Advanced Topics in Macro and Finance to Deal with Big Data

Alessia Paccagnini

Road Map

- ❑ Models used for forecasting
- ❑ MATLAB Toolboxes
- ❑ Codes for:

a) Factors Model

b) Lasso Regression

c) Ridge Regression

d) Elastic Net Regression

Empirical Framework

- In this lecture, we are going to use four models that deal with high dimensionality: Factor, Lasso, Ridge, and Elastic Net Regressions to forecast macro series.
- We rely on the FRED-MD monthly panel of US macroeconomic and financial variables from McCracken and Ng (2016).
- We forecast Inflation (Consumer Price Index).

Classic Least Square Estimator

The classic estimator for θ is the least squares estimator, defined as

$$\begin{aligned} \hat{\theta}^{LS} &= \arg\min_{\theta} \sum_{i=1}^{T} (Y_t - \theta' X_t)^2 \\ &= (X'X)^{-1} X'Y \end{aligned}$$

Factors Model

We add principal components («Factors») into the OLS estimation of the model

$$X_t = \Lambda^f F_t + \Lambda^y Y_t + v_t$$

X_t informational time series related to unobserved factors F_t and observed variables Y_t

Lasso Regression

The LASSO estimator of this model is defined as

$$\hat{\theta}^L_\lambda = \arg\min_\theta \sum_{t=1}^{T} (Y_t - \theta' X_t)^2 + \lambda \sum_{i=1}^{p} |\theta_i| \quad \lambda \geqslant 0$$

where λ is the LASSO tuning parameter

Ridge Regression

The ridge estimator of this model is defined as

$$\hat{\theta}_\lambda^R = \arg\min_\theta \sum_{t=1}^{T} (Y_t - \theta' X_t)^2 + \lambda \sum_{i=1}^{p} |\theta_i|^2 \quad \lambda \geqslant 0$$

where λ is the ridge tuning parameter

Lasso vs. Ridge Regression

- The LASSO has a major advantage over RIDGE regression, in that it produces simpler and more interpretable models that involved only a subset of predictors.

- The LASSO leads to qualitatively similar behavior to RIDGE regression, in that as λ increases, the variance decreases and the bias increases.

- The LASSO can generate more accurate predictions compared to RIDGE regression.

- Cross-validation can be used in order to determine which approach is better on a particular data set.

Elastic Net

The elastic net estimator is defined as

$$\hat{\theta}_{\lambda}^{EN} = \arg\min_{\theta} \sum_{t=1}^{T} (Y_t - \theta' X_t)^2 + \lambda \sum_{i=1}^{p} (\alpha|\theta_i|^2 + (1-\alpha)|\theta_i|)$$

where $\alpha \in (0,1)$, $\lambda \geqslant 0$

MATLAB Toolboxes

Users should also ensure that they have installed the Econometrics and Statistics and Machine Learning Toolboxes with MATLAB version R2021a+

General Setting

```
clear; close; clc;
addpath(genpath(pwd));

%% FORECASTING FRED-MD WITH FACTORS


%% READ IN AND PREPARE DATA

% read in fred-md stationary transformed series
data = readtable('fred_md_stationary.csv');

% read in pca factors
factors = readtable('fred_md_pca_factors.csv');
factors = factors{:,2:end};

% set dates
date = datetime(data.sasdate);

% set of forecasting targets
INDPRO = data.INDPRO;                      % industrial production
UNRATE = data.UNRATE;                      % unemployment rate
CPI    = data.CPIAUCSL;                    % inflation as captured by cpi
SPREAD = data.GS10 - data.FEDFUNDS;        % 10-year T rate - Fed funds rate
HOUST  = data.HOUST;                       % housing starts

% split data in-sample (training), hyper-parameter tuning (validation), and
% out-of-sample (test) -- (1/3, 1/3, 1/3) split
len = ceil(length(date) / 3);
train_ind = 1:len;
val_ind   = (len+1):(2*len);
test_ind  = (2*len+1):length(date);
```

- We upload data and factors
- We set the data line and select the target variables

