# Primo prompt con contesto

I am facing a problem translating a Python script into an MLflow recipe. To solve this issue, I need you to gather a detailed description and practical applications of the following technologies in the context of machine learning. Please retrieve relevant and detailed information for each topic:

1. **Python**
2. **MLflow**
3. **Recipe**
4. **Jinja2 YAML**
5. **Machine Learning**

These technologies are essential for addressing the problem of translating the Python script into an MLflow recipe.

# Prompt di spiegazione della traduzione – Ingest

To best identify the code to include in the **ingest step**, you need to locate where in the first portion of the script the dataset used for training the neural network is **created, retrieved, and aggregated**.

I’m providing you with example pseudocode for the ingest step, which you can use to address the translation problem I’ll present later. Here is the pseudocode:

**Pseudocode for the Ingest Step**

**Define the routines needed for a specific step in a process (e.g., 'ingest' in a regression recipe)**

1. **Import the necessary modules and libraries**, such as:
   * logging for log management
   * DataFrame from the pandas library for data manipulation
2. **Configure the logger** to record log messages:
   * Create a logger instance
3. **Define a function to load data** from a file in a specific format (e.g., CSV), or from the data source identified in the script to be translated:
   * Specify the function parameters (file path, file format)
   * Document the function with a docstring describing its purpose and parameters
4. **Implement the logic to load data based on the file format**:
   * If the file format is supported (e.g., CSV):
     + Import the necessary library (e.g., pandas)
     + Log a warning if needed
     + Load the file using the appropriate method (e.g., pandas.read\_csv)
     + Apply any necessary data transformations (e.g., add columns)
     + Return the resulting DataFrame
   * If the file format is not supported:
     + Raise an appropriate exception (e.g., NotImplementedError)
5. **Ensure the function is easily extendable or modifiable** to support new file formats or loading logic.
6. **Create the ingest step for the MLflow recipe**, including:
   * Type of ingestion file (e.g., CSV)
   * A location
   * The loader method

**End of Pseudocode**

Make sure the MLflow recipe and the step script are two separate files. Below is an example of the MLflow recipe file:

# Example MLflow recipe file

recipe: "recipename/v1"

target\_col: "targetcolumnlabel"

positive\_class: 1

primary\_metric: "f1\_score"

steps:

ingest: {{INGEST\_CONFIG}}

# Example local.yaml file

experiment:

name: "experiment\_name"

tracking\_uri: "sqlite:///metadata/mlflow/mlruns.db"

artifact\_location: "/metadata/mlflow/mlartifacts"

model\_registry:

model\_name: "red\_wine\_classifier" # for example

INGEST\_CONFIG:

using: csv

location: ["/data/winequality-white.csv", "/data/winequality-red.csv"]

loader\_method: method\_name

# Prompt di spiegazione della traduzione – Split

To best identify the code to include in the **split step**, you need to locate where in the first portion of the script the dataset is **split or filtered**, and also retrieve the **split parameters** to include them in the local.yaml file as SPLIT\_RATIOS.

I’m providing you with example pseudocode for the split step, which you can use to address the translation problem I’ll present later. This pseudocode will be necessary if the provided script involves dataset filtering. Here is the pseudocode:

**Define the routines needed for a specific step in a process (e.g., 'split' in a regression recipe)**

1. **Import the necessary modules and libraries**, such as:
   * DataFrame and Series from the pandas library for data manipulation
2. **Optionally configure the logger** to record log messages:
   * Create a logger instance
3. **Define a function to filter the data** into training, validation, and test datasets:
   * Specify the function parameters (e.g., dataset)
   * Document the function with a docstring describing its purpose and parameters
4. **Implement the logic to filter the data**:
   * Create a boolean Series indicating whether each row should be filtered
   * Return the resulting Series
5. **Ensure the function is easily extendable or modifiable** to support new filtering logic.

