DE VINCI

ESILV
ENGINEERING SCHOOL
DE VINCI PARIS NANTES

ADVANCED ML I

SAKINA TAKACHE, FALL 2024

PÔLE LÉONARD DE VINCI

CONTENTS

- ENSEMBLE LEARNING, IN 2 PARTS
- **HANDLING IMBALANCED DATA**
- **REINFORCEMENT LEARNING & AUTOML**

GRADING

30%

- 3 homework submissions (individual work at the end of the TD files) due one week after your last TD session (varies with group)
- Each instance of participation is equivalent to 1 perfect homework (10%)

→ **you can submit one less homework for each time you participate**

20%

- Research activity in groups of 3-4 on Variable Correlation & Bivariance (topic not discussed in class)
- To be submitted in whatever format you find suitable (.ipynb, PDF, .pptx, …)
- Due next week 04/12/2024 23:59 CET

50%

- Comprehensive project as individual or group work covering the lecture topics (detailed on DVL) due (?)
- To be submitted with executable code (.ipynb format, with markdown notes explaining your work, is preferred)

IMBALANCED DATA

HANDLING

(Image not by Author)

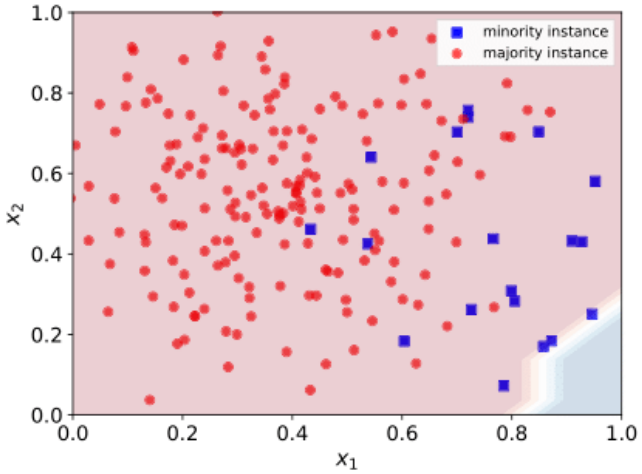
HANDLING IMBALANCED DATA

Contents

IMBALANCED DATA
WHAT IS IT?

Definition: Imbalanced data occurs when one class significantly outweighs the other(s) in a classification problem.

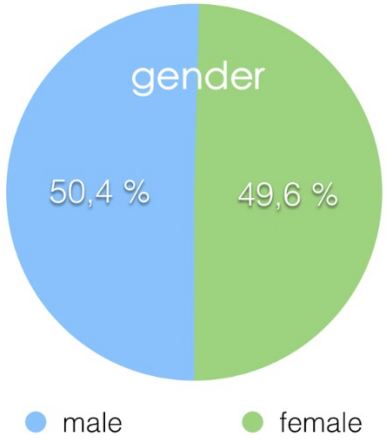
- **Examples**:
  - Fraud detection: very few fraudulent transactions compared to normal ones.
  - Medical diagnoses: rare disease occurrence.
- **Common Issue**: Standard machine learning models often perform poorly when the data is imbalanced, as they tend to favour the majority class.

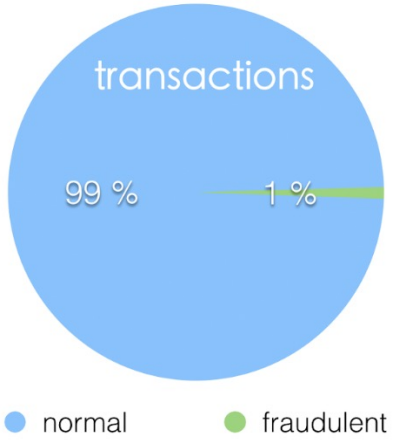


(Medium / Mario Dudjak)

IMBALANCED DATA
CHALLENGES

- **Biased Predictions**: Models predict the majority class too often.
- **Skewed Performance Metrics**: Accuracy can be misleading since it could be high even if the minority class is ignored.
- **Overfitting to Majority Class**: Models learn patterns mostly from the majority class and fail to generalize well for the minority class.
- **Poor Minority Class Detection**: Minority class is underrepresented, which may lead to under-detection in applications like fraud or medical diagnosis.





(Medium / German Lahera)

IMBALANCED DATA

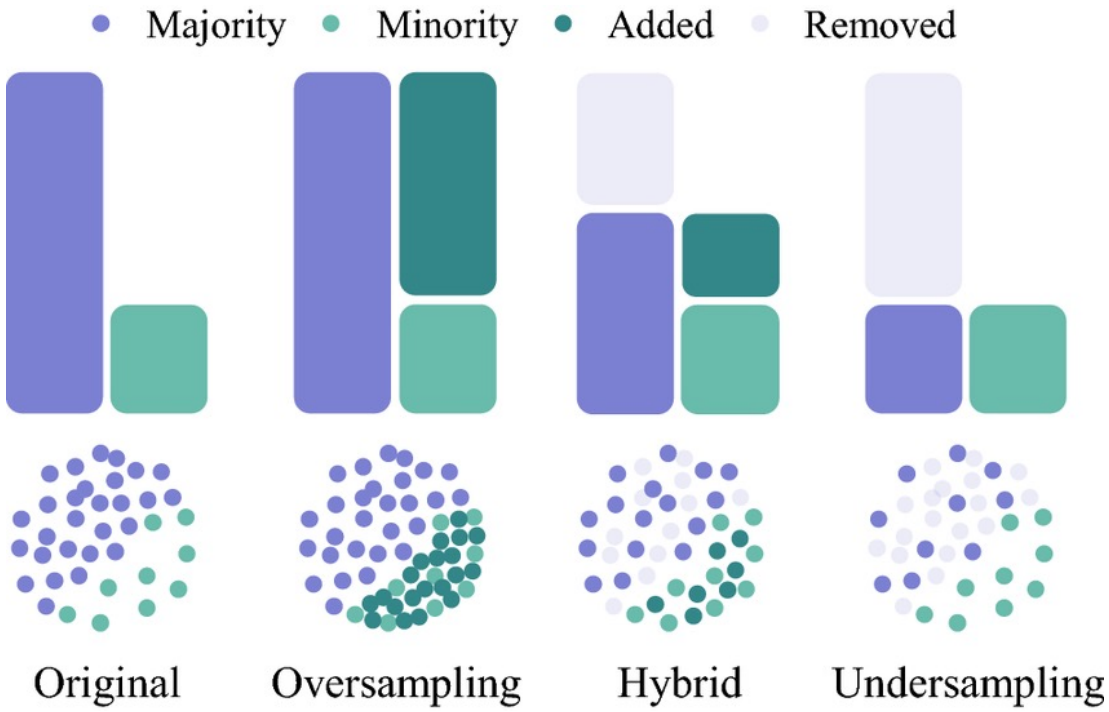
SOLUTIONS: RE-SAMPLING TECHNIQUES

- **Re-sampling** involves adjusting the training data by either oversampling the minority class or undersampling the majority class.
- Main methods:
  - **Oversampling**: Increase the number of minority class samples.
  - **Undersampling**: Reduce the number of majority class samples.

(Image not by author)

IMBALANCED DATA
SOLUTIONS: OVER-SAMPLING

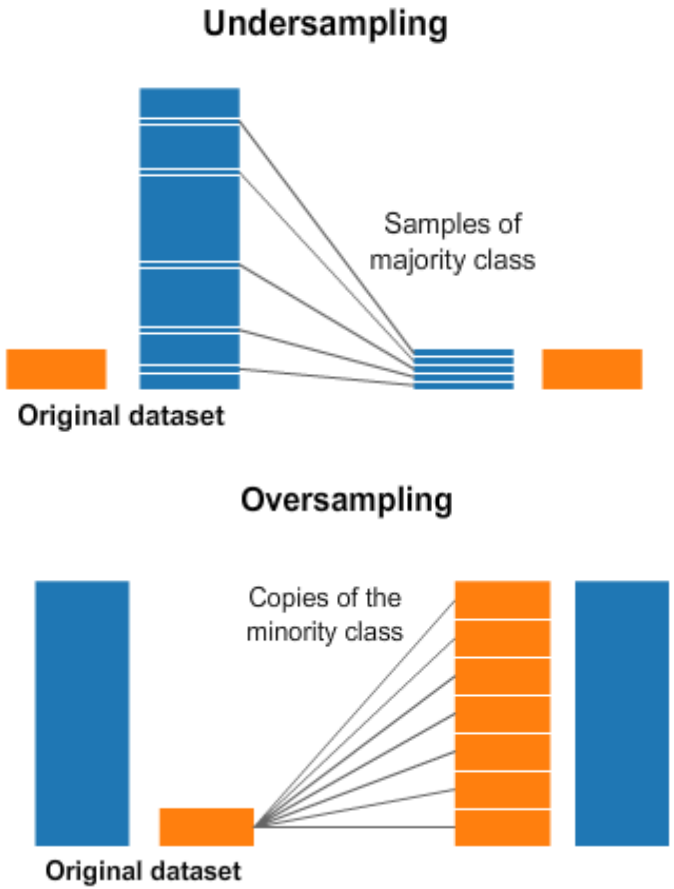
- **Oversampling** involves creating duplicates of minority class samples to balance the dataset.
- **Advantages**:
  - No information is lost from the majority class.
  - Helps models learn more from the minority class.
- **Disadvantages**:
  - Risk of overfitting since the same minority class samples are repeated multiple times.
  - Computationally expensive if the dataset is very large.



(ResearchGate / Vargas, V. et al.)

