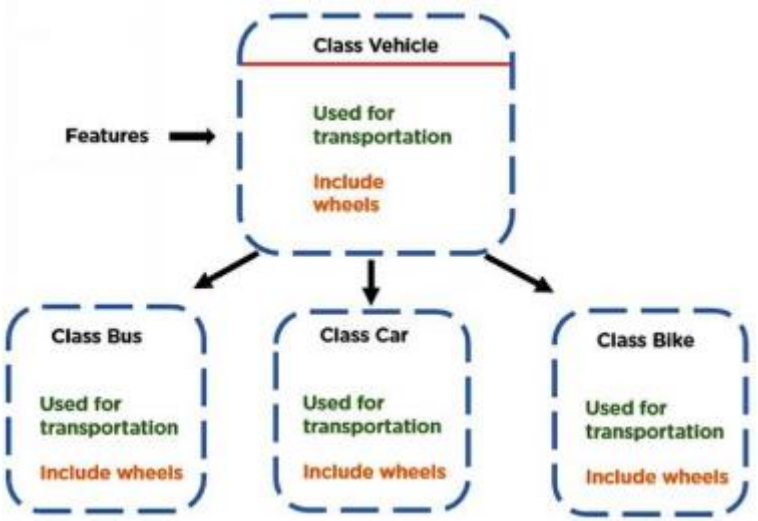
Introduction to Object-Oriented Programming (OOP)

What is Object-Oriented Programming (OOP)?

It helps organizes code into objects. Objects are created from classes, which are *templates* that define their properties and behavior. It helps make the code modular, reusable, and scalable.



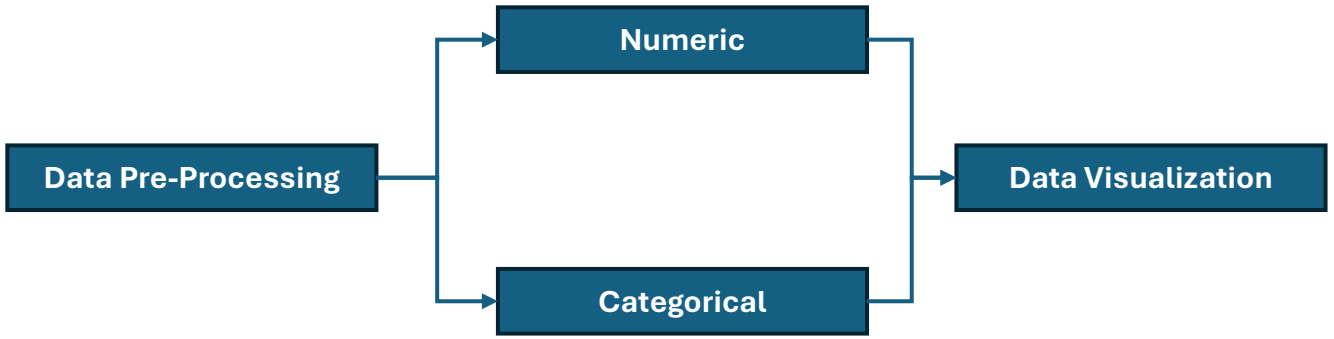
OOP in Data Science

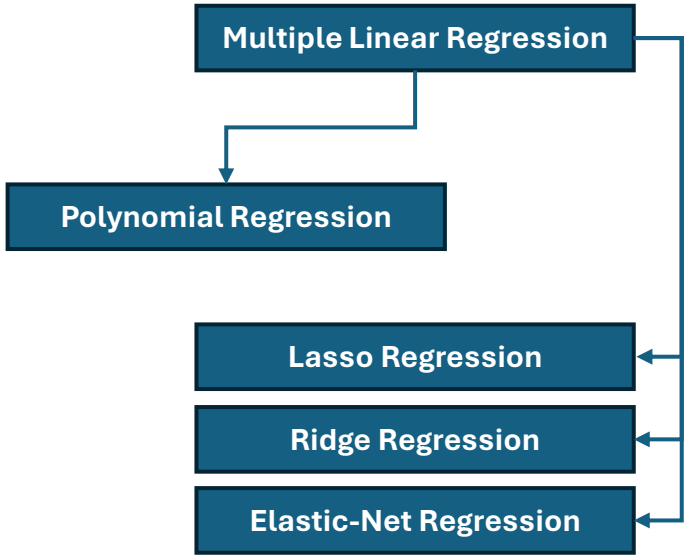
- Protect sensitive data in pre-processing and models through **encapsulation**.
- It allows reuse of functions for multiple ML models through **inheritance**.
- One can train different models using a **common interface**.