Code for Factor Model

```
%% SETTINGS

% pick the forecasting target
YY = CPI;

% pick the forecasting horizon
h = 1; % 3,9,12,24

% pick the maximum number of lags
p_max = 12;

% model selection method
%     ictr    = in-sample information criteria on training sample
%     icval   = in-sample information criteria on validation sample
%     pooscv  = pseudo oos cross-validation with expanding window (takes longer)
%     kfcv    = K-fold cross-validation
model_selection = 'icval';

% pick the in-sample information criterion for in-sample model selection
%   (only matters when model selection is icval or ictr)
%     Akaike (AIC)        = 1
%     Bayesian (BIC)      = 2
%     Hannan-Quinn (HQ)   = 3
icidx = 2;

% pick K if model selection method is K-Fold (default is 5)
```

```
K      = 5;

% rolling window indicator for pseudo-oos (default is expanding)
roll = 0;

% rolling window length (if roll=1);
wL = 36;

% specify benchmark for out-of-sample performance comparison (random walk
% (RW) or prevailing mean (PM)
benchmark = 'RW';

% max lag order in predictor
py_max      = 12;

% max lag order in factors
pf_max      = 12;

% max number of principal components
nf_max      = size(factors,2);

% index all possible combinations of hyperparameters
[a,b,c] = ndgrid(1:py_max,1:pf_max,1:nf_max);
combinations = [reshape(a,[],1), reshape(b,[],1), reshape(c,[],1)];
```

We set features

Code for Factor Model

```
%% AR with Factors MODEL - OPTIMIZE OVER HYPERPARAMETERS

% preallocate for in-sample model (lag-length) selection
insampIC = NaN(size(combinations,1),3);
val_err  = NaN(size(combinations,1),1);

% implement autoregressive model with in-sample lag length selection
for i = 1:size(combinations,1)

    tic;

    % hyperparams
    py        = combinations(i,1);
    pf        = combinations(i,2);
    nf        = combinations(i,3);

    % model evaluation
    switch model_selection

        case 'icval'

            % initialize target and predictors
            [Xtr,Ytr] = add_lags(YY(train_ind),factors(train_ind,1:nf),pf,h,0,py);

            % estimate OLS
            S = OLS(Xtr,Ytr);

            % evaluate on validation set
            [Xval,Yval] = add_lags(YY(val_ind),factors(val_ind,1:nf),pf,h,0,py);
            yhat = Xval * S.Beta;

            % store information criterial
            insampIC(i,:) = IC(Yval,yhat, length(Yval), size(Xval,2));

        case 'ictr'

            % initialize target and predictors
            [Xtr,Ytr] = add_lags(YY(train_ind),factors(train_ind,1:nf),pf,h,0,py);

            % estimate OLS
            S = OLS(Xtr,Ytr);
```

Code for Factor Model

```
        % in-sample evaluation on training set
        yhat = Xtr * S.Beta;
        insampIC(i,:) = IC(Ytr,yhat, length(Ytr), size(Xtr,2));

    case 'pooscv'

        % pseudo-oos evaluation (using last 25% of in-sample data)
        yhat = NaN(length(val_ind), 1);

        for t = 1:length(val_ind)

            % initialize data up to t in test set
            endInd = val_ind(t) - h;
            temp_ind = 1:(endInd-1);

            [Xtr,Ytr] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);

            % estimate OLS
            S = OLS(Xtr,Ytr);

            % forecast for t+h
            temp_ind = 1:endInd;
            [Xte,Yte] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);
            Xte = Xte(end,:);
            ythat = Xte * S.Beta;

            yhat(t) = ythat;
        end

        ytrue = YY(val_ind);
        eval_cv = MSPE(ytrue,yhat);
        val_err(i) = eval_cv.MSPE;

    case 'kfcv'

        % default is 5-Fold CV
        try K; catch; K=5; end
        split = floor(linspace(1,val_ind(end),K+2));

        tmp_errs = NaN(K,1);
```