IMBALANCED DATA
SOLUTIONS: UNDER-SAMPLING

- **Under-sampling** involves reducing the number of majority class samples to match the minority class.
- **Advantages**:
  - Simplifies the dataset, making training faster and more efficient.
  - Reduces the risk of overfitting by preventing a majority bias.
- **Disadvantages**:
  - Risk of losing valuable information by discarding a large number of majority class samples.
  - Can lead to underfitting if too much data is removed.

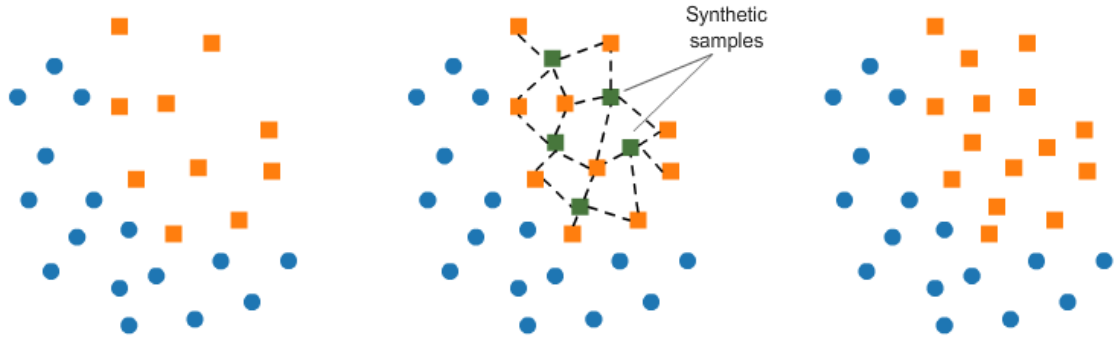


(kaggle / Rafael Alencar)

IMBALANCED DATA

OVER-SAMPLING TECHNIQUE: SMOTE

- **SMOTE**: **S**ynthetic **M**inority **O**ver-sampling **Te**chnique
- **Key Idea**: Instead of duplicating minority samples, SMOTE **generates synthetic samples** by interpolating between existing minority samples.
- **How it Works**:
  - For each minority sample, SMOTE finds its nearest neighbors.
  - New synthetic samples are created by randomly interpolating between the sample and its neighbors.
- **Advantage**: Introduces new minority examples without exact duplication, reducing the risk of overfitting.

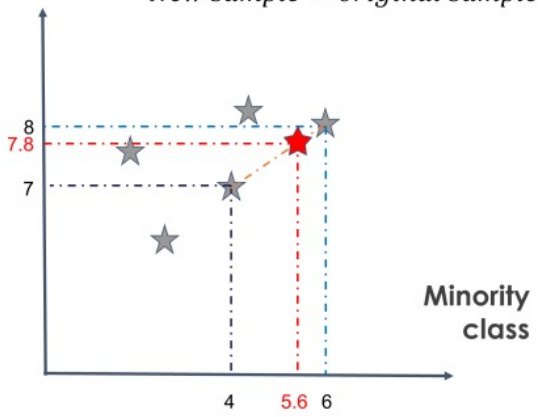


(kaggle / Rafael Alencar)

IMBALANCED DATA
SMOTE ALGORITHM

1. For each minority class instance, identify its k-nearest neighbours.
2. Randomly choose one or more of its neighbors.
3. Create a new synthetic sample by interpolating between the original instance and the chosen neighbor.
4. Repeat until the minority class is sufficiently oversampled.

$$New\ sample = original\ sample - factor * (original\ sample - neighbour)$$



$X_{ori} = (4,7)$

$X_{neig} = (6, 8)$

$$New\ sample = (4,7) - 0.8 * ((4,7) - (6,8))$$

$$New\ sample = (4,7) - 0.8 * ((-2,-1))$$

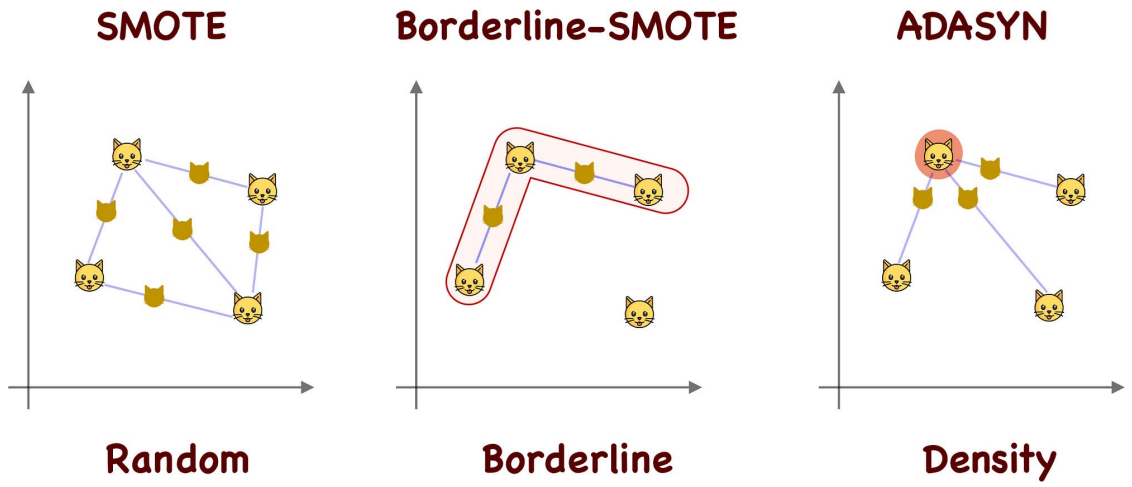
$$New\ sample = (4,7) - ((-1.6, -0.8))$$

$$New\ sample = (5.6, 7.8)$$

(Train in Data / Sole Galli)

IMBALANCED DATA
SMOTE VARIANTS

- **Borderline-SMOTE**: Focuses on creating synthetic samples near the decision boundary between classes, **improving model learning in critical regions**.
- **ADASYN (Ada**ptive **Syn**thetic Sampling**)**: Focuses on generating **more synthetic samples for harder-to-learn instances** (those with higher classification difficulty).
- **SMOTE-ENN**: Combines **SMOTE** with an **E**dited **N**earest **N**eighbor (ENN) algorithm that **removes misclassified samples** after synthetic generation, improving class separability.



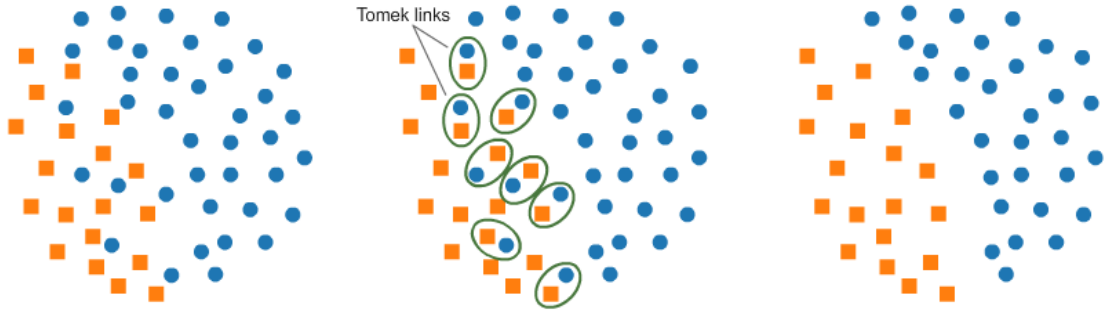
(Towards Data Science / Fernando Lopez)

IMBALANCED DATA
SMOTE-TOMEK

SMOTE-Tomek: Uses Tomek Links to remove noisy or borderline samples after applying SMOTE to improve the quality of synthetic data.

- **Dataset**: A binary classification dataset with severe class imbalance.
- **SMOTE**: Generates synthetic samples, but may introduce noise in areas with overlapping classes.
- **SMOTE-Tomek**: Applies Tomek Links to remove majority-class overlapping samples and reduce class noise.

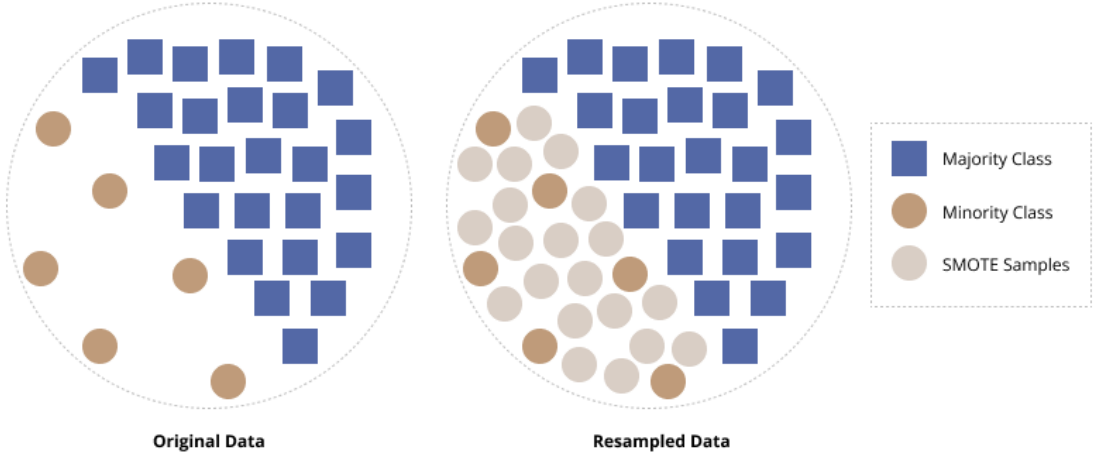
Result: **Improved decision boundary** with SMOTE-Tomek as compared to vanilla (original) SMOTE.



