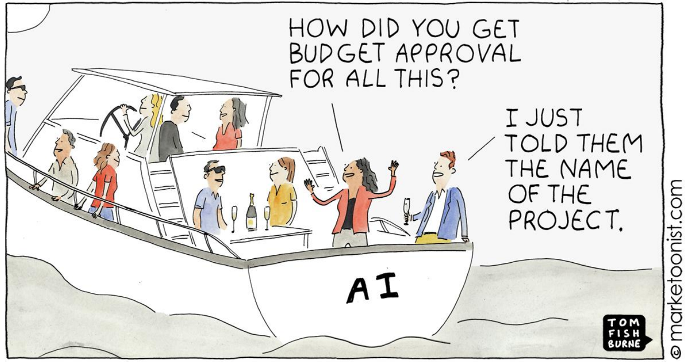


ML/AI for Software Engineers

17-313 Fall 2025

Foundations of Software Engineering

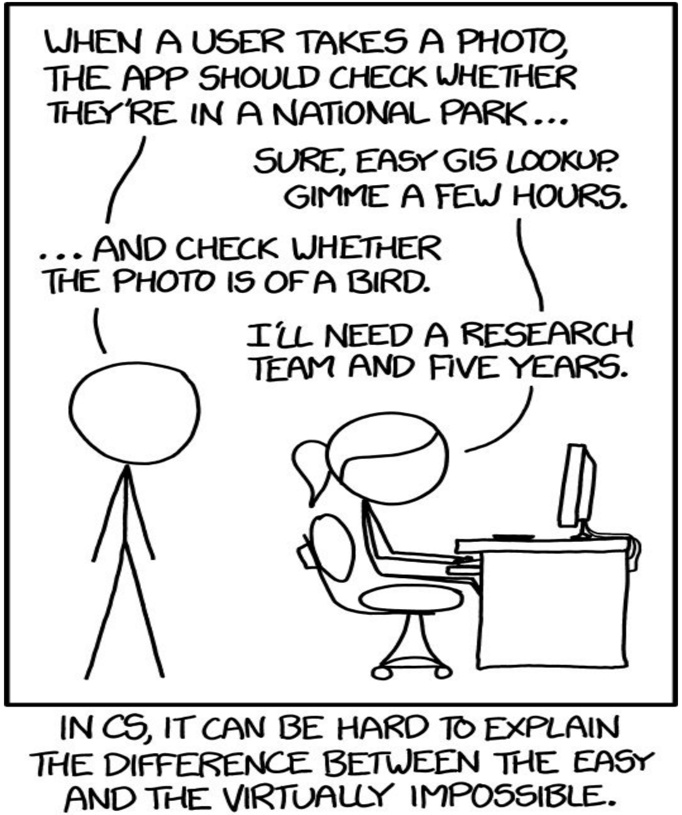
https://cmu-17313q.github.io

Eduardo Feo Flushing

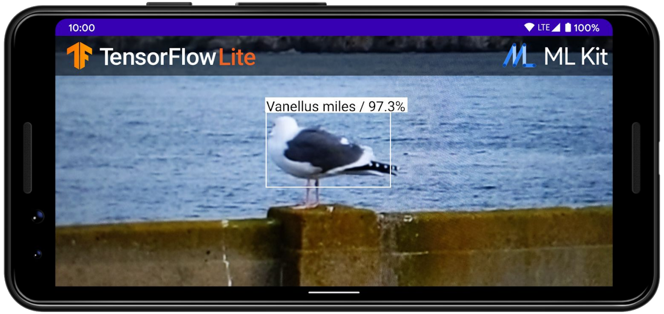
Learning goals

- Understand how machine learning (ML) components are parts of larger systems
- Explain the typical machine learning process
- Key differences from traditional software
- Distinguish between LLMs and traditional ML models in terms of flexibility, scalability, and application.
- Evaluate and improve LLM performance by understanding benchmarks, interpreting metrics like perplexity, and adjusting settings

2014



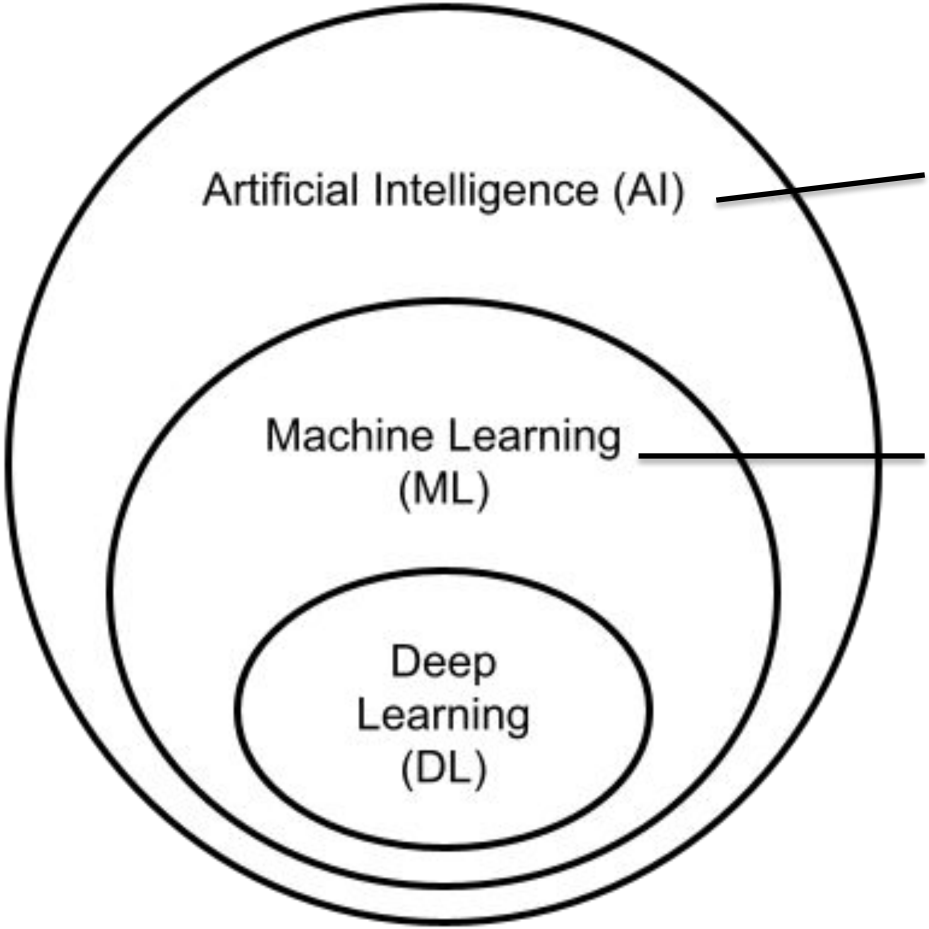
... a few years later



Definition of Artificial Intelligence (AI)

"the science and engineering of making intelligent machines”

- John McCarthy



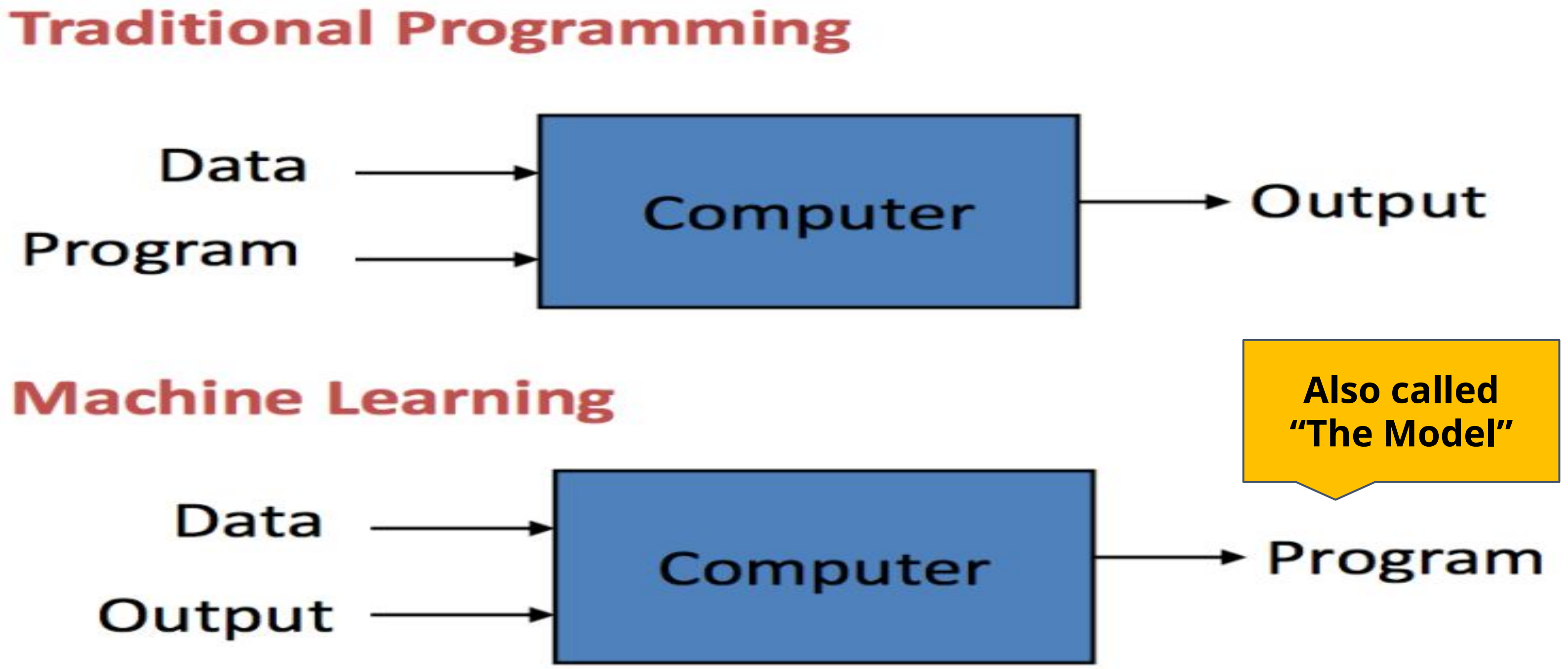
a wide range of technologies, strategies, and algorithms for machines to mimic human intelligence

subset of AI focused on the idea that machines can learn from observations or data

Outline

- Types of ML approaches
- ML Pipeline
  - Features
  - Model Building
  - Evaluation
- LLMs
  - What's the difference between traditional ML and LLMs?
  - Performance

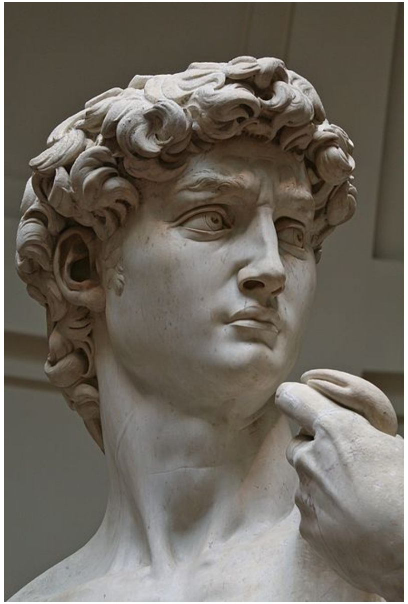
Traditional Programming vs ML



Traditional Programming

"It is easy. You just chip away the stone that doesn't look like David."

