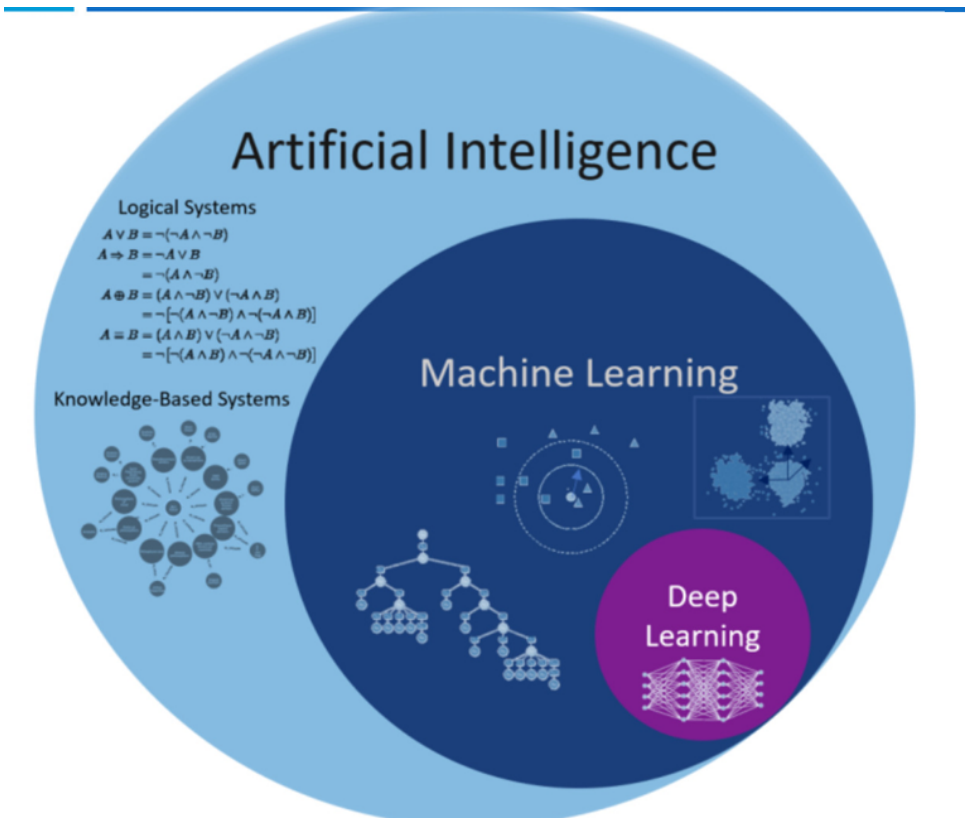
Overview on Machine Learning

▼ Introduction to Machine Learning

What is Machine Learning?

Machine Learning (ML) refers to the development of algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed. In other words, it's about teaching computers to learn from experience. Instead of relying on explicit instructions, ML systems learn patterns and relationships from data, allowing them to generalize to new situations.



Example: Consider a spam email filter. Instead of manually programming rules to identify spam emails, a machine learning algorithm can be trained on a dataset of emails labeled as spam or non-spam. The algorithm learns patterns in the data (e.g., common words or phrases in spam emails) and uses this knowledge to classify new, unseen emails as spam or not spam.

Understanding Intelligence

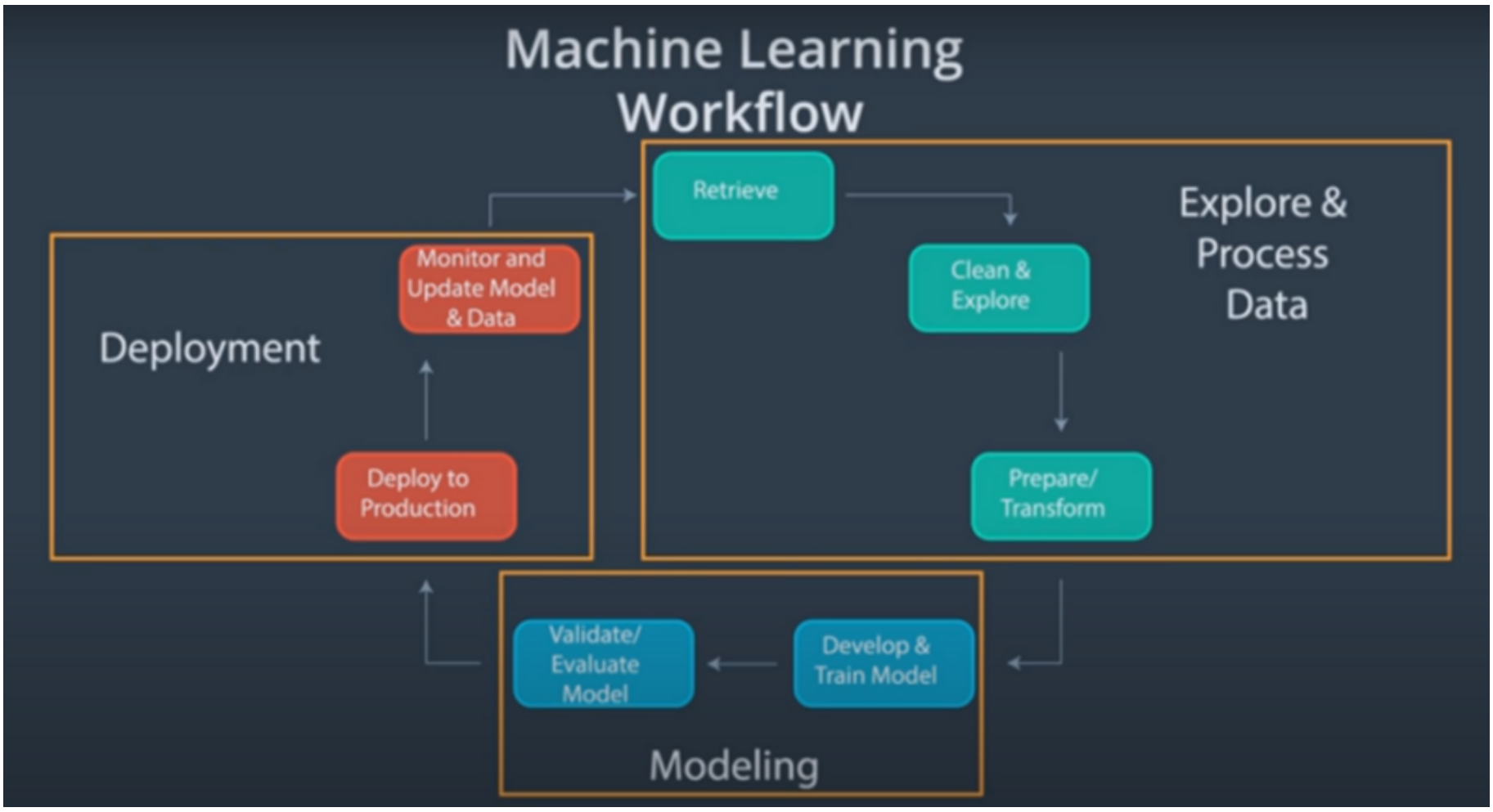
Intelligence is the ability to solve problems, learn from experience, and adapt to new situations. In the context of machine learning, the goal is to mimic or simulate human intelligence in computers. This involves various aspects such as reasoning, pattern recognition, and decision-making based on available information.

Example: Imagine a recommendation system used by online streaming platforms. By analyzing a user's past viewing history and preferences, the system can intelligently suggest movies or TV shows that the user is likely to enjoy. This mimics the way a human might recommend content based on someone's interests and past behavior.

▼ Basic Concepts and Workflow of ML

Workflow of Machine Learning

The workflow of machine learning typically involves several steps:



1. **Identifying the Problem**: Defining the task or problem to be solved.
2. **Data Collection**: Gathering relevant data that will be used to train the model.
3. **Model Selection and Training**: Choosing an appropriate model and training it on the collected data.
4. **Testing and Evaluation**: Assessing the performance of the trained model on a separate dataset to ensure it generalizes well to new data.
5. **Deployment**: Integrating the trained model into a production environment for real-world use.