(kaggle / Rafael Alencar)

IMBALANCED DATA
SMOTE ADVANTAGES

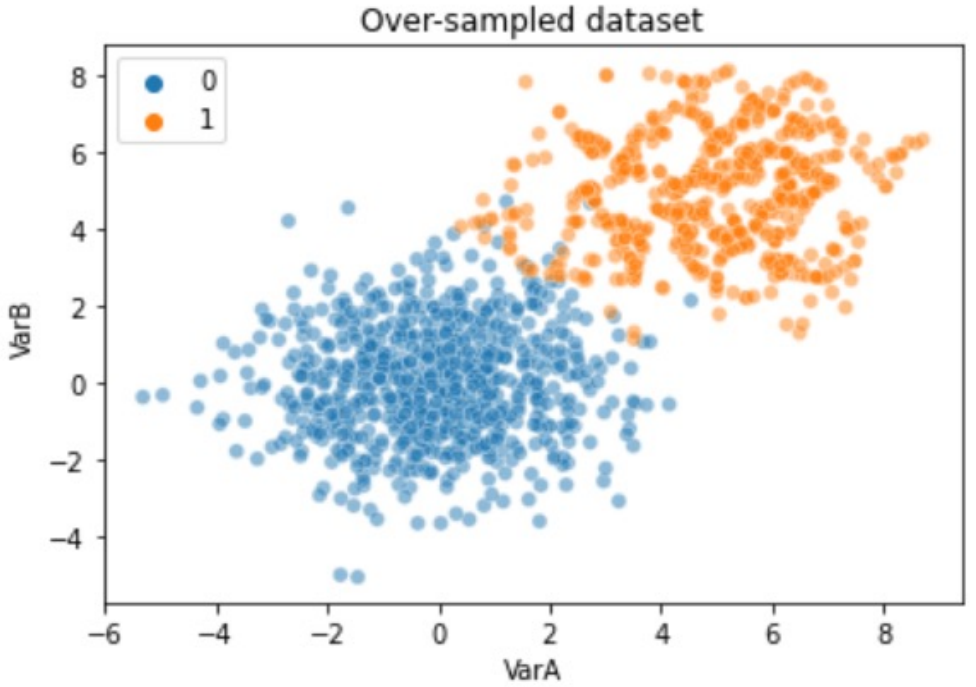
- Reduces overfitting by introducing synthetic, not duplicate, samples.
- Helps balance the dataset while keeping all majority class instances.
- Works well with classifiers sensitive to class imbalance (e.g., decision trees, SVMs).



(Data Knows All / Brian Roepke)

IMBALANCED DATA
SMOTE DISADVANTAGES

- Synthetic data may overlap with majority class, introducing noise.
- Computationally expensive, especially for large datasets.
- May lead to **class overlap**, where synthetic samples blur class boundaries.



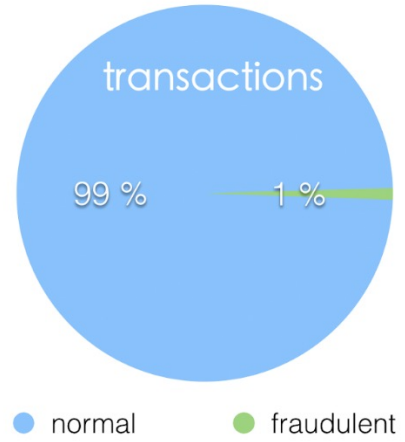
(Train in Data / Sole Galli)

IMBALANCED DATA

WHY ACCURACY IS MISLEADING

- Accuracy is often skewed towards the majority class.
- Even if the model predicts the majority class well, it may completely miss the minority class.
- Focus on metrics that provide a clearer picture of the performance for the minority class.

$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{all predictions}}$$



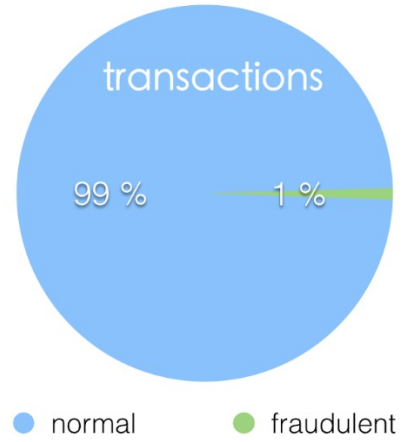
Do you see a problem?

IMBALANCED DATA

WHY ACCURACY IS MISLEADING

- Accuracy is often skewed towards the majority class.
- Even if the model predicts the majority class well, it may completely miss the minority class.
- Focus on metrics that provide a clearer picture of the performance for the minority class.

$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{all predictions}}$$



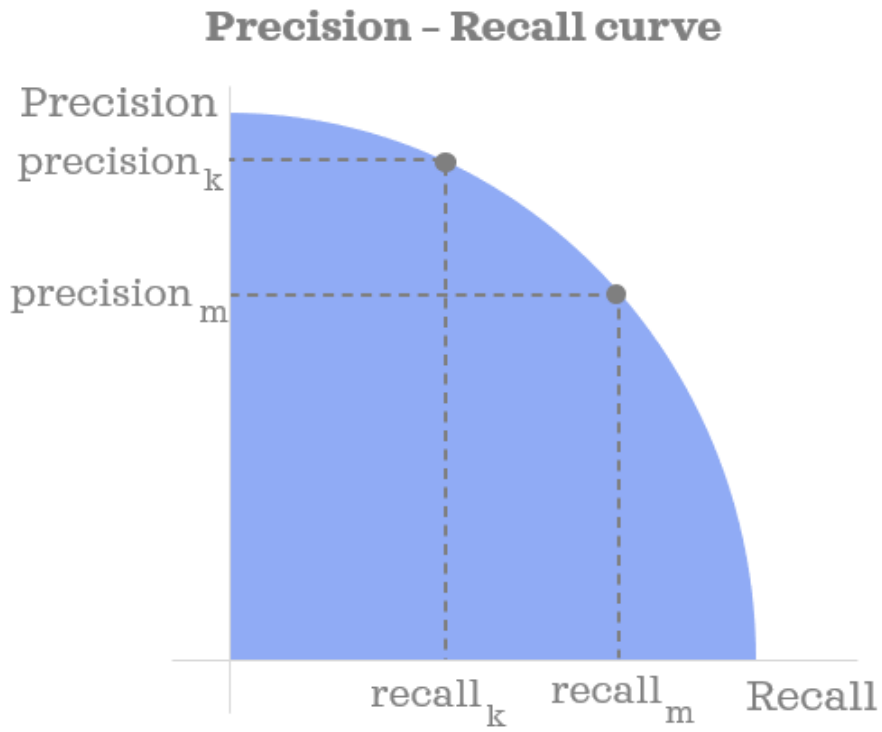
Do you see a problem?

→ 99% accuracy if it always predicts the majority class!

IMBALANCED DATA

BETTER EVALUATION METRICS

1. **Precision**: Fraction of true positives among predicted positives.
2. **Recall (Sensitivity)**: Fraction of true positives among actual positives.
3. **F1-Score**: Harmonic mean of precision and recall, balancing both metrics.
4. **ROC Curve & AUC**: Measures true positive rate vs false positive rate across different thresholds.
5. **Confusion Matrix**: Summarizes model performance by counting true positives, true negatives, false positives, and false negatives.



Precision-Recall Curve: More informative when the dataset is highly imbalanced

(Medium / Zolzaya Luvsandorj)

IMBALANCED DATA
COST-SENSITIVE LEARNING

Motivation: Instead of balancing the data, modify the learning algorithm to account for the **cost of misclassification**.

- **Approach**:
  - Assign higher penalties (costs) to misclassifying minority class instances.
  - Train classifiers to minimize the overall cost, instead of just maximizing accuracy.

- **Algorithm examples**:
  - **Cost-sensitive decision trees**: Adjusts split criteria by incorporating misclassification costs.
  - **Cost-sensitive boosting**: Adjusts boosting weight updates based on class-specific misclassification costs.

IMBALANCED DATA

ENSEMBLE METHODS FOR IMBALANCED DATA

- **Bagging**: Modifying traditional bagging approaches to give more focus to the minority class.
  - Example: **EasyEnsemble** and **BalanceCascade** — use bootstrapped samples of the minority class combined with majority class down-sampling.
- **Boosting**: Adaptively changes the weighting of instances to focus more on misclassified minority class samples.
  - Example: **AdaBoost.M1** — adjusts the weights of misclassified samples, making minority class samples more prominent in subsequent iterations.
- **RUSBoost**: Combines random under-sampling (RUS) with boosting, reducing the size of the majority class and boosting the remaining instances.
- **SMOTEBoost**: Integrates SMOTE with AdaBoost to generate synthetic minority class samples in the boosting process.

IMBALANCED DATA
BEST PRACTICES

- **Choosing the right approach**: No one-size-fits-all method. The choice of technique depends on the dataset, application, and the type of classifier.
- **Data preprocessing**: Always analyze the data distribution and try different sampling strategies (over/under-sampling, SMOTE, etc.).
- **Model tuning**: Fine-tune hyperparameters with cross-validation using metrics like F1-score or ROC-AUC to avoid overfitting to the majority class.
- **Performance trade-offs**: Understand trade-offs between precision and recall for your specific use case (e.g., detecting rare diseases where high recall is more important).
- **Computational complexity**: Some techniques like SMOTE variants or ensemble methods can increase computational complexity. Evaluate feasibility for large-scale datasets.

