We are starting at 13:00!

Grab a seat and get ready

Saturdays.AI
Kigali

Saturdays.AI
Kigali

#3 The ML Landscape

AI Saturdays Kigali

The Machine Learning Landscape

Saturdays.AI
Kigali

What is Machine Learning?

Machine Learning is the science (and art) of programming computers so they can learn from data.

Here is a slightly more general definition:

> [Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.
>
> —**Arthur Samuel, 1959**

Example: Spam Filter as a ML Program

- **Goal**: Automatically detect spam emails
- **Input**:
  - Labeled spam emails (e.g., flagged by users)
  - Labeled non-spam (ham) emails
- **Output**:
  - A model that learns to flag future spam emails

Why Use Machine Learning?

Machine Learning is great for:

- **Rule-heavy problems**
  When traditional solutions need constant fine-tuning or many rules, ML can simplify and outperform them.

```
if flat.price is <= 10.0:
      then send offer;

if flat is big...
if flat is new
...
```

```
try to find good flats:
adapt model to learn from err
repeat until threshold is ☑️;
```

- **Complex or unsolved problems**
  Where traditional methods fail, ML can uncover patterns and find workable solutions.

- **Changing environments**
  ML models adapt as new data arrives — ideal for dynamic or evolving systems.

Saturdays.AI
Kigali

Examples of Applications

- Analyzing images of products on a production line to automatically classify them
- Detecting tumors in brain scans
- Automatically classifying news articles
- Automatically flagging offensive comments on discussion forums
- Summarizing long documents automatically
- Creating a chatbot or a personal assistant
- Forecasting your company's revenue next year, based on many performance metrics
- Making your app react to voice commands
- Detecting credit card fraud
- Segmenting clients based on their purchases so that you can design a different marketing
- strategy for each segment
- Representing a complex, high-dimensional dataset in a clear and insightful diagram
- Recommending a product that a client may be interested in, based on past purchases
- Building an intelligent bot for a game

Saturdays.AI
Kigali

Types of Machine Learning Systems

Saturdays.AI
Kigali

Three types of machine learning

Supervised learning	Labeled data Direct feedback Predict outcome/future
Unsupervised learning	No labels/targets No feedback Find hidden structure in data
Reinforcement learning	Decision process Reward system Learn series of actions

Figure 1.1: The three different types of machine learning

Supervised Learning

- **Goal**: Learn a model from **labeled training data** to make predictions on **unseen/future data**
- **Why "Supervised"?**
  - Each training example comes with a **known label** (the "correct answer")
- **What it does**:
  - Models the relationship between **inputs** (features) and **outputs** (labels)
- **Another way to think about it**:
  - Supervised learning = "**learning from labels**"

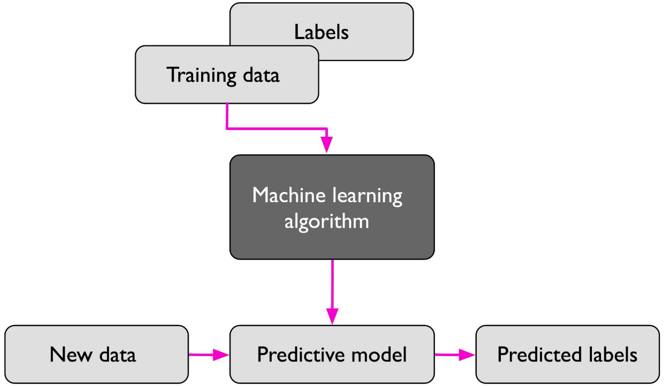


Figure 1.2: Supervised learning process

Saturdays.AI Kigali

Supervised Learning: Classification

- **Goal**: Predict categorical class labels for new data points
- **Labels** are **discrete**, **unordered values** representing group membership

Example: Binary Classification

- Task: **Spam detection**
- Two possible classes: **Spam** vs. **Not Spam**
- Model learns rules from labeled examples