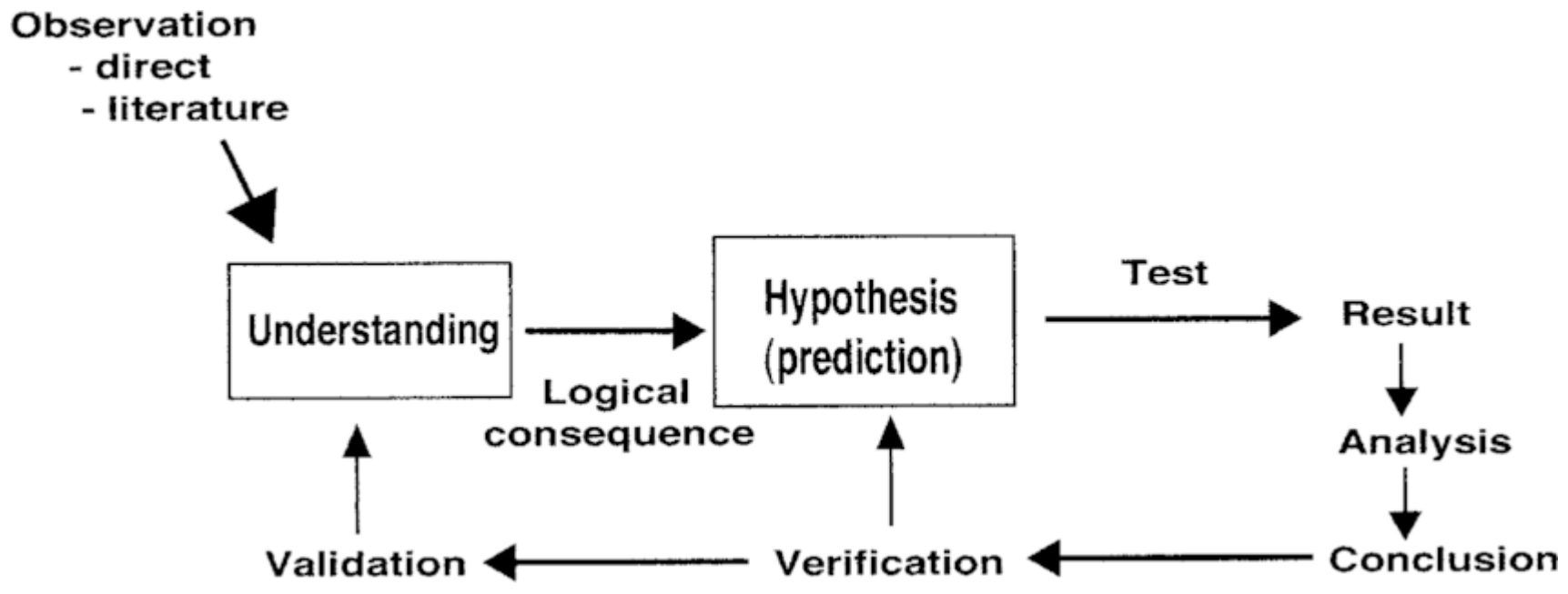
Learning Categories

Machine learning can be categorized into different types based on the approach used:

- **Logic IA (Reasoning-based)**: Involves using logical rules and reasoning to make decisions or draw conclusions.
- **Numeric IA (Knowledge-based)**: Relies on knowledge representation and processing numerical data to make predictions or classifications.

The Concept of Hypothesis/Assumption

In machine learning, a hypothesis or assumption is a proposed explanation for a phenomenon. It represents a conjecture about the relationship between variables in the data. During model training, the algorithm makes assumptions about the underlying patterns in the data and adjusts its parameters to fit the observed patterns.



Fuel of Machine Learning

The fuel of machine learning refers to the key components that drive the process:

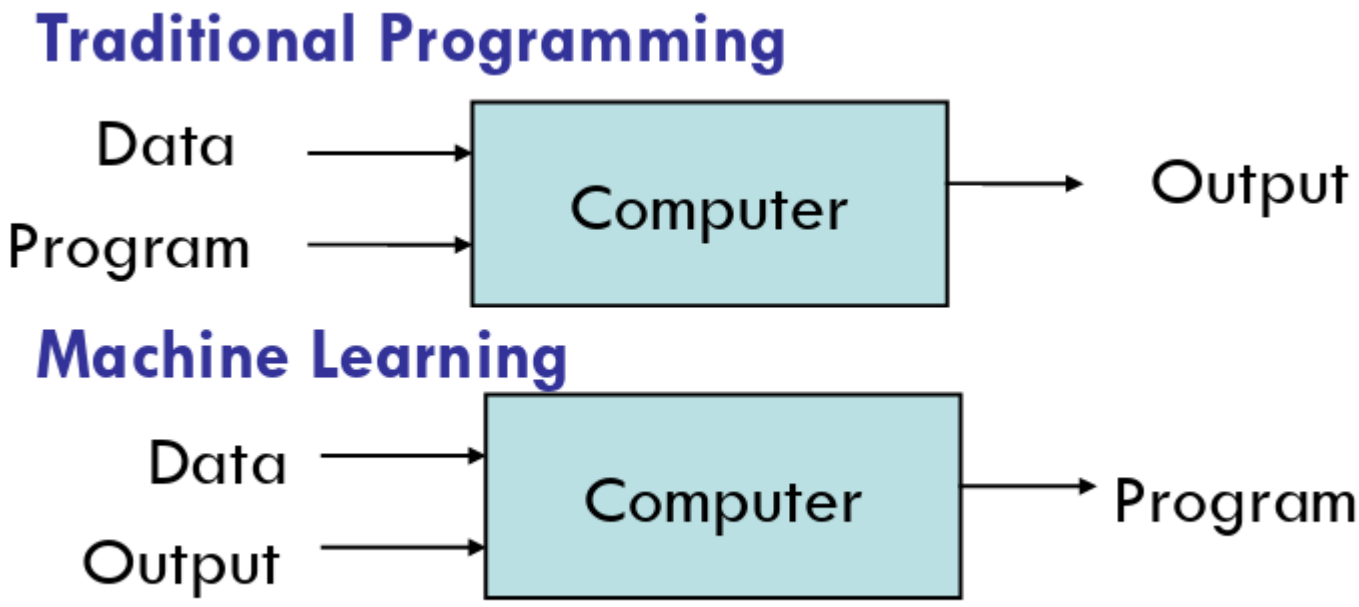
- **Data Availability**: Access to large and diverse datasets is essential for training accurate and robust machine learning models.
- **Computing Power**: High computational resources are required for processing large datasets and training complex models efficiently.

Example: Suppose you want to build a machine learning model to predict housing prices based on various features such as location, size, and number of bedrooms. You start by collecting a dataset containing historical housing data. Then, you choose an appropriate model and train it on this data. After testing its performance, you deploy the trained model to provide real-time predictions for new housing listings. Throughout this process, the availability of quality data and sufficient computing resources is crucial for the success of the machine learning project.

▼ Machine Learning Technologies

Traditional Programming vs. Programming Model

Traditional Programming involves explicitly instructing a computer on how to perform a task by writing rules and algorithms. This approach requires a deep understanding of the problem domain and manually coding solutions for specific scenarios. In contrast, **Programming Models** in machine learning allow computers to learn from data and generalize patterns without being explicitly programmed. Machine learning algorithms automatically adjust their parameters based on input data, enabling them to adapt to new situations and make predictions or decisions.



The Notion of Optimizing to Reduce the Number of Experiences

In machine learning, the notion of **optimizing to reduce the number of experiences** refers to the idea of minimizing the amount of trial and error required for a model to learn from data. Instead of relying solely on exhaustive search or brute-force methods, optimization techniques aim to efficiently navigate the solution space and find the optimal parameters or configurations for a given task. This approach accelerates the learning process and improves the efficiency of machine learning algorithms.

Example:

- **Traditional Programming**: Writing rules and conditions for a spam email filter to classify emails as spam or not spam based on keywords and patterns.
- **Programming Model**: Training a neural network to classify images of handwritten digits by learning patterns and features directly from labeled data.
- **Optimizing to Reduce the Number of Experiences**: Using techniques like gradient descent to iteratively update the parameters of a model and minimize the loss function, reducing the number of training iterations required for convergence.

▼ Model Construction and Training

Determine Tasks & Type of Problem

Before constructing a model, it's essential to **determine the task** you want the model to perform and the **type of problem** it falls into. Tasks can range from classification to regression to clustering, and understanding the problem type helps in selecting the appropriate algorithms and evaluation metrics.

Prediction by Approximation/Probability

In machine learning, predictions can be made through **approximation** or **probability**. Approximation involves estimating the output directly from the input data, while probability-based methods assign probabilities to different outcomes and make predictions based on likelihood.

