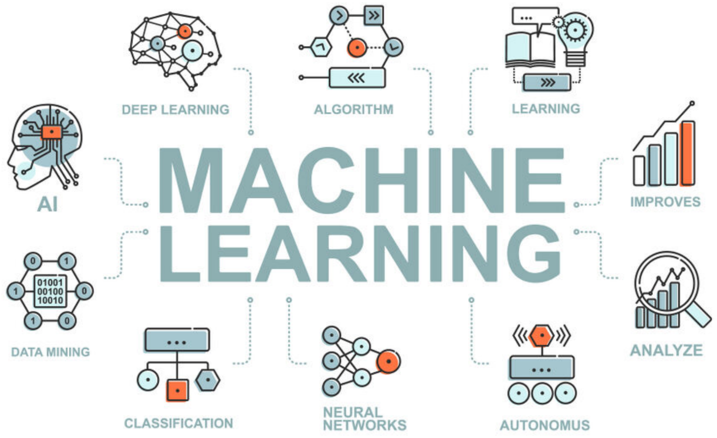
Introduction to machine learning

By Dr. Muhammad Alli

Dorset College Dublin
Educating since 1983

Machine learning

- We'll dive into the fundamental techniques that allow machines to learn from data, forming the backbone of modern AI systems.
- From self-driving cars to facial recognition, machine learning powers many of the intelligent technologies around us.



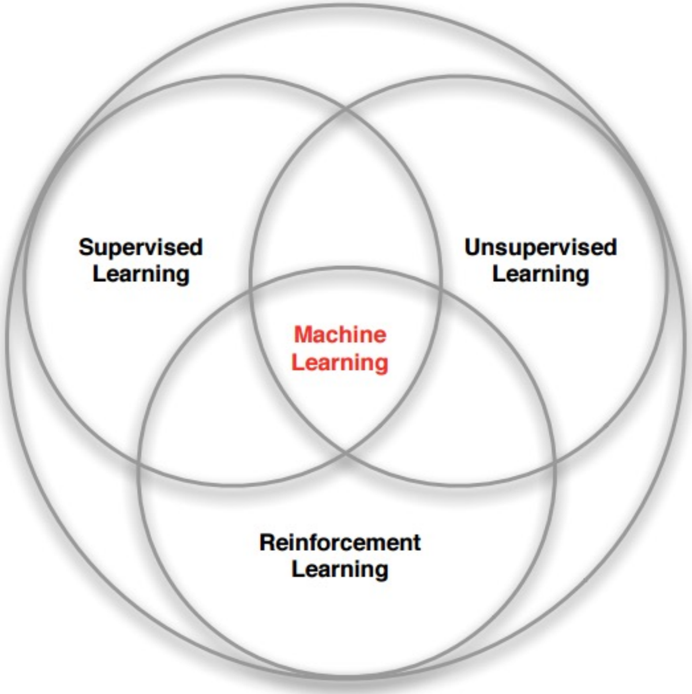
Dorset College Dublin
Educating since 1983

What is machine learning?

- Machine learning (ML) is a subfield of AI concerned with algorithms that enable computers to learn from data without explicit programming.
- **Learning from examples:** Instead of hand-coding rules, ML models identify patterns in data and make predictions or decisions.
- In traditional programming, we give computers instructions.
- In machine learning, we give them data and let them figure out the rules themselves.
- This ability to learn from examples makes ML a powerful tool for solving complex problems where the solution isn't easily programmable.

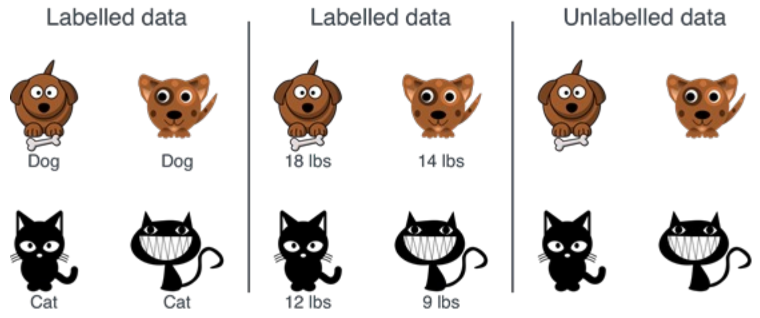
Types of machine learning

- Supervised learning: Learning from labeled data (input-output pairs).
- Unsupervised learning: Finding patterns in unlabeled data.
- Reinforcement learning: Learning through trial and error, maximizing rewards.
- ML can be broadly classified into these three types. Supervised learning is like having a teacher with example problems and correct answers. Unsupervised learning focuses on finding hidden structures in data. Reinforcement learning is akin to training a pet by giving rewards for good behavior.



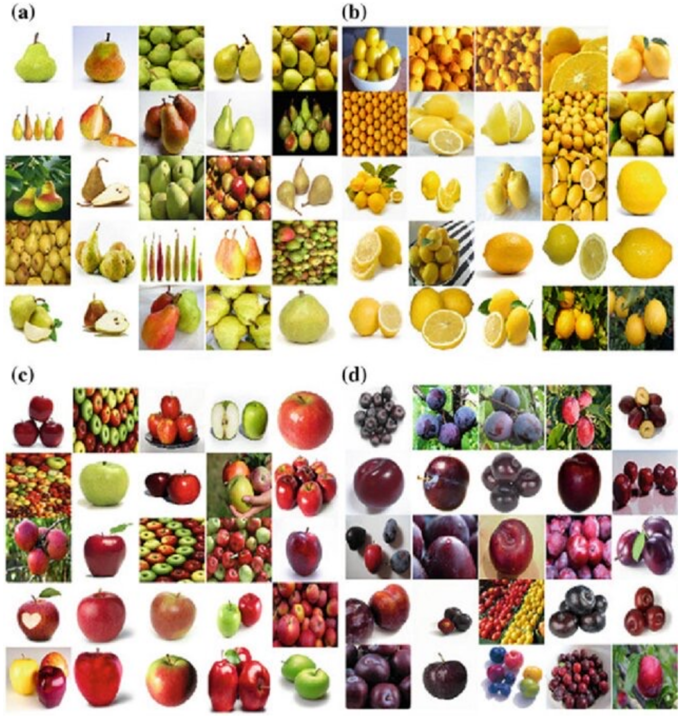
Supervised Learning: Learning from Labeled Data

- Labeled dataset: Examples with known input values and their corresponding desired outputs.
- Goal: Learn a function that maps inputs to outputs, allowing predictions on new data.
- Example tasks: Classification, regression
- Supervised learning is analogous to learning with a helpful teacher. We train a model using a dataset where correct answers are provided. The goal is to develop a function that accurately predicts the output for new, unseen input data.



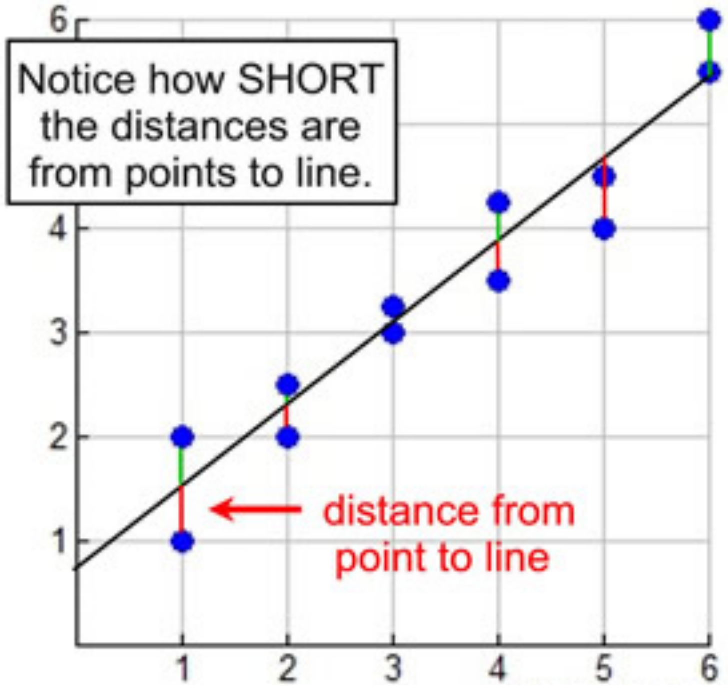
Classification

- Type of Supervised Learning
- Predicting discrete categories: Assigning data points into predefined classes (e.g., spam vs. not spam, types of flowers)
- Algorithms: Decision trees, support vector machines (SVMs), neural networks.
- Classification is about sorting data into buckets.
- A typical example is email spam filtering, where an algorithm learns to identify spam emails based on patterns in the text and other features.



Regression

- Type of Supervised Learning
- Predicting continuous values: Estimating a numerical output (e.g., predicting housing prices, stock market trends).
- Algorithms: Linear regression, polynomial regression, decision trees
- Regression is about predicting a smooth line across data.
- Think of predicting house prices based on square footage and other factors.
- Regression models try to find the best fit line to use for predictions

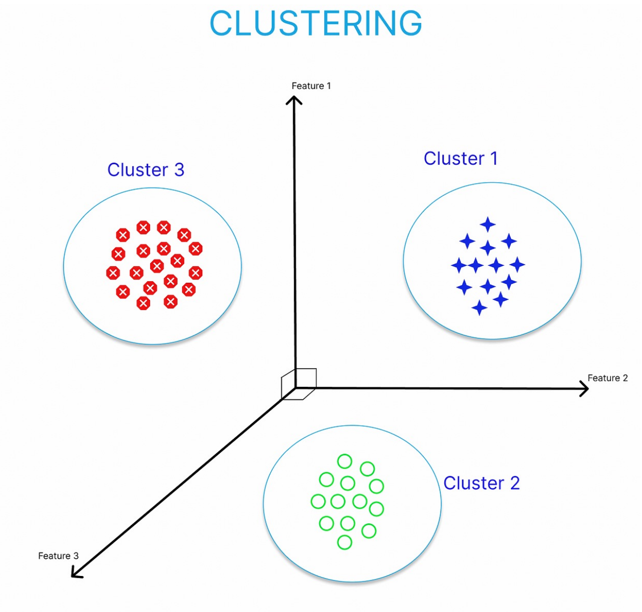


Unsupervised Learning: Finding Patterns in Data

- No labeled data: The algorithm explores the dataset without predefined answers
- Goal: Discover hidden structures, group similar data points, reduce dimensionality for better understanding.
- Example tasks: Clustering, dimensionality reduction
- Unsupervised learning is like trying to make sense of an unorganized library.
- Without labels, the algorithm aims to find natural groupings within the data or discover a more compact way to represent the information.

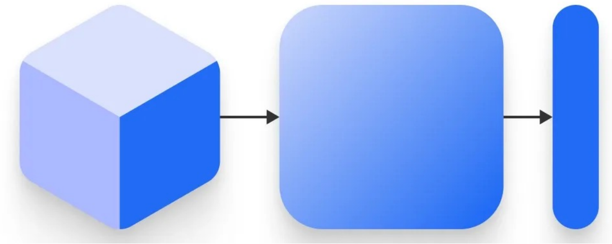
Clustering

- **Type of Unsupervised Learning**
- **Grouping similar data points:** Identifying clusters of data that share common traits.
- **Applications:** Customer segmentation, image analysis, anomaly detection
- **Algorithms:** K-means, hierarchical clustering
- Clustering helps us find natural groups. It's useful for analysing customer behaviour, grouping similar images, or detecting unusual activity in data.



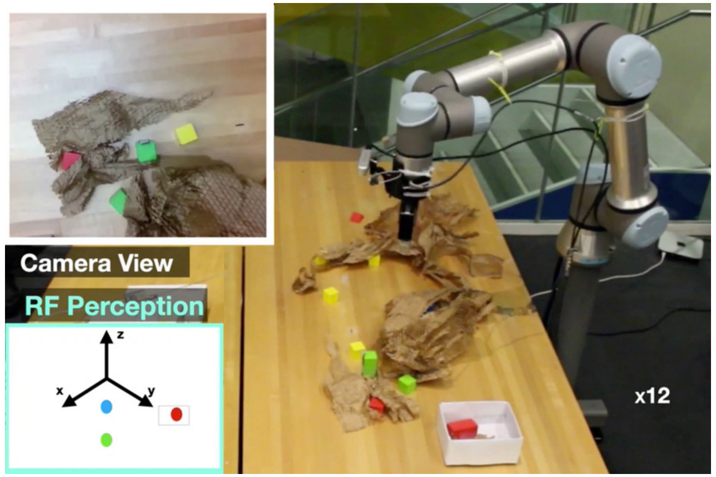
Dimensionality reduction

- Reducing the number of features: Techniques to represent data with fewer variables while preserving important information.
- Goals: Easier visualization, faster computation, combatting the "curse of dimensionality."
- Algorithms: Principal Component Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE).
- High-dimensional data can be difficult to process and interpret.
- Dimensionality reduction techniques help us project data onto a lower-dimensional space, making it easier to visualize patterns, train models more efficiently, and mitigate issues caused by too many features.



Reinforcement Learning: Learning by Doing

- Agent interacts with an environment: Takes actions, receives rewards or penalties based on outcomes.
- Policy optimization: The agent learns an optimal policy to maximize its long-term rewards.
- Applications: Game AI, robotics, self-driving cars, resource optimization.
- Reinforcement learning draws inspiration from how we learn through experience.
- An agent interacts with an environment, and through a system of rewards and feedback, it learns to perform actions that lead to the best outcomes.
- This method is powerful for problems where it's difficult to provide precise instructions, like teaching a robot to walk.



The Steps of Building an ML Model

- 1. Data collection and preparation: Gather relevant data and clean it for analysis.
- 2. Feature engineering: Select and transform features to improve model performance.
- 3. Model selection: Choose an appropriate algorithm based on the task.
- 4. Training: Feed the data to the algorithm to learn patterns.
- 5. Evaluation: Assess the model's performance on unseen data.
- 6. Hyperparameter Tuning: Adjust model settings for better results.
- 7. Deployment: Integrate the model into a real-world application.
- Machine learning is an iterative process. It starts with gathering and preparing data and involves the careful selection of features, algorithms, and tuning parameters. After training, it's important to evaluate the model thoroughly before deploying it for real-world use.

Data Preparation

- Data cleaning: Handling missing values, inconsistent formatting, outliers.
- Data transformation: Scaling, normalization, encoding categorical features.
- Data splitting: Dividing the dataset into training, validation, and testing sets.
- The quality of your data directly impacts model performance. Data cleaning and transformation ensure the data is consistent and compatible with the chosen algorithm. Data splitting is key to properly evaluate the model and prevent overfitting.

Feature Engineering

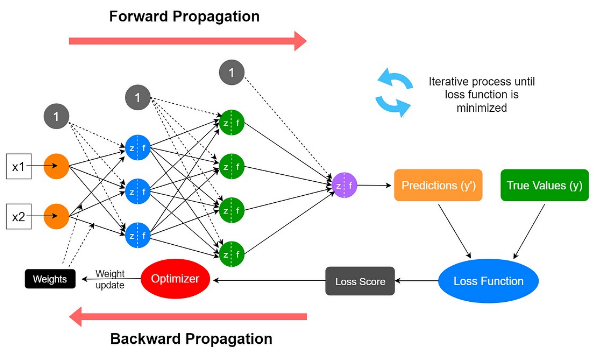
- Feature selection: Choosing the most important features for the task.
- Feature extraction: Creating new features from existing ones (e.g., combining or transforming features).
- Domain knowledge: Understanding the problem domain helps in deciding which features matter.
- Feature engineering is the art of selecting and transforming data to best represent the problem of interest. Incorporating knowledge about your specific application significantly impacts how well your model learns.

Model Selection

- Consider the problem type: Classification vs. regression, nature of the data.
- Algorithm characteristics: Linear vs. nonlinear, complexity, interpretability.
- Experimentation is key: Try different algorithms and compare performance.
- There is no single best ML algorithm. The right choice depends on the nature of your data and the task you're trying to solve. It's often necessary to try several options to find the best fit.

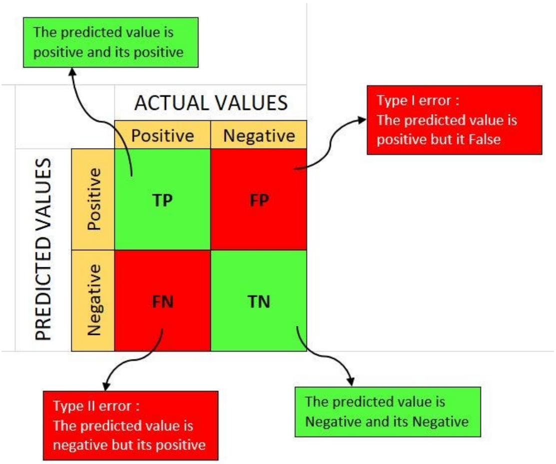
Model Training

- Iterative process: The model adjusts its parameters based on the training data.
- Loss function: Measures the error between model predictions and the true output.
- Optimization algorithm: Gradient descent and variants are commonly used to minimize the loss.
- Model training involves updating the model's internal parameters to reduce errors on the training data.
- By repeatedly calculating the error (loss function) and using optimization algorithms, the model learns to make better predictions.



Model Evaluation

- Assessing model performance: Measuring how well the model generalizes to unseen data.
- Metrics for classification: Accuracy, precision, recall, F1-score.
- Metrics for regression: Mean squared error (MSE), mean absolute error (MAE), R-squared.
- Overfitting vs. underfitting: Understanding when a model fails to generalize to new data.
- After training, we must thoroughly evaluate our model to understand its real-world performance.
- Different evaluation metrics give us insights into how well the model works.
- Recognizing signs of overfitting (too closely fitting the training data) or underfitting (not learning enough from the data) is crucial for model refinement.



