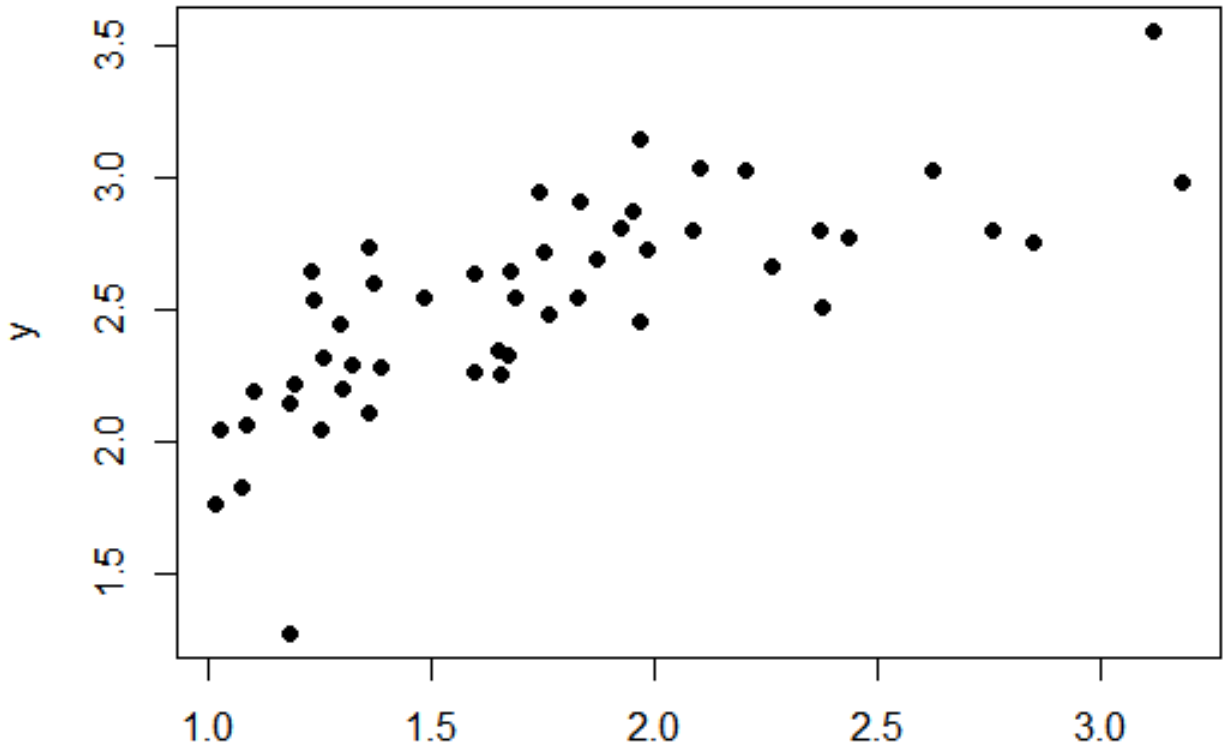
Session 2

Oleg Deev & Štefan Lyócsa

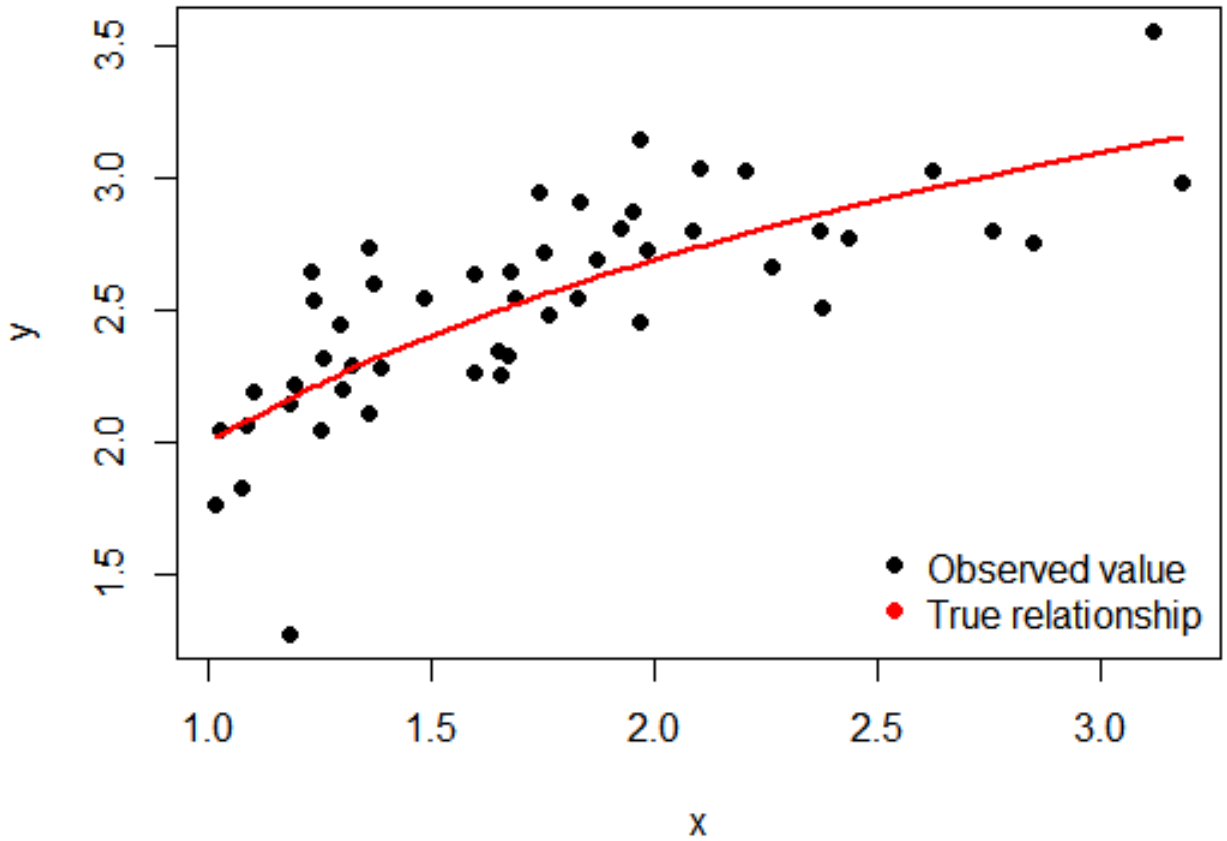
Masaryk University

The principle

Assume that we have the following observations available.



Assume that we **know** the true values (not contaminated by noise), are at the red line:



Sample splitting

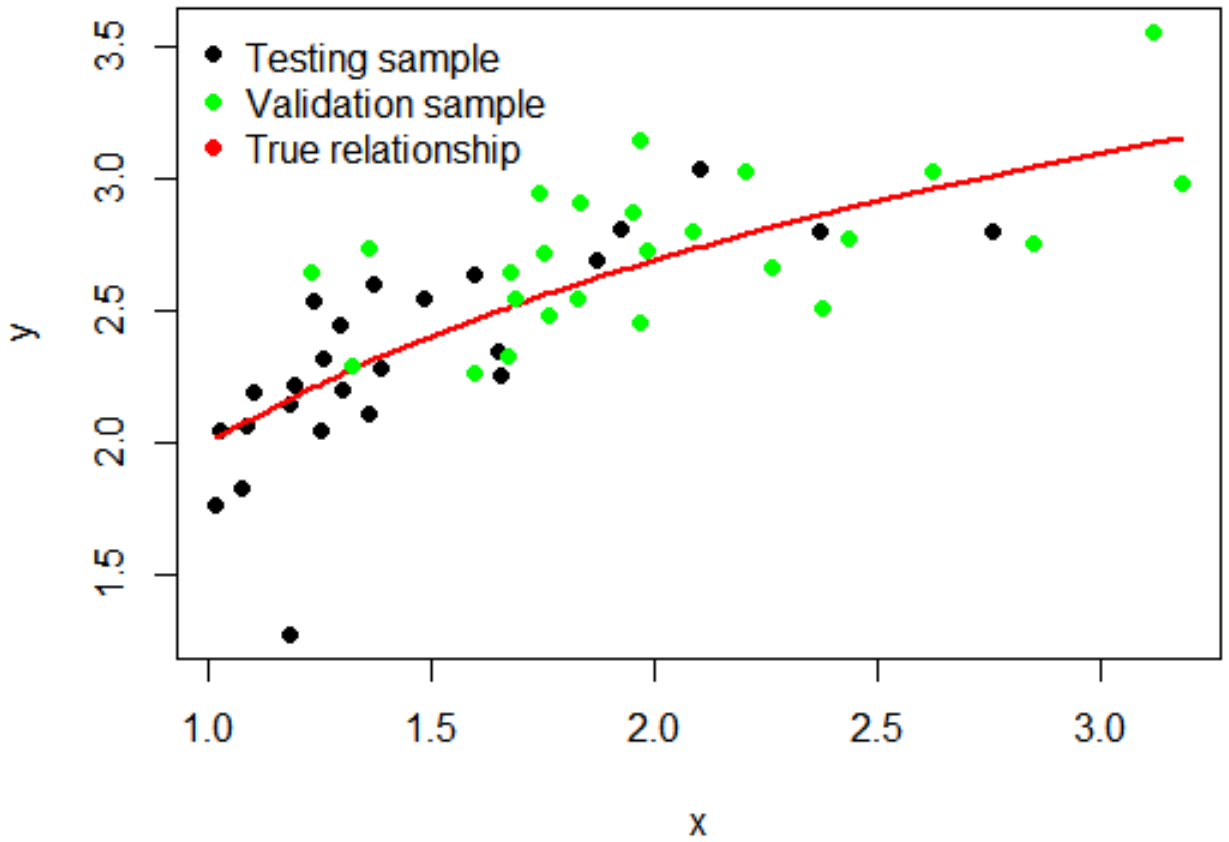
You want to estimate the relationship between x and y. Using the estimated model, you would like to make **predictions into future**.

A common strategy is to split the sample first into two parts:

- **Testing** sample - allow the model to learn.
- **Validation** sample - test the out-of-sample performance.

Different splitting strategies are possible. This is a basic one.