Find Connections/Relations in Dataset

During model construction, the goal is to identify and leverage **connections** or **relations** in the dataset. This involves exploring the data to understand how different variables interact with each other and how they contribute to the target variable.

Manipulate Data to Find Relations & Patterns

Data manipulation techniques are used to uncover **relations** and **patterns** in the dataset. This may include feature engineering, dimensionality reduction, or data transformation to highlight meaningful insights that can improve model performance.

Constructing the Model for Best Approximation

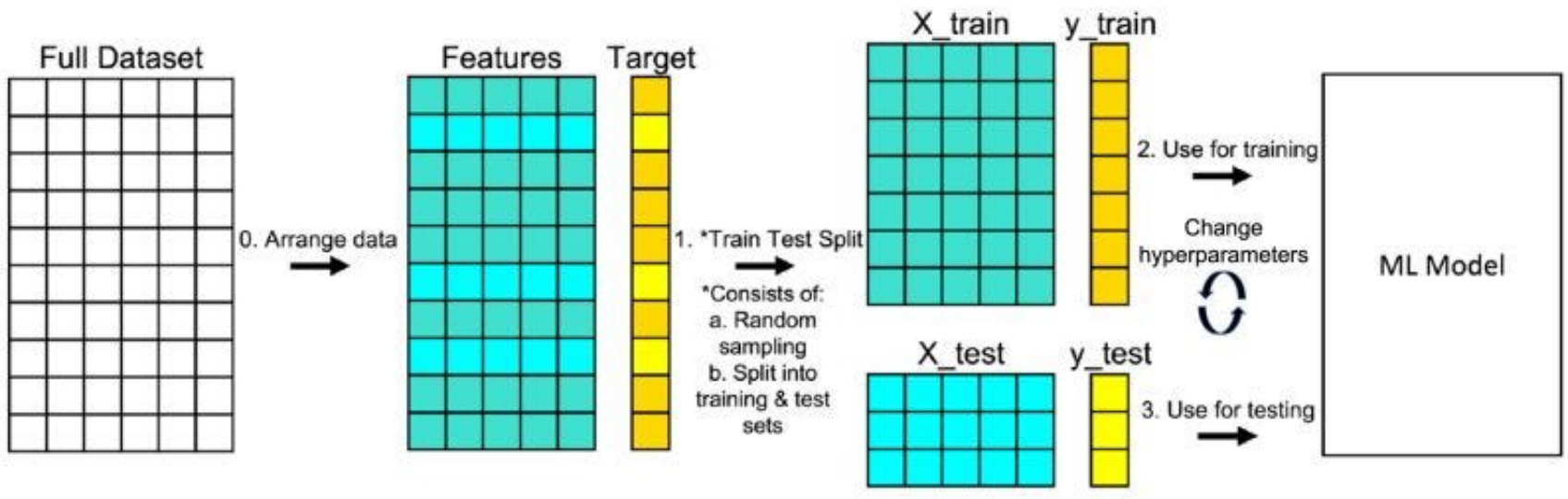
The process of constructing a model involves selecting the appropriate algorithm and fine-tuning its parameters to achieve the **best approximation** of the underlying patterns in the data. This requires balancing model complexity and generalization to ensure robust performance on unseen data.

Traditional Programming & Programming Model

Comparing **traditional programming** with **programming models** in machine learning highlights the shift from manual rule-based approaches to automated learning from data. Traditional programming relies on explicit instructions, while programming models enable machines to learn and adapt from experience.

Data Partition into Training & Testing Sets

To assess the performance of a model, the dataset is typically partitioned into **training** and **testing** sets. The training set is used to train the model, while the testing set is used to evaluate its performance on unseen data. This helps in estimating how well the model generalizes to new observations.



Performance of Model Depending on Modulization & Optimization

The **performance** of a model depends on various factors, including its modulization (structure) and the optimization techniques used during training. Model performance is evaluated using appropriate metrics, and optimization aims to improve performance by adjusting model parameters.

Quality & Quantity of Data

The **quality** and **quantity** of data significantly impact model performance. High-quality data with sufficient quantity ensures that the model can learn meaningful patterns and generalize well to new data.

Example:

- **Determine Tasks & Type of Problem**: Identifying whether the problem is a classification task (e.g., predicting spam or not spam) or a regression task (e.g., predicting house prices).
- **Prediction by Approximation/Probability**: Using a linear regression model to approximate the relationship between a house's features and its price, or using a logistic regression model to predict the probability of an email being spam.
- **Data Partition into Training & Testing Sets**: Splitting a dataset containing historical sales data into 70% training data and 30% testing data to train and evaluate a sales forecasting model.
- **Quality & Quantity of Data**: Ensuring that a dataset used for sentiment analysis of customer reviews contains a sufficient number of diverse reviews with accurate sentiment labels.

▼ Data Preparation

Prerequisites of ML: Raw Data Preparation

Before applying machine learning algorithms to raw data, it's essential to preprocess, transform, and clean the data to ensure its quality and suitability for modeling.

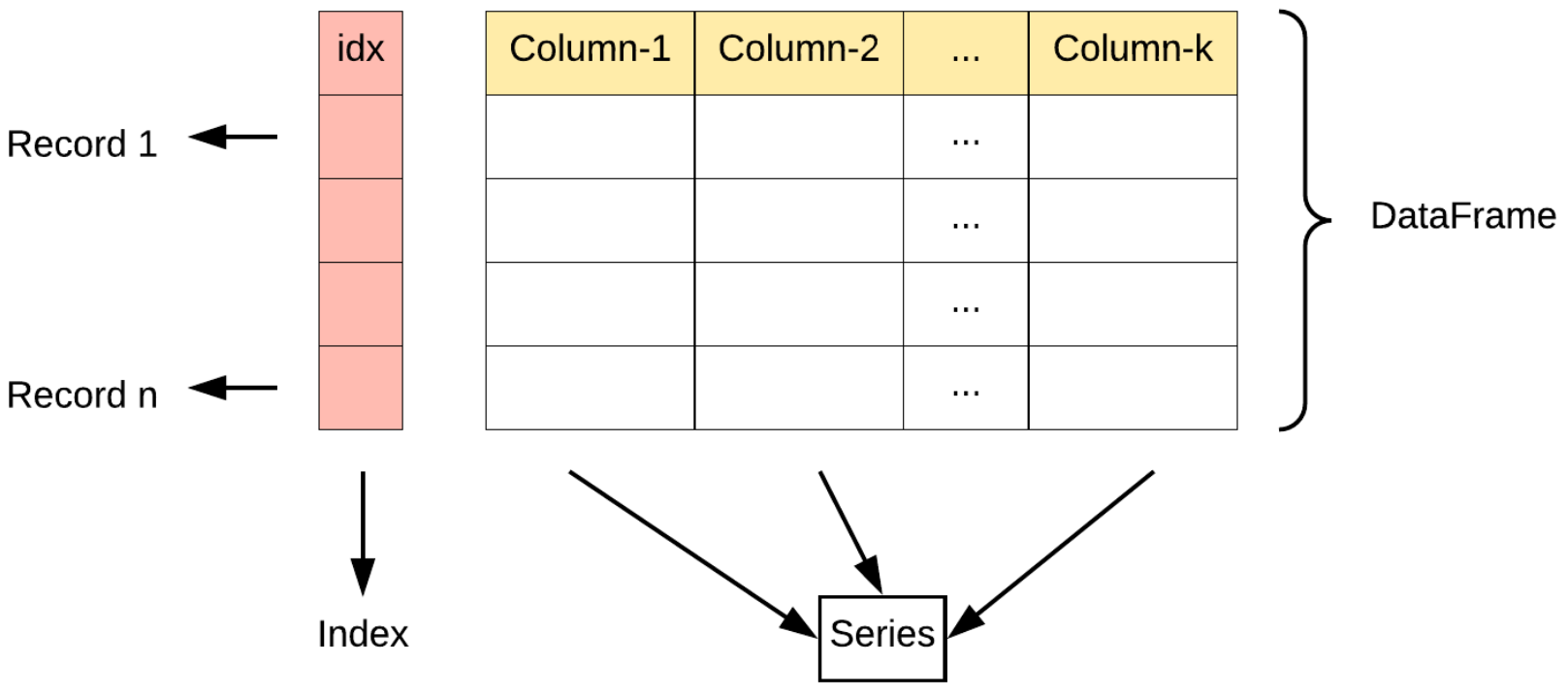
- **Preprocessing**: This involves tasks such as handling missing values, dealing with outliers, and scaling features to a similar range.
- **Transformation**: Data transformation may include converting categorical variables into numerical representations or performing feature engineering to create new informative features.
- **Cleaning**: Data cleaning focuses on identifying and correcting errors or inconsistencies in the dataset.

