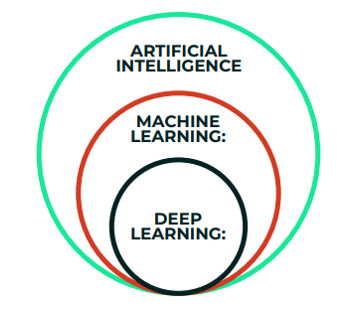
**What is Artificial Intelligence?**

* **Artificial Intelligence (AI)** refers to the simulation of human intelligence in machines that are programmed to think, learn, and make decisions.
* Science that empowers computers to mimic human intelligence such as decision making, text processing, and visual perception.
* AI enables machines to perform tasks that typically require human intelligence, such as reasoning, problem-solving, understanding natural language, recognizing patterns, and making decisions.
* AI is a broader field (i.e. the big umbrella) that contains several subfield such as machine learning, deep learning, robotics, and computer vision.



**Core Areas of AI**

**1) Machine Learning (ML):** Machines learn from data to make predictions or decisions without explicit programming.

**E.g.** Spam Email filters.

**2) Natural Language Processing (NLP):** Understanding and generating human language.

**E.g.** Chatbots and Language Translation tools.

**3) Computer Vision:** Interpreting Visual data from the world.

**E.g.** Facial Recognition systems.

**4) Robotics:** Physical Machines perform tasks in the real world.

**E.g.** Autonomous Vehicles.

**What is Machine Learning?**

1. Machine Learning (ML) is a branch of artificial intelligence (AI) that focuses on developing systems and algorithms that can learn from data and make decisions or predictions without being explicitly programmed.
2. ML uses Patterns and Insights derived from data to improve their performance over time.

**Key Concepts in Machine Learning:**

**1) Learning from Data:**

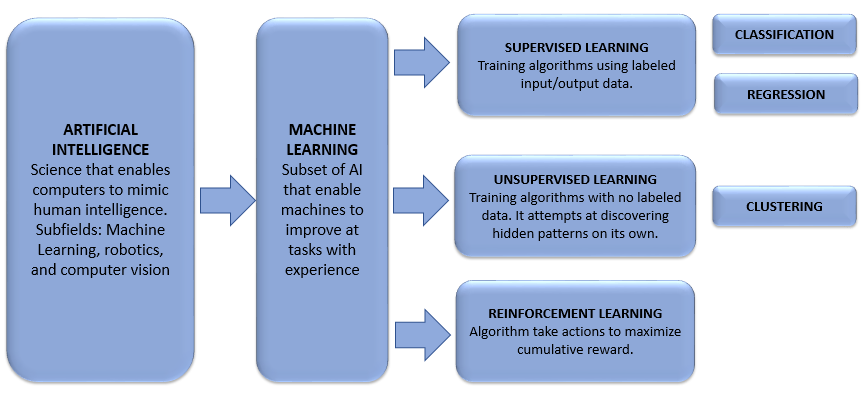
* ML systems process and analyze data to identify trends, patterns, and relationships.
* The system refines its predictions or actions as more data is available.