Code for Factor Model

```
        % in-sample evaluation on training set
        yhat = Xtr * S.Beta;
        insampIC(i,:) = IC(Ytr,yhat, length(Ytr), size(Xtr,2));

    case 'pooscv'

        % pseudo-oos evaluation (using last 25% of in-sample data)
        yhat = NaN(length(val_ind), 1);

        for t = 1:length(val_ind)

            % initialize data up to t in test set
            endInd = val_ind(t) - h;
            temp_ind = 1:(endInd-1);

            [Xtr,Ytr] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);

            % estimate OLS
            S = OLS(Xtr,Ytr);

            % forecast for t+h
            temp_ind = 1:endInd;
            [Xte,Yte] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);
            Xte = Xte(end,:);
            ythat = Xte * S.Beta;

            yhat(t) = ythat;
        end

        ytrue = YY(val_ind);
        eval_cv = MSPE(ytrue,yhat);
        val_err(i) = eval_cv.MSPE;

    case 'kfcv'

        % default is 5-Fold CV
        try K; catch; K=5; end
        split = floor(linspace(1,val_ind(end),K+2));

        tmp_errs = NaN(K,1);
```

Code for Factor Model

```matlab
    case 'kfcv'
        [~,best_i] = min(val_err);
end

%% AR with Factors MODEL - IN-SAMPLE CRITERIA, PSEUDO-OOS FIT

% set best parameters
py        = combinations(best_i,1);
pf        = combinations(best_i,2);
nf        = combinations(best_i,3);
p         = max(pf,py);

% preallocate oos forecast
yhat = NaN(length(test_ind),1);

if roll

    for t = 1:length(test_ind)

        % initialize data up to t in test set
        endInd = test_ind(t) - h
        temp_ind = (endInd-wL-p-h):(endInd-1);

        [Xtr,Ytr] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);

        % estimate OLS
        S = OLS(Xtr,Ytr);

        % forecast for t+h
        temp_ind = (endInd-wL-p-h):endInd;
        [Xte,Yte] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);
        Xte = Xte(end,:);
        ythat = Xte * S.Beta;
        yhat(t) = ythat;

    end

else

    for t = 1:length(test_ind)

        % initialize data up to t in test set
```

Code for Factor Model

```
    case 'kfcv'
        [~,best_i] = min(val_err);
end

%% AR with Factors MODEL - IN-SAMPLE CRITERIA, PSEUDO-OOS FIT

% set best parameters
py        = combinations(best_i,1);
pf        = combinations(best_i,2);
nf        = combinations(best_i,3);
p         = max(pf,py);

% preallocate oos forecast
yhat = NaN(length(test_ind),1);

if roll

    for t = 1:length(test_ind)

        % initialize data up to t in test set
        endInd = test_ind(t) - h
        temp_ind = (endInd-wL-p-h):(endInd-1);

        [Xtr,Ytr] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);

        % estimate OLS
        S = OLS(Xtr,Ytr);

        % forecast for t+h
        temp_ind = (endInd-wL-p-h):endInd;
        [Xte,Yte] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);
        Xte = Xte(end,:);
        ythat = Xte * S.Beta;
        yhat(t) = ythat;

    end

else

    for t = 1:length(test_ind)

        % initialize data up to t in test set
```

