Used design patterns

Anichitoaie Beatrice-Roxana:

Iterator (behavioral): The “for” loop from python uses this design pattern: the details about how a data structure is traversed is moved into an "iterator" object that, from the outside, simply yields one item after another without exposing the internals of how the data structure is designed. We will use the for loop for traversing the samples for train and test.

Prototype (creational): It creates objects by cloning existing objects. We will use this pattern in machine learning models (because they require much resources): we can keep the same model and change the value only for one or two variables. At the beginning of the application we will initialize a model, and then we will train for a certain data set. After some time, we will copy the model and update it (retrain with new data), to estimate the value of objects at a specific time.

Also, In Java it is supported through the clone() method defined in class Object and the use of java.lang.Cloneable interface to grant permission for cloning. We can also use în Java Hibernate ORM framework or JPA annotations to enable prototyping of database objects.

Cocei Tiberiu:

Composite (structural): Compose objects into tree structures to represent part-whole hierarchies. The javax.faces.component.UIComponent#getChildren() method is an example of this design pattern. It allows programmers to obtain the child UIComponents of a UIComponent. We will use this method while creating the GUI.

Singleton (creational): Ensure a class has only one instance, and provide a global point of access to it. The java.lang.Runtime#getRuntime() method is an example of this design pattern. We will be using this method indirectly, through the code that we will write. We may use this design pattern directly in Python for obtaining the database connection.

Pirlog Marcel Ionut:

Decorator (structural): Decorator pattern allows a user to add new functionality to an existing object without altering its structure. In the javax.servlet package, there are classes like javax.servlet.http.HttpServletRequestWrapp and HttpServletResponseWrapper that are based on Decorator design pattern.

Builder (creational): The intent of the Builder design pattern is to separate the construction of a complex object from its representation. In our project this design pattern is used through the annotation "@Builder" present in lombik.