–(probably not) Michelangelo

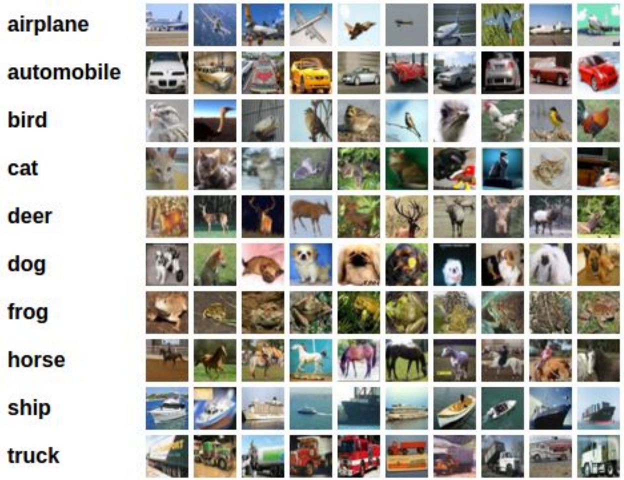


ML Development

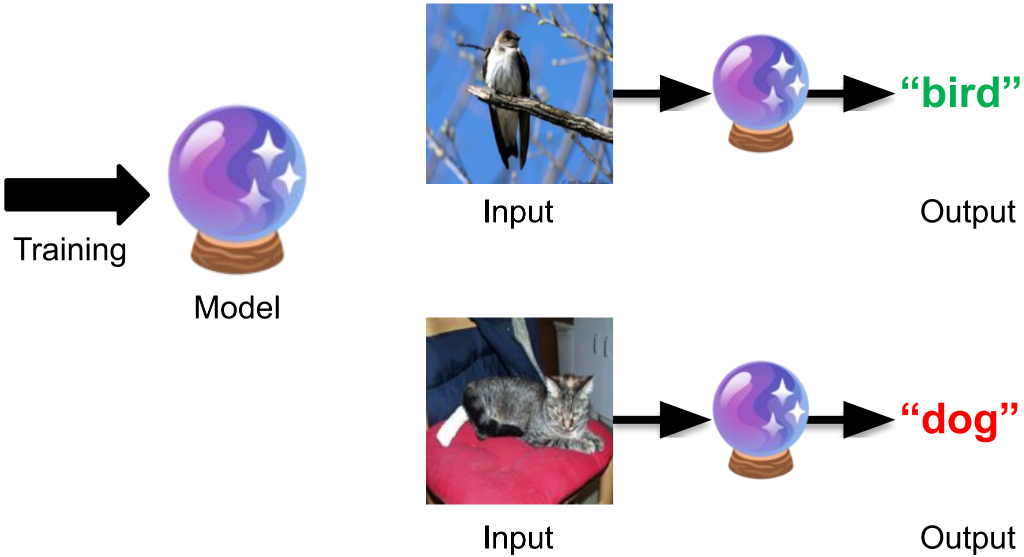
- Observation
- Hypothesis
- Predict
- Test
- Reject or Refine Hypothesis

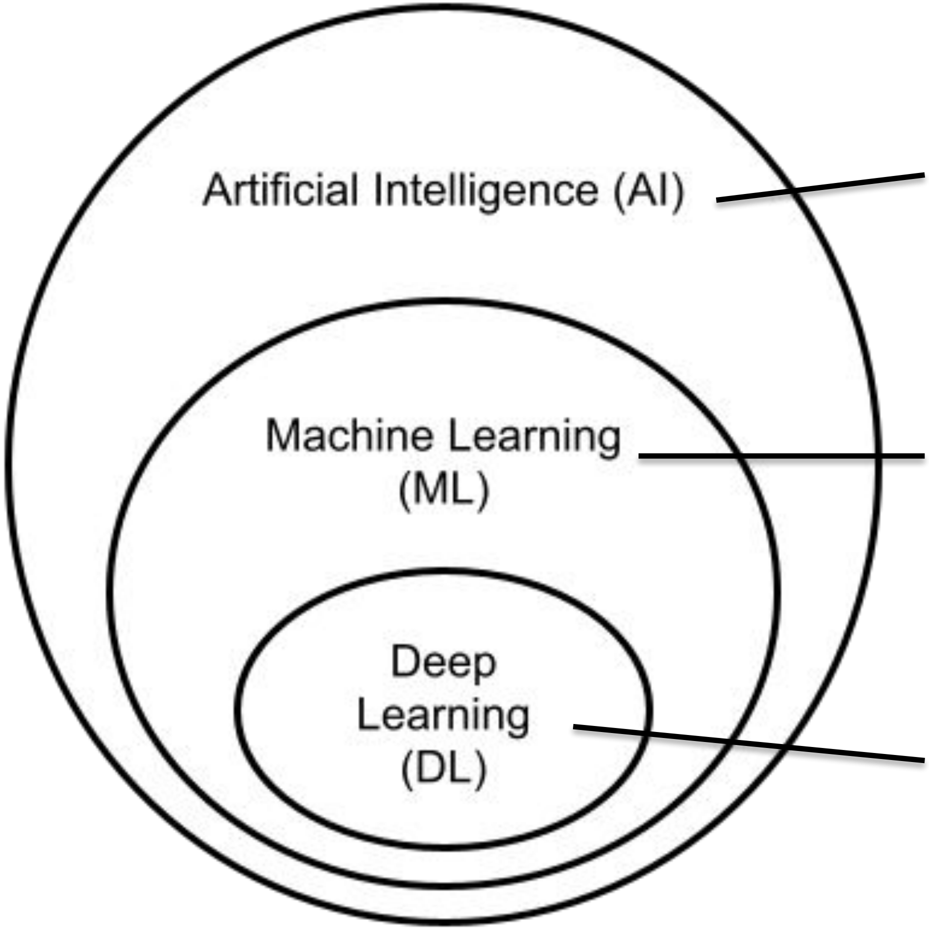
Machine Learning in One Slide

(Supervised)



Lots of labelled data
(Inputs, outputs)

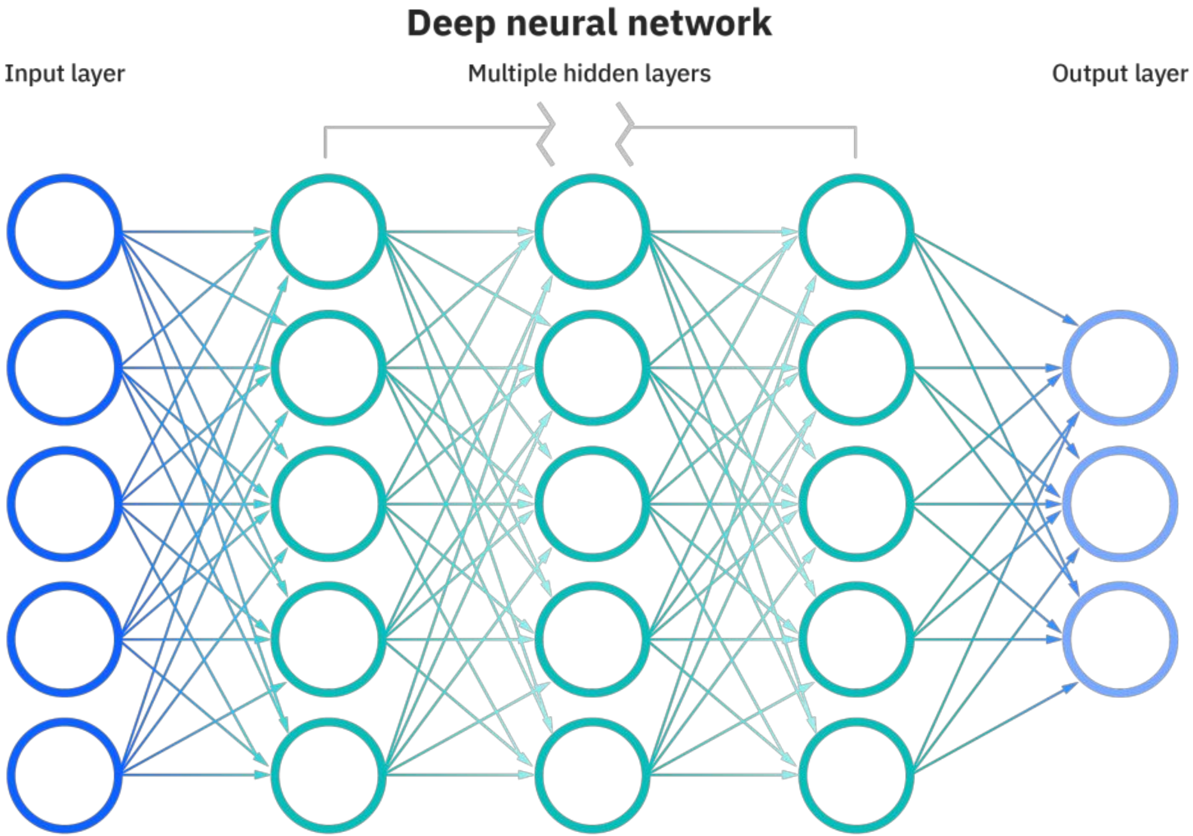




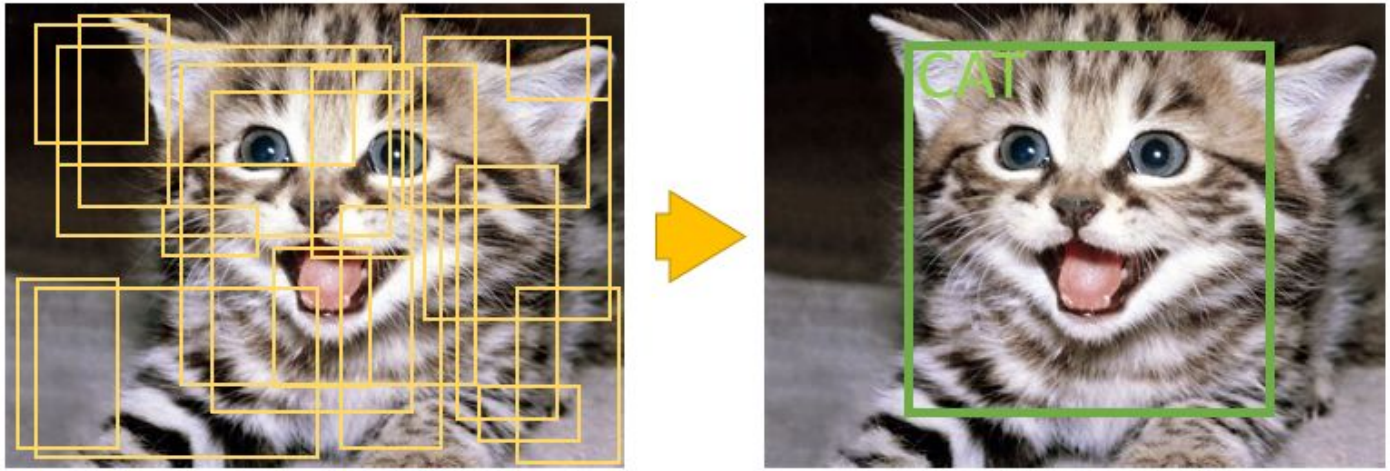
a wide range of technologies, strategies, and algorithms for machines to mimic human intelligence

subset of AI focused on the idea that machines can learn from observations or data

specialized subset of ML that uses neural networks with many layers (mimics the neural networks of the human brain)


