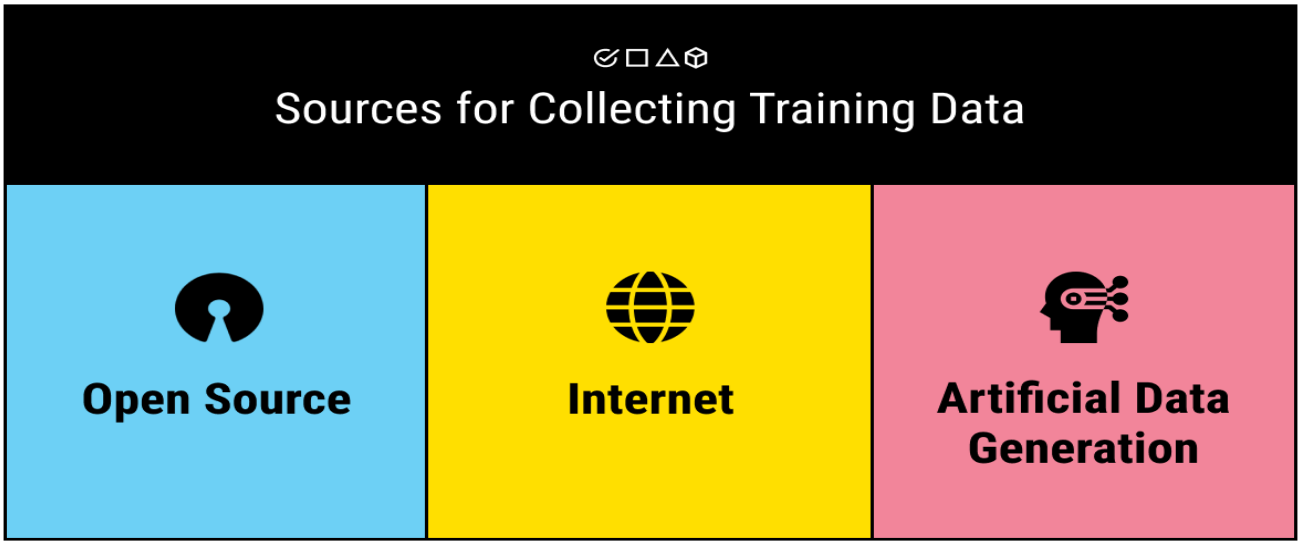
Three steps of data processing in machine learning

1. **Collect.** The first thing to do when you're looking for a dataset is deciding on the sources you'll be using for data collection in ML. Usually, there are three types of sources you can choose from: the freely available open-source datasets, the Internet, and the generators of artificial data. Each of these sources has its pros and cons and should be used for specific cases. We'll talk about this step in more detail in the next section of this article.
2. **Preprocess.** There's a principle in data science that every experienced professional adheres to. Start by answering this question:

has the dataset you're using been used before? If not, assume this dataset is flawed. If yes, there's still a high probability you'll need to re-appropriate the set to fit your specific goals. After covering the sources, we'll talk more about the features that constitute a proper dataset (you can click here to skip to that section now).

3. **Annotate.** After you've ensured your data is clean and relevant, you also need to make sure it's understandable for a computer to process. Machines do not understand the data the same way as humans do (they aren't able to assign the same meaning to the images or words as we). This step is where a lot of businesses often decide to outsource the task to experienced data tagging services, since keeping a trained annotation professional is not always viable. We have a great article on building an in-house labeling team vs. outsourcing this task to help you understand which way is the best for you.

Quest for a Dataset in Machine Learning: Where to Find It and What Sources Fit Your Case Best?



The three sources of the dataset collection

The sources for collecting an AI/ML dataset vary and strongly depend on your project, budget, and size of your business. The best option is to get help from professional data collection services that directly correlate with your business goals. However, while this way you have the most control over the data that you collect, it may prove complicated and demanding in terms of financial, time, and human resources.

Other ways like automatically generated datasets require significant computational powers and are not suitable for any project. For the purposes of this article, we'd like to specifically distinguish the free, ready-to-use datasets for machine learning. There are large, comprehensive repositories of public datasets that can be freely downloaded and used for the training of your machine learning algorithm.
The obvious advantage of free datasets is that they're, well, free. On the other hand, you'll most likely need to tune any of such downloadable datasets to fit your project, since they were built for other purposes initially and won't fit precisely into your custom-built ML model. Still, this is an option of choice for many startups, as well as small and medium-sized businesses, since it requires fewer resources to collect a proper dataset.

The Features of a Proper, High-Quality Dataset in Machine Learning

A good dataset combines high quality with sufficient quantity

However, before you decide on what sources to use while collecting a dataset for your ML model, consider the following features of a good dataset.

Quality of a Dataset: Relevance and Coverage

High data quality is the essential thing to take into consideration when you collect a dataset for a machine learning project. But what does this mean in practice? First, the data pieces should be relevant to your goal. If you are designing an ML algorithm for an autonomous vehicle, you will have no need even for the best of datasets that consist of celebrity photos.

Furthermore, it's important to ensure the pieces of data are of sufficient quality. While there are ways of cleaning the data and making it uniform and manageable before annotation and training processes, it's best to have the data correspond to a list of required features. For example, when building a facial recognition model, you will need the training photos to be of good enough quality.

In addition, even for relevant and high-quality datasets, there is a problem of blind spots and biases that any data can be subject to. An imbalanced dataset in ML poses the dangers of throwing off the prediction results of your carefully built ML model. Let's say you're planning to build a text classification model to arrange a database of texts by topic. But if you only use NLP datasets that don't cover enough topics, your model will likely fail to recognize the rarer ones.

Tip: try to use live data and expert text annotation services. Fake data might seem like a good idea when you're building your model (it is cheaper, cleaner, and is available in large volumes). But if you try to cut costs by using a fake dataset, you might end up with a weirdly trained algorithm. Fake data might turn out to be too predictable or not predictable enough. Either way, it's not a great start for your AI project.

Sufficient Quantity of a Dataset in Machine Learning

Not only quality but quantity matters, too. It's important to have enough data to train your algorithm properly. There's also a possibility of overtraining an algorithm (known as overfitting), but it's more likely you won't get enough high-quality data.

There's no perfect recipe for how much data you need. It's always a good idea to get advice from a data scientist. Professionals with extensive experience usually can roughly estimate the volume of the dataset you'll need for a specific AI project.

Alas, it is not sufficient to collect your dataset and make sure it corresponds to all the features we've listed above. There is one more step you need to take before starting the training of your ML model: analysis of the dataset.

There are cases that range from hilarious to horrifying about how strongly an ML algorithm depends on the exhaustive analysis of its dataset. One of such cases told by Martin Goodson, a guru of data science, shows the story of a hospital that decided to cut treatment costs for pneumonia patients. The highly accurate neural network that was built based on the clinic data could determine the patients with a low risk of developing complications. These patients could just take antibiotics at home without the need to visit the hospital.

However, when the model was considered for practical use, it was found that it sent all patients with asthma home even though these patients were actually at high risk of developing fatal complications. The problem was that human doctors knew this and always sent such patients to intensive care. For this reason, the historic dataset of the hospital had no recorded deaths for asthmatics with pneumonia, which resulted in the algorithm deciding asthma was not an aggravating condition. If employed in a practical setting, the algorithm would potentially result in human deaths, even though the dataset was relevant, comprehensive, and of high quality.

This case demonstrates that machines still cannot do the analytic work of humans and are merely tools that require supervision and control. When your dataset is collected, cleansed, annotated, and seems ready, analyze it before deploying the data as a training tool for your model.

Collecting different types of datasets in machine learning might seem like an easy task that can be done in the background while you pour most of your time and resources into building the machine learning model. However, as practice shows, time and time again, dealing with data might take most of your time due to the sheer scale that this task might grow to. For this reason, it's important to understand what a dataset in machine learning is, how to collect the data, and what features a proper dataset has.

A machine learning dataset is, quite simply, a collection of data pieces that can be treated by a computer as a single unit for analytic and prediction purposes. This means that the data collected should be made uniform and understandable for a machine that doesn't see data the same way as humans do. For this, after collecting the data, it's important to preprocess it by cleaning and completing it, as well as annotate the data by adding meaningful tags readable by a computer.

Moreover, a good dataset should correspond to certain quality and quantity standards. For smooth and fast training, you should make sure your dataset is relevant and well-balanced. Try to use live data whenever possible and consult with experienced professionals about the volume of the data and the source to collect it from.

Following these tips won't guarantee you collect a perfect dataset for your ML project. However, it will help you avoid some major pitfalls on your way to success.