IMBALANCED DATA

SUMMARY OF TECHNIQUES FOR HANDLING IMBALANCED DATA

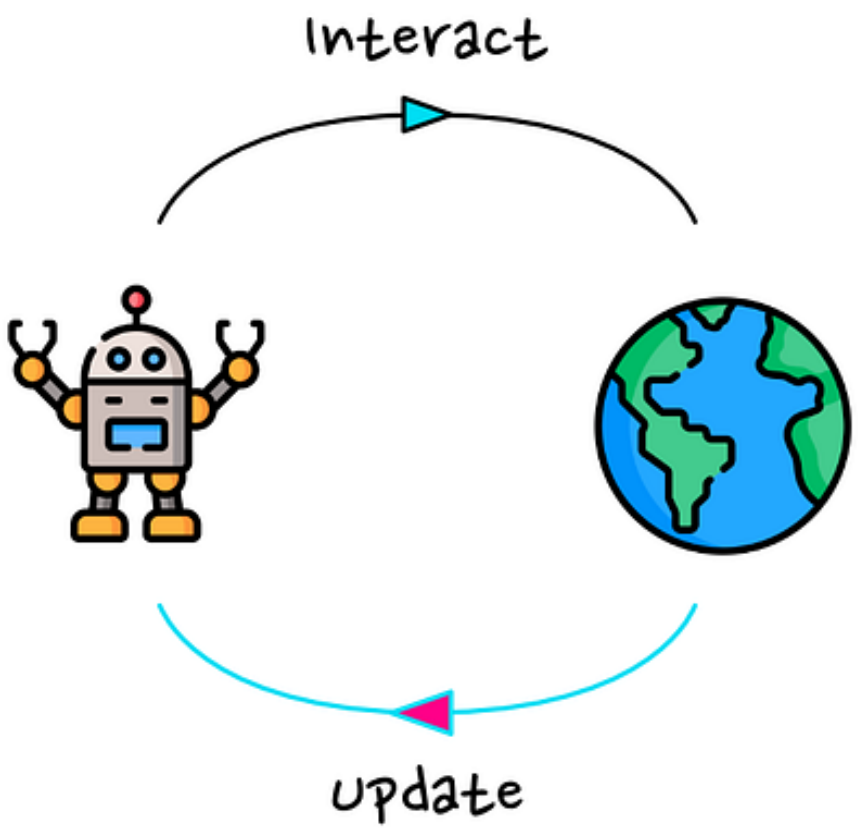
- **Data-Level Methods**: Over-sampling (SMOTE, ADASYN) and Under-sampling (Random, Tomek Links).
- **Algorithm-Level Methods**: Cost-sensitive learning, ensemble methods (EasyEnsemble, RUSBoost, SMOTEBoost).
- **Evaluation Metrics**: Precision, Recall, F1-score, ROC-AUC, Precision-Recall AUC.

Final Thought: Combining multiple methods (e.g., SMOTE with cost-sensitive learning) often yields the best results, but the best approach depends on the specific problem.

ESILV
ENGINEERING SCHOOL
DE VINCI PARIS NANTES

REINFORCEMENT LEARNING

METHODS



(Image not by Author)

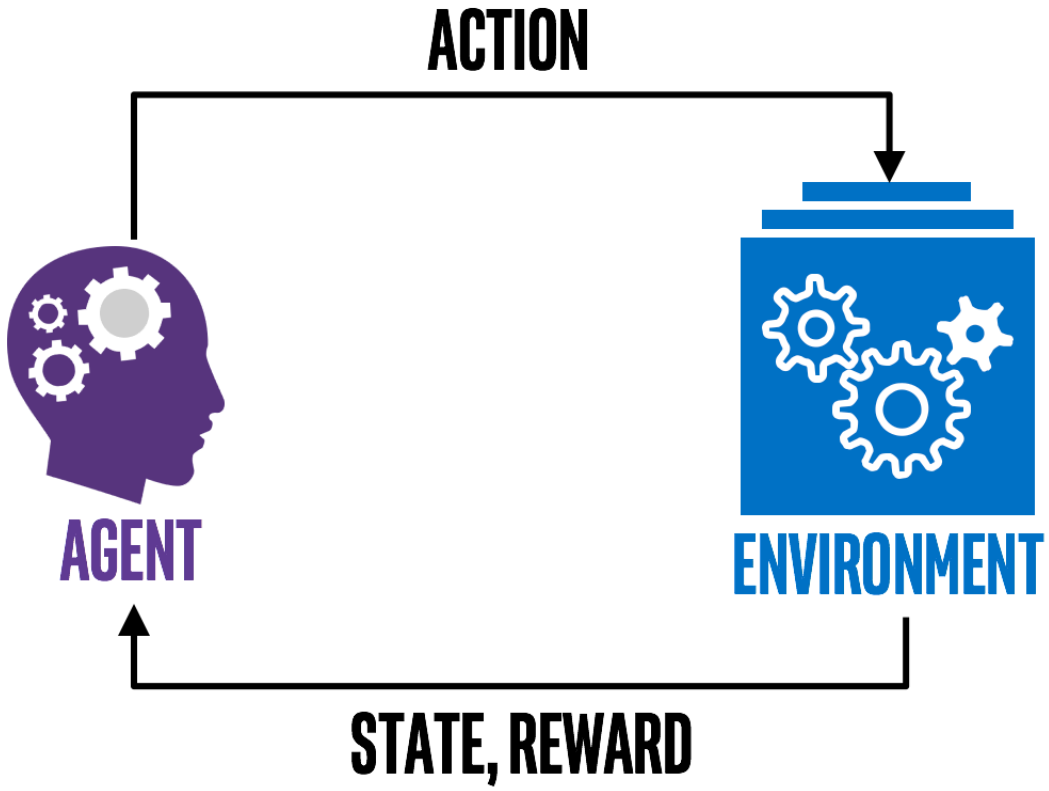
REINFORCEMENT LEARNING (RL)

Contents

REINFORCEMENT LEARNING
WHAT IS IT?

Definition: RL is an area of Machine Learning where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards.

- **Agent**: the decision-maker
- **Environment**: The world with which the agent interacts
- **Goal**: Maximize long-term rewards by learning optimal behavior through trial and error

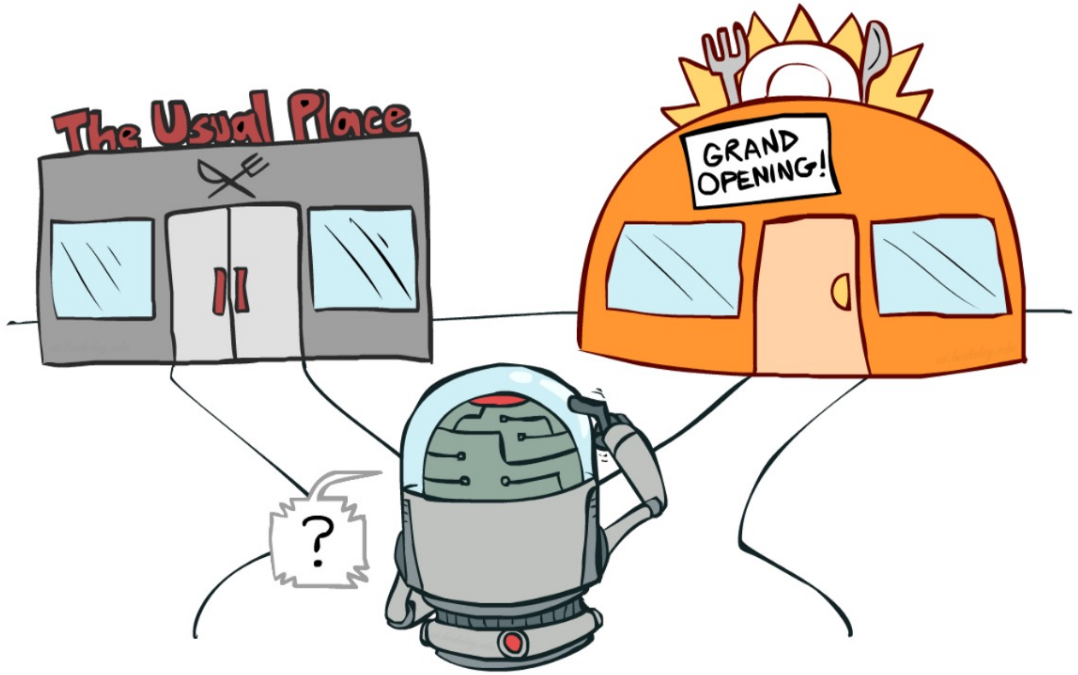


(Medium / Dan Lee)

REINFORCEMENT LEARNING
EXPLORATION VS EXPLOITATION

- **Exploration**: Trying new actions to discover their effects and learn more about the environment.
- **Exploitation**: Using current knowledge to make the best possible decision based on known rewards.

Dilemma: The agent must balance exploration (to discover potentially better rewards) and exploitation.



(Berkley AI Course)

REINFORCEMENT LEARNING
THE MARKOV DECISION PROCESS (MDP)

- **MDP** is the mathematical framework used to model decision-making in RL problems.
- **Components of MDP**:
  - **States (S)**: Set of all possible situations.
  - **Actions (A)**: Set of all possible actions.
  - **Transition Function (P)**: Probability of moving from one state to another based on an action.
  - **Rewards (R)**: Immediate reward received after a state-action transition.
  - **Discount Factor (γ)**: Determines the importance of future rewards ($0 < \gamma \leq 1$).

Goal: Find a policy that maximizes cumulative rewards over time.

$$\mathrm{M} = (S, A, P, r, \gamma)$$

MDP Tuple

REINFORCEMENT LEARNING
BELLMAN EQUATION FOR OPTIMALITY

- The **Bellman Equation** defines the value of a state as the immediate reward plus the discounted value of future states
- This recursive equation helps the agent compute the value of states to make optimal decisions.