Example: Multiclass Classification

- Task: **Handwritten character recognition**
- Classes: Letters **A–Z**
- Model predicts the correct letter from input
- Limitation: Can't recognize digits (0–9) unless seen during training

Saturdays.AI
Kigali

Supervised Learning: Classification

- **Scenario**: 30 training examples
  - 15 labeled **Class A,** 15 labeled **Class B**
- **Goal**:
  - Use a supervised ML algorithm to learn a **decision boundary**
  - This boundary (dashed line) separates Class A from Class B

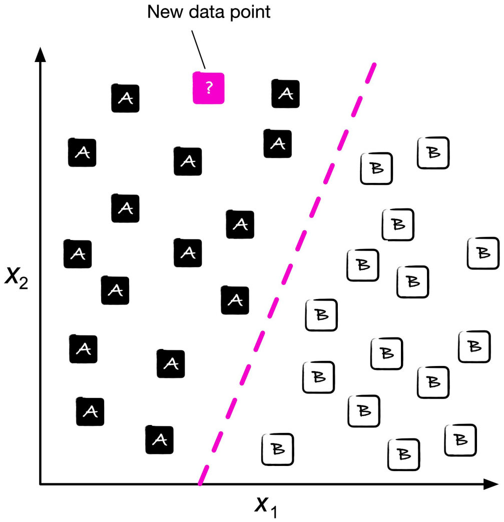


Figure 1.3: Classifying a new data point

Saturdays.AI
Kigali

Supervised Learning: Regression

- **Unlike classification**, regression predicts **continuous values**, not categories
- Also known as **regression analysis**

Example: Predicting a student's SAT Math Scores

- **Feature**: Time spent studying
- **Target**: Final score

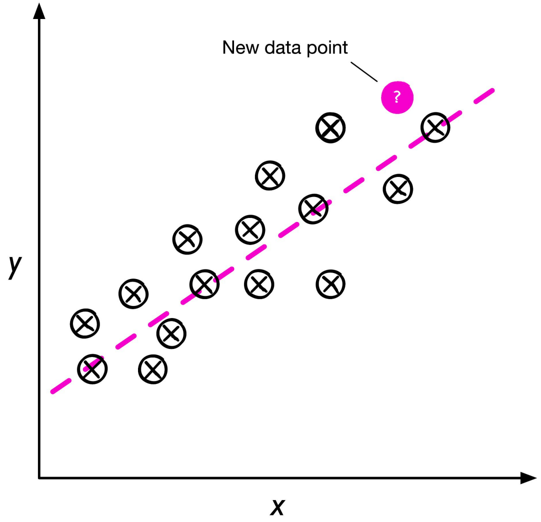


Figure 1.4: A linear regression example

Saturdays.AI Kigali

Unsupervised Learning

- Works with **unlabeled data**
- No predefined **outcomes** or **target variables**
- Goal: **Discover hidden patterns** or **underlying structure** in the data
- No external supervision or feedback (like rewards or labels)

Examples of Unsupervised Learning Tasks:

- **Clustering**: Grouping similar data points
- **Dimensionality reduction**: Simplifying data while preserving structure

Saturdays.AI
Kigali

Unsupervised Learning: Clustering

- Clustering is an exploratory technique used to **organize data into meaningful groups** (clusters)
- No prior knowledge of group memberships required
- Groups are formed based on **similarity within clusters** and **dissimilarity between clusters**

Why It's Useful:

- Helps **reveal hidden patterns** or relationships
- Also known as **unsupervised classification**

Example:

- **Marketing**: Identify distinct customer segments based on interests to tailor marketing strategies