**End of Pseudocode**

**MLflow Recipe Split Step Template**

split:

split\_ratios: {{SPLIT\_RATIOS|default([0.75, 0.125, 0.125])}}

**local.yaml File Specification**

SPLIT\_RATIOS: [0.80, 0.10, 0.10]

# Prompt di spiegazione della traduzione – Transform

To best identify the code to include in the **transformation step**, you need to locate where in the central portion of the script the data is **transformed**. It is also important to retrieve the **transformation parameters** to include them in the local.yaml file as TRANSFORM\_PARAMS.

I’m providing you with example pseudocode for the transformation step, which you can use to address the translation problem I’ll present later. This pseudocode will be necessary if the provided script involves dataset transformation. Here is the pseudocode:

1. **Define a function named transformer\_fn:**
   * This function should return an **untrained transformer**.
2. **Inside the transformer\_fn function:**
   * Initialize a transformer object.
   * Define a method called fit that accepts training data and trains the transformer.
   * Define a method called transform that accepts input data and transforms it according to the defined logic.
3. **Assign the fit and transform methods to the transformer object:**
   * Ensure the transformer has both fit and transform methods properly defined.
4. **Return the untrained transformer object or None:**
   * The transformer\_fn function should return the transformer with the fit and transform methods ready to be used, or None if the original script did not use a transformer.

# Prompt di spiegazione della traduzione – Train

To best identify the code to include in the **training step**, you need to locate where in the central portion of the script the **model is trained**.

During model training, the transformation step is crucial to ensure that the data is properly prepared. Here is an example of pseudocode:

1. **Module Description:**
   * This module defines a routine used in the **'train' step** of a model.
   * The main routine is, for example, estimator\_fn, which defines the customizable estimator type and the parameters used during training to produce a model.
2. **Imports:**
   * For example, Dict and Any are imported from the typing module to specify the data types of the parameters.
3. **Definition of the estimator\_fn function:**
   * The estimator\_fn function accepts an optional parameter estimator\_params of type Dict[str, Any].
   * If estimator\_params is not provided (i.e., is None), it is initialized as an empty dictionary.
4. **Importing the SGDClassifier class:**
   * Inside the function, the SGDClassifier class is imported from the sklearn.linear\_model module.
5. **Creating and returning the estimator:**
   * The function creates an instance of SGDClassifier, using random\_state=42 to ensure reproducibility.
   * Any additional parameters provided in estimator\_params are passed to the SGDClassifier constructor.
   * The function returns the **unfitted estimator**, which has fit() and predict() methods compatible with scikit-learn estimators.

# Prompt di spiegazione della traduzione – Evaluate, Register, Ingest\_scoring, Predict

Based on the script that I will provide later, you will need to extract the evaluation parameters to be included as **thresholds** in the recipe.yaml file, along with other parameters (which I will describe in detail) that must be placed in separate steps.

* **Validation Thresholds**: Identify the metrics used to evaluate the model’s performance (e.g., f1\_score, precision\_score, recall\_score) and determine the thresholds used for each metric present in the script I will provide.
* **Registration Permissions**: Specify whether a model that does not meet the thresholds can still be registered (set a boolean value for allow\_non\_validated\_model). This should be identified in the code if it is handled; otherwise, set a default value such as false.
* **Ingest Configurations**: Provide any configurations for the ingest\_scoring section (e.g., input data path or format). If an ingest\_scoring section is present in the script I will provide, you must extract the parameters and insert them into the local.yaml file, while in the recipe.yaml file you must include the reference using {{}} to point to the parameters.
* **Prediction Configurations**: If the script I will provide includes a prediction section, it must be analyzed to extract parameters such as the data format type and data location. These parameters must be added to the local.yaml file and referenced using {{}} in the recipe.yaml file.

# Prompt di sottomissione della traduzione

Based on the information obtained earlier, take this Python script and translate it into an **MLflow recipe**. Remember to:

* Separate different scripts for each step.
* Create the recipe.yaml and local.yaml files as previously described.
* Include the necessary **Jinja dependencies** and refer to the **previous outputs**.
* From the script, you will be able to determine the correct value for "recipename" — either classification or regression.

Here is the script: