

ARV ML Dataset Export — User Manual

Product: ARV ML Dataset Export for MetaTrader 5

Component: ARV_ML_Export .ex5 (script)

Version: 1.00

Category: Utilities

Contents

1. What it does	1
2. Why it is built this way	1
3. Installation	2
4. Running the script	2
5. Inputs	2
6. Output format	2
7. Using the dataset for training	3
8. Troubleshooting	3
9. Disclaimer	4

1. What it does

ARV ML Dataset Export turns price history into a machine-learning dataset that is ready for training an ONNX model outside the terminal. One run on any chart produces two files:

File	Content
ARV_ML_<SYMBOL>_H1.csv	feature matrix, TP/SL labels, excursions, train/test split
ARV_ML_Norm_<SYMBOL>_H1.json	normalization statistics fitted on the training part only , plus a causality report

The script reads history and writes files into the terminal sandbox. It never trades, never places orders and never sends data anywhere.

2. Why it is built this way

Two design decisions matter more than the code itself.

Normalization is fitted on the training window only. Fitting mean/std on the whole history and then training on a part of that same history leaks future information into the model: the features of 2016 are scaled by statistics computed from 2026. The exported JSON therefore also contains `full_means`, `full_std`s and a `causal_check` block that quantifies the difference. If the difference exceeds the tolerance you set in `InpCausalToLPct`, the journal prints a FAIL line and you know that full-sample normalization would bias the model.

The train/test split is a date, not a random shuffle. Financial series are not independent; a random split puts neighbouring bars from the same market regime into both parts. InpSplitDate keeps the split chronological.

3. Installation

1. Copy ARV_ML_Export.ex5 to <terminal data folder>\MQL5\Scripts\.
2. In MetaTrader 5 press F5 in the Navigator so the new script appears.
3. No libraries, no DLL, no WebRequest, no external services are used.

4. Running the script

1. Open a chart of the instrument you want to export. The chart period does not matter: all data is taken from H1, H4 and M15 series of that symbol.
2. Drag ARV_ML_Export from the Navigator onto the chart, or double-click it.
3. Check the inputs and press OK.

Wait for the line ARV ML Export: done, ... rows in the journal (Toolbox -> Experts).

5. Inputs

Input	Default	Meaning
InpFrom	2015.01.01	Export start date, server time
InpSplitDate	2023.01.01	Boundary between the train and test parts
InpHorizonBars	48	How many H1 bars are scanned for a TP/SL outcome
InpSL_ATR_Mult	1.5	Stop loss distance in ATR(14,H1) multiples
InpTP_ATR_Mult	3.0	Take profit distance in ATR(14,H1) multiples
InpAmbigPessimistic	true	Bar touched both TP and SL: count it as a loss
InpUseCommon	true	Write to the shared terminal folder instead of MQL5\Files
InpPrefix	ARV_ML	Prefix of the output file names
InpCausalTolPct	5.0	Tolerance of the causality report, percent

6. Output format

CSV header (comma separated, ASCII):

```
time,f0,f1,f2,f3,f4,f5,close,atr,dir,label,mfe_r,mae_r,split
```

Column	Meaning
time	open time of the signal bar, YYYY.MM.DD HH:MM
f0..f5	raw, unnormalized features of the closed H1 bar
close	close price of the signal bar
atr	ATR(14) of the H1 series at the signal bar
dir	trade direction 1 / -1 / 0 (sign of f1)
label	1 if TP was reached before SL, 0 if SL was reached first, empty if neither happened inside the horizon
mfe_r, mae_r	maximum favourable and adverse excursion inside the horizon, in R units
split	train or test

Feature definitions (all values as of the closed H1 bar; H4 and M15 values are taken from the last bar closed at that moment):

```
f0 = |EMA20(H1) - EMA50(H1)| / ATR14(H1)
f1 = (EMA20(H1) - EMA20(H1) 5 bars ago) / ATR14(H1)
f2 = (Close(H1) - EMA200(H1)) / ATR14(H1)
f3 = (BB_upper(H4) - BB_lower(H4)) / EMA20(H4)           Bollinger 20, 2.0
f4 = ATR14(M15) / Close(M15) * 100
f5 = StdDev20(H1, SMA) / EMA20(H1)
```

JSON keys, in order:

```
version, means[6], stds[6], schema, symbol, timeframe, split_date,
rows_train, rows_test, features[], full_means[6], full_stds[6],
causal_check{max_rel_diff, threshold, passed}
```

7. Using the dataset for training

Training is done outside the terminal, for example with Python and NumPy: load the CSV, normalize the features with means/stds from the JSON, train a small network and export it to ONNX. The export is deliberately framework-agnostic: any tool that can read CSV, JSON and write ONNX will do.

An example project that also writes the ONNX file without any ONNX package (a small protobuf writer) is published with the companion article; the models it produces load in MetaTrader 5 through `OnnxCreateFromBuffer`.

8. Troubleshooting

Symptom	Reason / fix
indicator buffers are not ready	the terminal is still loading history; wait and run again
too few rows	the symbol has short history or InpFrom is too late
empty file	the terminal has no H4/M15 history for that symbol; open those timeframes once to download them
many skipped rows	normal: bars without enough history for EMA200/ATR are skipped
causal check ... FAIL	train statistics differ from the full sample; keep normalization on train rows only

9. Disclaimer

The script performs no trading and gives no signals. It only produces data files. Backtest results and machine-learning models built on this data do not guarantee future performance.