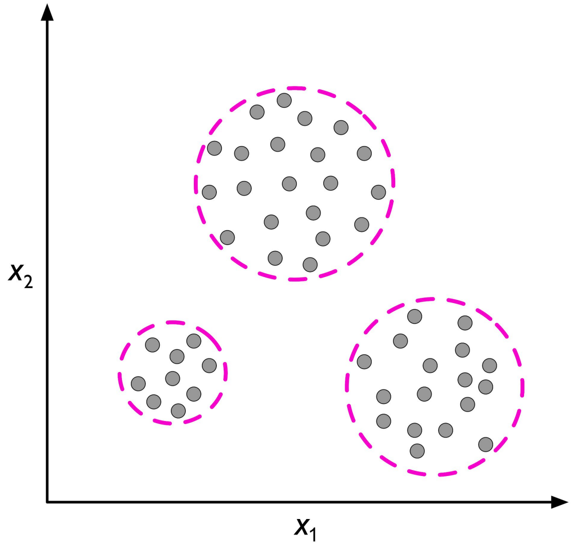


Figure 1.5: How clustering works

Saturdays.AI Kigali

Reinforcement Learning

- An agent learns by interacting with an **environment**
- Learns to take actions that **maximize cumulative reward** over time
- Feedback comes as a **reward signal**, not a correct label

How it Works

- Agent observes the current **state**
- Takes an **action**
- Receives a **reward** based on that action
- Learns through **trial-and-error** or **planning**

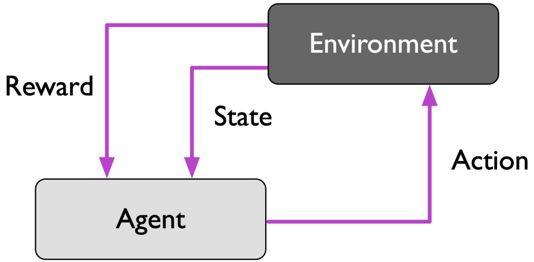


Figure 1.6: Reinforcement learning process

Saturdays.AI
Kigali

Reinforcement Learning

Key Differences from Supervised Learning

- No ground truth label for each input
- Feedback is **delayed and indirect** (via reward function)
- Goal: Learn a **policy** that maps states to optimal actions

Example: Chess AI

- **State**: Current board position
- **Action**: Chosen move
- **Reward**: +1 for win, 0 for draw, -1 for loss
- Learns strategies by playing many games

Saturdays.AI
Kigali

Machine Learning Terminology

Training Example

- A **row** in a dataset (a single data point)
- Also known as: **observation, record, instance**, or **sample**

Training

- The process of **model fitting**
- In parametric models: similar to **parameter estimation**

Feature (x)

- A **column** in the dataset (an input variable)
- Also known as: **predictor, input, variable, attribute**, or **covariate**

Target (y)

- The value to be predicted (the output variable)
- Also known as: **outcome, label, response, dependent variable, ground truth**

Loss Function / Cost Function

- Measures the **error** between predicted and actual values
- Synonyms: **error function, objective function**

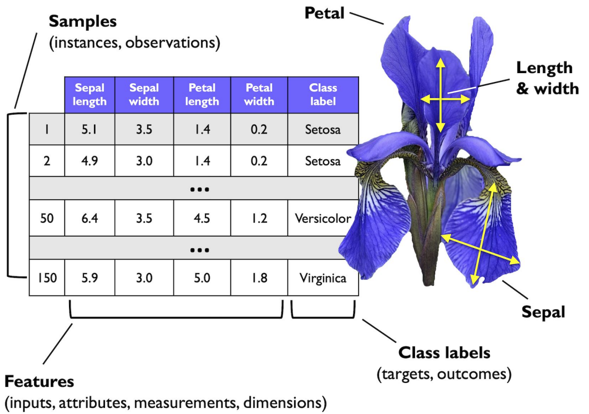


Figure 1.7: The Iris dataset

Saturdays.AI
Kigali

Main Challenges of Machine Learning

Saturdays.AI
Kigali

Machine Learning Challenges

Since your task is to select a learning algorithm and train it on some data, **two key things can go wrong**:

Bad Data

- Insufficient data quantity
- Data that does not represent the real-world problem
- Poor-Quality Data
- Irrelevant or redundant features

Bad Algorithm

- Wrong choice of model for the task
- Overfitting or underfitting
- Poor hyperparameter tuning

Even the best algorithm can fail on poor data — and great data can be wasted on the wrong model.