Deep neural network
Input layer
Multiple hidden layers
Output layer

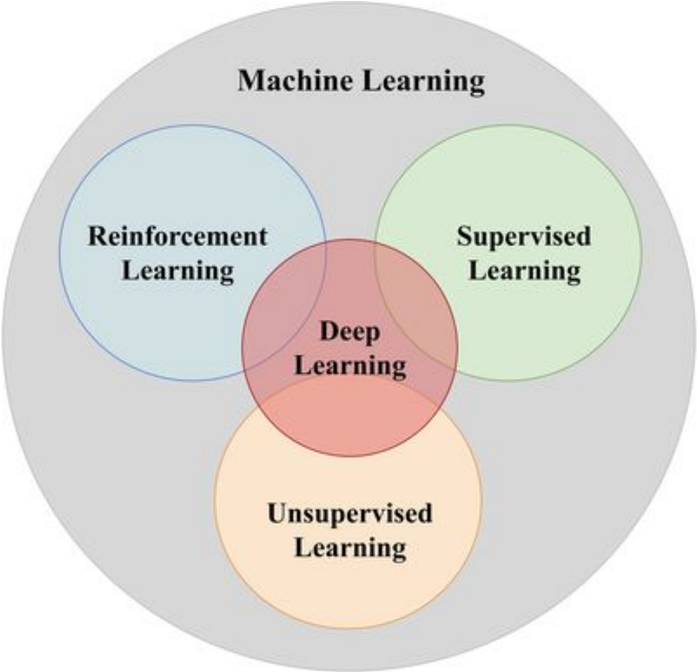
Tons of Features

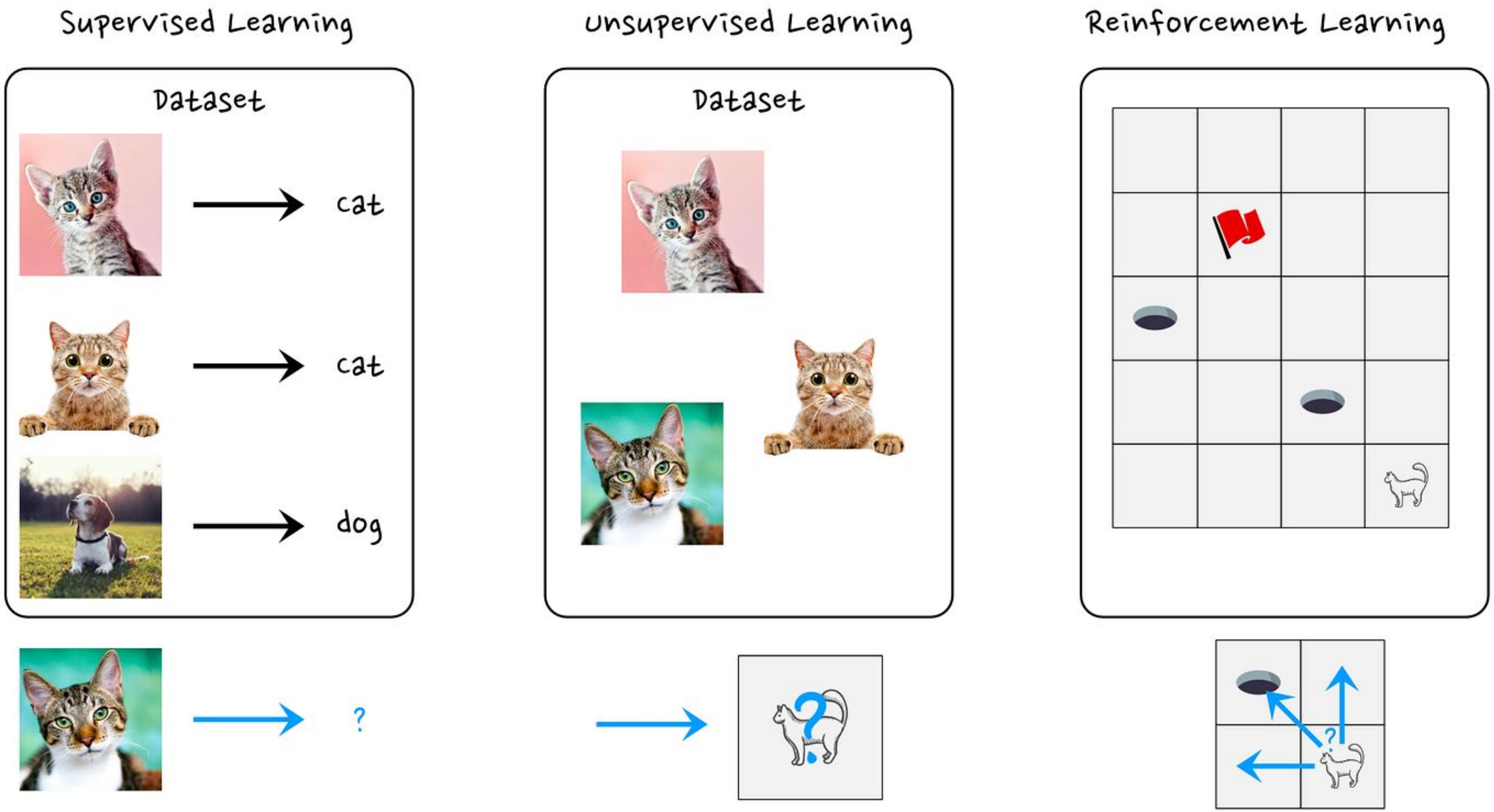


DL automates feature extraction -- handles raw data without needing human-designed features.

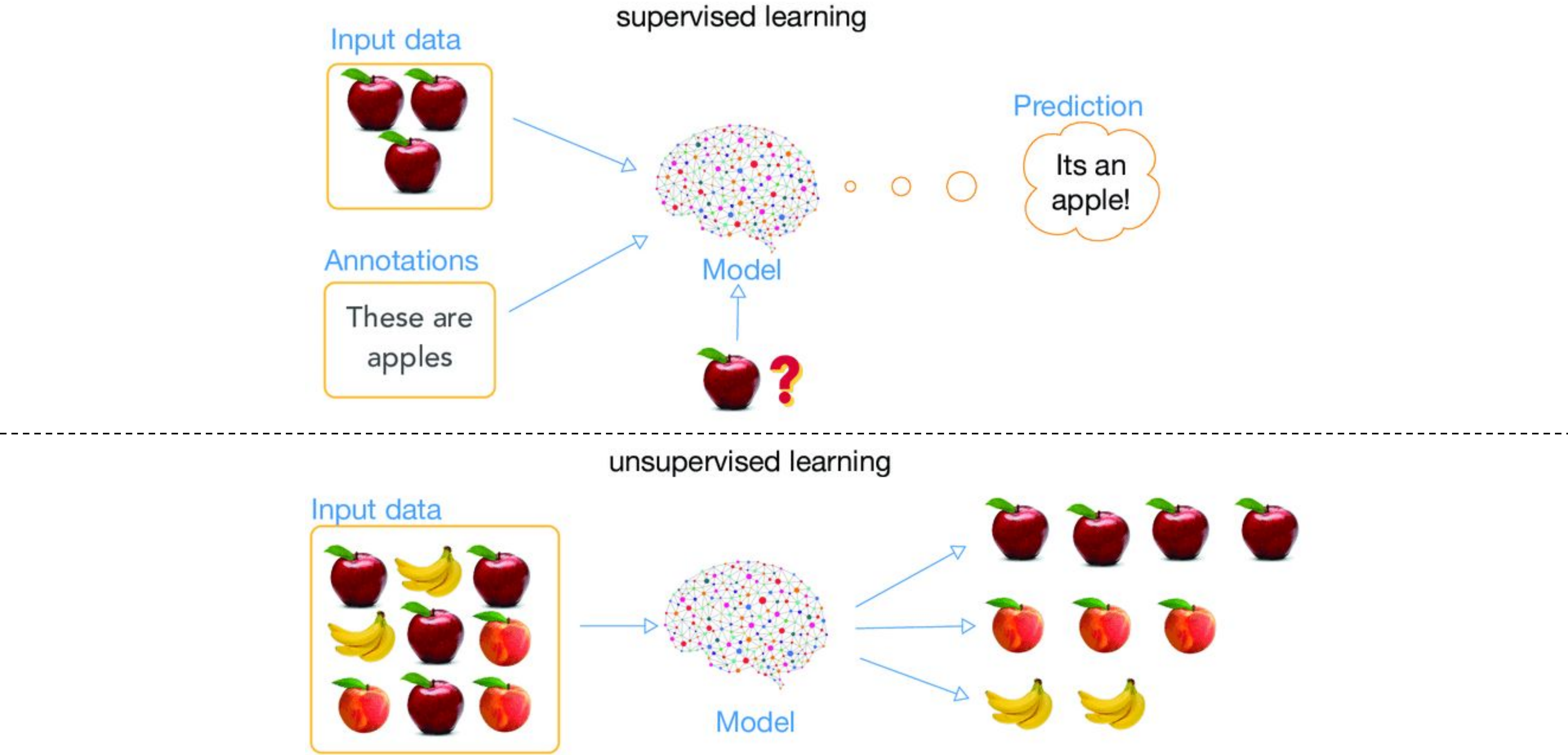
Different Categories of ML Algorithms

- Supervised
- Unsupervised
- Reinforcement Learning

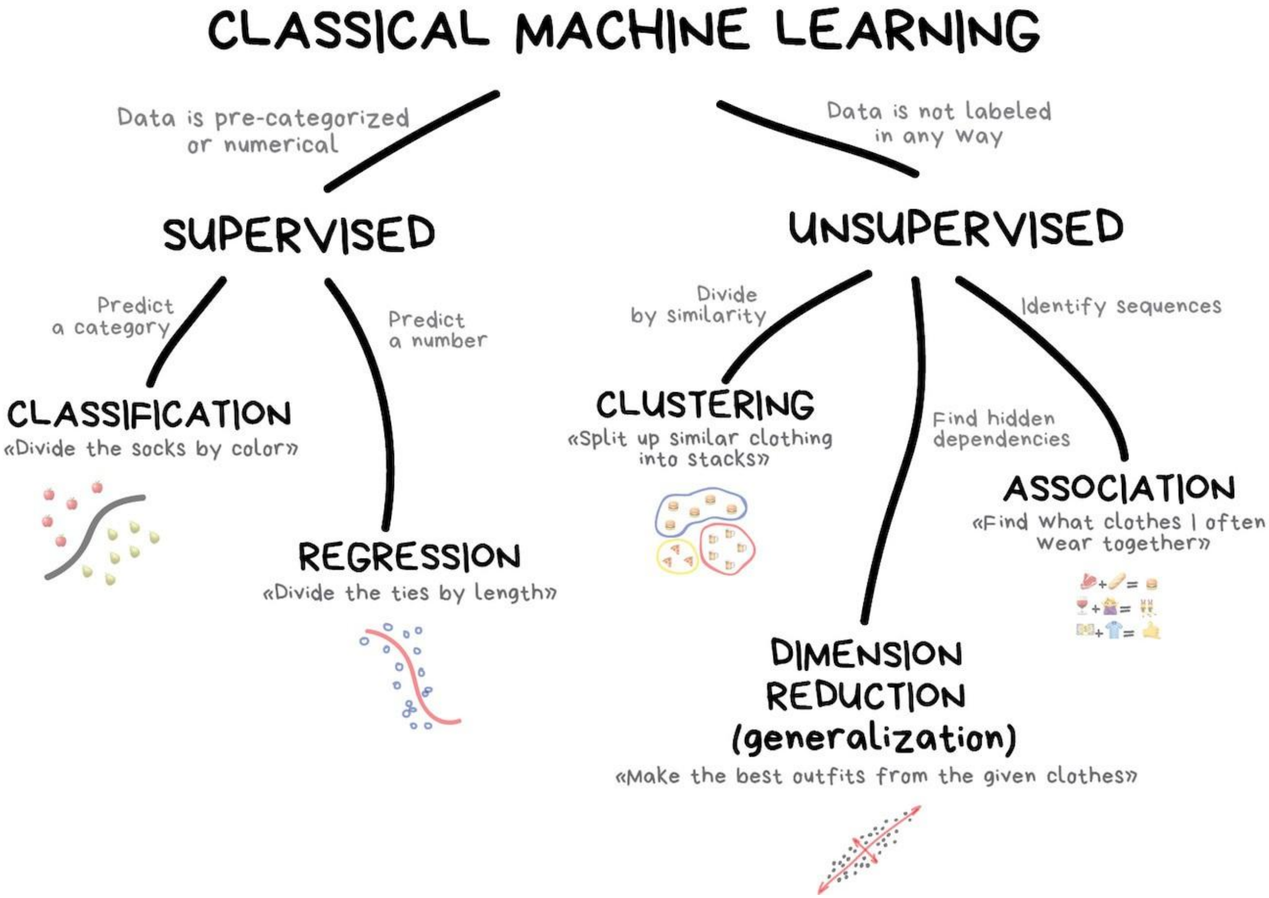




https://medium.com/@cedric.vandelaer/reinforcement-learning-an-introduction-part-1-4-866695deb4d1

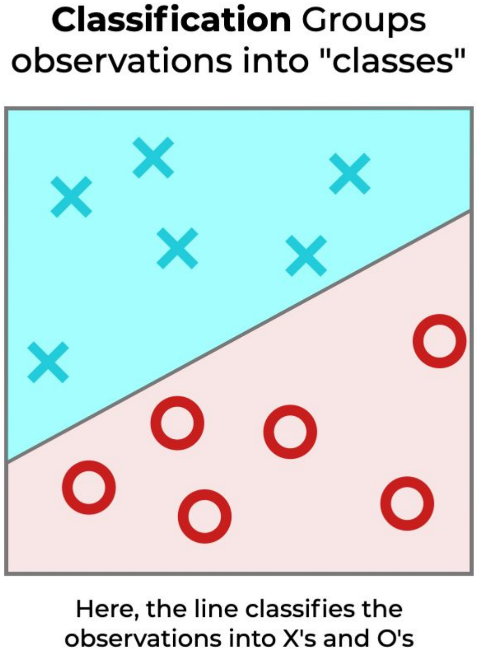


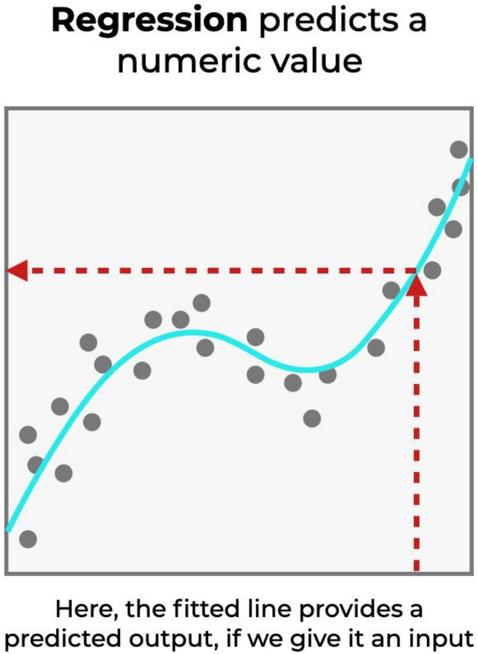
https://devopedia.org/supervised-vs-unsupervised-learning



