

1) One-time: add preprocessing_sers.py to your repo

Use the preprocessing_sers.py module I gave you earlier (SG + IModPoly + MinMax fit-on-train). Put it next to your scripts (or in a utils/ folder).

2) Then apply it to each attached code (exactly where)

A) 1D-CNN-Simple.py

Insert immediately after:

```
X_train, Y_train = load_dir(train_data_dir, cultivar_cys_map, input_length)
X_test, Y_test = load_dir(test_data_dir, cultivar_cys_map, input_length)
```

Add:

```
from preprocessing_sers import PreprocessConfig, SERSPreprocessor

cfg = PreprocessConfig(
    input_length=input_length,
    use_savgol=True, sg_window_length=11, sg_polyorder=3, sg_mode="interp",
    use_imodpoly=True, imod_deg=7, imod_kmad=2.5, imod_max_iter=30, imod_tol=1e-6, imod_min_keep=0.05,
    use_minmax=True, minmax_lo=0.0, minmax_hi=1.0,
)

pp = SERSPreprocessor(cfg)
X_train = pp.fit_transform(X_train) # fit minmax on TRAIN only
X_test = pp.transform(X_test)
pp.save_minmax("minmax_params.npz")
```

B) LR.py

Insert immediately after:

```
X_train, Y_train = load_dir(train_data_dir, cultivar_cys_map, input_length)
X_val, Y_val = load_dir(val_data_dir, cultivar_cys_map, input_length)
```

Add the same block as above, but use X_val instead of X_test:

```
pp = SERSPreprocessor(cfg)
X_train = pp.fit_transform(X_train)
X_val = pp.transform(X_val)
pp.save_minmax("minmax_params.npz")
```

C) RFR.py

Same as LR.py (train/val). Insert right after its load_dir(...) calls and preprocess X_train (fit) + X_val (transform).

D) SVR.py **important: remove the existing StandardScaler**

Right after its train/val load_dir(...) calls, add preprocessing exactly like LR.py.

Then **change your SVR pipeline** because it currently does:

```
pipe = Pipeline([
    ("scaler", StandardScaler(...)),
    ("svr", SVR(...))
])
```

If you keep that scaler, you're **double-scaling** (MinMax + StandardScaler).
So update to:

```
pipe = Pipeline([
    ("svr", SVR(kernel="rbf", cache_size=1000, tol=1e-3, max_iter=-1))
])
```

(or set "scaler" to "passthrough").

E) PLS.py **important: keep scale=False**

In PLS.py you load like:

```
X_train, Y_train, train_labels, expected_length = load_dir(...)
X_val, Y_val, val_labels, expected_length = load_dir(...)
```

Insert preprocessing immediately after both loads, using `expected_length`:

```
from preprocessing_sers import PreprocessConfig, SERSPreprocessor

cfg = PreprocessConfig(
    input_length=expected_length,
    use_savgol=True, sg_window_length=11, sg_polyorder=3, sg_mode="interp",
    use_imodpoly=True, imod_deg=7, imod_kmad=2.5, imod_max_iter=30, imod_tol=1e-6, imod_min_keep=0.05,
    use_minmax=True, minmax_lo=0.0, minmax_hi=1.0,
)

pp = SERSPreprocessor(cfg)
X_train = pp.fit_transform(X_train)
X_val = pp.transform(X_val)
pp.save_minmax("minmax_params.npz")
```

Then ensure **both places** where you create `PLSRegression(...)` use:

```
pls = PLSRegression(scale=False)
```

Because `scale=True` would again apply extra scaling on top of your MinMax.

Summary (what changes in every file)

- Add:
 - `from preprocessing_sers import PreprocessConfig, SERSPreprocessor`
- Add preprocessing block **right after loading X/y**, before training.
- Remove/disable any **extra scaling** inside the model pipeline (`StandardScaler`, `PLSRegression(scale=True)`) if you want the preprocessing to be exactly SG + IModPoly + MinMax.