Goal: Maximize the expected cumulative reward over time.

$$V(s) = \max_a \left(R(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s') \right)$$

Bellman Equation, with:
V, the value of the state
R, the reward
s, the state
a, the action
γ, the discount factor
P, the probability of reaching s' from s via a
s', the next state

REINFORCEMENT LEARNING

MODEL-FREE RL: Q-LEARNING

- **Q-Learning** is a popular off-policy algorithm that learns the optimal action-value function (Q-function).
- **Key Idea**: Update the Q-value for each state-action pair by adjusting it based on the immediate reward and the maximum future reward.
- **Advantages**:
  - Model-free: no model of the environment.
  - Can handle large state spaces.
- **Exploration Strategy**: Often balances exploration and exploitation (e.g., epsilon-greedy approach).

$$Q(s,a) \leftarrow Q(s,a) + \propto \left(R + \gamma \max_a Q(s',a') - Q(s,a)\right)$$

Q-Value Update Rule, with:

Q, the quality

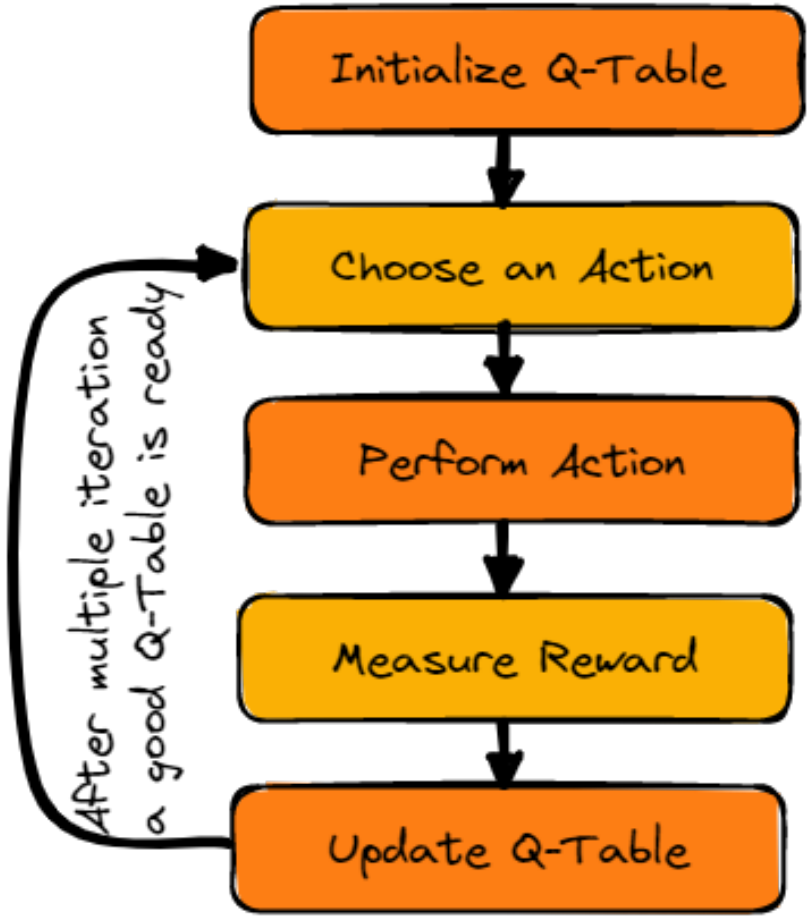
$\propto$, the learning rate

a', possible future action

REINFORCEMENT LEARNING

Q-LEARNING ALGORITHM

1. Initialize Q-table with random values.
2. Agent selects an action based on the Q-table (e.g., epsilon-greedy).
3. Execute the action and observe the reward and new state.
4. Update the Q-value for the state-action pair.
5. Repeat the process until the Q-values converge to the optimal policy.

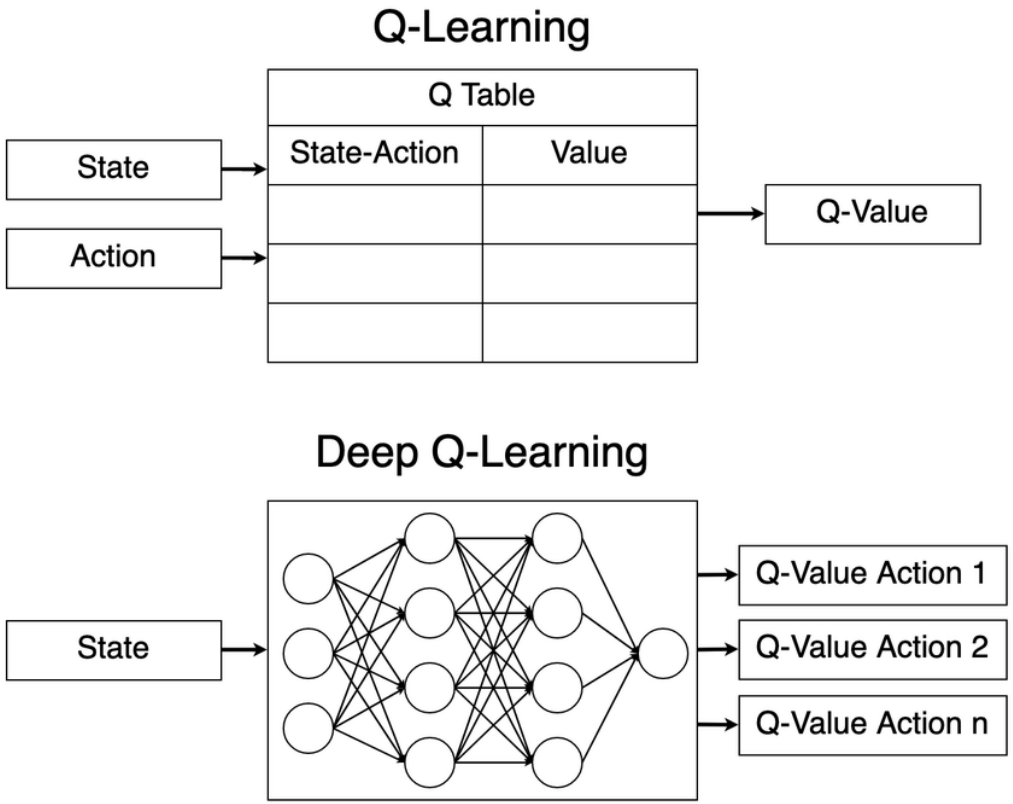


(datacamp / Abid Ali Awam)

REINFORCEMENT LEARNING
DEEP-Q NETWORKS (DQN)

- **Deep Q-Networks (DQN)** are an extension of Q-learning that use a neural network to approximate the Q-function when the state-action space is too large to represent with a table.
- **Key Components**:
  - **Neural Network**: Approximates the Q-function.
  - **Experience Replay**: Stores past experiences (state, action, reward, next state) and samples them to train the network, reducing correlation between samples.
  - **Target Network**: A copy of the Q-network used to calculate target values, updated periodically to stabilize learning.

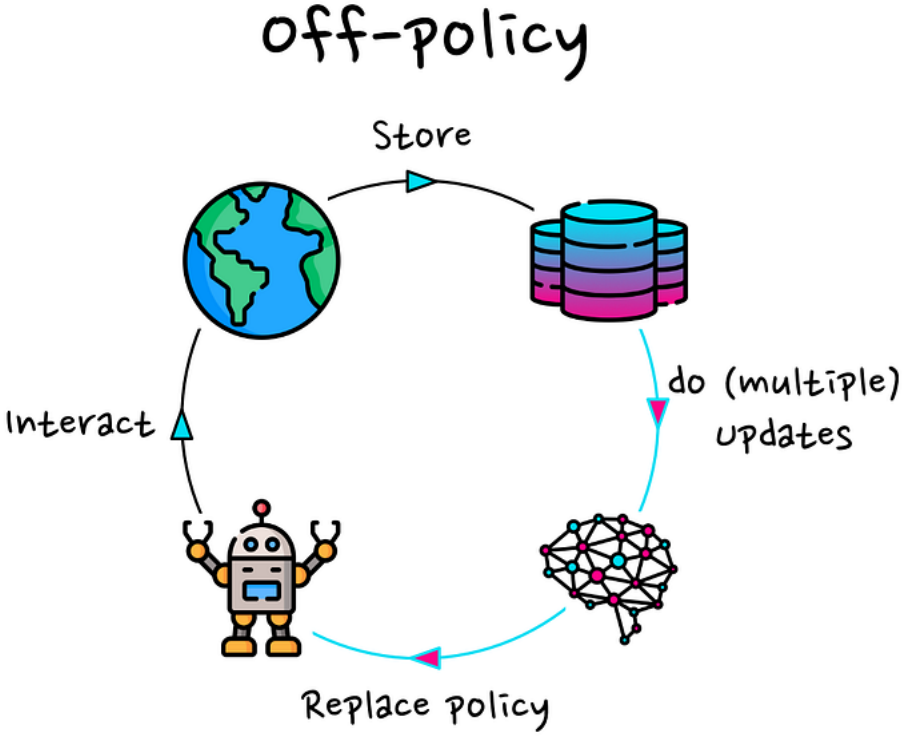
Advantages: Can handle high-dimensional state spaces like images (e.g., Atari games).



(Sebastianelli, A., et al., 2021)

REINFORCEMENT LEARNING
DQN ALGORITHM

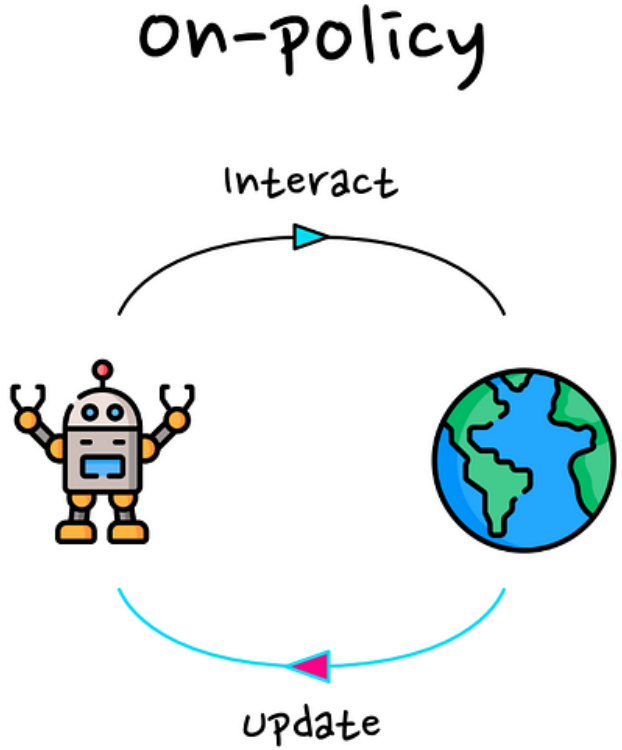
1. Initialize Q-network with random weights.
2. For each step:
   - Select action using epsilon-greedy strategy.
   - Execute action and observe the reward and next state.
   - Store experience (state, action, reward, next state) in a replay buffer.
   - Sample a mini-batch from the replay buffer.
   - Update the Q-network by minimizing the loss between predicted Q-values and target Q-values.
   - Periodically update the target network.
3. Repeat until convergence.



(Medium / Cédric Vandelaer)

REINFORCEMENT LEARNING
POLICY-BASED METHODS

- **Policy-based methods** learn the policy directly instead of learning the value function.
- **Policy Gradient**: A method to optimize the policy by maximizing the expected cumulative reward.
- **Advantages**:
  - Can handle continuous action spaces.
  - Learns stochastic policies (e.g., softmax).
- **Disadvantages**:
  - High variance in policy gradient estimates.
  - Can be unstable without proper tuning.



(Medium / Cédric Vandelaer)

REINFORCEMENT LEARNING

« REINFORCE »: A BASIC POLICY GRADIENT ALGORITHM

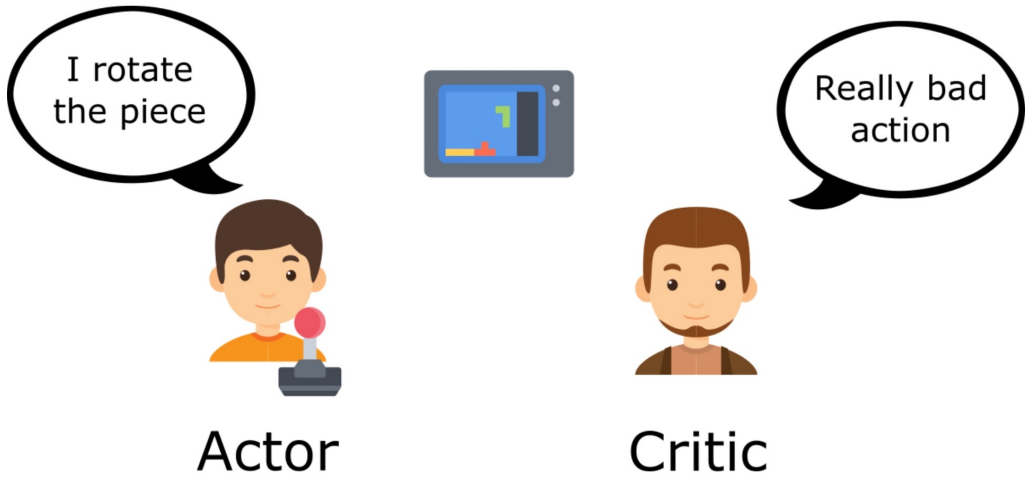
- **REINFORCE** is one of the simplest policy gradient algorithms.
- **Key Idea**: Update the policy parameters to increase the probability of actions that result in higher rewards.
- **Advantages**: Directly optimizes the policy, easy to implement.
- **Disadvantages**:
  - High variance in policy gradient estimates.
  - Slow convergence and instability without proper techniques like baseline subtraction.

$$\theta \leftarrow \theta + \propto \nabla_\theta \log \pi_\theta(a|s) G_t$$

Policy Update Rule
where θ in a policy parameter,
$\pi_\theta(a|s)$ is the policy,
and G_t is the cumulative reward

REINFORCEMENT LEARNING
ACTOR-CRITIC METHODS

- **Actor-Critic** combines both value-based and policy-based methods.
- **Key Components**:
  - **Actor**: Learns the policy (i.e., which action to take).
  - **Critic**: Learns the value function (i.e., how good the action taken was).
- The **Critic** evaluates the actions chosen by the **Actor**, providing feedback to help adjust the policy.
- **Advantage**: Reduces variance in policy updates by incorporating value estimates from the Critic.
- **Disadvantage**: Can be more complex to train and tune.



(Hugging Face / Advantage Actor Critic A2C)

REINFORCEMENT LEARNING
VALUE-BASED VS POLICY-BASED ALGORITHMS

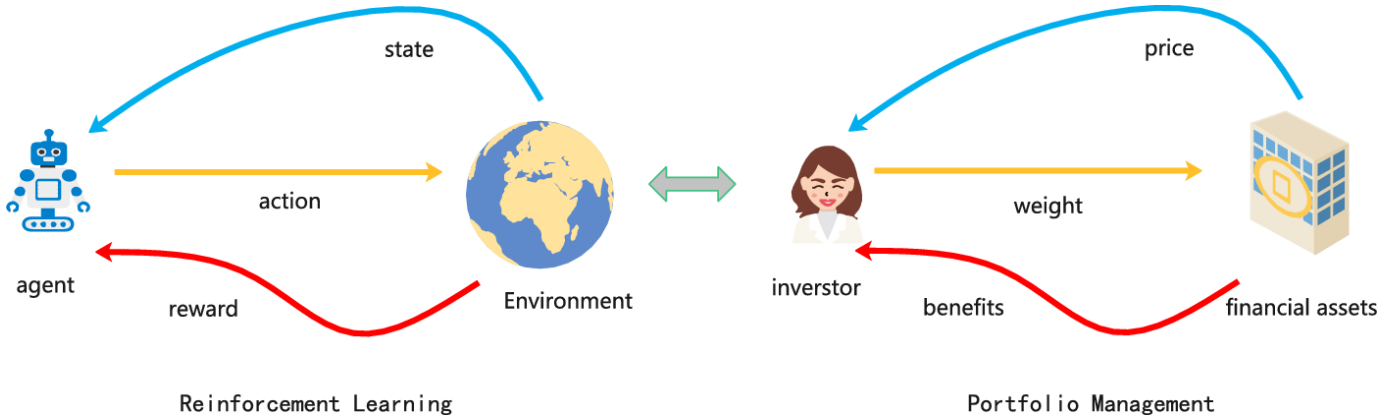
- **Value-Based (e.g., Q-Learning, DQN)**:
  - Focuses on learning the value of state-action pairs.
  - Works well in discrete action spaces.
  - Easier to understand and implement.

- **Policy-Based (e.g., REINFORCE, Actor-Critic)**:
  - Learns the policy directly.
  - Can handle continuous action spaces.
  - Generally more flexible but more difficult to train.

- **Hybrid Methods**:
  - Actor-Critic combines advantages of both.
  - Typically more stable and effective in large or complex environments.

REINFORCEMENT LEARNING
REAL-WORLD APPLICATIONS

- **Game Playing**: complex games like Go and Chess; Atari games.
- **Robotics**: RL used to train robots to perform tasks like walking, grasping, and manipulating objects.
- **Autonomous Vehicles**: for path planning, obstacle avoidance, and self-driving decisions.
- **Healthcare**: Optimizing treatment plans for patients based on evolving conditions.
- **Finance**: Algorithmic trading and portfolio optimization using RL to maximize long-term returns.



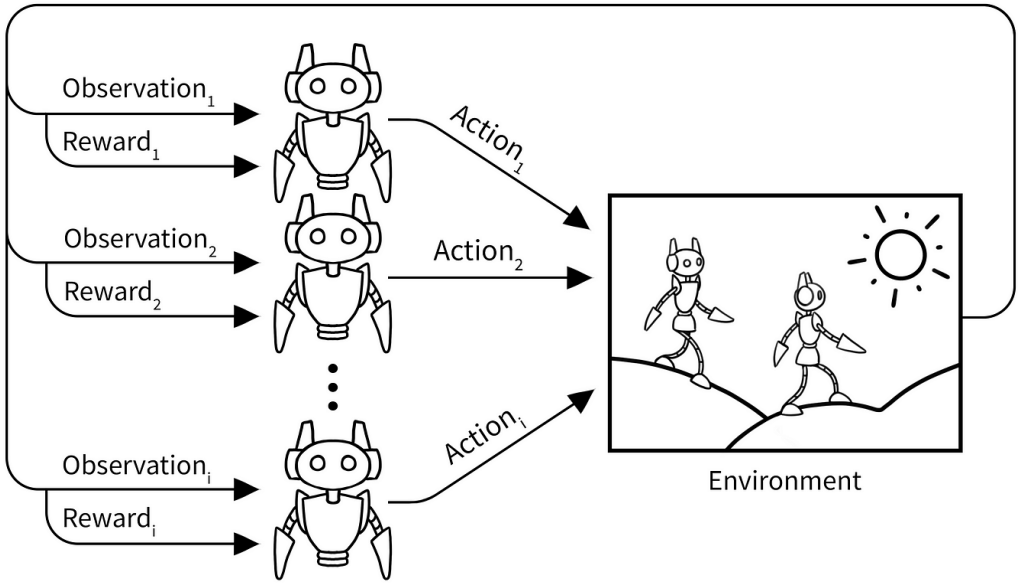
(Haifeng Li & Mo Hai, 2024)

REINFORCEMENT LEARNING
CHALLENGES

- **Sample Efficiency**: RL algorithms often require many interactions with the environment, which can be costly or time-consuming.
- **Exploration vs. Exploitation**: Balancing the need for discovering new strategies (exploration) versus using known strategies (exploitation) is difficult.
- **Reward Sparsity**: In many tasks, rewards are rare or delayed, making it hard for the agent to learn.
- **High Variance in Policy Gradients**: Policy gradient methods can suffer from high variance, leading to unstable learning.
- **Scalability**: RL algorithms can struggle with environments that have high-dimensional state or action spaces.

REINFORCEMENT LEARNING
RECENT ADVANCEMENTS

- **Deep Reinforcement Learning**: Combining deep learning with RL for handling complex, high-dimensional environments.
- **Proximal Policy Optimization (PPO)**: An efficient policy gradient algorithm that strikes a balance between exploration and exploitation, used by OpenAI.
- **AlphaZero**: Advances in self-play and learning from scratch, surpassing human-level performance in games like Go, Chess, and Shogi.
- **Model-Based RL**: New methods focusing on using models of the environment to improve sample efficiency and learning speed.
- **Multi-Agent RL**: Cooperative and competitive settings where multiple agents learn and interact in shared environments.



(Medium / Justin Terry)

REINFORCEMENT LEARNING
ETHICAL CONSIDERATIONS

- **Reward Hacking**: Agents may find unintended ways to maximize rewards, leading to unethical or dangerous behavior.
- **AI Safety**: Ensuring RL agents act safely in real-world environments, especially in critical applications like autonomous vehicles or healthcare.
- **Bias in Rewards**: Reward functions designed by humans might embed biases, leading to unfair or unintended outcomes.
- **Transparency and Interpretability**: RL models can be difficult to interpret, making it hard to understand why certain decisions are made.

(Medium / Utku Sen)

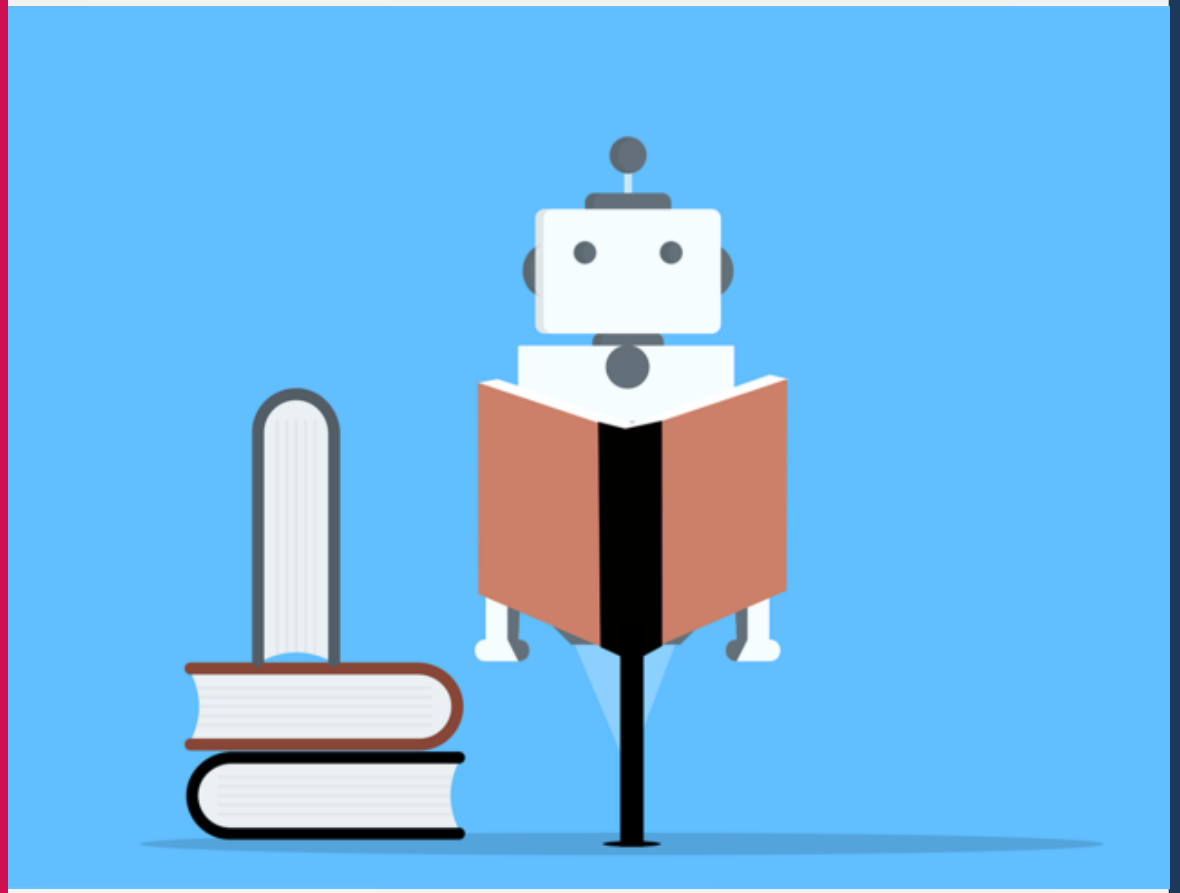
REINFORCEMENT LEARNING
SUMMARY

- **Key Concepts**: Reinforcement Learning enables agents to learn from interaction with an environment by balancing exploration and exploitation.
- **Algorithms**: Q-learning, DQN, Policy Gradient, and Actor-Critic are foundational approaches in RL.
- **Applications**: RL is transforming fields like gaming, robotics, autonomous vehicles, and finance.
- **Challenges**: RL faces issues with sample efficiency, reward design, and scalability, but advancements like deep RL and PPO are helping address these challenges.
- **Future**: RL continues to grow in importance, especially in real-world applications with complex decision-making tasks.

ESILV
ENGINEERING SCHOOL
DE VINCI PARIS NANTES

AUTO ML

SNEAK-PEAK

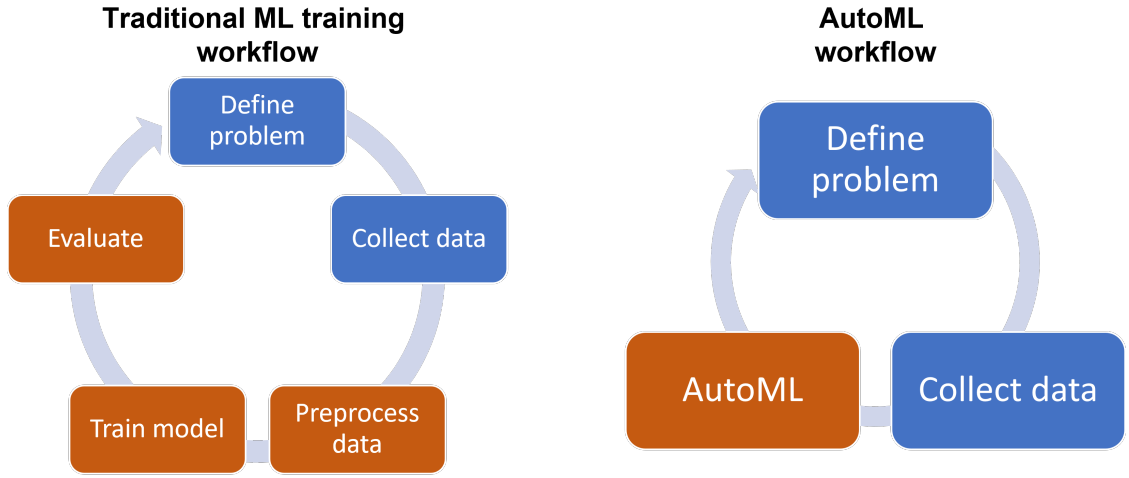


(Image not by Author)

AUTOML
WHAT IS IT?

- **Automated Machine Learning** is a process that automates the steps of applying machine learning to real-world problems.

Goal: Make machine learning accessible to non-experts and increase the efficiency of experts.



(Microsoft Learn / luisquintanilla & gewarren)

AUTOML
WHY USE IT?