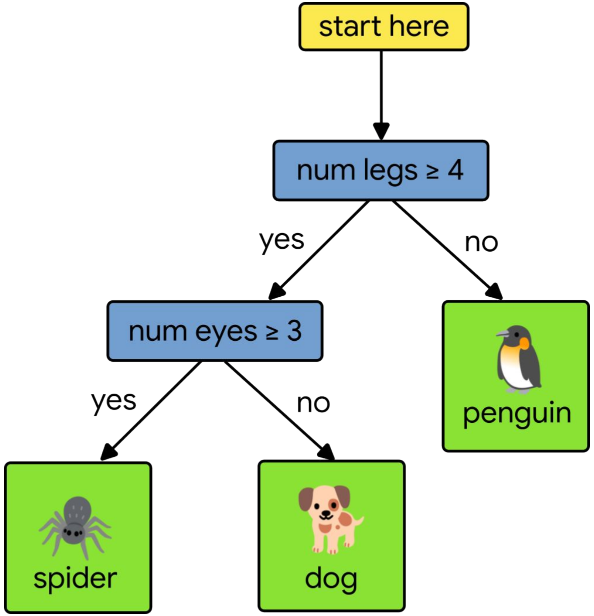
https://devopedia.org/supervised-vs-unsupervised-learning

Supervised Learning

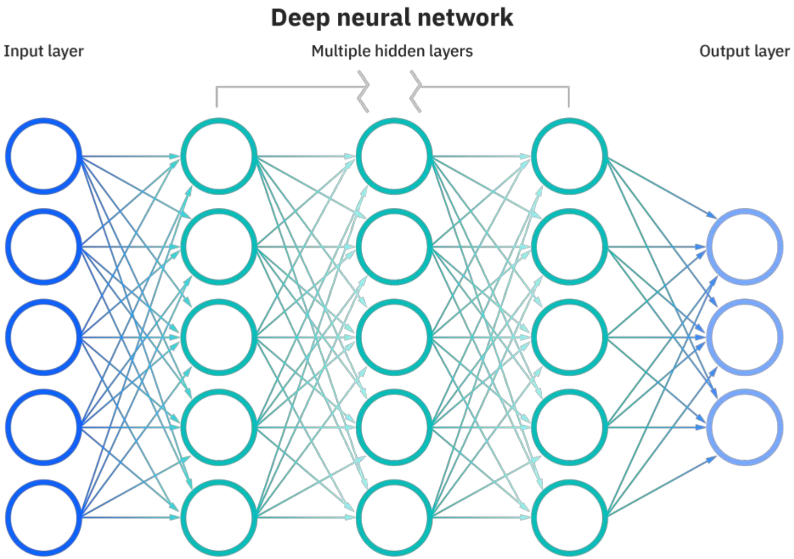




Supervised Learning: Different Complexities and Capabilities

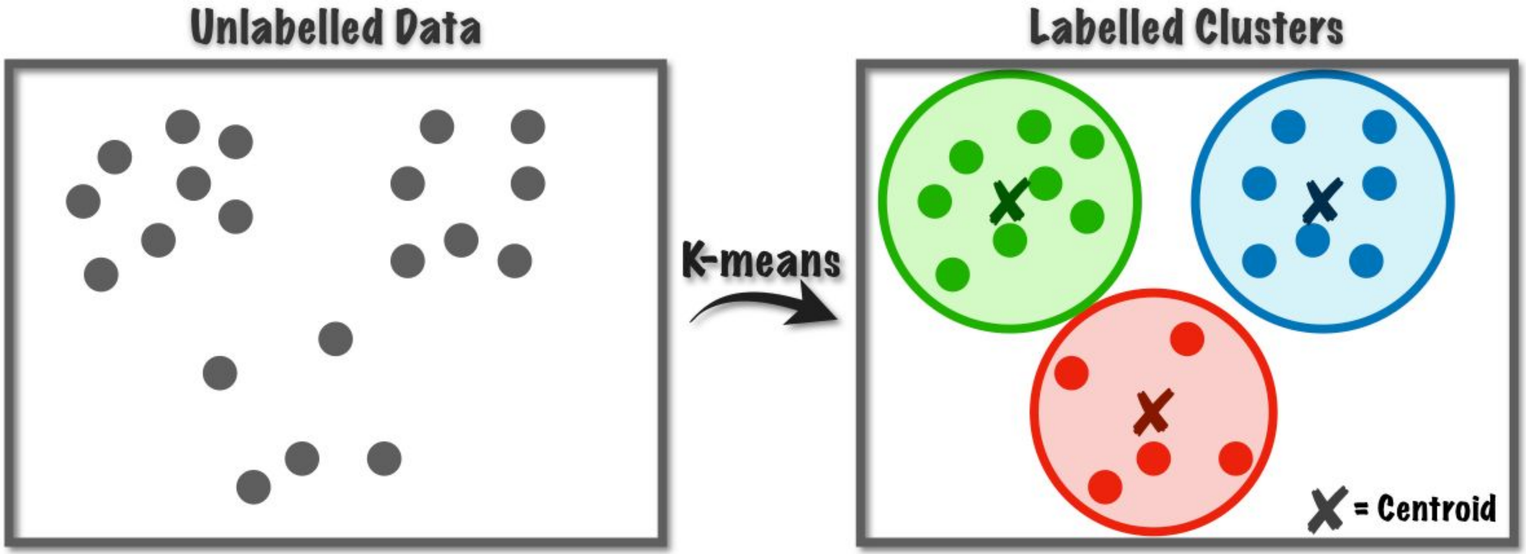


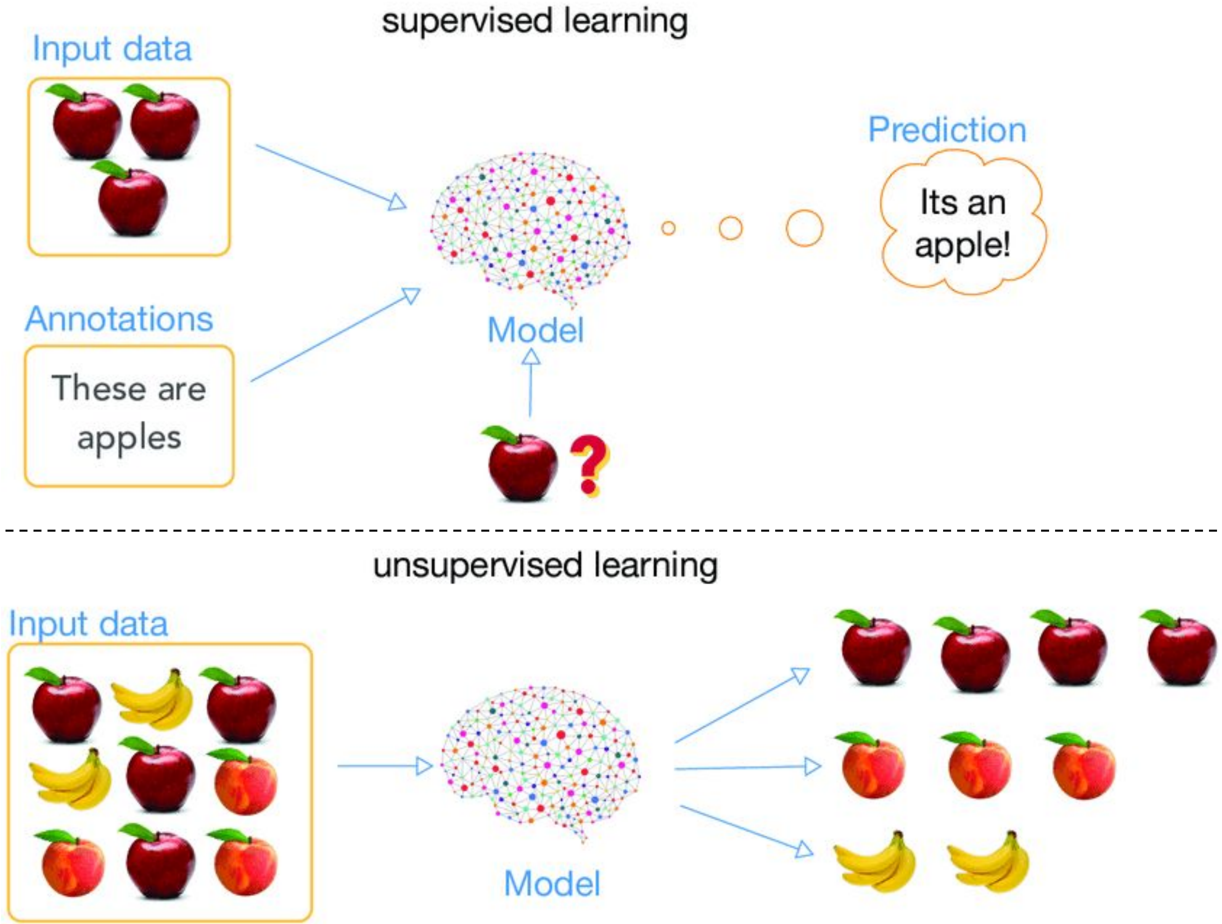
Decision Tree



Deep Neural Network

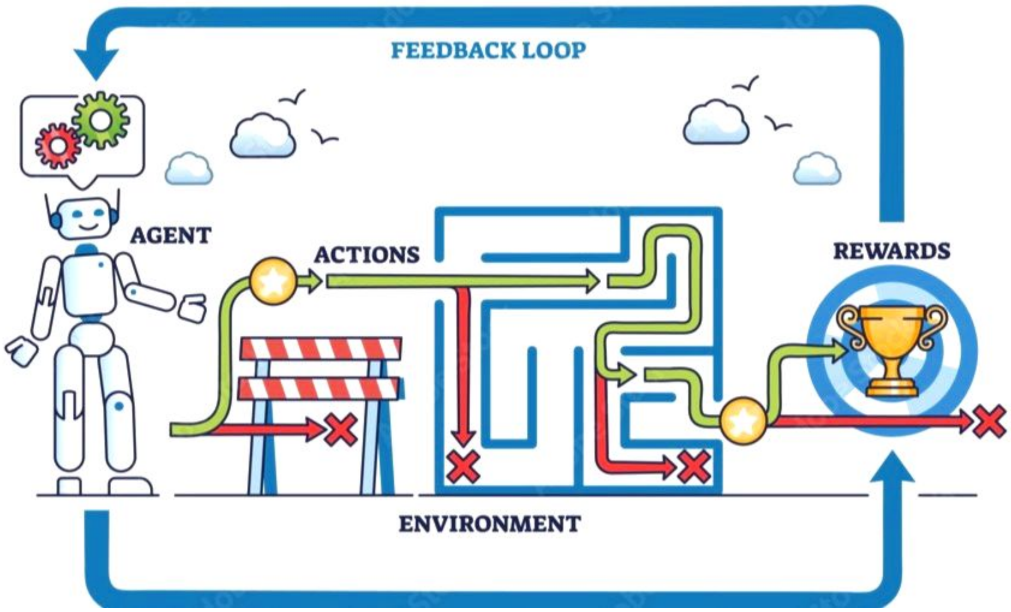
Unsupervised Learning





https://devopedia.org/supervised-vs-unsupervised-learning

Reinforcement learning



Agent: The decision-maker (the ML algorithm)

Environment: The problem space that the agent interacts with

Action: A step the agent takes to navigate the environment

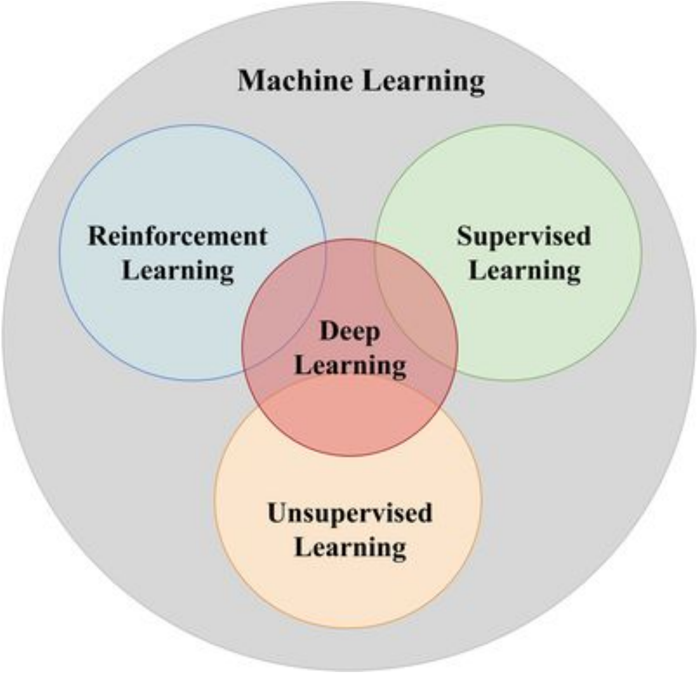
Reward: The feedback the agent receives after taking an action


Since LW-RCP is based on Matlab/Simulink,
it is easy to observe real-time data from the
real system using the 'Scope' block.