Role of Data: 70% Effort on Data / 30% on Model

The **quality** and **quantity** of data play a significant role in the success of a machine learning project. It's often said that **70%** of the effort in machine learning goes into data preparation, while the remaining **30%** is dedicated to model selection, training, and evaluation. This highlights the importance of investing time and resources into collecting, preprocessing, and curating high-quality datasets.

Representation of Data: Matrix

In machine learning, data is typically represented as a **matrix**, where each row corresponds to an observation or sample, and each column represents a feature or attribute of the data. This structured format allows algorithms to efficiently process and analyze large datasets. Additionally, data can also be represented as a set of **vectors**, where each vector represents a data point in a high-dimensional space.



Representation of Valuable Features/Attributes of Dataset

Identifying valuable features or attributes within a dataset is crucial for building effective machine learning models. These features should contain relevant information that is predictive of the target variable. Feature selection or extraction techniques can be used to identify the most informative features, leading to more efficient and accurate models.

Example: Suppose you're building a machine learning model to predict customer churn for a telecommunications company. Before training the model, you preprocess the raw data to handle missing values and encode categorical variables. Then, you transform the dataset by creating new features such as customer tenure or average monthly usage. After cleaning the data, you represent it as a matrix where each row represents a customer and each column represents a feature. Finally, you identify the most valuable features (e.g., customer satisfaction score, contract type) that significantly influence churn prediction.

▼ Types of Learning

Supervised Learning $\left(x \in \mathbb{R}^b,\, y \in \mathbb{R}\right)^N$

Supervised learning is a type of machine learning where the model is trained on a labeled dataset, meaning the dataset includes both input data and corresponding output labels. The goal is for the model to learn the mapping between inputs and outputs, allowing it to make predictions or decisions on new, unseen data.

Unsupervised Learning $\left(x \in \mathbb{R}^b\right)^N$

Unsupervised learning involves training a model on an unlabeled dataset, where only input data is provided without corresponding output labels. The model learns to find patterns, structures, or relationships in the data without explicit supervision. Common tasks include clustering similar data points or dimensionality reduction.

Semi-supervised Learning $\left(x \in \mathbb{R}^b, [\, y \in \mathbb{R}]\right)^N$

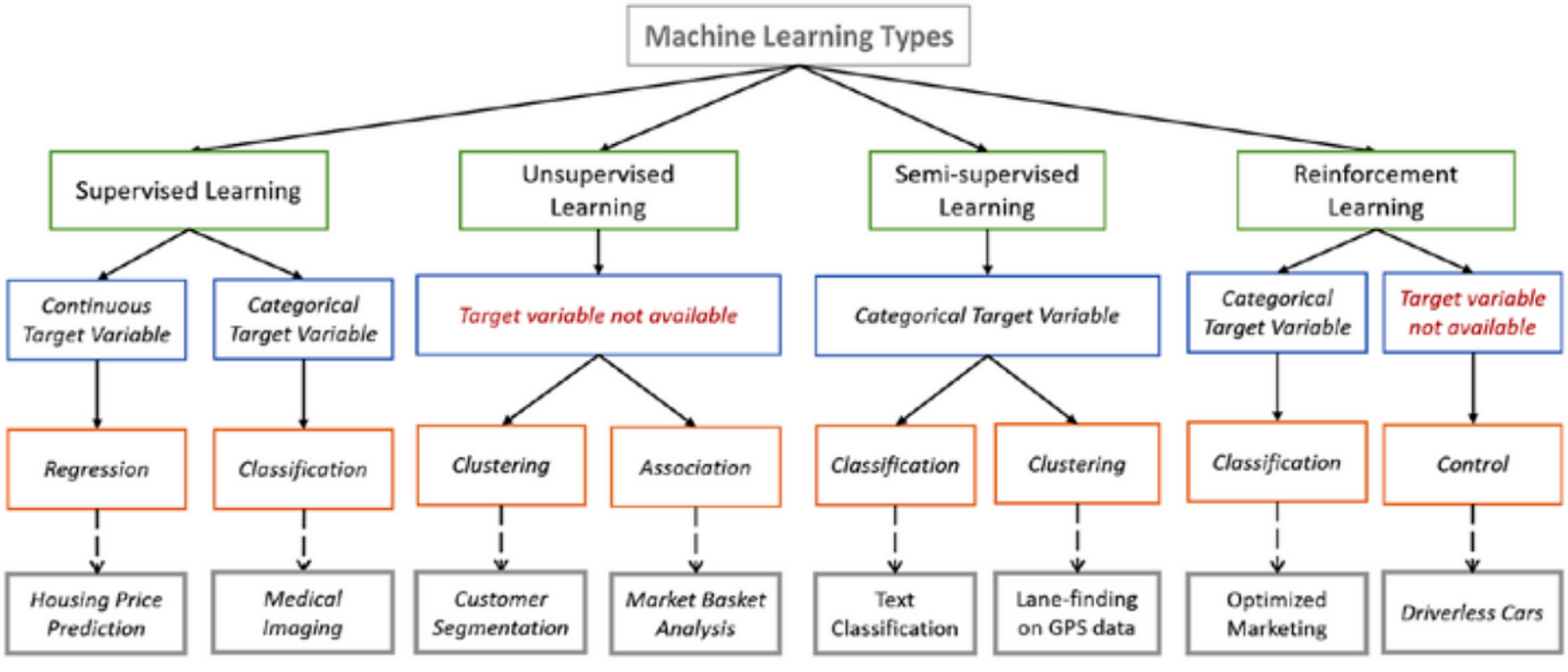
Semi-supervised learning combines elements of both supervised and unsupervised learning. It uses a small amount of labeled data along with a larger amount of unlabeled data for training. The model leverages the labeled data to guide the learning process and generalize to unlabeled data, potentially improving performance compared to purely unsupervised methods.

Reinforcement Learning

Reinforcement learning involves training an agent to interact with an environment and learn from feedback in the form of rewards or penalties. The agent takes actions in the environment to maximize cumulative rewards over time. Reinforcement learning is commonly used in tasks such as game playing, robotics, and autonomous vehicle control.

Examples:

- **Supervised Learning**: Predicting house prices based on features like size, location, and number of bedrooms, using a dataset with historical sales data where prices are labeled.
- **Unsupervised Learning**: Grouping customers based on their purchasing behavior without prior knowledge of customer segments, using clustering algorithms.
- **Semi-supervised Learning**: Training a spam email filter with a small labeled dataset of spam and non-spam emails, along with a larger unlabeled dataset, to improve classification accuracy.
- **Reinforcement Learning**: Training a robot to navigate through a maze by rewarding successful movements and penalizing collisions, allowing it to learn an optimal path over time.



▼ Machine Learning Components

Representation

Representation in machine learning refers to how data is represented and structured to be fed into a machine learning algorithm. It involves encoding input data into a format that can be understood and processed by the model. This step is crucial as the choice of representation can significantly impact the performance of the model.

Evaluation

Evaluation in machine learning involves assessing the performance of a trained model on unseen data. It measures how well the model generalizes to new data and can make accurate predictions or classifications. Evaluation metrics vary depending on the task, such as accuracy, precision, recall, or mean squared error, and are used to quantify the model's performance.