Code for Forecasting

```
%% EVALUATE OOS PERFORMANCE

% true value
Y = YY(test_ind);

% plot forecast vs. true
plot_ts(date(test_ind), [Y, yhat], '', '',1, {'True', 'Forecast'}, 0)

% MSPE calculations
% 3 year window for rolling rmse
wL = 36;

switch benchmark
    case 'RW'
        ytemp      = forecast_RW(YY);
        bench_eval = MSPE(YY(test_ind), ytemp(test_ind),wL);
    case 'PM'
        ytemp      = forecast_PM(YY);
        bench_eval = MSPE(YY(test_ind), ytemp(test_ind),wL);
end


eval = MSPE(Y,yhat,wL);
disp(join(['Out-of-sample total MSPE is: ', num2str(eval.MSPE, "%.4f")]))

% Cumulative MSPE
plot_ts(date(test_ind),[eval.CUM_MSPE, bench_eval.CUM_MSPE], ...
    '', 'Cumulative RMSE',2, {'ARDI', benchmark}, 0)

% Rolling RMSPE
plot_ts(date(test_ind((1+wL):end)),[eval.ROLL_RMSPE, bench_eval.ROLL_RMSPE],...
    '', 'Rolling RMSE',3, {'ARDI', benchmark}, 0 )

%% OOS PERFORMANCE TESTS

% Diebold-Mariano test(1995)
e1 = eval.errors;
e2 = bench_eval.errors;
[DM,pval_L,pval_LR,pval_R]= dmtest(e1, e2, h);
```

```
% Mincer and Zarnowitz test(1969)
[MZstat, MZpval]=MZtest(Y,yhat);

% White and Hansen P-vals : c = consistent, u = upper, l = lower
[c,u,l]=bsds(e2,e1,1000,12,'STUDENTIZED','STATIONARY');

% model confidence set
[includedR,pvalsR,excludedR,includedSQ,pvalsSQ,excludedSQ]=mcs([e1,e2],0.05,1000,12,'STATIONARY');

%% SAVE OOS ERROR FOR LATER ANALYSIS
ARDI_errors = e1;
if isfile('FORECAST_ERRORS.mat'), save('FORECAST_ERRORS.mat', 'ARDI_errors','-append'); end
```

Code for Lasso Regression

```
% lasso hyperparameters
%     lambda = regularization parameter
lambda_max = 1e-4;
lambda_vec = linspace(0,lambda_max,5);

% index all possible combinations of hyperparameters
[a,b,c,d] = ndgrid(1:length(lambda_vec),1:py_max,1:pf_max,nf_max:nf_max);
combinations = [reshape(a,[],1), reshape(b,[],1), reshape(c,[],1), ...
    reshape(d,[],1)];

disp(join(['Testing ', num2str(size(combinations,1), "%.0f"), ' combinations of hyperparameters.']))

%% LASSO MODEL - OPTIMIZE OVER HYPERPARAMETERS

insampIC = NaN(size(combinations,1),3);
val_err  = NaN(size(combinations,1),1);

for i = 1:size(combinations,1)

    tic;

    % set hyperparameters
    lambda = lambda_vec(combinations(i,1));
    py     = combinations(i,2);
    pf     = combinations(i,3);
    nf     = combinations(i,4);

    % model evaluation
    switch model_selection

        case 'icval'

            % estimate model on in-sample part
            [Xtr,Ytr] = add_lags(YY(train_ind),factors(train_ind,1:nf),pf,h,0,py);
            B_lasso   = lasso(Xtr,Ytr,'Lambda', lambda);

            % evaluate the model on the validation set
            [Xval,Yval] = add_lags(YY(val_ind),factors(val_ind,1:nf),pf,h,0,py);
            yhat        = Xval*B_lasso;

            insampIC(i,:) = IC(Yval,yhat, length(Yval), size(Xval,2));

        case 'ictr'
```

Code for Lasso Regression

```
        % estimate model on in-sample part
        [Xtr,Ytr] = add_lags(YY(train_ind),factors(train_ind,1:nf),pf,h,0,py);
        B_lasso   = lasso(Xtr,Ytr,'Lambda', lambda);

        % in-sample evaluation on training set
        yhat = Xtr*B_lasso;
        insampIC(i,:) = IC(Ytr,yhat, length(Ytr), size(Xtr,2));

    case 'pooscv'

        % pseudo-oos evaluation (using last 25% of in-sample data)
        yhat = NaN(length(val_ind), 1);

        for t = 1:length(val_ind)
            % initialize data up to t in test set
            endInd = val_ind(t) - h;
            temp_ind = 1:(endInd-1);

            % estimate KRR on expanding window
            [Xtr,Ytr] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);
            B_lasso   = lasso(Xtr,Ytr,'Lambda', lambda);

            % forecast for t+h
            temp_ind = 1:(endInd);
            [Xte,Yte] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);
            Xte = Xte(end,:);
            ythat = Xte*B_lasso;

            yhat(t) = ythat;
        end

        ytrue = YY(val_ind);
        eval_cv = MSPE(ytrue,yhat);
        val_err(i) = eval_cv.MSPE;

    case 'kfcv'

        % default is 5-Fold CV
        try K; catch; K=5; end
        split = floor(linspace(1,val_ind(end),K+2));

        tmp_errs = NaN(K,1);

        for k = 1:K
```