021 3 1

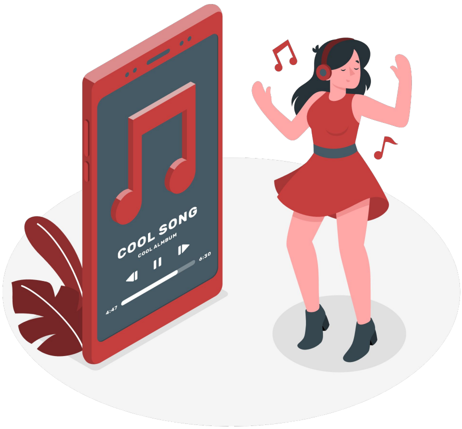
Different Categories of ML Algorithms

- Supervised
- Unsupervised
- Reinforcement Learning



Activity: Choosing the Algorithm

Three Scenarios:



Scenario A: **Music Recommendation App**

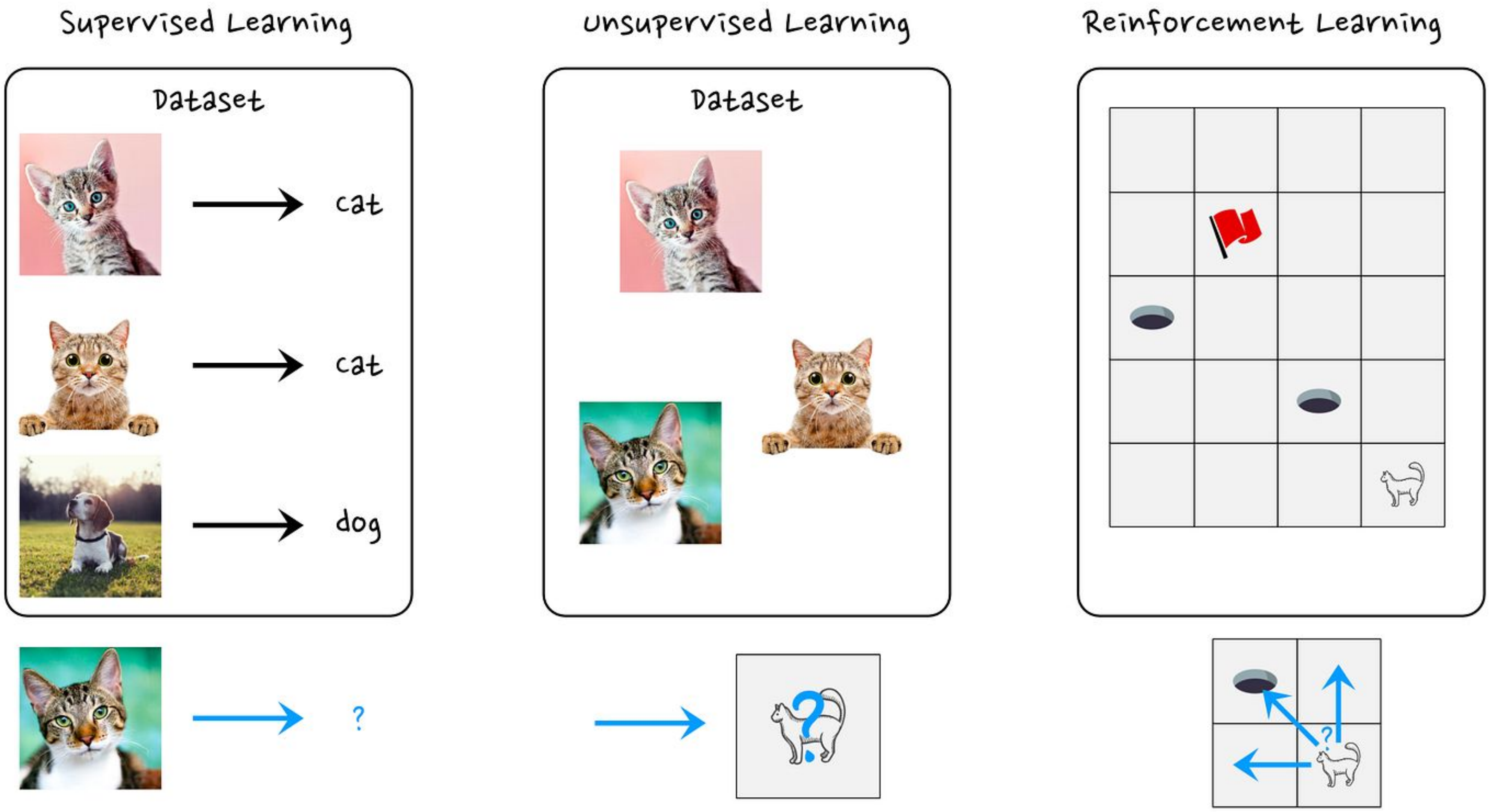
Scenario B: **Analyzing Sales Data**

Scenario C: **Adaptive Game Difficulty**

Activity: Choosing the Algorithm

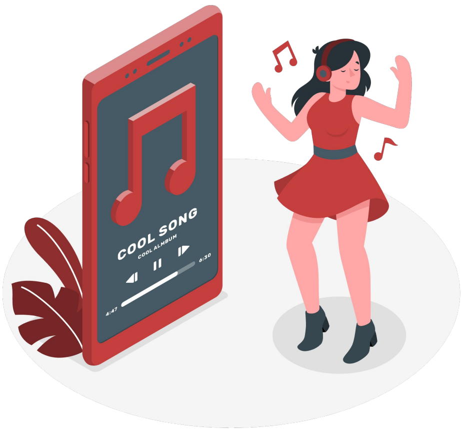
In a team of 3-4 students, for one assigned scenario:

- Discuss which learning strategies (**supervised**, **unsupervised**, or **reinforcement**) might be suitable for their scenario
- Determine why one might be more appropriate than the others.
- Consider the **nature of the data**, the **problem objectives**, and any aspects of adaptability or exploration required.



https://medium.com/@cedric.vandelaer/reinforcement-learning-an-introduction-part-1-4-866695deb4d1

Activity: Choosing the Algorithm



Scenario A: **Music Recommendation App**

Supervised Learning: train model on historical data; use labeled data of past user preferences to predict new songs they might like.

Unsupervised Learning: use clustering techniques to group similar music or users to offer recommendations within those clusters.

Reinforcement Learning: adapt to user feedback (likes/dislikes) over time to improve recommendations, learning optimal strategies through reward signals.

Activity: Choosing the Algorithm

Scenario B: **Analyzing Sales Data**

Supervised Learning: use historical sales data to train predictive models for forecasting future sales based on labeled outcomes (e.g., sales figures).

Unsupervised Learning: cluster analysis can identify groupings or patterns in products frequently purchased together without prior labels.

Reinforcement Learning: not a typical choice

Activity: Choosing the Algorithm

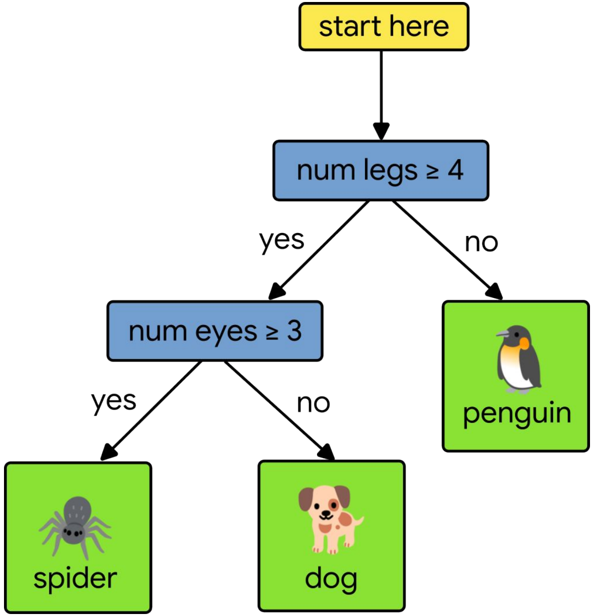
Scenario C: **Adaptive Game Difficulty**

Supervised Learning: use labeled outcomes of previous game sessions for modeling difficulty adjustments based on historical performance data

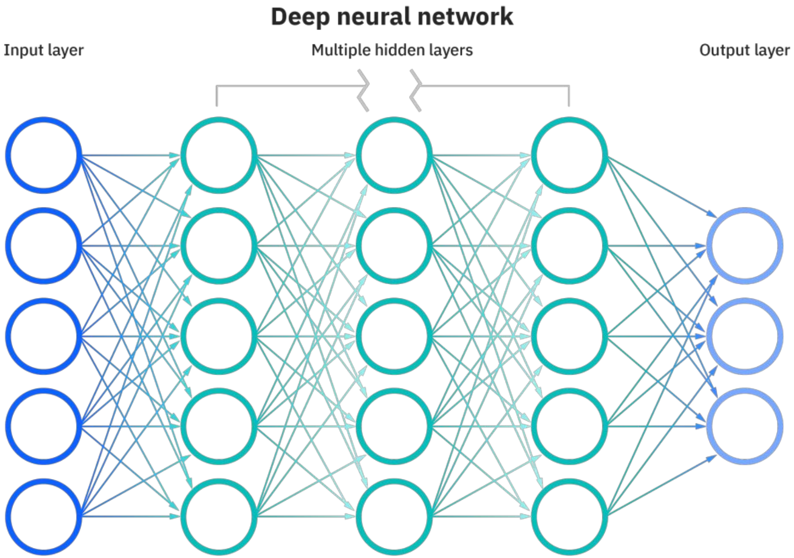
Unsupervised Learning: not typically the primary approach.