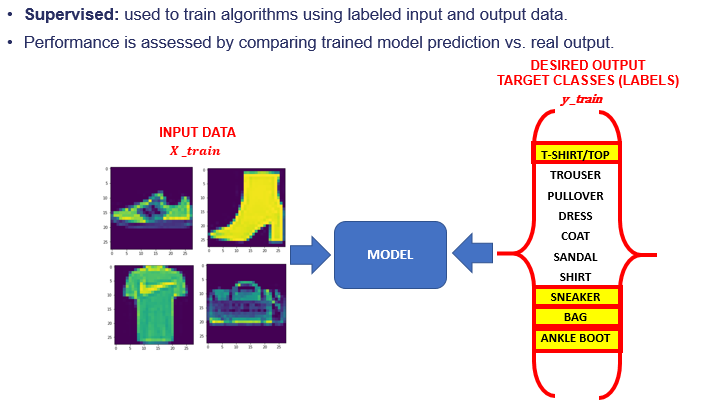
**2) Algorithms:**

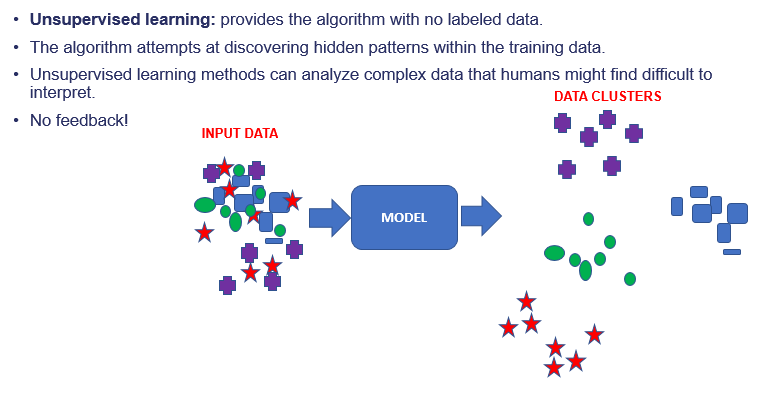
* ML uses algorithms like Linear Regression, Decision Trees, Random Forest, Neural Networks and Support Vector Machines to analyze and make sense of data.
* Algorithms are chosen based on the nature of the problem and data.

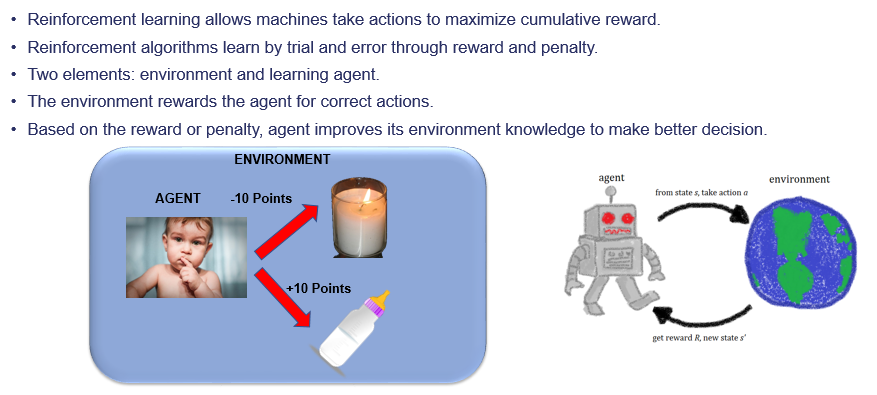
1. **Types of Learning:**

* **Supervised Learning:** The model is trained on labelled data (data with input-output pairs). Examples include classification (e.g., email spam detection) and regression (e.g., predicting house prices).
* **Unsupervised Learning:** The model is trained on unlabelled data to find hidden patterns (e.g., clustering customer behaviour or detecting anomalies).
* **Reinforcement Learning:** The model learns by interacting with its environment and receiving feedback in the form of rewards or penalties.







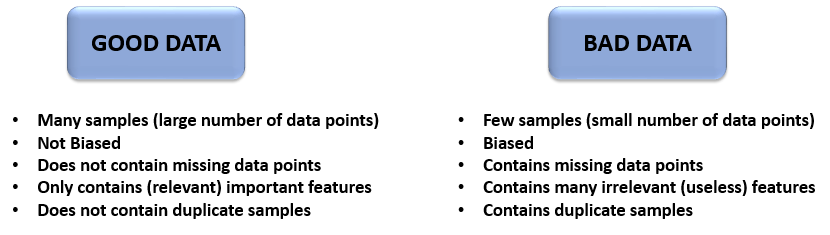


1. **Applications:**

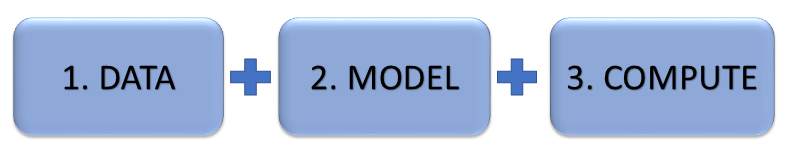
* Speech and Image recognition.
* Autonomous Vehicles.
* Natural Language Processing (e.g., chatbots, translation).
* Fraud detection and Predictive analytics.
* Recommender systems (e.g., Netflix or Amazon suggestions).