Code for Lasso Regression

```
            cvidxtr = split(1):split(k+1);
            cvidxte = split(k+1):split(k+2);

            % train up to fold k
            [Xtr,Ytr] = add_lags(YY(cvidxtr),factors(cvidxtr,1:nf),pf,h,0,py);
            B_lasso   = lasso(Xtr,Ytr,'Lambda', lambda);

            % evaluate on fold k+1
            [Xte,Yte] = add_lags(YY(cvidxte),factors(cvidxte,1:nf),pf,h,0,py);
            yhat = Xte*B_lasso;

            eval_cv = MSPE(yhat,Yte);
            tmp_errs(K) = eval_cv.MSPE;

        end

        val_err(i) = mean(tmp_errs);

    end

    % estimate time remaining
    elapsed   = toc;
    remaining = ceil(elapsed * (size(combinations,1) - i) / 60);

    % progress updates
    if mod(i,100) == 0
        disp(join([i, "/", size(combinations,1)]))
        disp(join(["Approximately" remaining, "minutes remaining."]))
    end

end

% pick best model
switch model_selection
    case 'ictr'
        [~,best_i] = min(insampIC(:,icidx));
    case 'icval'
        [~,best_i] = min(insampIC(:,icidx));
    case 'pooscv'
        [~,best_i] = min(val_err);
    case 'kfcv'
        [~,best_i] = min(val_err);
end
```

Code for Lasso Regression

```
%% LASSO MODEL: IN-SAMPLE CRITERIA, PSEUDO-OOS FIT AND EVALUATION

% set hyperparameters based on best validation performance
lambda = lambda_vec(combinations(best_i,1));
py     = combinations(best_i,2);
pf     = combinations(best_i,3);
nf     = combinations(best_i,4);
p      = max(pf,py);

% preallocate oos forecast
yHat = NaN(length(test_ind),1);

if roll

    for t = 1:length(test_ind)

        % initialize data up to t in test set
        endInd = test_ind(t) - h;
        temp_ind = (endInd-wL-p-h):(endInd-1);

        % estimate lasso on rolling window
        [Xtr,Ytr] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);
        B_lasso   = lasso(Xtr,Ytr,'Lambda', lambda);

        % forecast for t+h
        temp_ind = (endInd-wL-p-h):(endInd);
        [Xte,Yte] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);
        Xte = Xte(end,:);
        ythat = Xte * B_lasso;

        yHat(t) = ythat;

    end

else

    for t = 1:length(test_ind)

        % initialize data up to t in test set
        endInd = test_ind(t) - h;
        temp_ind = 1:(endInd-1);

        % estimate lasso on expanding window
```

```
        [Xtr,Ytr] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);
        B_lasso   = lasso(Xtr,Ytr,'Lambda', lambda);

        % forecast for t+h
        temp_ind = 1:(endInd);
        [Xte,Yte] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);
        Xte = Xte(end,:);
        ythat = Xte * B_lasso;

        yHat(t) = ythat;

    end

end
```

Code for Ridge Regression

```
%% KRR MODEL - OPTIMIZE OVER HYPERPARAMETERS

insampIC = NaN(size(combinations,1),3);
val_err  = NaN(size(combinations,1),1);

for i = 1:size(combinations,1)

    tic;

    % set hyperparameters
    lambda = lambda_vec(combinations(i,1));
    sigma  = sigma_vec(combinations(i,2));
    py     = combinations(i,3);
    pf     = combinations(i,4);
    nf     = combinations(i,5);

    % model evaluation
    switch model_selection

        case 'icval'

            % estimate model on in-sample part
```

Code for Ridge Regression

```
            [Xtr,Ytr] =
add_lags(YY(train_ind),factors(train_ind,1:nf),pf,h,
0,py);
            alpha_tr =
train_KRR(Xtr,Ytr,lambda,sigma);

            % in-sample evaluation on validation set
            [Xval,Yval] =
add_lags(YY(val_ind),factors(val_ind,1:nf),pf,h,0,py
);
            yhat = forecast_KRR(alpha_tr, Xtr, Xval,
sigma);
            insampIC(i,:) = IC(Yval,yhat,
length(Yval), size(Xval,2));
```