Overfitting and underfitting

- Overfitting: Model performs too well on training data, but poorly on unseen data. It memorizes the noise rather than learning general patterns.
- Underfitting: Model performs poorly on both training and unseen data. It's too simple to capture the underlying pattern in the data.
- Finding the balance: Aim for a model that generalizes well, performing adequately on new, unseen data.
- Imagine overfitting like memorizing answers for a specific test, but then doing poorly on a different exam. Underfitting is like not studying enough and failing both. The key is finding a balance where your model has learned general concepts without getting sidetracked by the specifics of the training data.

Regularization

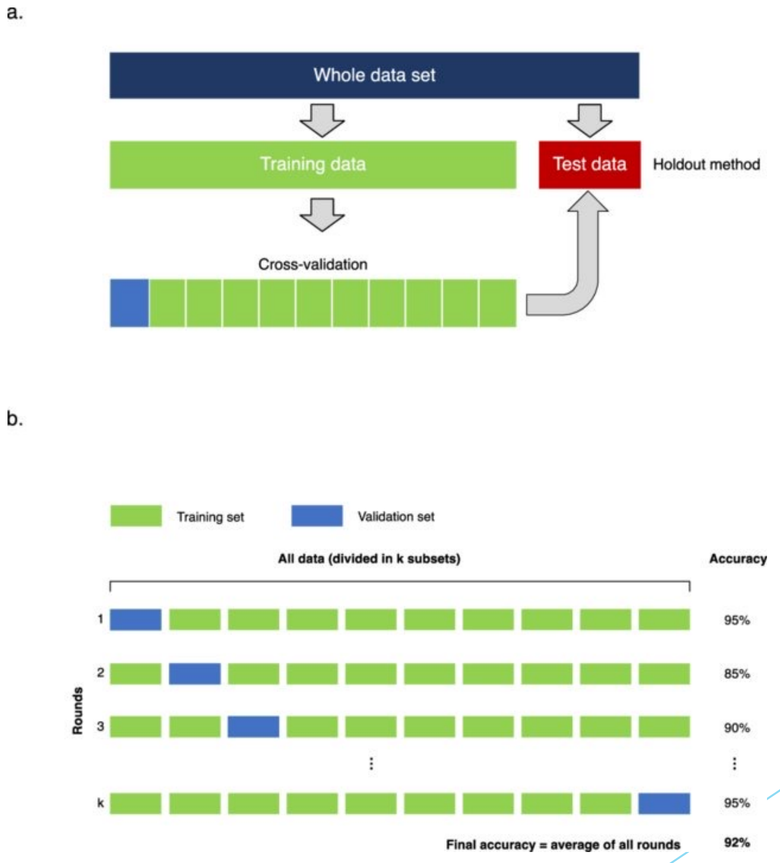
- Preventing overfitting: Techniques to encourage simpler models, reducing their tendency to overfit.
- L1 regularization (Lasso): Adds a penalty term to the loss function, encouraging some features to have zero weights.
- L2 regularization (Ridge): Adds a penalty term that shrinks feature coefficients towards zero.
- Dropout (for neural networks): Randomly deactivates neurons during training, preventing over-reliance on specific patterns.
- Regularization helps combat overfitting. Think of it as giving the model a little nudge towards simpler explanations, reducing the chance of it getting fixated on the training examples. L1 and L2 regularization penalize complex models, while dropout specifically targets overfitting in neural networks.

Hyperparameter Tuning

- Hyperparameters: Settings that control the behavior of the learning algorithm (not learned from data).
- Examples: Learning rate, number of layers in a neural network, regularization strength
- Methods: Grid search, random search, Bayesian optimization
- Goal: Finding the optimal hyperparameter configuration that maximizes model performance.
- Hyperparameters are like the knobs and dials on a machine learning algorithm.
- They influence things like how quickly the model learns or its structural complexity.
- Finding the best set of hyperparameter values can significantly enhance model performance and is a crucial step in the machine learning process.