1. **Model Evaluation:**

* ML models are evaluated using metrics like accuracy, precision, recall, and F1-score to ensure their reliability and efficiency.



**ML Components:**



**What is PySpark MLib?**

1. **PySpark MLlib** is the ML library for **Apache Spark**, a powerful open-source distributed computing framework.
2. It provides scalable and easy-to-use tools for machine learning tasks, leveraging Spark's distributed architecture to handle large-scale data efficiently.

**Key Features of PySpark MLlib**

1. **Distributed Machine Learning:**

MLlib enables distributed training and testing of machine learning models, making it suitable for large datasets that don't fit into a single machine's memory.

1. **Ease of Use:**

Supports Python, along with Java, Scala, and R, making it accessible for a wide range of developers and data scientists.

1. **Integrated Workflow:**

Works seamlessly with other Spark components like SQL, DataFrames, and GraphX for integrated data processing and analysis.

1. **Performance:**

Optimized for in-memory processing, ensuring high performance for iterative machine learning algorithms.

**Components of PySpark MLlib:**

1. **Data Types:**

Provides data structures like Vectors and Matrices to handle features and labels.

1. **Algorithms:**

* **Classification:** Logistic Regression, Decision Trees, Random Forests, and Support Vector Machines (SVM).
* **Regression:** Linear Regression.
* **Clustering:** K-Means, Gaussian Mixture etc.

1. **Feature Engineering:**

Tools for feature extraction, transformation, dimensionality reduction, and selection. **For E.g.**

* TF-IDF for text data.
* StandardScaler and MinMaxScaler for normalization.
* Principal Component Analysis (PCA) for dimensionality reduction.

1. **Pipelines:**

A high-level API for building machine learning workflows. The Pipeline concept helps manage the sequence of data transformation and model training steps.

1. **Evaluation Metrics:**

Tools to evaluate models, including metrics for classification, regression, and clustering.

**What is VectorAssembler in PySpark MLlib?**

1. VectorAssembler is a **feature transformer** in PySpark MLlib that **combines multiple columns** of a DataFrame into a **single feature vector column**.
2. It is essential when building machine learning pipelines, as many PySpark ML algorithms expect features to be provided as a single vector.

**Syntax:**

*from pyspark.ml.feature import VectorAssembler*

*assembler = VectorAssembler (*

*inputCols = ["feature1", "feature2", "feature3"], # Columns to combine*

*outputCol = "features", # Name of the new feature vector column*

*handleInvalid = "error" # How to handle invalid data ('error', 'skip', or 'keep'))*

**What is Vectors in PySpark MLlib?**

1. In **PySpark MLlib**, **Vectors** represent feature vectors used in machine learning models.
2. They are key components for storing and processing numeric data efficiently.
3. PySpark supports two types of vectors:

* **Dense Vectors** and **Sparse Vectors**, both of which are part of the pyspark.ml.linalg module.

**What is StandardScaler in PySpark MLlib?**

1. **StandardScaler is a feature scaling transformer in PySpark MLlib that standardizes features by removing the mean and scaling to unit variance.**
2. **It ensures that features with different scales or ranges contribute equally to a machine learning model, improving its performance.**

**Syntax:**

from pyspark.ml.feature import StandardScaler

scaler = StandardScaler(

inputCol="features", # Input vector column

outputCol="scaled\_features", # Scaled output column

withMean=False, # Centers data (default: False)

withStd=True # Scales data (default: True))

**What is StringIndexer in PySpark MLlib?**

1. StringIndexer is a feature transformer in PySpark MLlib that **encodes categorical string columns** into **numerical indices**.
2. This transformation is essential because many machine learning algorithms work with numerical data only.