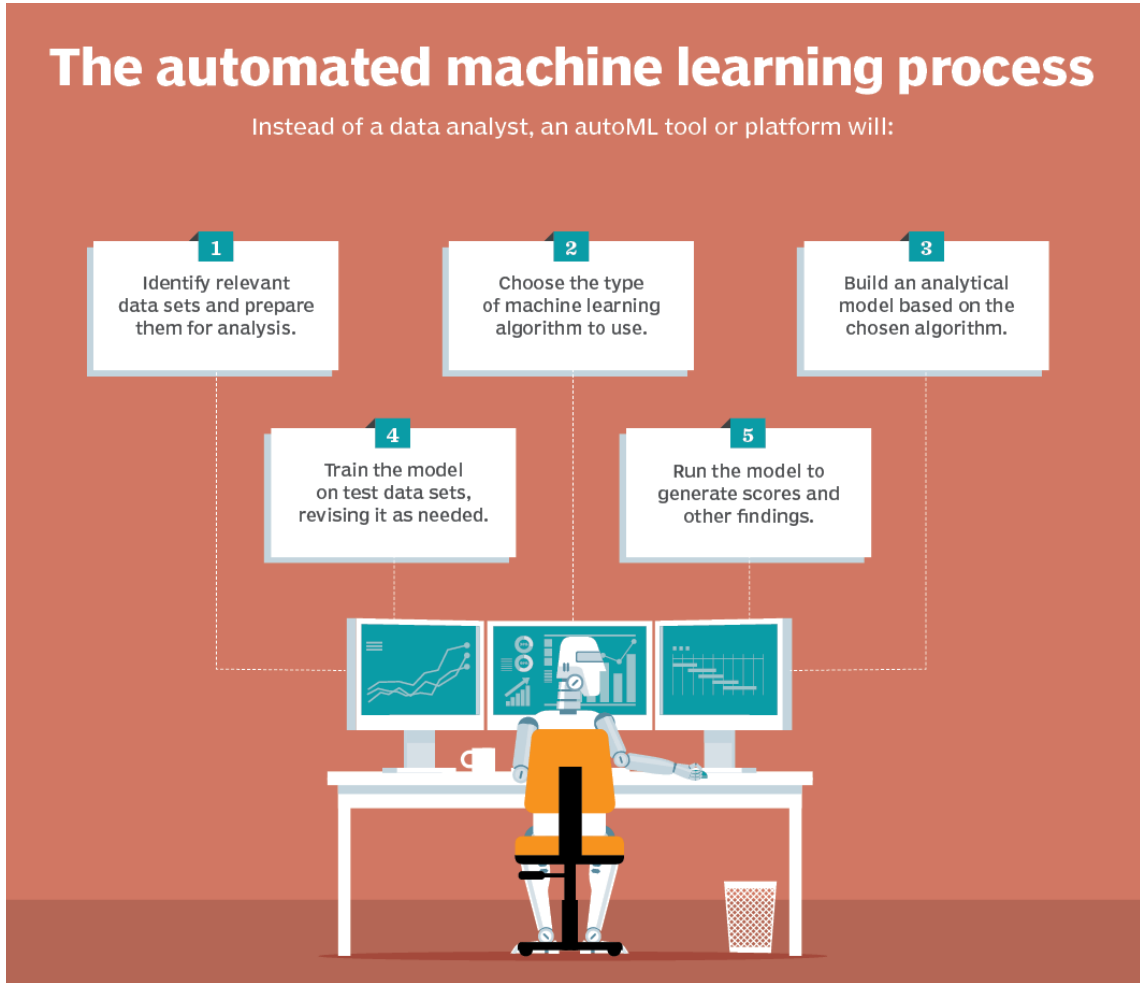
- **Challenges in ML Development**:
  - Manual data preparation and model tuning are time-consuming.
  - Expertise required in model selection, hyperparameter tuning, and evaluation.

- **AutoML solves these challenges**:
  - **Efficiency**: Reduces the need for manual intervention.
  - **Performance**: Finds high-performing models without extensive manual tuning.
  - **Accessibility**: Lowers the barrier to entry for non-experts.

- **Applications**: Data science competitions, business applications with rapid deployment, and organizations without dedicated ML teams.

AUTOML
KEY COMPONENTS

1. **Data Preprocessing**:
   - Handling missing values, normalization, encoding categorical variables.
   - Automatic feature generation and transformation.
2. **Model Selection**:
   - Evaluating different algorithms (e.g., tree-based models, deep learning, etc.).
   - Cross-validation to ensure generalizability.
3. **Hyperparameter Optimization**:
   - Techniques such as grid search, random search, or advanced methods like Bayesian optimization.
4. **Ensemble Methods**:
   - Combining different models to improve performance (stacking, blending).
5. **Model Evaluation**:
   - Automatically choosing appropriate evaluation metrics (accuracy, F1-score, AUC).



(TechTarget / Ben Luktevich)

AUTOML
TECHNIQUES

1. **Random Search**:
   - Explores random hyperparameter configurations and selects the best-performing model.

2. **Grid Search**:
   - Systematically tries combinations of hyperparameters but can be inefficient for large parameter spaces.

3. **Bayesian Optimization**:
   - Builds a probabilistic model to predict which hyperparameters will work best and optimize accordingly.

4. **Neural Architecture Search (NAS)**:
   - Automates the design of neural network architectures.

5. **Meta-Learning**:
   - Uses information from previous learning tasks to make model selection and optimization faster for new tasks.

AUTOML

POPULAR FRAMEWORKS

- **Auto-Sklearn**: Built on top of Scikit-Learn, uses meta-learning and Bayesian optimization.
- **TPOT**: Uses genetic programming to optimize pipelines.
- **H2O.ai**: Provides automatic machine learning for various tasks (classification, regression, etc.).
- **Google Cloud AutoML**: Automates model building on the cloud with minimal user input.
- **Microsoft Azure AutoML**: Automated ML for building and deploying models on the Azure platform.

AUTOML
WORKFLOW EXAMPLE

- **Example** of using Auto-Sklearn to solve a classification task:

1. **Data Loading**: Import dataset (e.g., classification task).
2. **AutoML Initialization**: Define the AutoML system (set time limits, evaluation metrics).
3. **Model Search**: Automatically searches for the best model pipeline.
4. **Evaluation**: Returns the model with the highest validation performance.

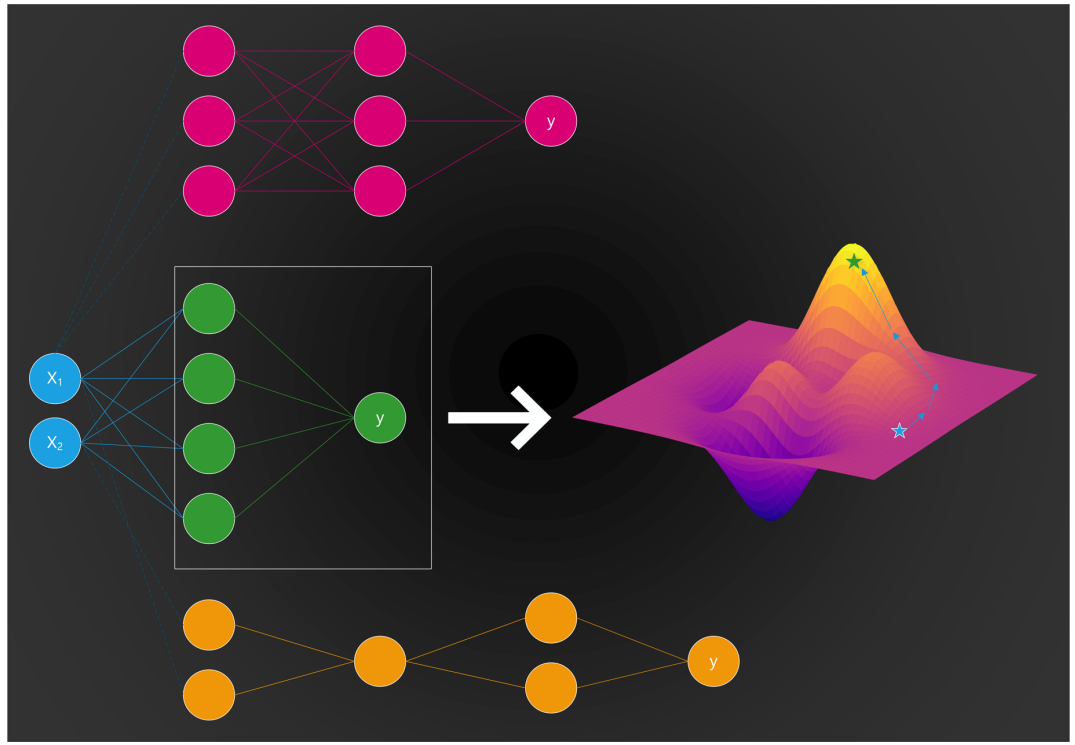
Python Code Example:

```
from autosklearn.classification
import AutoSklearnClassifier
automl =
AutoSklearnClassifier(time_left_fo
r_this_task=3600)
automl.fit(X_train, y_train)
print(automl.show_models())
```

AUTOML

DEEP LEARNING: NEURAL ARCHITECTURE SEARCH

- **Neural Architecture Search (NAS)**:
  - Automates the design of deep learning architectures.
  - Instead of manually defining neural network layers, NAS finds optimal architectures.
- **Techniques**:
  - **Reinforcement Learning**: A controller network learns to generate architectures.
  - **Evolutionary Algorithms**: Uses genetic algorithms to evolve architectures.
- **Example**: Google's AutoML used NAS to design a state-of-the-art image classification model.



(Towards Data Science / Alex Adam)

AUTOML
CHALLENGES

1. **Computationally Expensive**:
   - Training and evaluating multiple models requires significant computational resources.

2. **Model Interpretability**:
   - AutoML often creates complex models, which may be harder to interpret.

3. **Handling Small Datasets**:
   - AutoML works better on large datasets. For small datasets, traditional manual tuning might still perform better.

4. **Bias and Fairness**:
   - AutoML systems need to be monitored to ensure they do not introduce bias in the models.

5. **Customization Limits**:
   - AutoML abstracts many decisions, but experts may want more control in specific cases.

AUTOML
FUTURE

- **Democratization of AI**: AutoML will make machine learning more accessible to non-experts.
- **Increased Use of NAS**: Advances in Neural Architecture Search will lead to better and faster deep learning models.
- **Integration with Cloud Platforms**: AutoML integrated into cloud systems (AWS, Azure, Google Cloud) will make deployment seamless.
- **Specialized AutoML**: AutoML systems will evolve to handle specialized tasks (e.g., time-series forecasting, NLP).

PÔLE LÉONARD

DE VINCI

ESILV
ENGINEERING SCHOOL
DE VINCI PARIS NANTES