Bad Data

Insufficient Quantity of Training Data

Human Learning vs. Machine Learning

For a toddler to learn what an apple is, all it takes is for you to point to an apple and say "apple" (possibly repeating this procedure a few times). Now the child is able to recognize apples in all sorts of colors and shapes. Genius. Machine Learning is not quite there yet;

ML Needs Lots of Data

- **Thousands** of examples for simple tasks
- **Millions** for complex tasks (e.g., image or speech recognition - unless you can reuse parts of an existing model known as ***transfer learning***)
- More data = better generalization, less overfitting

Nonrepresentative Training Data

In order to generalize well, it is crucial that your training data be representative of the new cases you want to generalize to.

Common Pitfalls

- **Sampling bias**: Data collected from one group or condition but applied to another
- **Temporal drift**: Training data is outdated and no longer reflects the current environment
- **Coverage gaps**: Important cases or variations are missing from training examples
- **Sampling noise:** Random variations in the sample that cause it to misrepresent the true distribution of the population (e.g., over- or under-representing certain groups or outcomes)

Saturdays.AI
Kigali

Poor-Quality Data

- A major part of a data scientist’s time is spent **cleaning and preprocessing data**
- High-quality data often matters **more than complex algorithms**

Garbage In, Garbage Out

- Errors, noise, and outliers in training data can **hide real patterns**
- Leads to poor generalization and weak model performance

Examples of Data Issues:

- **Outliers**: Remove or manually correct obvious errors
- **Missing values**:
  - Drop the feature or the affected instances
  - **Impute** missing values (e.g., use median or average)
  - Train **separate models** with and without the feature

Saturdays.AI
Kigali

Irrelevant Features

Garbage In, Garbage Out

- Your model's success depends on having **relevant, meaningful features**
- Too many irrelevant features = noise, overfitting, and poor performance

The Art of Feature Engineering

A key step in building effective ML models. It includes:

- **Feature selection**: Choosing the most useful features from existing ones
- **Feature extraction**: Creating better features by combining or transforming existing ones
    - (e.g., via dimensionality reduction techniques like PCA)
- **Feature creation**: Gathering new data or constructing new features from domain knowledge

Saturdays.AI
Kigali

Bad Algorithm

Saturdays.AI
Kigali

Overfitting the Training Data

- Happens when a model learns the **training data too well**, including its **noise and outliers**
- Performs well on training data but **poorly on new, unseen data**

Example:

- A complex model fits every fluctuation in training data, but **fails to generalize** (*like memorizing answers instead of understanding the material*)