Reinforcement Learning: adapt difficulty levels dynamically based on player performance feedback using reward signals (e.g., player scores or game duration)

Tradeoffs



Decision Tree

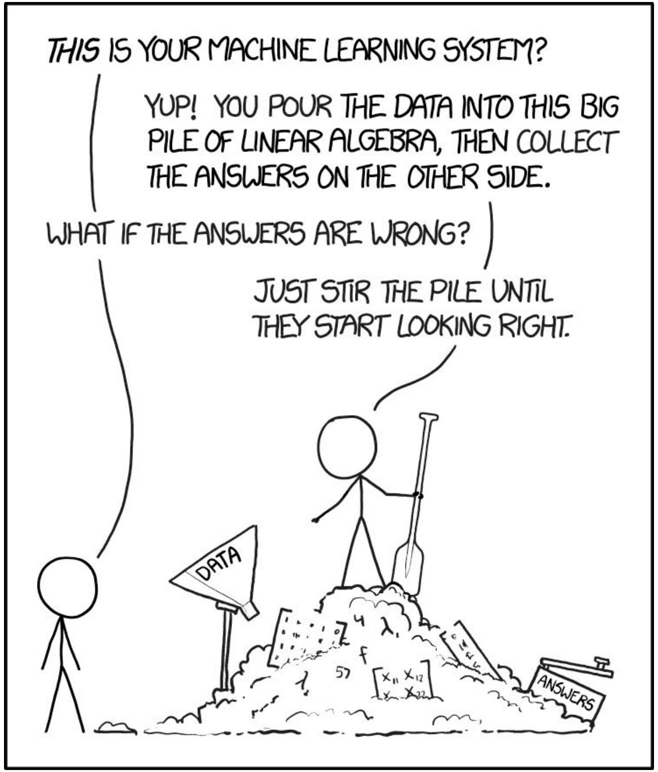


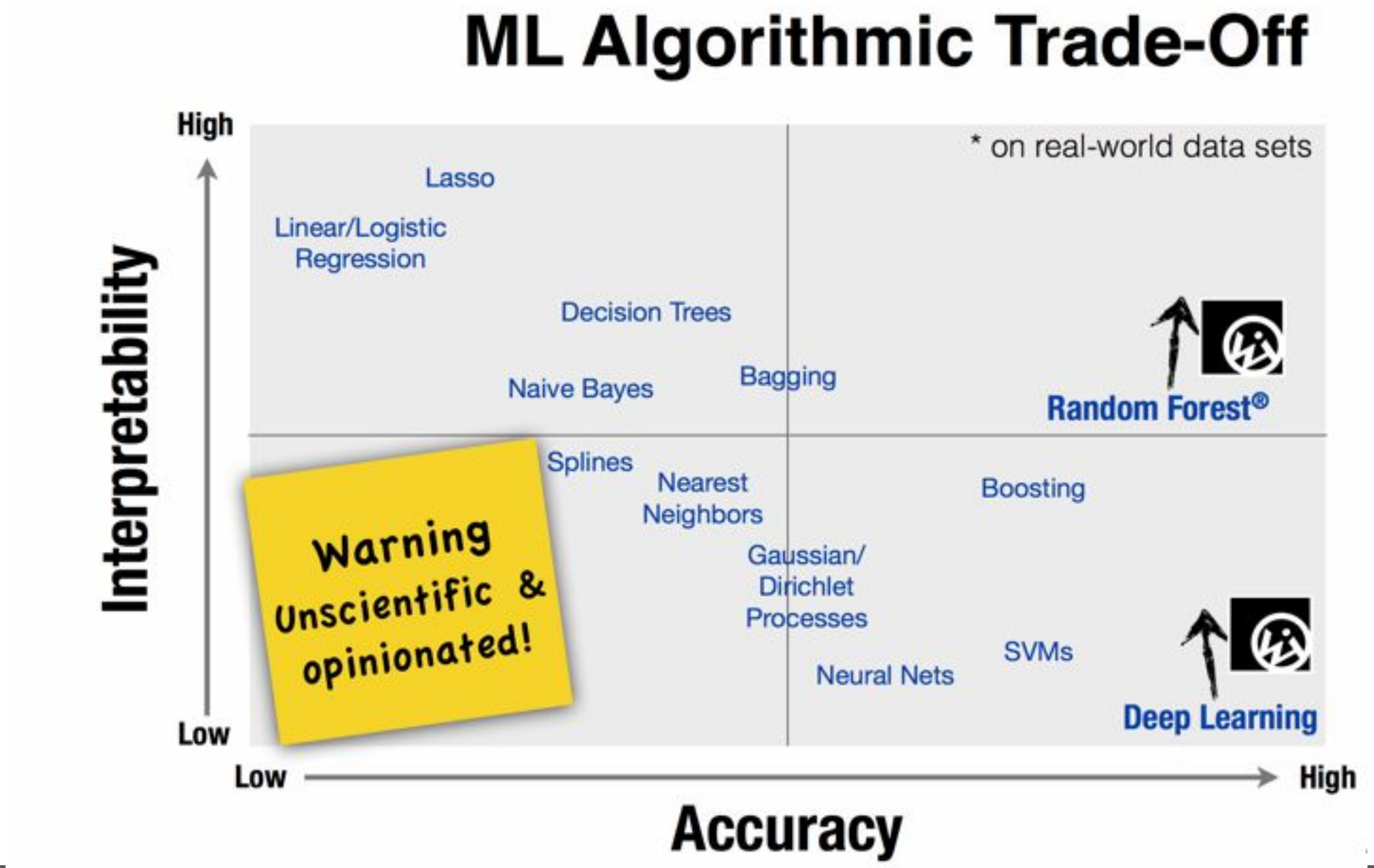
Deep Neural Network

Tradeoffs

- Accuracy
- Capabilities (e.g. classification, recommendation, clustering…)
- Amount of training data needed
- Inference latency
- Learning latency
- Model size
- Explainable
- …

Black-box view of ML

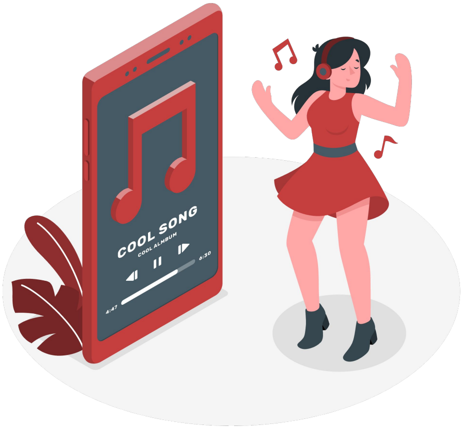




ML Algorithmic Trade-Off
High
Interpretability
Low
* on real-world data sets
Lasso
Linear/Logistic
Regression
Decision Trees
Naive Bayes
Bagging
Random Forest®
Splines
Nearest
Neighbors
Boosting
Gaussian/
Dirichlet
Processes
SVMs
Neural Nets
Deep Learning
Warning
Unscientific &
opinionated!
Low
High
Accuracy

Which ones are more important?

Accuracy, latency, model size, explainability



Scenario A: **Music Recommendation App**

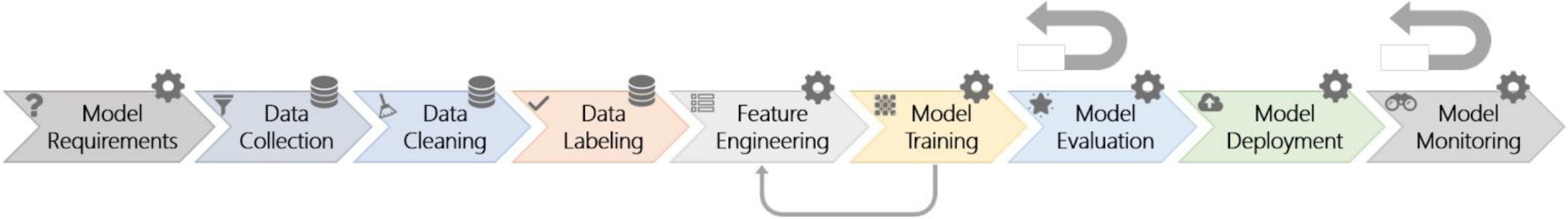
Scenario B: **Analyzing Sales Data**

Scenario C: **Adaptive Game Difficulty**

Outline

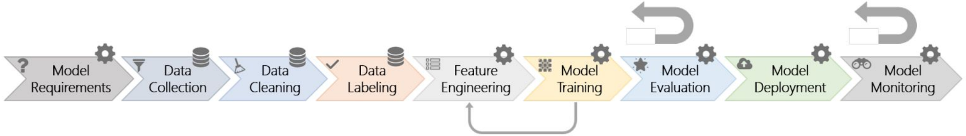
- Types of ML approaches
- **ML Pipeline**
  - Features
  - Model Building
  - Evaluation
- LLMs
  - What's the difference between traditional ML and LLMs?
  - Performance

ML Development Process (ML Pipeline)



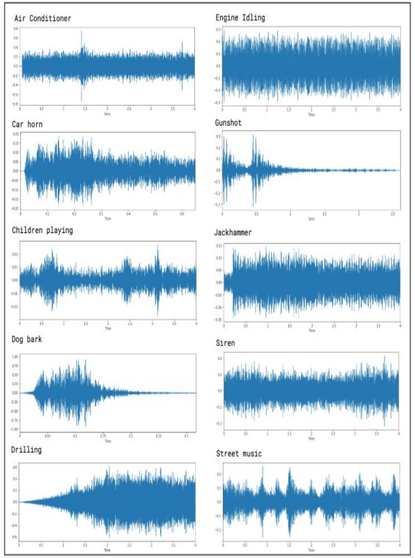
Source: “Software Engineering for Machine Learning: A Case Study” by Amershi et al. ICSE 2019

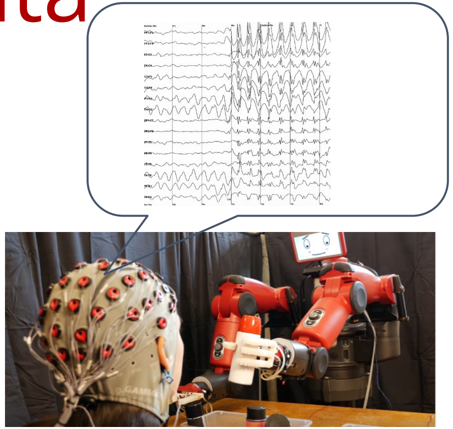
Typical ML Pipeline



- Static
  - **Get** labeled data (data collection, cleaning and, labeling)
  - Identify and **extract** features (feature engineering)
  - **Split** data into training and evaluation set
  - Learn model from training data (model **training**)
  - **Evaluate** model on evaluation data (model evaluation)
  - **Repeat**, revising features
- In production
  - Evaluate model on production data; **monitor** (model monitoring)
  - Select production data for **retraining** (model training + evaluation)
  - **Update** model regularly (model deployment)

Example Data





UserId	PickupLocation	TargetLocation	OrderTime	PickupTime
5	....	...	18:23	18:31
...				

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	disease
63	M	3	145	233	1	0	150	0	2.3	0	0	1	Y
37	M	2	130	250	0	1	187	0	3.5	0	0	2	Y
41	F	1	130	204	0	0	172	0	1.4	2	0	2	Y
56	M	1	120	236	0	1	178	0	0.8	2	0	2	N
57	F	0	120	354	0	1	163	1	0.6	2	0	2	Y
57	M	0	140	192	0	1	148	0	0.4	1	0	1	Y
56	F	1	140	294	0	0	153	0	1.3	1	0	2	N
44	M	1	120	263	0	1	173	0	0	2	0	3	N
52	M	2	172	199	1	1	162	0	0.5	2	0	3	N
57	M	2	150	168	0	1	174	0	1.6	2	0	2	Y
54	M	0	140	239	0	1	160	0	1.2	2	0	2	N
48	F	2	130	275	0	1	139	0	0.2	2	0	2	Y
49	M	1	130	266	0	1	171	0	0.6	2	0	2	Y

Feature Engineering (non DL)

- Convert raw data into a functional form
  - Transform raw data into a more compact representation that captures the most important information in the data.
- Improve the performance of models by focusing on the most relevant information in the data
  - Remove misleading things

ML Evaluation (Static)

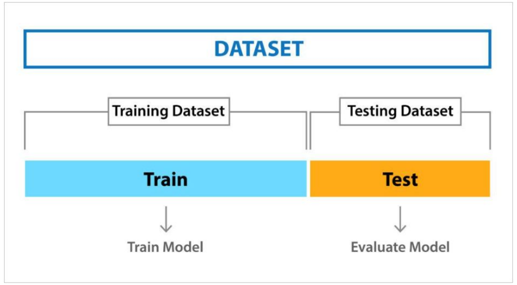
- Prediction accuracy on learned data vs unseen data
  - Separate learning set, not used for training
- For binary predictors: false positives vs. false negatives, precision vs. recall
- For numeric predictors: average (relative) distance between real and predicted value
- For ranking predictors: top-K, etc.

Evaluation

- Prediction accuracy on learned data vs. unseen data
- Why?