Code for Elastic Net Regression

```
% elastic net hyperparameters
%     lambda = regularization parameter
%     alpha  = weight of lasso versus ridge optimization (0 = ridge, 1 = lasso)
lambda_max = 1e-4;
alpha_max  = 1;
lambda_vec = linspace(0,lambda_max,5);
alpha_vec  = linspace(0.01,alpha_max,5);

% index all possible combinations of hyperparameters
[a,b,c,d,e] = ndgrid(1:length(lambda_vec),1:length(alpha_vec),1:py_max,1:pf_max,nf_max:nf_max);
combinations = [reshape(a,[],1), reshape(b,[],1), reshape(c,[],1), ...
    reshape(d,[],1), reshape(e,[],1)];

disp(join(['Testing ', num2str(size(combinations,1), "%.0f"), ' combinations of hyperparameters.']))

%% ELASTIC NET MODEL - OPTIMIZE OVER HYPERPARAMETERS

insampIC = NaN(size(combinations,1),3);
val_err  = NaN(size(combinations,1),1);

for i = 1:size(combinations,1)

    tic;

    % set hyperparameters
    lambda = lambda_vec(combinations(i,1));
    alpha  = alpha_vec(combinations(i,2));
    py     = combinations(i,3);
    pf     = combinations(i,4);
    nf     = combinations(i,5);

    % model evaluation
    switch model_selection

        case 'icval'

            % estimate model on in-sample part
            [Xtr,Ytr] = add_lags(YY(train_ind),factors(train_ind,1:nf),pf,h,0,py);
            B_lasso   = lasso(Xtr,Ytr,'Lambda', lambda,'Alpha', alpha);

            % evaluate the model on the validation set
            [Xval,Yval] = add_lags(YY(val_ind),factors(val_ind,1:nf),pf,h,0,py);
            yhat        = Xval*B_lasso;
```

Code for Elastic Net Regression

```
        insampIC(i,:) = IC(Yval,yhat, length(Yval), size(Xval,2));

    case 'ictr'

        % estimate model on in-sample part
        [Xtr,Ytr] = add_lags(YY(train_ind),factors(train_ind,1:nf),pf,h,0,py);
        B_lasso    = lasso(Xtr,Ytr,'Lambda', lambda,'Alpha', alpha);

        % in-sample evaluation on training set
        yhat = Xtr*B_lasso;
        insampIC(i,:) = IC(Ytr,yhat, length(Ytr), size(Xtr,2));

    case 'pooscv'

        % pseudo-oos evaluation (using last 25% of in-sample data)
        yhat = NaN(length(val_ind), 1);

        for t = 1:length(val_ind)
            % initialize data up to t in test set
            endInd = val_ind(t) - h;
            temp_ind = 1:(endInd-1);

            % estimate ENET on expanding window
            [Xtr,Ytr] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);
            B_lasso    = lasso(Xtr,Ytr,'Lambda', lambda,'Alpha', alpha);

            % forecast for t+h
            temp_ind = 1:(endInd);
            [Xte,Yte] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);
            Xte = Xte(end,:);
            ythat = Xte*B_lasso;

            yhat(t) = ythat;
        end

        ytrue = YY(val_ind);
        eval_cv = MSPE(ytrue,yhat);
        val_err(i) = eval_cv.MSPE;

    case 'kfcv'

        % default is 5-Fold CV
        try K; catch; K=5; end
        split = floor(linspace(1,val_ind(end),K+2));
```

```
tmp_errs = NaN(K,1);

for k = 1:K

    cvidxtr = split(1):split(k+1);
    cvidxte = split(k+1):split(k+2);

    % train up to fold k
    [Xtr,Ytr] = add_lags(YY(cvidxtr),factors(cvidxtr,1:nf),pf,h,0,py);
    B_lasso    = lasso(Xtr,Ytr,'Lambda', lambda,'Alpha', alpha);
```