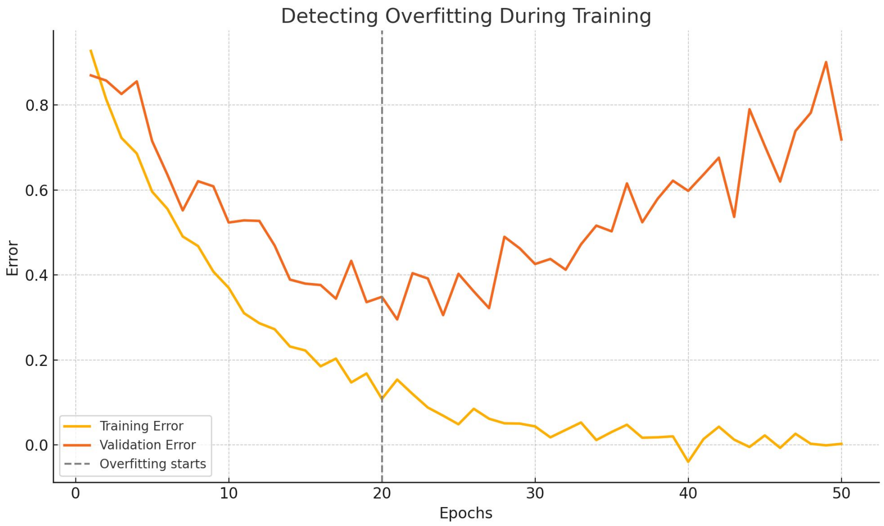
How to Fix Overfitting

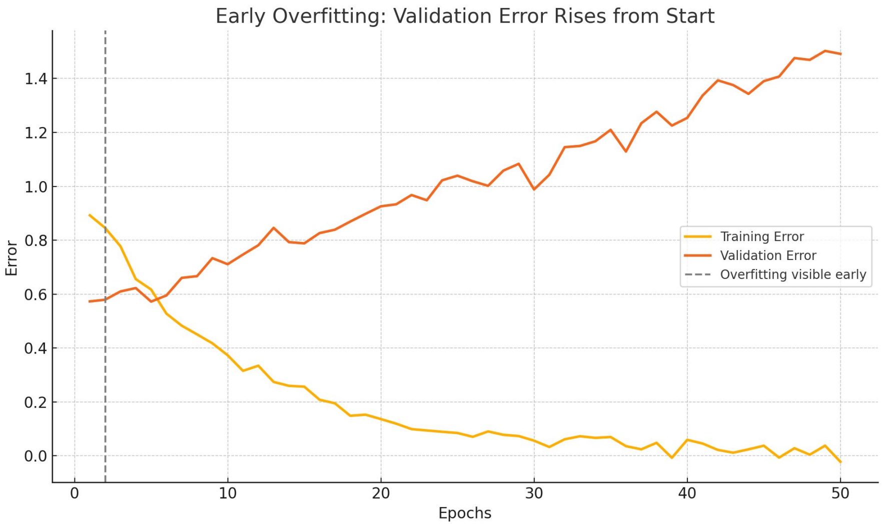
- Use a **simpler model** (fewer parameters)
- **Regularize** the model (e.g., L1 or L2 regularization) - to reduce the model complexity
- Use more **training data** if possible
- Apply **cross-validation** to monitor generalization
- Use techniques like **early stopping** or **dropout** (in neural networks)

Saturdays.AI
Kigali

Signs of Overfitting! Validation set

- Detecting overfitting is only possible once we move to the testing phase.
- Don't trust yourself
- Overfitting is massive for smaller datasets
- Ideally have a part of the dataset you don't have access to (validation set)
- Even some signs for *huge* datasets (imagenet)





Underfitting the Training Data

- Occurs when a model is **too simple** to capture the underlying patterns in the data
- Leads to **poor performance**, even on the training set

Example:

- A **linear model** trying to predict life satisfaction — too simplistic for a complex reality

How to Fix Underfitting

- Select a **more powerful model**, with more parameters.
- Feed better features to the learning algorithm (**feature engineering**).
- **Reduce the constraints** on the model (e.g., reduce the regularization hyperparameter).