Both samples visualized:



Using data from the testing sample, let's fit a linear line model:

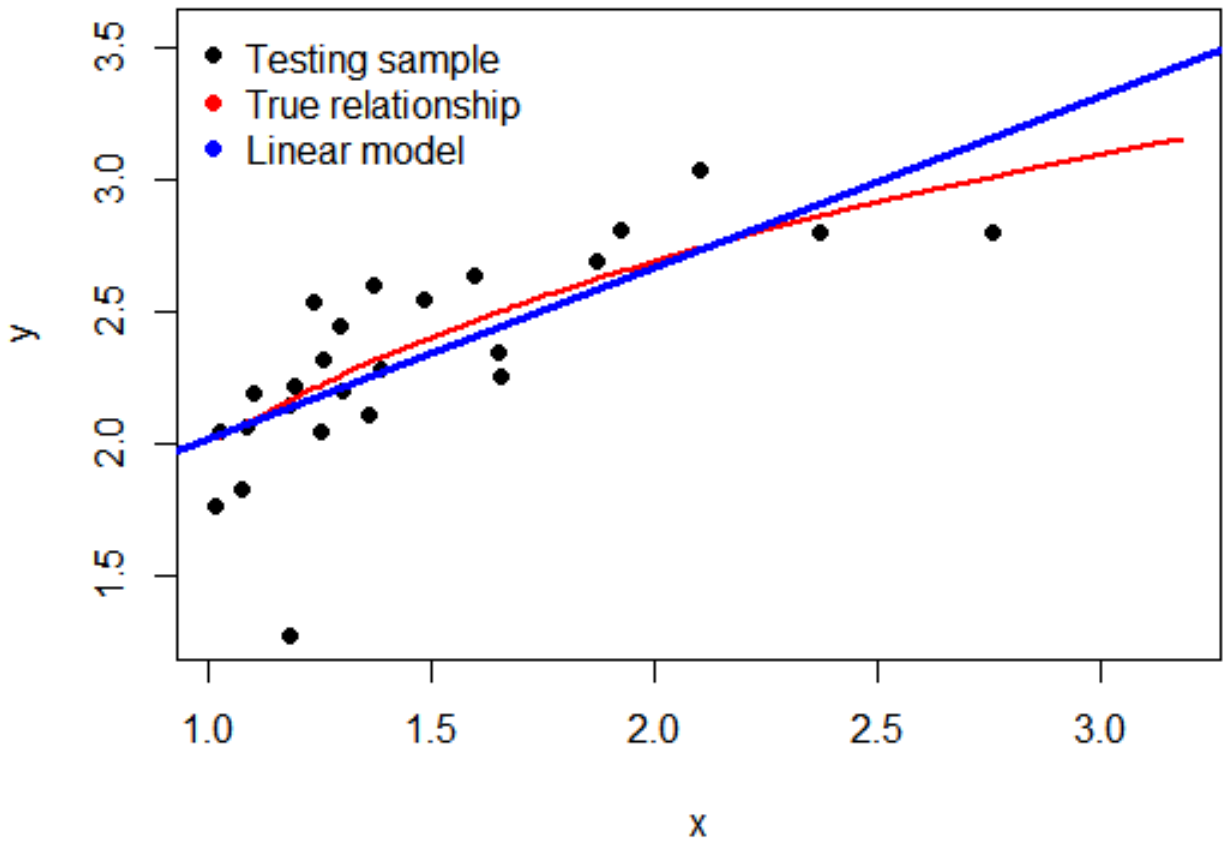
$$y_{i,test} = \beta_0 + \beta_1 x_{i,test} + u_{i,test}$$

The estimated coefficients are:

$$y_{i,test} = 1.37 + 0.65 x_{i,test} + \hat{u}_{i,test}$$

We know that the model is **ill specified**, no way a line is going to fit these data very well. But for prediction purposes, it might a good-enough **approximation** to the reality.

This is how the line looks like:



Using data from the testing sample, let's fit a polynomial model:

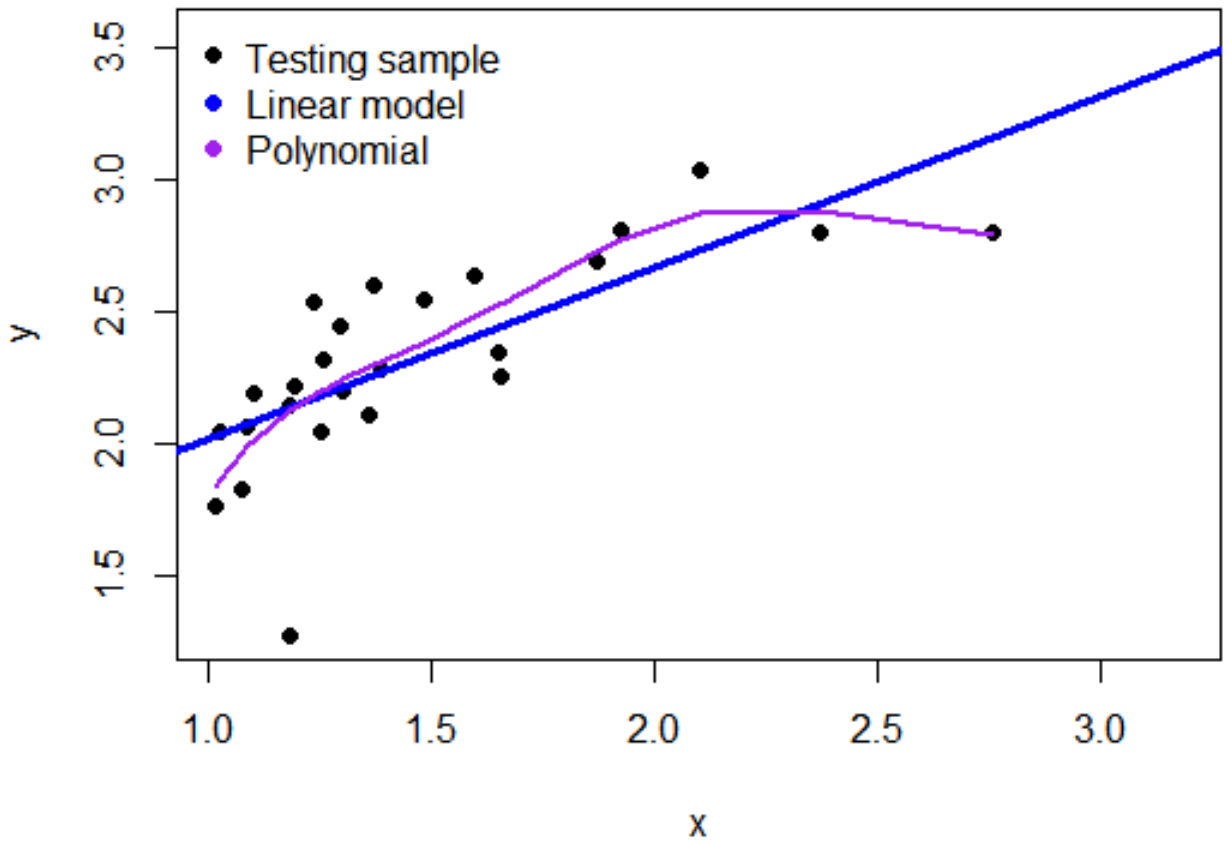
$$Y_{i,test} = \beta_0 + \sum_{p=1}^{5} \beta_p X_{i,test}^p + u_{i,test}$$
$$Y_{i,test} = \beta_0 + \beta_1 X_{i,test} + \beta_2 X_{i,test}^2 + \beta_3 X_{i,test}^3 + \beta_4 X_{i,test}^4 + \beta_5 X_{i,test}^5 + u_{i,test}$$

The estimated coefficients are:

$$Y_{i,test} = -24.33 + 75.59 X_{i,test} - 85.93 X_{i,test}^2 + 48.28 X_{i,test}^3 - 13.28 X_{i,test}^4 - 1.41 X_{i,test}^5 + \hat{u}_{i,test}$$

This polynomial is going to fit the data **much** better.

This is how the curve looks like:



Model comparison

We can compare which model fits the data better, e.g. R^2. Instead, we calculate a related measure, teh **mean square error** for the first model:

$$MSE_1 = N_{test}^{-1} \sum_i (Y_{i,test} - \hat{Y}_{i,1})^2 = 0.06676$$

The smaller the value, the better the fit. Now for the second model:

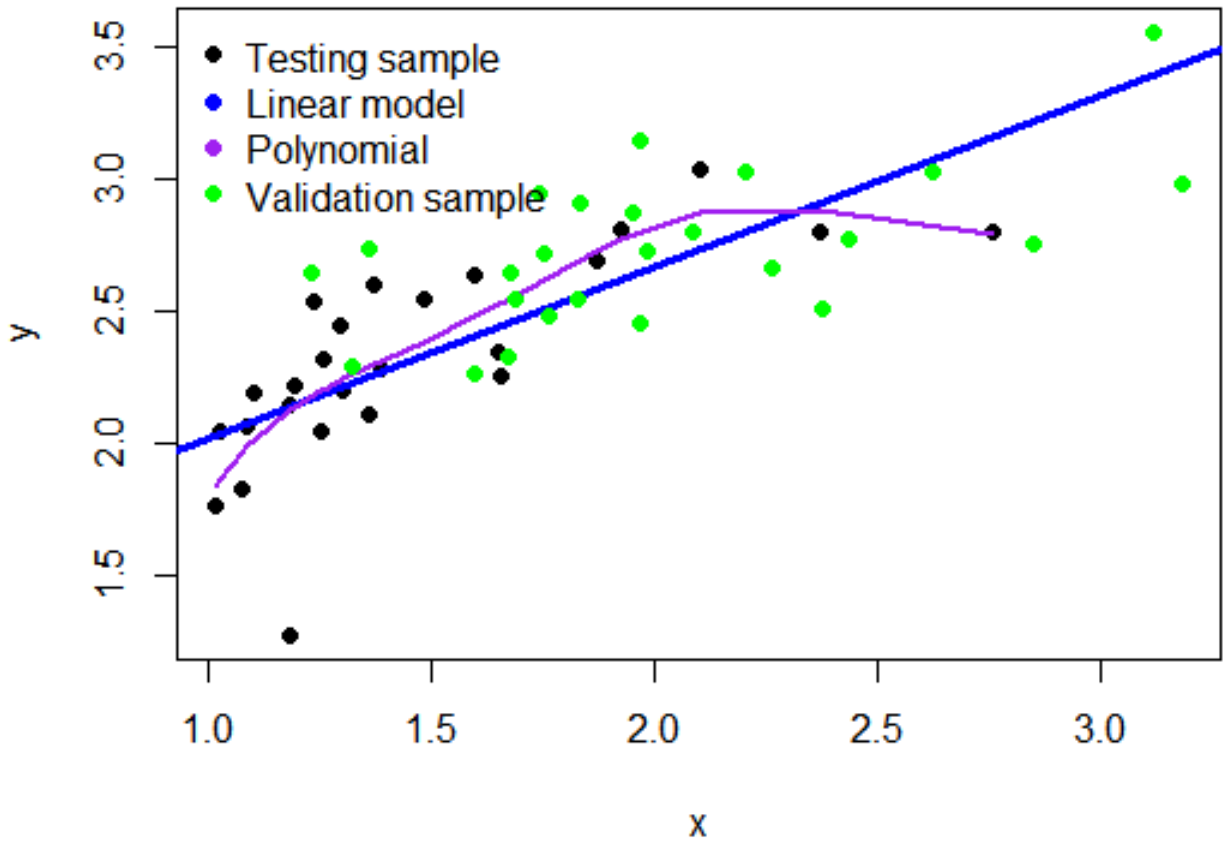
$$MSE_2 = N_{test}^{-1} \sum_i (Y_{i,test} - \hat{Y}_{i,2})^2 = 0.05401$$