**Syntax:**

from pyspark.ml.feature import StringIndexer

indexer = StringIndexer(

inputCol="categorical\_col", # Column with categorical strings

outputCol="indexed\_col", # Name of the transformed output column

handleInvalid="error" # How to handle invalid data ('error', 'skip', or 'keep'))

**What is** Pipeline **in PySpark MLlib?**

1. A **Pipeline** in PySpark MLlib is a mechanism for building and managing a **machine learning workflow**.
2. It allows you to assemble multiple stages such as **data transformation**, **feature engineering**, and **model training** into a single object that can be trained and evaluated as one unit.
3. This simplifies the machine learning process by making it modular, reproducible, and easy to maintain.

**Syntax:**

from pyspark.ml import Pipeline

pipeline = Pipeline(stages=[

transformer\_1, # Example: StringIndexer

transformer\_2, # Example: VectorAssembler

estimator\_1 # Example: LinearRegression

])

**What is LinearRegression in PySpark MLlib?**

1. LinearRegression in PySpark MLlib is a supervised learning algorithm used for **regression tasks**.
2. It predicts a continuous numeric value based on input features by learning a linear relationship between features (independent variables) and the target variable (dependent variable).

**Syntax:**

from pyspark.ml.regression import LinearRegression

**Initialize the model:**

lr = LinearRegression(

featuresCol="features", # Feature vector column

labelCol="label", # Target column

predictionCol="prediction", # Output column

maxIter=100, # Maximum iterations

regParam=0.1, # Regularization parameter (L2 penalty)

elasticNetParam=0.5 # ElasticNet mixing (0 for L2, 1 for L1)

)

**What is RegressionEvaluator in PySpark MLlib?**

1. RegressionEvaluator in PySpark MLlib is a tool for **evaluating the performance** of regression models.
2. It computes evaluation metrics based on the difference between predicted and actual values.
3. This is essential for assessing the accuracy of regression models in tasks like predicting house prices, stock prices, or any continuous target variable.

**Syntax:**

from pyspark.ml.evaluation import RegressionEvaluator

evaluator = RegressionEvaluator(

labelCol="label", # Actual values

predictionCol="prediction", # Predicted values

metricName="rmse" # Evaluation metric (rmse, mse, mae, r2))

What is **RMSE (Root Mean Squared Error)** in PySpark MLlib?

1. **RMSE (Root Mean Squared Error)** in PySpark MLlib is a metric used to evaluate the accuracy of a machine learning model especially in regression tasks.
2. It measures average magnitude of errors between predicted and actual values.
3. A lower RMSE indicates a better fit of the model to the data.

What is **MSE (Root Mean Squared Error)** in PySpark MLlib?

1. **MSE (Mean Squared Error)** in PySpark MLlib is a evaluation metric used to measure the average squared difference between the predicted values and the actual target values in regression tasks.
2. It indicates how well a regression model fits the data.

What is **MAE (Root Mean Squared Error)** in PySpark MLlib?

1. **MAE (Mean Absolute Error)** in PySpark MLlib is a regression evaluation metric that measures the average magnitude of errors between predicted and actual values, without considering their direction.
2. It represents the average absolute difference between the predicted and actual target values.

What is **r2 (R-Squared)** in PySpark MLlib?

1. **R² (R-Squared)** in PySpark MLlib is a statistical metric used to evaluate the performance of regression models.
2. It measures how well the predicted values from the model explain the variance in the actual target values.
3. R² is also known as the **coefficient of determination**.

**How to Interpret Accuracy for Regression:**

If you still want something similar to "accuracy," you can use **R² (R-Squared)**, which indicates how well the model fits the data:

**R² = 1:** Perfect fit (100% accuracy).

**R² = 0:** No predictive power (similar to guessing the mean).

**R² < 0:** Worse than the mean-based prediction.