Saturdays.AI
Kigali

Underfitting vs Overfitting

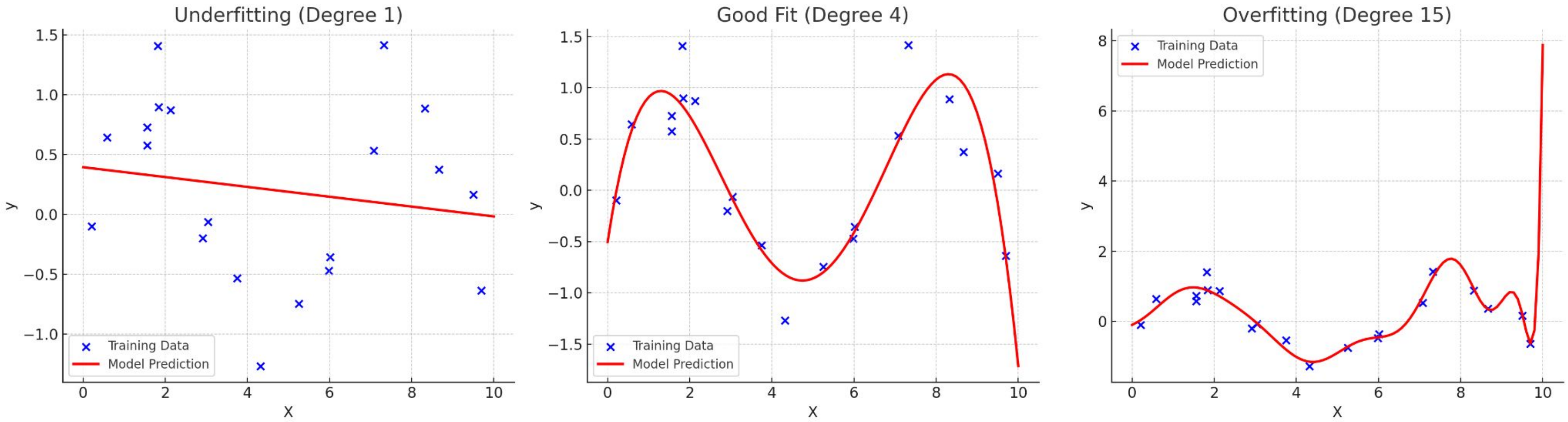


Figure 1.9: Underfitting vs Overfitting

Saturdays.AI Kigali

Testing and Validating

Saturdays.AI
Kigali

Testing and Validating your Model

The only way to know how well a model will generalize to new cases is to actually try it out on new cases.

Goal: Estimate Generalization Performance

- We want to know how well the model performs on **unseen data**
- The safest way: **test it on data it hasn't seen during training**

Train/Test Split

- **Training Set**: Used to train the model
- **Test Set**: Used to evaluate how well the model generalizes
- The error on the test set is called the **generalization error** (or **out-of-sample error**)
- **Common Practice**: 80% training, 20% testing
    - For very large datasets, even a 1% test set can be enough for reliable evaluation

Overfitting Warning

- If **training error is low** but **test error is high** → your model is overfitting

Saturdays.AI
Kigali

Testing and Validating your Model

The Problem:

- Choosing the best model and hyperparameters **by testing repeatedly on the test set** can lead to **overfitting to the test set**
- This makes your final model **less reliable** on truly unseen data

The Solution: Holdout Validation

1. **Split training data** into:
   - A training subset
   - A validation set (aka dev set)
2. **Use the validation set** to:
   - Compare different models
   - Tune hyperparameters (e.g., regularization strength)
3. Evaluate **only once** on the **test set** for a true estimate of generalization error