The second model has better fit, by approx. 19%!

Model comparison

The first model fit the data poorly. It is linear. The data are curved. It is a **biased** model. The second model fits the data better. The higher the order of the polynomial, the better the fit and lower the bias (in-sample).

Is the model with better fit on the **testing** sample going to be better in the **validation** sample?


Testing sample
Linear model
Polynomial
Validation sample
y
x
1.5
2.0
2.5
3.0
3.5
1.0
1.5
2.0
2.5
3.0

Model comparison

Using coefficients from model 1 and model 2, and given new x from validation sample, we can predict y. Next we compare which model forecasts better using the MSE, but now we use predicted values. This is called the **M**ean **F**orecasted **S**quared **E**rror:

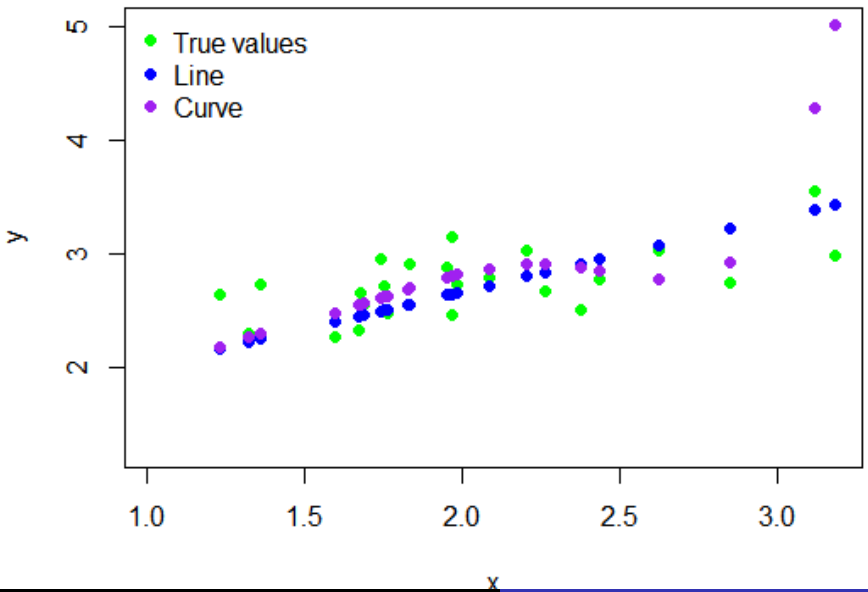
$$MSFE_1 = N_{validation}^{-1} \sum_i (Y_{i,validation} - \hat{Y}_{i,1})^2 = 0.0802$$

Now for the second model:

$$MSFE_2 = N_{validation}^{-1} \sum_i (Y_{i,validation} - \hat{Y}_{i,2})^2 = 0.2357$$

- The forecasts are less accurate on the validation sample.
- The linear model (although biased) performs **much** better.

Why? The polynomial model is **over-fitting** the data, e.g. fits too well on the expense of parameters. Parameters are not estimated with certainty - they suffer from variance. This leads to an increase in the **variance** of the predictions.



The goal of the **Machine learning** is to find an optimum between model bias and the variance of predictions. Many strategies, two standard ones:

- **Regularization** (Ridge regression, Lasso, Elastic net).
- **Boosting** (Regression trees, Random forest,...).

One strategy is to **allow** small bias (e.g. less parameters in the model) while **lowering** the variance. The accuracy of predictions might improve.

The common theme is to sacrifice in-sample fit in hope for a better out-of-sample prediction. Recall a multiple linear regression model:

$$Y_i = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + ... + \beta_p X_{i,p} + u_i$$

Using OLS, parameters of interest are estimated by minimizing the sum of squared residuals:

$$\underset{\hat{\beta}_0,...\hat{\beta}_p}{min} \rightarrow \sum_{i=1}^{n} \hat{u_i}^2 = \sum_{i=1}^{n} (Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_{i,1} - ... - \hat{\beta}_p X_{i,p})^2$$

In short:

$$\underset{\hat{\beta}_0,...\hat{\beta}_p}{min} \rightarrow \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

OLS approach:

$$\underset{\hat{\beta}_0,\ldots\hat{\beta}_p}{min} \rightarrow \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Ridge regression:

$$\underset{\hat{\beta}_0,\ldots\hat{\beta}_p}{min} \rightarrow \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

- $\lambda > 0$,
- X are standardized (0 mean, 1 variance),
- Y is centered around 0.

Ridge regression:

$$\underset{\hat{\beta}_0,\ldots\hat{\beta}_p}{min} \rightarrow \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

The higher the λ the lower the β coefficients, i.e. stronger the penalty.

Why might Ridge regression actually work? The **higher** the λ, the **less** sensitive is Y, the dependent variable, to the changes in the X_j explanatory variable(s). The Ridge regression model is more 'robust' to changes in explanatory variables.

How to find λ? Standard approach is to use 10-fold cross-validation technique. See the next *Case study*.

Case study 4

What factors drive the rate of return on a loan? We use the same model as in the Case study 3. Now, instead of OLS, we estimate it via penalized 'Ridge' estimator.

- Can Ridge out-perform (out-of-sample) the OLS model?

$$RR2_i = \beta_0 + \beta_1 new_i + \beta_2 ver3_i + ... + \beta_p nrodep_i + u_i$$

1. Split the sample into two. Leave last 100 observations for out-of-sample (validation).
2. Estimate OLS and calculate MSFE using the out-of-sample data.
3. Perform $k - fold$ cross-validation to estimate λ for the Ridge regression models.
4. Calculate MSFE using the out-of-sample data.

Sample split

- `NF = 100`
- `N = dim(DT)[1]`
- `tst = DT[1:(N-NF),]`
- `val = DT[((N-NF)+1):N,]`

k-fold Cross validation

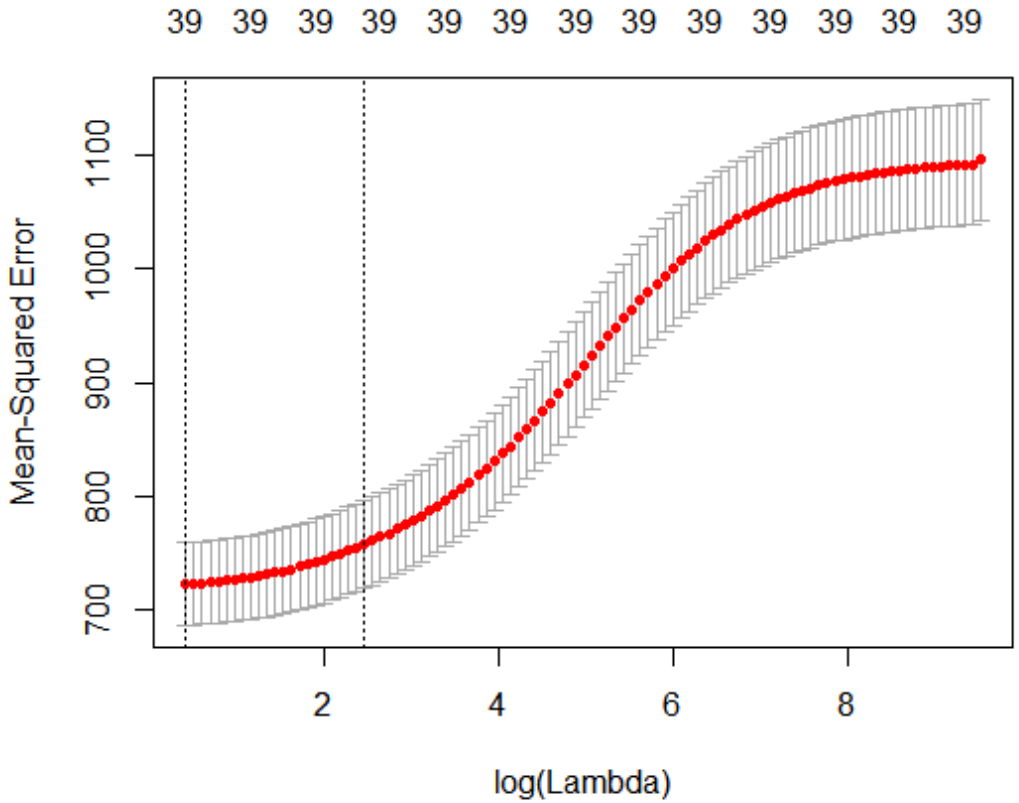
We need to prepare the data for the *glmnet* functions. See the codes....