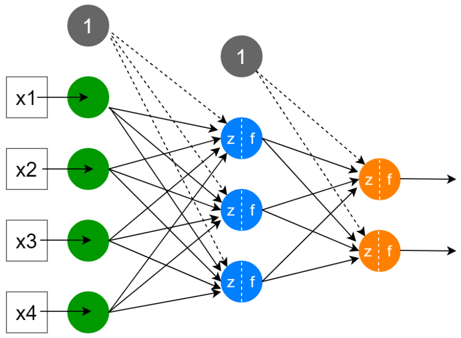
Cross validation

- More reliable evaluation: Technique for assessing model performance across varying data splits.
- Process: Dataset split into multiple folds. The model is trained on all folds except one, which is used as a validation set.
- Advantages: Reduces variance in performance estimation due to how the data was initially split into training and testing sets.
- Cross-validation ensures more robust model evaluation. Think of it as running multiple mini experiments within your dataset, where you get a better picture of how well the model would perform on different unseen data samples.



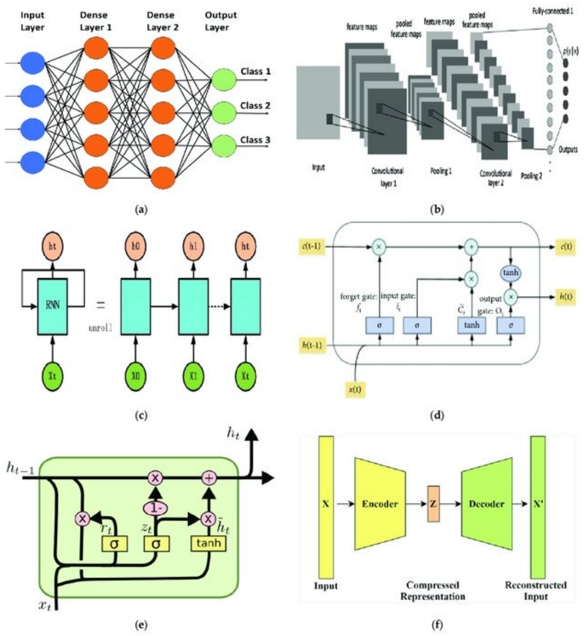
Neural Networks

- Inspired by the brain: Interconnected artificial neurons process information.
- Non-linear models: Capture complex relationships between inputs and outputs.
- Deep Learning: Neural networks with multiple hidden layers, enabling hierarchical feature learning.
- Applications: Image recognition, natural language processing, complex prediction tasks.
- Speaker Notes: Neural networks are powerful, biologically-inspired algorithms. Modeled loosely on the way neurons work in our brains, they can learn complex patterns and represent non-linear relationships between data. The development of deep learning, where neural networks use multiple layers, has revolutionized many AI tasks.



Deep learning Architectures

- Convolutional Neural Networks (CNNs): Specialized for image and spatial data, excel at tasks like image classification.
- Recurrent Neural Networks (RNNs): Designed for sequential data (e.g., text, time series), used in natural language processing and forecasting.
- Transformer Architectures: State-of-the-art for NLP tasks, revolutionizing machine translation, text generation, and more.
- Deep learning offers a variety of architectures tailored for specific tasks.
- CNNs extract features from image data in a hierarchical way.
- RNNs are designed to handle sequences and maintain memory of previous information.
- Transformer models have had a massive impact on how computers process and understand language.



The architecture of different neural networks. (a) Fully Connected Neural Network (FCNN) , (b) Convolutional Neural Network (CNN) , (c) Recurrent Neural Network (RNN) , (d) Long Short–Term Memory (LSTM) , (e) Gated Recurrent Units (GRU) , (f) Autoencoders (AE) .

Machine Learning Libraires and frameworks

- Simplify ML development: Tools that provide pre-built algorithms, data structures, and convenient abstractions.
- Popular options: TensorFlow, PyTorch, Scikit-learn, Keras, and more.
- Accelerated prototyping and deployment: Focus on building and experimenting with models.
- These libraries and frameworks are like power tools for machine learning practitioners.
- They streamline the development process, letting us focus on problem-solving and experimentation instead of reinventing the wheel with code implementation.

Machine Learning in the Cloud

- Scalability and Accessibility: Powerful cloud-based platforms for training and deploying ML models.
- Managed services: Pre-trained models, automated ML pipelines, tools for easy deployment and monitoring.
- Benefits: Reduced infrastructure costs, access to advanced compute resources, and simpler management.
- Cloud computing has democratized machine learning access. With scalable resources and managed services, individuals and organizations can train sophisticated models and deploy AI applications without large investments in infrastructure.

Conclusions

- Very particular rules on how to setup and implement these algorithms!
- The wild west days of putting math together to solve problems is over, there is a homogenized approach to compare models