Saturdays.AI
Kigali

A roadmap for building machine learning systems

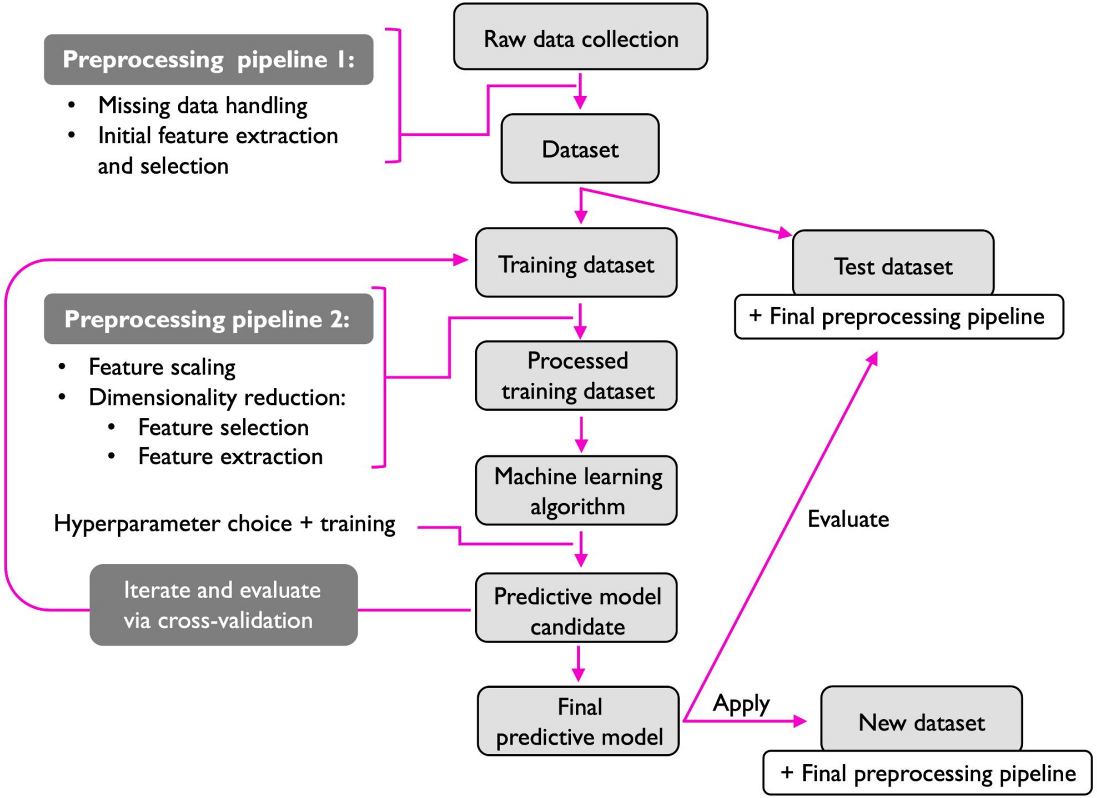


Figure 1.10: Predictive modeling workflow

Saturdays.AI Kigali

Packages for Machine Learning

- NumPy >1.21.2
- SciPy >1.7.0
- Scikit-learn >1.0
- Matplotlib >3.4.3
- Pandas >1.3.2

Check Your Understanding

Saturdays.AI
Kigali

Training and Test Sets

Explore the options below.

We looked at a process of using a test set and a training set to drive iterations of model development. On each iteration, we'd train on the training data and evaluate on the test data, using the evaluation results on test data to guide choices of and changes to various model hyperparameters like learning rate and features. Is there anything wrong with this approach? (Pick only one answer.)

A. This is computationally inefficient. We should just pick a default set of hyperparameters and live with them to save resources.
B. Totally fine, we're training on training data and evaluating on separate, held-out test data.
C. Doing many rounds of this procedure might cause us to implicitly fit to the peculiarities of our specific test set.

Training and Test Sets

Explore the options below.

We looked at a process of using a test set and a training set to drive iterations of model development. On each iteration, we'd train on the training data and evaluate on the test data, using the evaluation results on test data to guide choices of and changes to various model hyperparameters like learning rate and features. Is there anything wrong with this approach? (Pick only one answer.)

A. This is computationally inefficient. We should just pick a default set of hyperparameters and live with them to save resources.
B. Totally fine, we're training on training data and evaluating on separate, held-out test data.
C. Doing many rounds of this procedure might cause us to implicitly fit to the peculiarities of our specific test set.

Break

Practice

We give you code. What happens?

Run. Learn. Repeat.

Saturdays.AI
Kigali

Challenges & Next steps!

Best part of the day! 🥳

Scan this beautiful code!

Saturdays.AI
Kigali

Kahoot!

Any questions?

Saturdays.AI
Kigali

Saturdays.AI
Kigali

THANKS

kigali@saturdays.ai
coming soon
coming soon