- `CV = cv.glmnet(x=indep,y=dep,nfolds=30,alpha=0)`
- `plot(CV)`
- `CV$lambda.min`
- `CV$lambda.1se`
- `round(cbind(coefficients(m7),coef(CV,s='lambda.min'), coef(CV,s='lambda.1se')),4)`

OLS estimation and prediction

- `m7 = lm(RR2 ~ new+ver3+ver4+lfi+lee+luk+lrs+lsk+age+un`
`female+lamt+int+durm+educprim+educbasic+educvocat+`
`educsec+msmar+msco+mssi+msdi+nrodep+espem+esfue+`
`essem+esent+esret+dures+exper+linctot+noliab+ lliatot+nor`
`lamteprl+nopearlyrep,data=tst)`
- `yOLS = predict(m7,new=val)`
- `ytrue = val[,"RR2"]`
- `MSEOLS = mean((yOLS-ytrue)^2)`

How λ (actually $log(\lambda)$) and MSE are related. **Increasing** penalization is very expensive as it increases MSE considerably.



Prediction and evaluation

We need to prepare the data for the *glmnet* functions. See the codes....

- `yRIDGEmin=predict(CV,newx=pred,s=CV$lambda.min)`
- `MSER1 = mean((ytrue-yRIDGEmin)^2)`
- `yRIDGE1se=predict(CV,newx=pred,s=CV$lambda.1se)`
- `MSER2 = mean((ytrue-yRIDGE1se)^2)`
- `cbind(MSEOLS, MSER1, MSER2)`

OLS approach:

$$\underset{\hat{\beta}_0,\ldots\hat{\beta}_p}{min} \rightarrow \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Ridge regression:

$$\underset{\hat{\beta}_0,\ldots\hat{\beta}_p}{min} \rightarrow \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Least **A**bsolute **S**hrinkage and **S**election **O**perator (LASSO):

$$\underset{\hat{\beta}_0,\ldots\hat{\beta}_p}{min} \rightarrow \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

As before:

- $\lambda > 0$,
- X are standardized (0 mean, 1 variance),
- Y is centered around 0.

Least **A**bsolute **S**hrinkage and **S**election **O**perator (LASSO):

$$\underset{\hat{\beta}_0,\ldots\hat{\beta}_p}{min} \rightarrow \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

As with Ridge, the **higher** the λ, the **lower** the β coefficients, i.e. stronger the penalty.

With LASSO, coefficients might be reduced to 0. This is **useful** as LASSO reduces the model complexity, which in turn is known to be helpful for forecasting purposes.

Which to use? LASSO or Ridge?

- Ridge is useful when **many** variables are supposed to be useful (they might be highly correlated as well).
- LASSO is useful when only **few** variables are useful.

Why not to select only useful variables and run OLS?

Case study 5

What factors drive the rate of return on a loan? We use the same model as in the Case study 3 and 4. Now, instead of OLS and Ridge, we estimate it via penalized 'LASSO' estimator.

- Can LASSO out-perform (out-of-sample) the OLS and Ridge model?

$$RR2_i = \beta_0 + \beta_1 new_i + \beta_2 ver3_i + ... + \beta_p nrodep_i + u_i$$

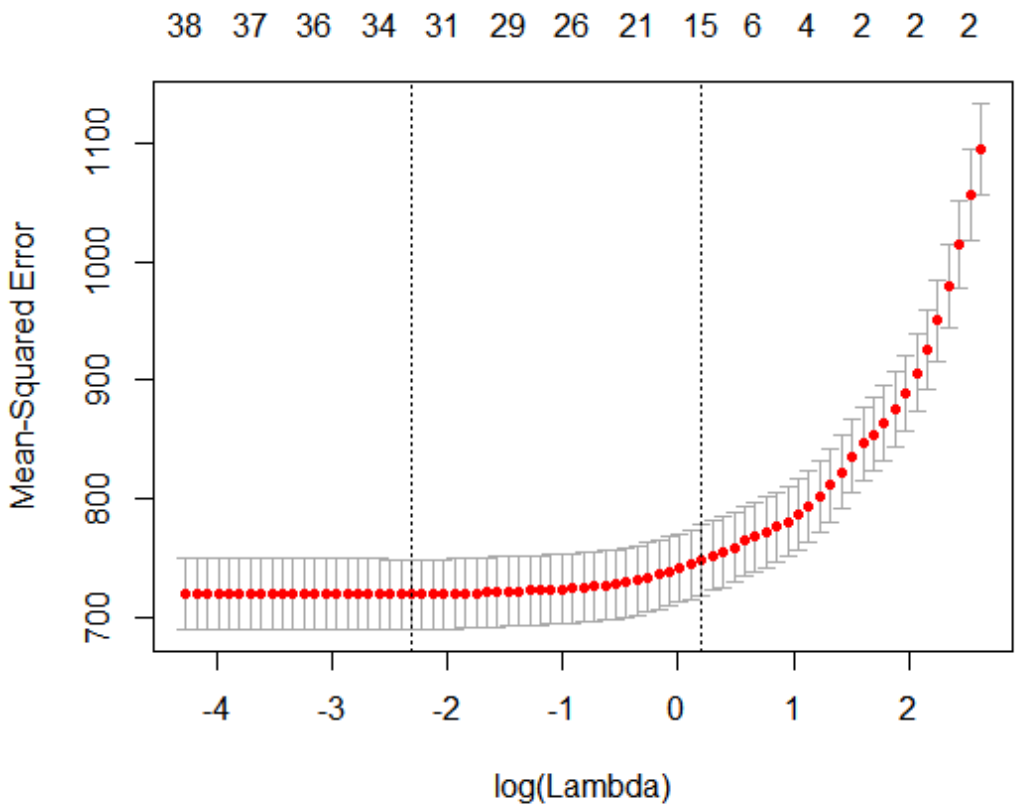
1. Split the sample into two. Leave last 100 observations for out-of-sample (validation).
2. Estimate OLS and calculate MSFE using the out-of-sample data.
3. Perform $k - fold$ cross-validation to estimate λ for the Ridge regression models.
4. Calculate MSFE using the out-of-sample data.
5. Perform $k - fold$ cross-validation to estimate λ for the LASSO regression models.
6. Calculate MSFE using the out-of-sample data.

k-fold Cross validation

We need to prepare the data for the *glmnet* functions. See the codes....

