Dataset Description

You are given two CSV files:

- **development.csv** – labeled dataset, containing 20 attributes and one target column.
- **evaluation.csv** – unlabeled dataset with the same schema as the development set.

The evaluation labels are hidden and will be used by the evaluator for grading.

Note on Feature Types

The dataset columns follow this convention:

- **var_0 → var_9**: Continuous numerical features.
- **var_10 → var_14**: Ordinal features.
  For each column, the possible values are ordered as: `val_0, val_1, ..., val_N`.
  Even if the same string values appear across different columns, their semantic meaning and ordering are column-specific and must be treated independently.
- **var_15 → var_20**: Categorical features, without intrinsic ordering.

Correct handling of these feature types is a core part of the evaluation.

Task

You must implement a complete regression pipeline that:

1. Loads and inspects the development dataset.
2. Applies appropriate preprocessing.
3. Trains a model of your choice.
4. Generates predictions for all rows in `evaluation.csv`.
5. Saves the predictions to a file named **submission.csv**, following the provided template.

The performance of your solution will be evaluated using the coefficient of determination (**R²**) on the hidden evaluation labels.

Implementation Requirements

You must create a ZIP file containing the following files:

- **main.py**: a single executable Python script.
- **submission.csv**: the file containing predictions for the test set.
- **explore.ipynb** or **explore.py**: code for exploratory data analysis and model selection. Note that this part should contain exploratory code and/or code for hyperparameter tuning; it will not be subject to execution. It will be checked to verify that it is consistent with the exploratory part of your textual solution.

Your `main.py` file should be self-consistent and not rely on other scripts, artifacts, or checkpoints.
You can assume that the only available files will be the development and evaluation sets.

The training of all models must occur **on-the-fly** when `main.py` is executed. You can explore multiple configurations of hyperparameters during the exam; however, `main.py` should only run your **best** configuration.

Your `main.py` script is expected to run for **at most 150 seconds**.

Evaluation Process

An evaluation process will be run on your output files. The evaluation process will:

1. Re-run `main.py` in a clean, offline environment and verify it completes within 150 seconds.
2. Regenerate `submission.csv`.
3. Verify schema, row count, and reproducibility.

If the evaluator cannot reproduce your results, the submission will be penalized or discarded.

If your solution involves any randomness (e.g., data splitting, model initialization), you should **fix a random seed** to ensure deterministic results.
Failure to ensure reproducibility may invalidate your submission.

You can refer to the provided code snippet to fix seeds for the most common libraries you may use.

```python
import random
import numpy as np
import torch

random.seed(42)
np.random.seed(42)
torch.manual_seed(42)
# for skelearn, use random_state=42 in all relevant functions
```

Invalid or Penalized Submissions

A submission will be considered invalid if any of the following occurs:

- `main.py` crashes or fails to run.
- Execution exceeds the time limit.
- `submission.csv` is missing or malformed.
- Wrong filenames or wrong directory location.
- Non-reproducible results (mismatch between uploaded CSV and generated CSV file).
- Use of pre-trained models or pre-computed artifacts.