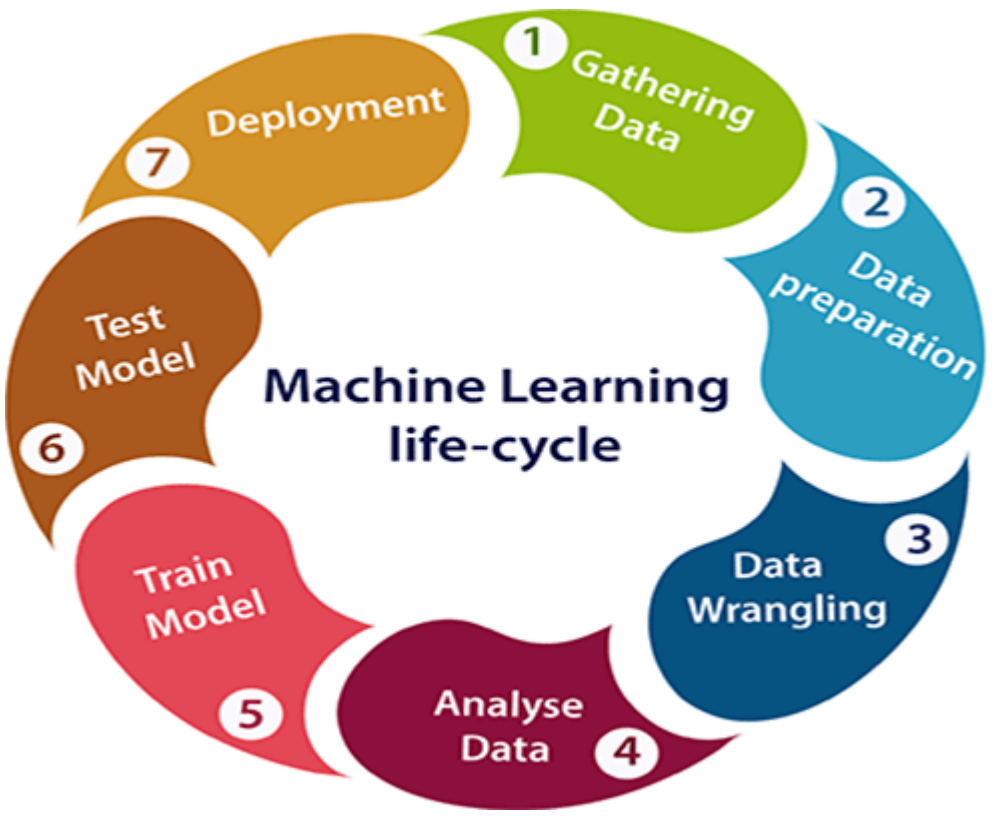
Optimization

Optimization in machine learning focuses on fine-tuning the parameters or hyperparameters of a model to improve its performance. This process involves iteratively adjusting the model's parameters based on feedback from the evaluation metrics. Optimization techniques aim to minimize errors or maximize performance metrics by optimizing the model's predictions or decisions.

Example:

- **Representation**: Representing images as pixel values or text documents as bag-of-words vectors before feeding them into a neural network for classification.
- **Evaluation**: Assessing the accuracy of a spam email filter by measuring its performance on a test dataset containing labeled spam and non-spam emails.
- **Optimization**: Tuning the learning rate and regularization parameters of a logistic regression model to minimize the classification error on a validation dataset.

▼ Steps in Developing a Machine Learning Application



Collecting Data

Collecting data is the first step in building a machine learning application. It involves gathering relevant datasets that contain examples of input data and corresponding output labels (if applicable). Data collection methods vary depending on the application and may involve sources such as databases, APIs, or manual data entry.

Preparing Input Data

Preparing input data involves preprocessing and cleaning the collected data to ensure its quality and suitability for training the machine learning model. This may include tasks such as handling missing values, encoding categorical variables, and scaling features to a similar range.

Analyzing Input Data

Analyzing input data involves exploring the dataset to gain insights into its characteristics, distributions, and relationships between variables. Data analysis techniques such as visualization and statistical analysis help identify patterns and trends that may inform feature selection or model design.

Filtering Garbage

Filtering garbage refers to removing irrelevant or noisy data points from the dataset that could negatively impact the performance of the machine learning model. This step helps improve the quality of the training data and ensures the model focuses on relevant patterns and relationships.

Training the Algorithm

Training the algorithm involves using the prepared dataset to train the machine learning model. During training, the model learns from the input data and adjusts its parameters to minimize errors or maximize performance metrics. The training process typically involves iterative optimization algorithms that update the model's parameters based on feedback from the training data.

Testing the Algorithm

Testing the algorithm is done to evaluate the performance of the trained model on unseen data. A separate dataset, called the test set, is used to assess how well the model generalizes to new examples. Evaluation metrics such as accuracy, precision, recall, or mean squared error are calculated to quantify the model's performance.

Using the Trained Model

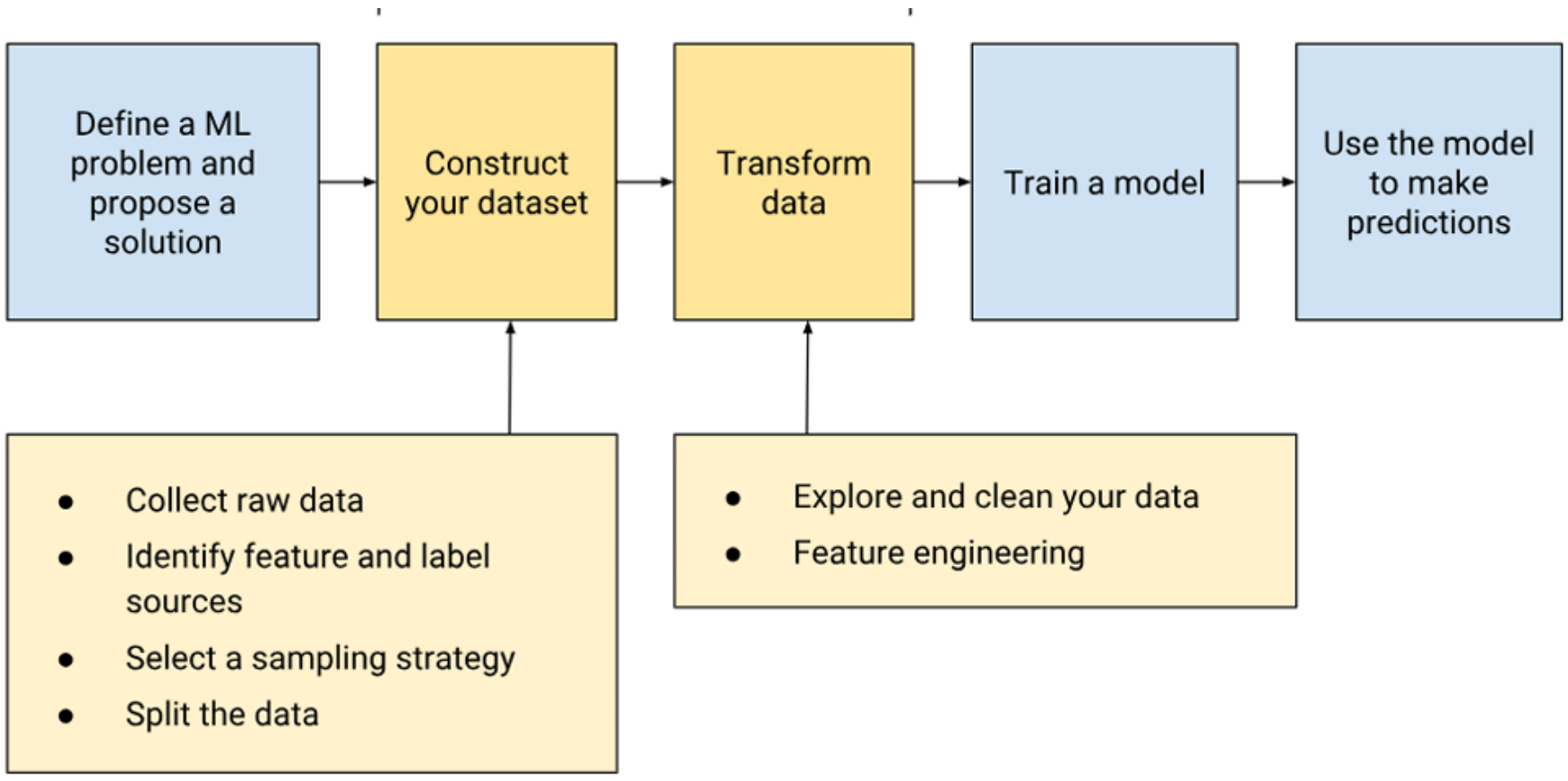
Using the trained model involves deploying it into a real-world application to make predictions or decisions based on new input data. The trained model can be integrated into software systems, mobile apps, or other platforms where it can provide insights or automate tasks based on its learned patterns and relationships.

Example:

Suppose you're developing a machine learning application to predict customer churn for a subscription-based service. You start by collecting relevant data on customer demographics, usage patterns, and subscription history. After

preparing and analyzing the data, you filter out irrelevant or noisy data points. Then, you train a machine learning model using the cleaned dataset. After testing its performance on a separate test set, you deploy the trained model into the production environment, where it can generate predictions on new customer data to identify potential churners.

▼ Data Preparation and Feature Engineering Process



ML Problem & Solution

ML Problem & Solution involves identifying the problem you want to solve with machine learning and determining the appropriate solution approach. This step requires understanding the business or domain context, defining the problem statement, and selecting the most suitable machine learning techniques or algorithms to address the problem effectively.

Constructing Dataset

Constructing Dataset involves gathering and assembling the necessary data for training the machine learning model. This includes collecting relevant features and labels (if applicable) from various sources and organizing them into a structured format suitable for analysis and modeling.

Transforming Data

Transforming Data refers to preprocessing and transforming the raw dataset to prepare it for training the machine learning model. This may involve tasks such as handling missing values, encoding categorical variables, scaling features, or performing feature engineering to create new informative features that capture relevant information from the data.

Training Model

Training Model involves using the transformed dataset to train the machine learning model. During training, the model learns patterns and relationships in the data by adjusting its parameters iteratively to minimize errors or maximize performance metrics. This step typically requires selecting appropriate algorithms and optimization techniques tailored to the specific problem domain.

Using Model to Make Predictions

Using Model to Make Predictions occurs after the model has been trained and validated. In this step, the trained model is deployed into production or operational environments where it can generate predictions or make decisions based on new input data. The model's predictions can be used to provide insights, automate tasks, or support decision-making processes in real-world applications.

Example:

Suppose you're developing a machine learning system to classify images of handwritten digits. The ML problem involves identifying the digit represented in each image. You construct a dataset by gathering a collection of labeled images of digits from zero to nine. After preprocessing the images to resize them and normalize pixel values, you train a convolutional neural network (CNN) model on the transformed dataset. Once trained, the model can be used to predict the digit in new images with high accuracy, making it useful for tasks like optical character recognition (OCR). A diagram illustrating this process could visualize the flow from data collection to model deployment, highlighting each step's importance in achieving accurate predictions.

▼ Steps to Constructing Your Dataset

Collecting Raw Data

Collecting raw data is the initial step in constructing a dataset. It involves gathering relevant data from various sources such as databases, APIs, sensors, or manual data entry. Raw data can include a wide range of information depending on the specific problem domain, such as text, images, numerical measurements, or categorical attributes.

Identifying Feature and Label Sources

Identifying feature and label sources entails determining the variables or attributes that will serve as input features for the machine learning model and identifying the target variable or label that the model aims to predict. Features are the input variables used to make predictions, while the label is the output variable that the model learns to predict.

Selecting a Sampling Strategy

Selecting a sampling strategy involves deciding how to sample data from the collected dataset. Depending on the dataset's size and characteristics, various sampling methods can be used, such as random sampling, stratified sampling, or over-sampling/under-sampling techniques to address class imbalances.

Splitting the Data

Splitting the data into training, validation, and test sets is essential to evaluate the model's performance accurately. The training set is used to train the model, the validation set is used to tune hyperparameters and assess model performance during training, and the test set is used to evaluate the final model's performance on unseen data.

Balancing Dataset Size to Avoid Overfitting & Underfitting

Balancing dataset size involves ensuring that the distribution of classes or labels in the dataset is representative and balanced. Imbalanced datasets with unequal class distributions can lead to model biases and affect performance metrics. Balancing the dataset size helps prevent overfitting or underfitting by *providing sufficient examples* for each class.

Quality of Data

> **Your model is only as good as your data**

Ensuring the **quality of data** is crucial for constructing a reliable dataset. This includes:

- **Reliability of Data Sources**: Assessing the reliability of data sources ensures that the collected data is accurate, consistent, and free from errors or biases. Reliable data sources contribute to *the trustworthiness and validity of the dataset*, enhancing the robustness of the machine learning model.
- **Feature Representation**: Ensuring proper feature representation involves selecting and encoding features in a way that captures relevant information and patterns in the data. Well-represented features provide *meaningful input to the model*, improving its ability to learn and make accurate predictions.
- **Minimizing Skew (Training vs Prediction)**: Minimizing skew between the *training and prediction datasets* is essential for model generalization. Skew refers to differences in *data distributions* between these datasets *(training & testing)*, which can lead to biased predictions and poor performance in real-world applications. By minimizing skew, the model can effectively generalize to unseen data and maintain its predictive accuracy.

Example:

Suppose you're building a spam email classifier. You collect raw email data from various sources, including labeled spam and non-spam emails. After identifying relevant features such as email content, sender address, and subject line,

you split the dataset into training, validation, and test sets. To address class imbalances, you use oversampling techniques to balance the dataset size. Additionally, you ensure data quality by verifying the reliability of email sources and representing features appropriately. Finally, you minimize skew between the training and prediction datasets to ensure the model's generalization ability.

▼ Identifying Labels and Sources/variables

1. **Identifying Labels**: Labels are the target variables that the model aims to predict. They represent the outcome or the variable of interest in the dataset. Identifying labels involves determining what specific aspect of the data you want the model to learn and predict. For example, in a dataset of images of handwritten digits, the label might be the actual digit represented by each image (e.g., 0, 1, 2, ..., 9).
2. **Identifying Sources**: Sources refer to the data from which features and labels are derived. This involves understanding where the data comes from, how it's collected, and its reliability. It may include sources such as databases, APIs, sensor data, user input, or external datasets. Identifying sources involves ensuring that the data is relevant, accurate, and representative of the problem domain.
3. **Direct Labels**:
    - **Definition**: Direct labels are the target variables that directly represent the outcome or prediction of interest.
    - **Examples**: In a classification task, direct labels could be binary (e.g., spam or not spam) or categorical (e.g., classifying images of animals into different species). In a regression task, direct labels could be continuous variables (e.g., predicting house prices based on features like size, location, etc.).
4. **Derived Labels**:
    - **Definition**: Derived labels are created or computed based on some transformation or combination of the original features or labels.
    - **Examples**: In a sentiment analysis task, the sentiment score (positive, neutral, negative) could be derived from text reviews using natural language processing techniques. In a predictive maintenance scenario, the time until the next failure could be derived from sensor data and historical failure records.

▼ Sampling and Splitting Data

Selection Strategies

When dealing with vast amounts of data in a machine learning project, selecting a suitable subset for training becomes imperative. This subset's selection criteria are dictated by the problem at hand and the specific features necessary for predictive modeling.

Example: E-commerce Recommendation System

Consider building a recommendation system for an e-commerce platform. Here, data sampling could involve selecting subsets at various levels:

1. **Product Level**: If the goal is to recommend similar products based on user interactions, sampling at the product level could involve selecting similar items or product categories.
2. **User Session Level**: Sampling at the user session level allows capturing sequences of user interactions, such as browsing history or items added to the cart during a session.
3. **User Level**: Sampling at the user level aggregates data over multiple sessions, providing insights into long-term preferences and behavior patterns.

Tailoring Sampling to Objectives

The choice of sampling granularity depends on the specific features required for analysis. For instance, to predict user preferences based on recent interactions, sampling at the user session level may be more relevant. Conversely, if the focus is on understanding overall user behavior over time, sampling at the user level would be appropriate.

▼ Down-sampling & Up-weighting for imbalanced data

Imbalanced Data in Machine Learning

Imbalanced data refers to a situation where the classes or categories within a dataset are not represented equally. Instead, one or more classes dominate the dataset, while others are underrepresented. This scenario is common in various real-world applications, such as fraud detection, medical diagnosis, or rare event prediction.

Challenges of Imbalanced Data

1. **Bias Towards Majority Class**: Machine learning models trained on imbalanced datasets tend to exhibit a bias towards the majority class, as they prioritize accuracy without properly accounting for the minority classes.
2. **Poor Generalization**: Models trained on imbalanced data may struggle to generalize well to new data, especially for the minority classes, leading to suboptimal performance in real-world scenarios.
3. **Evaluation Metrics**: Traditional evaluation metrics like accuracy can be misleading in imbalanced datasets since a model could achieve high accuracy by simply predicting the majority class most of the time.

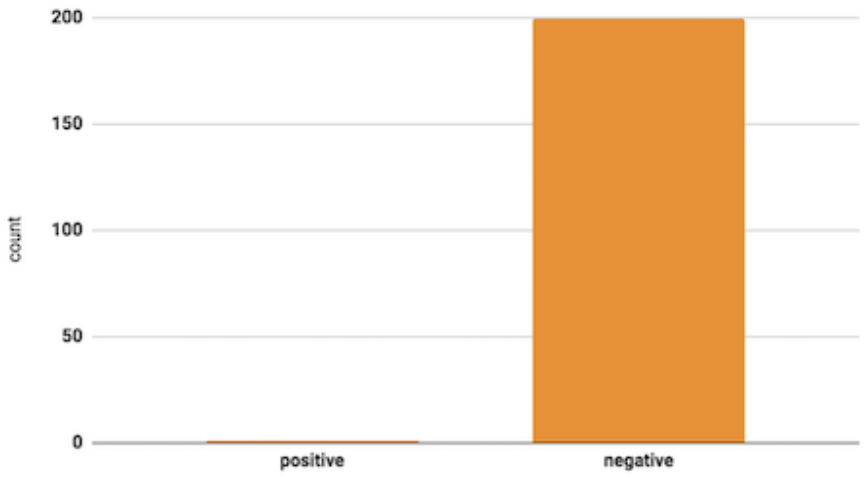
Techniques to Address Imbalanced Data [Resampling Methods]

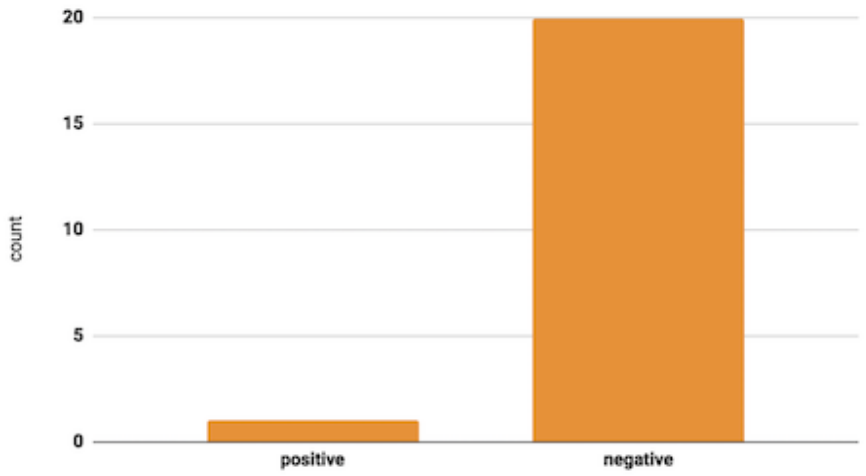
An effective way to handle imbalanced data is to down-sample and up-weight the majority class

- **Up/Over-sampling**: Increase the number of instances in the minority class by generating synthetic samples or replicating existing ones.
- **Down/Under-sampling**: Reduce the number of instances in the majority class to balance the dataset, typically by randomly removing instances.

Step 1. Down-sample the majority class

Consider the example of a fraud dataset, with 1 positive to 200 negatives. Down-sampling by a factor of 10 the majority class improves the balance to 1 positive over 20 negatives (5%). Although the resulting training set is still moderately imbalanced, the proportion of positives to negatives is much better than the original **extremely imbalanced** proportion (0.5%).

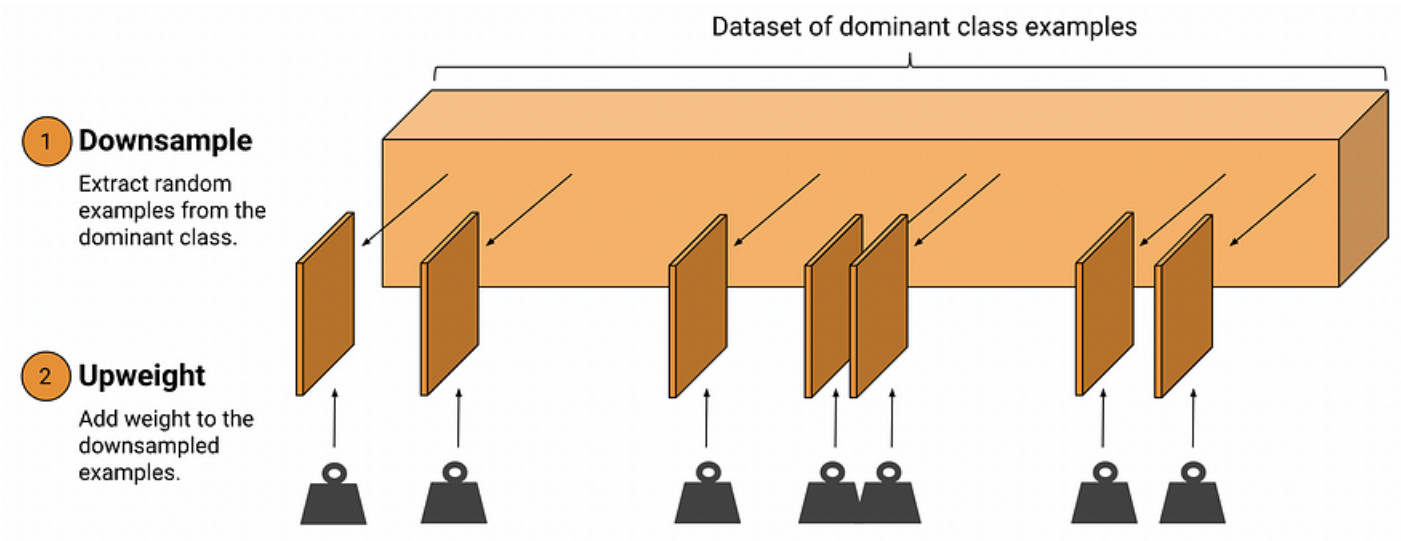




Step 2. Upweight the down-sampled majority class

The next step is to add weights to the down-sampled class. Since we down-sampled by a factor of 10, the weight should be 10. A weight of 10 means the model treats the sample as 10 times as important (when computing loss) as it would a sample of weight 1.

The weight should be equal to the factor used to down-sample:



The weight should be equal to the factor you used to downsample:

{example weight} = {original example weight} × {downsampling factor}

It may seem odd to add sample weights after down-sampling. We were trying to make the model improve on the minority class, why would upweight the majority? These are the resulting changes:

- **Faster convergence**: during training, the minority class is seen more often, which will help the model converge faster.
- **Disk space**: by consolidating the majority class into fewer samples with larger weights, we consume less disk space. This savings allow more disk space for the minority class, so we can collect a greater number and a wider range of samples for that class.
- **Modification**: Overweighting ensures the model remains accurately adjusted, allowing the outputs to retain their interpretability as probabilities.

▼ When Random Splitting isn't the Best Approach

While random splitting is the best approach for many ML problems, it isn't always the right solution. For example, consider data sets in which the examples are naturally clustered into similar examples.

Suppose you want your model to classify the topic from the text of a news article. Why would a random split be problematic?

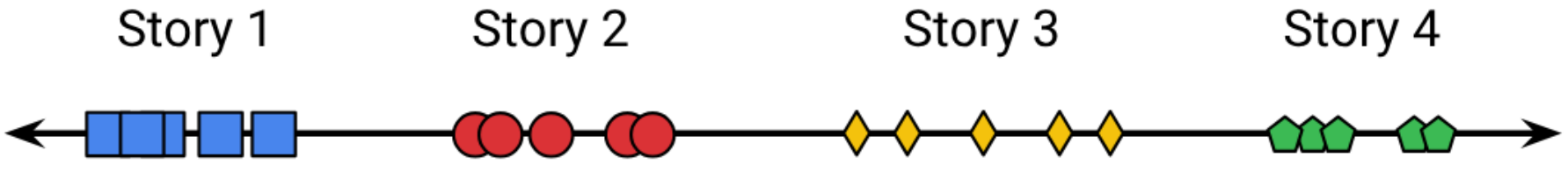


Figure 1. News Stories are Clustered.

News stories appear in clusters: multiple stories about the same topic are published around the same time. If we split the data randomly, therefore, the test set and the training set will likely contain the same stories. In reality, it wouldn't work this way because all the stories will come in at the same time, so doing the split like this would cause skew.

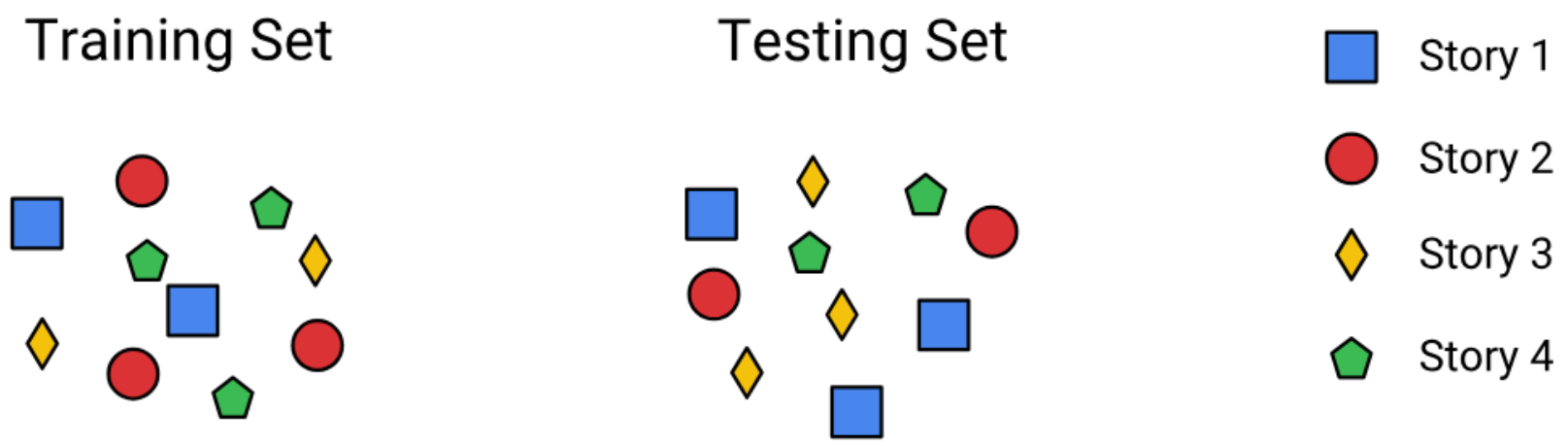


Figure 2. A random split will split a cluster across sets, causing skew.

A simple approach to fixing this problem would be to split our data based on when the story was published, perhaps by day the story was published. This results in stories from the same day being placed in the same split.

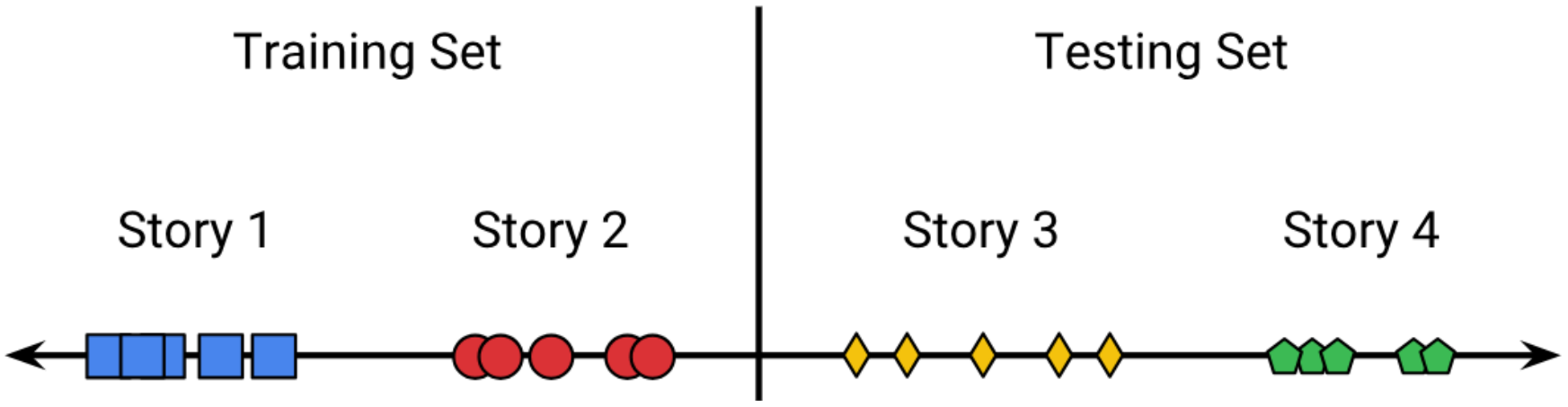
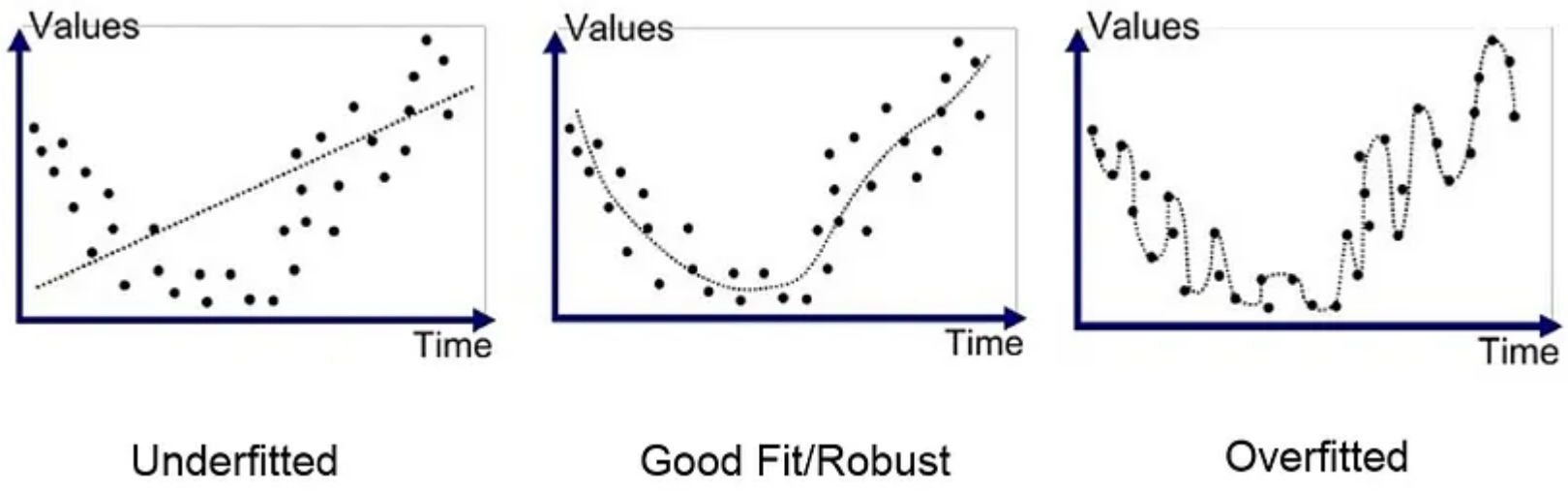


Figure 3. Splitting on time allows the clusters to mostly end up in the same set.

With tens of thousands or more news stories, a percentage may get divided across the days. That's okay, though; in reality these stories were split across two days of the news cycle. Alternatively, you could throw out data within a certain distance of your cutoff to ensure you don't have any overlap. For example, you could train on stories for the month of April, and then use the second week of May as the test set, with the week gap preventing overlap.

▼ Why Our model perform poor sometime



When our model performs poorly, it could be attributed to different issues:

1. **Overfitting**: This happens when the model tries too hard to fit the training data perfectly, capturing even the noise or random variations that are not relevant. It's like memorizing the training data rather than understanding the underlying patterns. As a result, the model may perform well on the training data but poorly on new, unseen data.

   Example: Imagine you're studying for a test by memorizing every question and answer in your textbook without understanding the concepts. You might do well on practice quizzes but struggle on the actual test because you haven't truly learned the material.

2. **Underfitting**: In contrast, underfitting occurs when the model is too simplistic and fails to capture the essential patterns in the data. It's like oversimplifying a complex problem and missing important details. An underfit model performs poorly both on the training data and new data because it hasn't learned enough from the training process.

   Example: Suppose you're trying to predict a friend's mood based on factors like their sleep, exercise, and social interactions. An underfit model might just guess the same mood for everyone, ignoring the individual differences and nuances that affect mood.

Ideal solution

- **Good Fit / Robust Model**: This is the ideal scenario where the model strikes the right balance between complexity and simplicity. It captures the relevant patterns in the data without getting bogged down by noise or oversimplification. A robust model performs well on both the training data and new, unseen data, making accurate predictions without overfitting or underfitting.

  Example: Think of a chef who creates a recipe that perfectly balances flavors, textures, and cooking techniques to create a delicious dish. The recipe isn't overly complicated with unnecessary ingredients, nor is it too basic to be interesting. It's just right, resulting in a meal that consistently delights diners.