Evaluation

- Prediction accuracy on learned data vs. unseen data
  - Separate learning set, not used for training



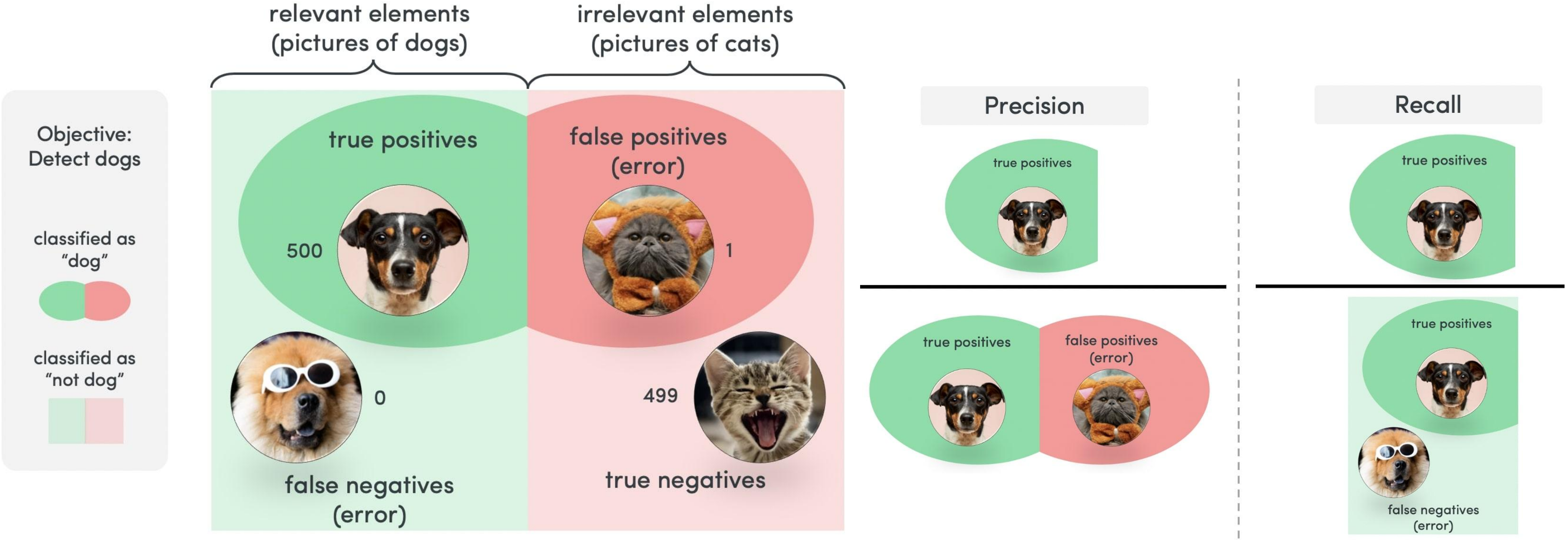
Evaluation

- Binary classification: Positive / Negative
- Possible classification outcomes:
  - TN: True Negatives
  - TP: True Positives
  - FN: False Negatives
  - FP: False Positives

Actual Class	Predicted Class	
	Negative	Positive
Negative	**TN**	**FP**
Positive	**FN**	**TP**

Accuracy is calculated as the total number of two correct predictions (TP + TN) divided by the total number of a dataset (TP + TN + FP + FN).

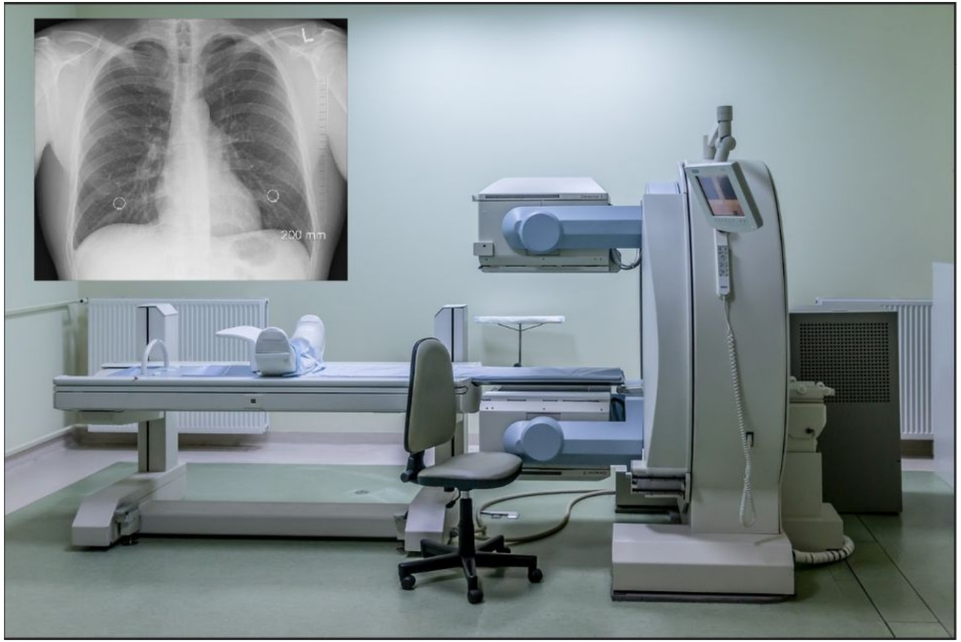
ML Evaluation (Static)



https://levity.ai/blog/precision-vs-recall

Evaluation: is model accuracy enough?

Q. Are false positives and false negatives equally bad?



Activity: False positives and false negatives, equally bad?

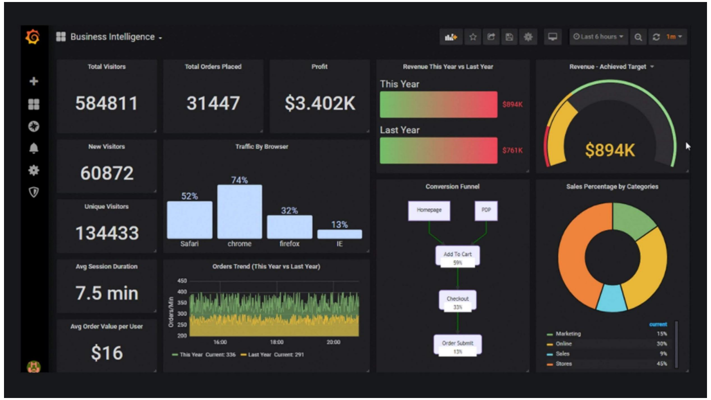
Discuss in groups these scenarios:

- Recognizing cancer
- Suggesting products to buy on e-commerce site
- Identifying human trafficking at the border
- Predicting high demand for ride sharing services
- Predicting recidivism chance
- Approving loan applications

ML Evaluation (In Production)

- Beyond static data sets, **build telemetry**
- Identify mistakes in practice
- Use sample of live data for evaluation
- Retrain models with sampled live data regularly
- Monitor accuracy and intervene

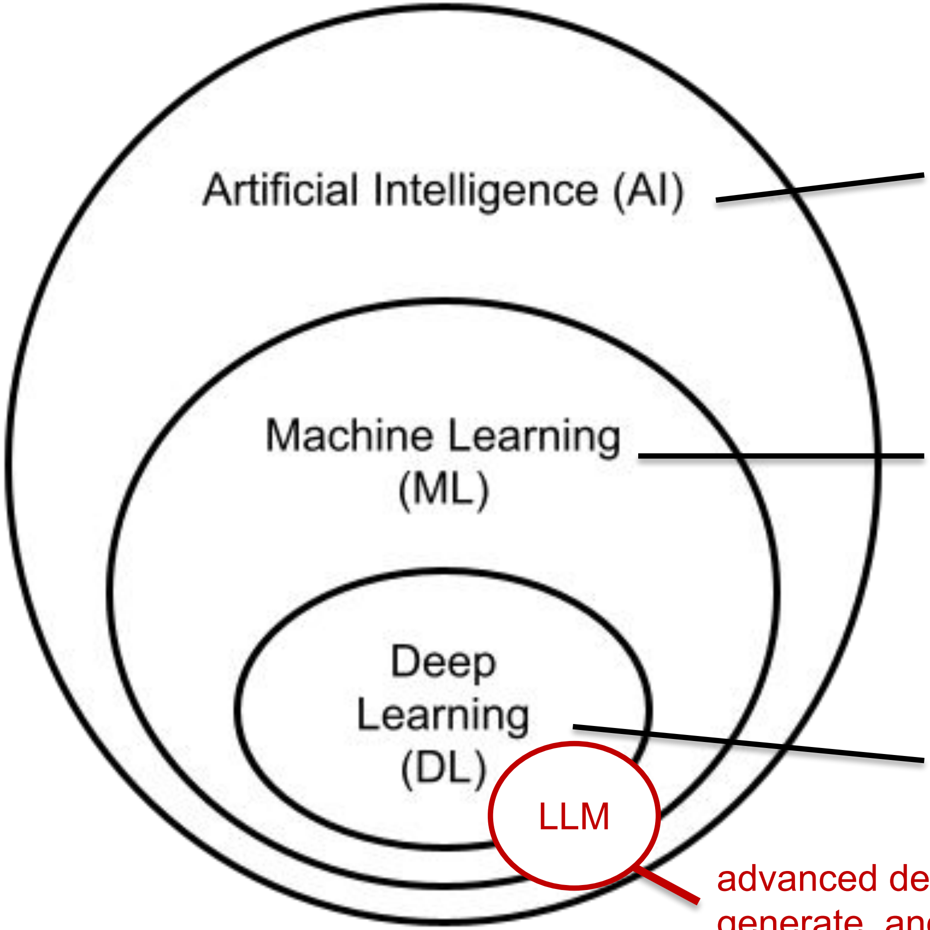
Prometheus Grafana



Outline

- Traditional Programing vs. ML
- Case Studies
- ML Pipeline
  - Features
  - Model Building
  - Evaluation
- **LLMs**
  - **What's the difference between traditional ML and LLMs?**
  - **Performance**

Large Language Models (LLMs)



a wide range of technologies, strategies, and algorithms for machines to mimic human intelligence

subset of AI focused on the idea that machines can learn from observations or data

specialized subset of ML that uses neural networks with many layers (mimics the neural networks of the human brain)

advanced deep learning models designed to comprehend, generate, and manipulate human language

Large Language Models

- Language Modeling: Learning to predict the probability of word sequences.
    - Input: Text sequence
    - Output: Most likely next word

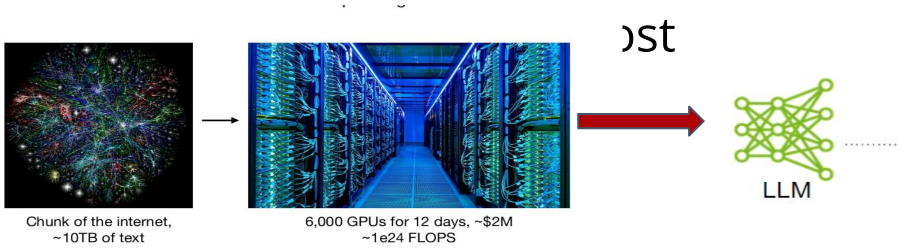
P(Eduardo, loves, his, cat) = 0.02
P(Eduardo, cat, loves, his) = 0.0001
P(Eduardo, hates, his, cat) = 0.00001

Syntactic knowledge

Semantic knowledge

Large Language Models

- Language Modeling: Learning to predict the probability of word sequences.
- LLMs are… large
  - GPT-3 has 175B parameters
  - GPT-4 is estimated to have ~1.24 Trillion
- Pre-trained with up to a PB of Internet text data
  - Massive fir ost



Large Language Models

- Language Modeling: Learning to predict the probability of word sequences.
  - Probability distribution of a sequence of words

$$\mathbf{P}(\text{Eduardo, loves, his, cat}) = \mathbf{P}(\text{cat}|\text{Eduardo, loves, his})\,\mathbf{P}(\text{Eduardo, loves, his})$$
$$= \mathbf{P}(\text{cat}|\text{Eduardo, loves, his})\,\mathbf{P}(\text{his}|\text{Eduardo, loves})\,\mathbf{P}(\text{Eduardo, loves})$$
$$= \mathbf{P}(\text{cat}|\text{Eduardo, loves, his})\,\mathbf{P}(\text{his}|\text{Eduardo, loves})\,\mathbf{P}(\text{loves}|\text{Eduardo})\mathbf{P}(\text{Eduardo})$$

- Generative Models

Large Language Models

- Autoregressive Generative Models

Eduardo loves his

Chickens	**0.99**
Wife	0.9
Job	0.85
Cat	0.8
…	< 0.8

Eduardo loves his chickens

Language Models are Pre-trained

- Only a few people have resources to train LLMs
- Access through API calls
  - OpenAI, Google Vertex AI, Anthropic, Hugging Face

- We will treat it as a **black box that can make errors**!

LLMs are far from perfect

- Hallucinations
  - Factually Incorrect Output
- High Latency
  - Output words generated one at a time
  - Larger models also tend to be slower
- Output format
  - Hard to structure output (e.g. extracting date from text)
  - Some workarounds for this (later)

USER print the result of the following Python code:

```
def f(x):
 if x == 1:
   return 1
 return x * (x - 1) * f(x-2)

f(2)
```

ASSISTANT The result of the code is 2.

Traditional ML vs LLMs

Focus and Versatility

- Traditional ML Models:
  - Broadly adaptable (e.g., image classification, fraud detection)
  - Flexible but needs task-specific feature engineering
- LLMs:
  - Specialized for language tasks
  - Ideal for chatbots, text summarization, translation

Traditional ML vs LLMs

Scale and Complexity

- Traditional ML Models:
  - Range from simple to complex; millions of parameters max
  - Optimization and fine-tuning are often simpler, with a focus on hyperparameters.
- LLMs:
  - Billions of parameters; high computational demands
  - Extensive training time on vast datasets; may take days or weeks to complete.

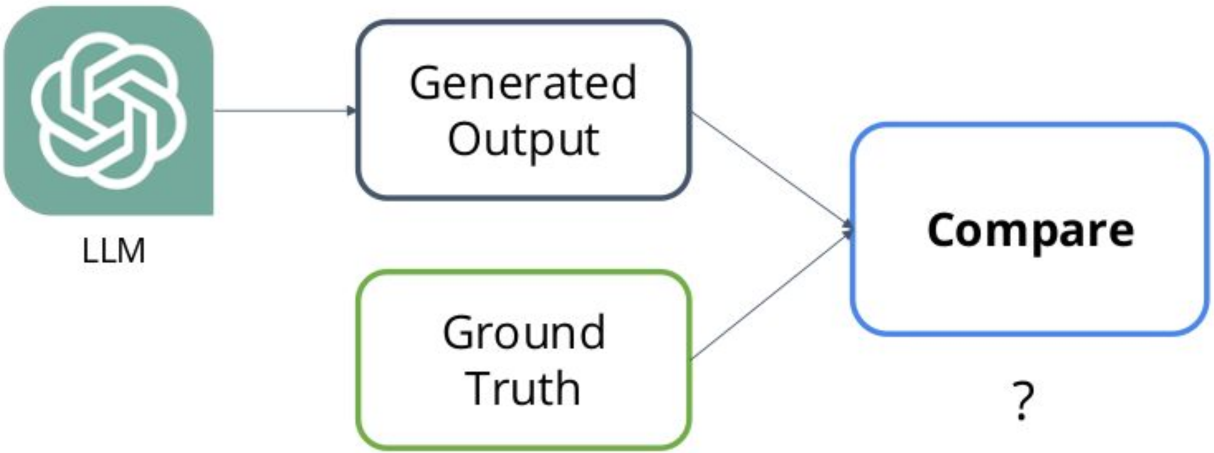
Traditional ML vs LLMs

Performance and Generalization

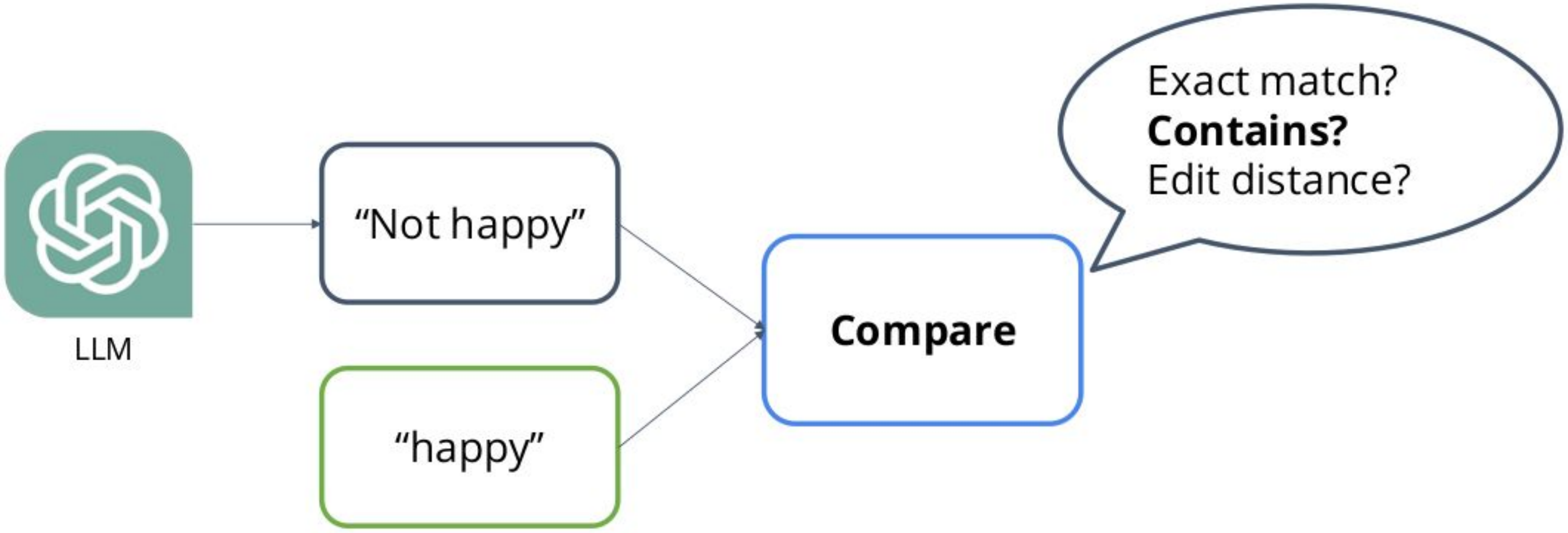
- Traditional ML Models:
  - Performance depends on feature engineering and task-specific data
- LLMs:
  - Strong generalization; adaptable to new tasks with minimal fine-tuning

Evaluation: is the LLM good at our task?

First, do we have a labeled dataset?

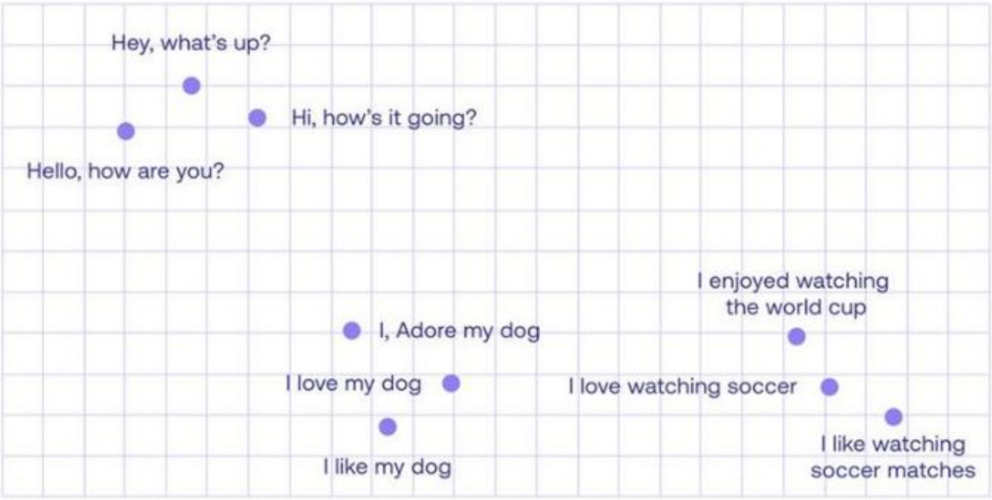


Textual Comparison: Syntactic Checks



Textual Comparison: Embeddings

Embeddings are a representation of text aiming to capture semantic meaning.



https://txt.cohere.com/sentence-word-embeddings/

LLM Evaluation

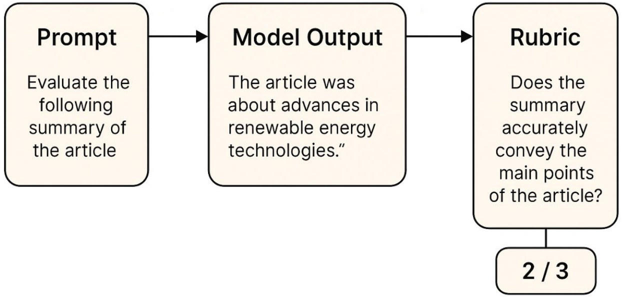
Suppose we don't have an evaluation dataset.
What do we care about in our output?

Example: Creative Writing

- Lexical Diversity (unique word counts)
- Semantic diversity (pairwise similarity)
- Bias

LLM-as-a-Judge

- Uses an LLM to evaluate other model outputs
- Works best when guided by a rubric
- Enables faster iteration and large-scale evaluation
- Risks: bias, inconsistency, requires human calibration



Improving LLM Performance

Prompt Engineering

- Rewording text prompts to achieve desired output.
- Low-hanging fruit to improve LLM performance.
- Popular prompt styles
  - Zero-shot: instruction + no examples
  - Few-shot: instruction + examples of desired input-output pairs
- Don’t be too afraid of prompt length: 1000+ words is OK

Improving LLM Performance

Chain of Thought Prompting

- Few-shot prompting strategy
- Example responses include reasoning
- Useful for solving more complex word problems [arXiv]
- Example:
  - Q: A person is traveling at 20 km/hr and reached his destiny in 2.5 hr then find the distance? Answer Choices: (a) 53 km (b) 55 km (c) 52 km (d) 60 km (e) 50km
  - A: The distance that the person traveled would have been 20 km/hr * 2.5 hrs = 50 km. The answer is (e).

Improving LLM Performance

Fine-Tuning

- Retrain part of the LLM with your own data
- Create dataset specific to your task
    - Provide input-output examples (>= 100)
    - Quality over quantity!
- Generally not necessary: try prompt engineering first.

Improving LLM Performance

Fine-Tuning Output via LLM Model Settings:

Temperature

- Controls randomness in output
- Higher values (e.g., 1.0) make responses more diverse, while lower values (e.g., 0.2) make responses more focused and deterministic.

Top-k Sampling

- Limits output choices to the top k highest-probability words, reducing unlikely words. Lower k (e.g., 10) makes responses more predictable.

Top-p (Nucleus) Sampling

- Restricts choices to a dynamic set of words with a cumulative probability threshold (e.g., 0.9). This setting balances creativity and coherence.

Improving LLM Performance

Fine-Tuning Output via LLM Model Settings:

Max Tokens

- Sets the maximum length of the output, useful for limiting responses to a specific length.

Frequency and Presence Penalties

- Frequency Penalty: Discourages word repetition by reducing the likelihood of words already used.
- Presence Penalty: Encourages or discourages certain words based on how frequently they appear.

Tailoring settings allows for better control over response style, making outputs more suitable for creative tasks, factual responses, or concise summaries

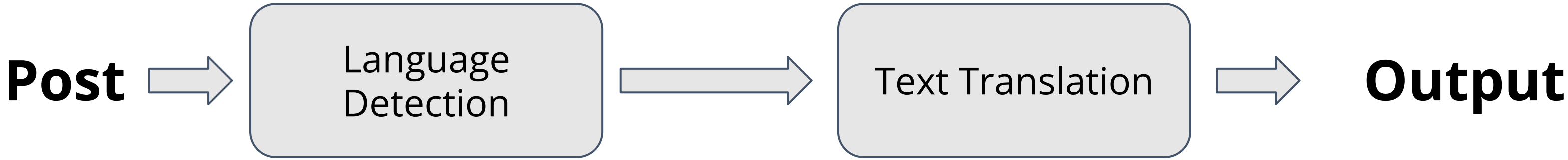
Agentic AI

- What’s an AI Agent?

  “AI-Powered software that accomplishes a goal”

  \- Dharmesh Shah

- Agentic AI systems use **multi-step workflows** that combine **one or more LLM calls**, tool use, and human input to accomplish tasks autonomously.



Agentic AI: Example Workflows

- Reflection
  - Agents critique & improve their own output
- Tool Use
  - Agents call APIs, retrieve information from databases, or write code
- Planning
  - Agents autonomously break complex goals into sub-tasks & execute adaptively
- Multi-Agent Collaboration
  - Specialized agents coordinate

Next class …

Why ML/AI projects fail?

What's wrong with the model-centric pipeline?

Are there any new challenges?

What is ML Ops?