Application of OOP in ML

Why do we need OOP in ML?

Many algorithms in ML are extensions of other algorithms. As a result, re-writing the code for individual algorithms can lead to inconsistencies.





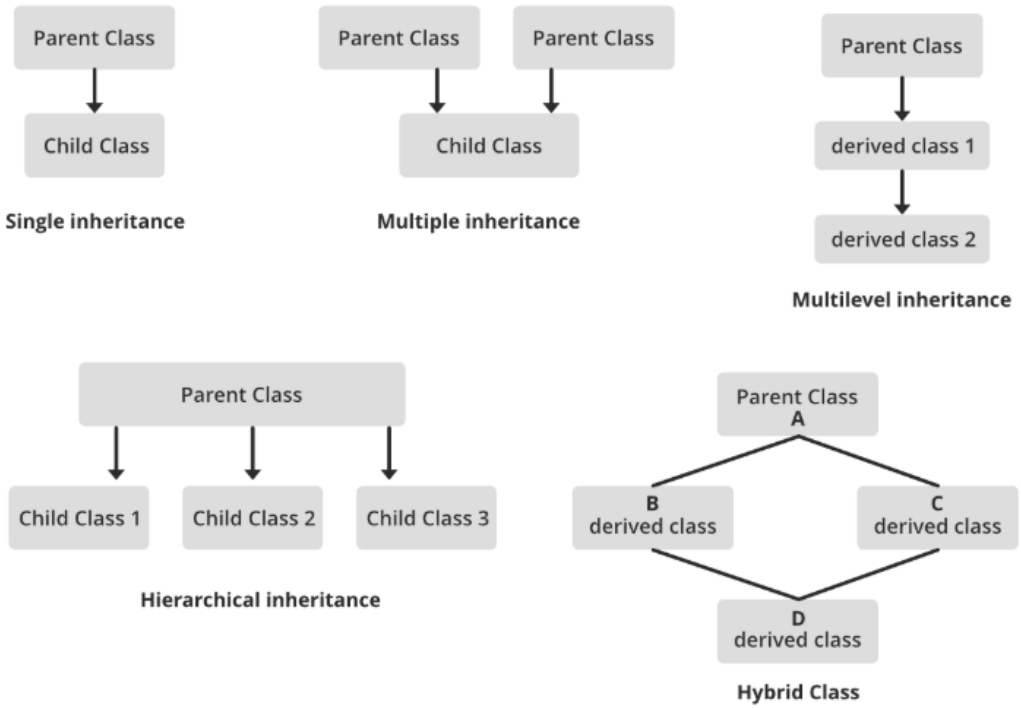
OOP in Data Science

- Protect sensitive data in pre-processing and models through **encapsulation**.
- It allows reuse of functions for multiple ML models through **inheritance**.
- One can train different models using a **common interface**.

Introduction to Object-Oriented Programming (OOP) for Data Science

Inheritance can be quite flexible

Child classes can inherit from more than one classes, significantly extending its functionality



Key Characteristics of Object-Oriented Programming

Encapsulation

It helps **restricts direct access** to certain details of an object and only allowing controlled interaction through methods.

- Protects Sensitive Data
- Hides implementation details while exposing only what's necessary

Inheritance

It allows a new child class to **inherit** attributes and methods from an existing (parent) class, resulting in code reusability.

- Avoids duplication of code by reusing logic
- Allows hierarchical organization of classes
- Makes it easier to extend and maintain large codebases

Polymorphism

It allows the same methods name to have **different implementations** depending upon the class it is used in.

- Provides flexibility, allowing different objects to be treated in a unified way
- Support method overriding, where child classes provide their own implementations of a method