- `CV = cv.glmnet(x=indep,y=dep,nfolds=30,alpha=1)`
- `plot(CV)`
- `CV$lambda.min`
- `CV$lambda.1se`
- `round(cbind(coefficients(m7),coef(CV,s='lambda.min'), coef(CV,s='lambda.1se')),4)`

How λ (actually $log(\lambda)$) and MSE are related.



Prediction and evaluation

We need to prepare the data for the *glmnet* functions. See the code in Case study 3. Next, we can run the predictions:

- `yLASSOmin=predict(CV,newx=pred,s=CV$lambda.min)`
- `MSEL1 = mean((ytrue-yLASSOmin)^2)`
- `yLASSO1se=predict(CV,newx=pred,s=CV$lambda.1se)`
- `MSEL2 = mean((ytrue-yLASSO1se)^2)`
- `cbind(MSEOLS, MSER1, MSER2, MSEL1, MSEL2)`

OLS approach:

$$\underset{\hat{\beta}_0,\ldots\hat{\beta}_p}{min} \rightarrow \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Ridge regression:

$$\underset{\hat{\beta}_0,\ldots\hat{\beta}_p}{min} \rightarrow \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Least **A**bsolute **S**hrinkage and **S**election **O**perator (LASSO):

$$\underset{\hat{\beta}_0,\ldots\hat{\beta}_p}{min} \rightarrow \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Elastic net:

$$\underset{\hat{\beta}_0,\ldots\hat{\beta}_p}{min} \rightarrow \frac{1}{2n} \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \lambda(\frac{1-\alpha}{2} \sum_{j=1}^{p} \beta_j^2 + \alpha \sum_{j=1}^{p} |\beta_j|)$$

Elastic net:

$$\underset{\hat{\beta}_0,\ldots\hat{\beta}_p}{min} \rightarrow \frac{1}{2n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \lambda(\frac{1-\alpha}{2}\sum_{j=1}^{p}\beta_j^2 + \alpha\sum_{j=1}^{p}|\beta_j|)$$

It gives a combined penalization of Ridge and LASSO. The new parameter α shows which of the two penalization forms gets higher weight.

- If $\alpha = 1$ it is a LASSO model.
- If $\alpha = 0$ it is a Ridge model.
- With $0 \leq \alpha \leq 1$, we have the Elastic net.

As before, the *optimal* α and λ is determined via cross-validation.

Case study 6

What factors drive the rate of return on a loan? We use the same model as in the Case study 3, 4 and 5. Now, instead of OLS, Ridge, LASSO we estimate it via 'Elastic net' estimator.

- Can Elastic net out-perform (out-of-sample) the OLS, Ridge, LASSO model?

$$RR2_i = \beta_0 + \beta_1 new_i + \beta_2 ver3_i + ... + \beta_p nrodep_i + u_i$$

For $\alpha = 0.25$

- `CV = cv.glmnet(x=indep,y=dep,nfolds=30,alpha`
- `yNET025min=predict(CV,newx=pred,s=CV$lambda.min)`
- `MSEEN1.1 = mean((ytrue-yNET025min)^2)`
- `yNET0251se=predict(CV,newx=pred,s=CV$lambda.1se)`
- `MSEEN1.2 = mean((ytrue-yNET0251se)^2)`

For $\alpha = 0.50$

- `CV = cv.glmnet(x=indep,y=dep,nfolds=30,alpha`
- `yNET050min=predict(CV,newx=pred,s=CV$lambda.min)`
- `MSEEN2.1 =mean((ytrue-yNET050min)^2)`
- `yNET0501se=predict(CV,newx=pred,s=CV$lambda.1se)`
- `MSEEN2.2 =mean((ytrue-yNET0501se)^2)`

For $\alpha = 0.75$

- `CV = cv.glmnet(x=indep,y=dep,nfolds=30,alpha`
- `yNET075min=predict(CV,newx=pred,s=CV$lambda.min)`
- `MSEEN3.1 = mean((ytrue-yNET075min)^2)`
- `yNET0751se=predict(CV,newx=pred,s=CV$lambda.1se)`
- `MSEEN3.2 = mean((ytrue-yNET0751se)^2)`

We can compare results:

	MSEs
EN75_1	868.86
LASSO_1	871.52
EN25_1	874.82
EN50_1	876.54
Ridge_1	929.69
Ridge_M	968.78
EN75_M	976.86
EN25_M	976.97
EN50_M	977.07
LASSO_M	977.57
OLS	995.62

Session 2

Oleg Deev & Štefan Lyócsa

Masaryk University