Code for Elastic Net Regression

```
        insampIC(i,:) = IC(Yval,yhat, length(Yval), size(Xval,2));

    case 'ictr'

        % estimate model on in-sample part
        [Xtr,Ytr] = add_lags(YY(train_ind),factors(train_ind,1:nf),pf,h,0,py);
        B_lasso   = lasso(Xtr,Ytr,'Lambda', lambda,'Alpha', alpha);

        % in-sample evaluation on training set
        yhat = Xtr*B_lasso;
        insampIC(i,:) = IC(Ytr,yhat, length(Ytr), size(Xtr,2));

    case 'pooscv'

        % pseudo-oos evaluation (using last 25% of in-sample data)
        yhat = NaN(length(val_ind), 1);

        for t = 1:length(val_ind)
            % initialize data up to t in test set
            endInd = val_ind(t) - h;
            temp_ind = 1:(endInd-1);

            % estimate ENET on expanding window
            [Xtr,Ytr] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);
            B_lasso   = lasso(Xtr,Ytr,'Lambda', lambda,'Alpha', alpha);

            % forecast for t+h
            temp_ind = 1:(endInd);
            [Xte,Yte] = add_lags(YY(temp_ind),factors(temp_ind,1:nf),pf,h,0,py);
            Xte = Xte(end,:);
            ythat = Xte*B_lasso;

            yhat(t) = ythat;
        end

        ytrue = YY(val_ind);
        eval_cv = MSPE(ytrue,yhat);
        val_err(i) = eval_cv.MSPE;

    case 'kfcv'

        % default is 5-Fold CV
        try K; catch; K=5; end
        split = floor(linspace(1,val_ind(end),K+2));
```

References

P. G. Coulombe, M. Leroux, D. Stevanovic, and S. Surprenant. How is machine learning useful for macroeconomic forecasting?, 2020.

F. Diebold and R. Mariano. Comparing predictive accuracy. *Journal of Business Economic Statistics*, 13(3):253–63, 1995. URL `https://EconPapers.repec.org/RePEc:bes:jnlbes:v:13:y:1995:i:3:p:253-63`.

G. Elliott, A. Gargano, and A. Timmermann. Complete subset regressions. *Journal of Econometrics*, 177(2):357–373, 2013. ISSN 0304-4076. doi: https://doi.org/10.1016/j.jeconom.2013.04.017. URL `https://www.sciencedirect.com/science/article/pii/S0304407613000948`. Dynamic Econometric Modeling and Forecasting.

P. R. Hansen. A test for superior predictive ability. *Journal of Business Economic Statistics*, 23(4):365–380, 2005. ISSN 07350015. URL `http://www.jstor.org/stable/27638834`.

P. R. Hansen, A. Lunde, and J. M. Nason. The model confidence set. *Econometrica*, 79(2):453–497, 2011. doi: https://doi.org/10.3982/ECTA5771. URL `https://onlinelibrary.wiley.com/doi/abs/10.3982/ECTA5771`.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

M. W. McCracken and S. Ng. Fred-md: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics*, 34(4):574–589, 2016. doi: 10.1080/07350015.2015.1086655. URL `https://doi.org/10.1080/07350015.2015.1086655`.

J. A. Mincer and V. Zarnowitz. *The Evaluation of Economic Forecasts*, pages 3–46. NBER, 1969. URL `http://www.nber.org/chapters/c1214`.

D. Pettenuzzo and A. Timmermann. Forecasting macroeconomic variables under model instability. *Journal of Business Economic Statistics*, 35:0–0, 06 2015. doi: 10.1080/07350015.2015.1051183.

J. Stock and M. Watson. Macroeconomic forecasting using diffusion indexes. *Journal of Business Economic Statistics*, 20(2):147–62, 2002. URL `https://EconPapers.repec.org/RePEc:bes:jnlbes:v:20:y:2002:i:2:p:147-62`.

H. White. A reality check for data snooping. *Econometrica*, 68(5):1097–1126, 2000. doi: https://doi.org/10.1111/1468-0262.00152. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/1468-0262.00152`.