Machine Learning - Exercise 1

Robin Perälä 175910, Alexander Norrgran 164620, Robert Tommola 194000

5.4.2022

Note: We are using the older version of exercise 1

Task 1

Indicate whether a more flexible or a less flexible model is likely to be better or worse for prediction performance in the scenarios below. Motivate your answers.

a) *The true relationship between y and x is linear.*

Less flexible. We know that the true relationship is linear, so a linear model (which is less flexible) will yield the most accurate predictions.

b) *The true relationship between y and x is highly non-linear.*

More flexible. We know that the true relationship is non-linear, so a non-linear model (which is more flexible) will yield the most accurate predictions.

c) *The variance of the error term is very high.*

Less flexible. If the variance is very high, this might indicate that the model is too flexible and overfitting would be an issue. A flexible model would capture noise in the data due to the large variance in the error term. Therefore a less flexible model is preferred (We are assuming that the error term is of the reducible form. If the error term is totally irreducible, neither a more flexible or less flexible model will generate an improvement)

Task 2

2. i. ***Explain how k-fold cross-validation works. Answer with no more than 10 sentences.***

In k-fold cross validation we divide the sample into k random folds of about the same size. Each fold is in turn used as the test set while the rest k-1 folds form the training set. A statistical model is fitted on the training set, which is then tested on the test set to obtain a test MSE (mean squared error). Because we have k folds, we obtain k test MSEs. The average of the k test MSEs is called the cross-validation estimate. The procedure can be repeated for different statistical models to choose between competing models. The model with the lowest test MSE is preferred. (Example below of a 5-fold cross-validation)

Figure 1: 5-fold cross-validation (ISLR 2021, 203)

2. ii. ***In general, what can we say about the bias-variance tradeoff when using LOOCV (Leave One Out Cross-Validation) versus 10-fold cross validation? Answer with no more than 10 sentences.***

LOOCV leads to a lower bias, while 10-fold CV leads to lower variance. Both methods divide the data into folds, but a different number of folds. 10-fold CV into ten folds, LOOCV into n folds (where n is the sample size). This leads to LOOCV using a larger training set while 10-fold CV uses a larger test set. If we want to minimize bias we should have a sufficiently large training set. If we want to minimize testing variance we should have a large test set. This is the bias-variance tradeoff. Because of this bias-variance tradeoff, 10-fold cross validation is commonly used (Machine learning for finance lecture 2: Resampling, slide 32).

Task 3 (Forward Stepwise Selection)

Load data, split data and preparation

```
rm(list=ls())#clear environment

library(haven)  #For importing Stata data
library(leaps)  #For best subset selection
library(glmnet) #For Lasso
library(knitr)  #For formatting tables

#Load Stata data with read_dta function from haven
USCompanies_data_winsorized = read_dta("USCompanies_data_winsorized.dta")
USData = subset(USCompanies_data_winsorized, select = -c(conm))
USData = na.omit(USData)

#Split data into 10 folds
k=10; set.seed(707)
folds=sample(1:k, nrow(USData), replace=TRUE)
upToOrder = 21
cv.errors = matrix(NA, k, upToOrder, dimnames=list(NULL, paste(1:upToOrder)))

#Create predict function for later use.
#Regsubsets doesn't include its own prediction function
predict.regsubsets = function(object, newdata, id ,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form, newdata)
  coefi=coef(object, id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}
```

3. i. ***Use Forward Stepwise Selection to find the variables included in the M1, . . . , Mk, where k is the number of variables. Report the variables for at least 20 of the first models. Note that the total number of independent variables in the dataset is 44.***

```
#Forward Stepwise Selection
#Using regsubsets from library leaps
regfit_best=regsubsets(roa_w ~., data=USData, nvmax=upToOrder, method="forward")
regfit_summary = summary(regfit_best)

report = 1:upToOrder
#Display variables in a table
#Format table using knitr::kable
kable(t(regfit_summary$outmat[report,]),
             col.names=report,
             caption="Forward Stepwise Selection: Variables for 21 of the first models")
```

Table 1: Forward Stepwise Selection: Variables for 21 of the first models

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
pe_w													*	*	*	*	*	*	*	*	*
aco_w																					
ap_w																	*	*	*	*	*
aqc_w																					
at_w										*	*	*	*	*	*	*	*	*	*	*	*
bkvlps_w				*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
caps_w																					
capx_w																					*
ceq_w																					
ch_w					*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
chech_w																					
ci_w											*	*	*	*	*	*	*	*	*	*	*
cidergl_w																					
cogs_w																					
dlc_w																					
dp_w														*	*	*	*	*	*	*	*
dvt_w																					
ebitda_w									*	*	*	*	*	*	*	*	*	*	*	*	*
epspi_w		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
fiao_w																					
fopo_w																					
gdwl_w																					
ib_w																					
icapt_w																					
intano_w							*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
invt_w																					
ivncf_w																				*	*
ivst_w																			*	*	*
lco_w																					
lt_w																					
np_w																					
oancf_w																		*	*	*	*
ppent_w															*	*	*	*	*	*	*
re_w						*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
rect_w																					
sale_w																*	*	*	*	*	*
siv_w																					
teq_w								*	*	*	*	*	*	*	*	*	*	*	*	*	*
tstk_w																					
txt_w																					
roe_w			*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
profit_margin_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
asset_turnover_w												*	*	*	*	*	*	*	*	*	*
operating_margin_w																					

3. ii. ***Use 10-fold cross-validation to decide which of the M1, . . . , Mk models is the best model.***

```
#Perform cross-validation. Calculate test MSE for all 20x10 models
for(j in 1:k){
  best.fit=regsubsets(roa_w~.,data=USData[folds!=j,], nvmax=upToOrder, method="forward")
  for(i in 1:upToOrder){
    pred=predict(best.fit, USData[folds==j,], id=i) #Using earlier created predict function
    cv.errors[j,i]= mean((USData$roa_w[folds==j]-pred)^2)
  }
}

#Calculate mean MSE for each order
mean.cv.errors=apply(cv.errors, 2, mean)
bestOrder = which.min(mean.cv.errors)[[1]]
smallestMSE = min(mean.cv.errors)

#Best model order and its MSE
cat("Order:", bestOrder, "\t", "MSE:", signif(smallestMSE, 3))
```

```
## Order: 21     MSE: 0.0311
```

3. iii. ***Report the coefficients for the variables included in the final best model that you have chosen using Forward Stepwise Selection.***

```
#Coefficients of the best model. Fitted on full data
best.fit=regsubsets(roa_w~.,data=USData, nvmax=upToOrder, method="forward")
signif(coef(best.fit, bestOrder), 3)
```

```
##      (Intercept)            pe_w            ap_w            at_w
##        -4.77e-02        6.45e-05       -1.44e-05       -5.72e-06
##         bkvlps_w          capx_w            ch_w            ci_w
##         2.11e-03       -6.21e-05        5.23e-05       -9.49e-05
##             dp_w        ebitda_w         epspi_w        intano_w
##        -1.28e-04        1.08e-04        1.49e-02        2.38e-05
##          ivncf_w          ivst_w         oancf_w         ppent_w
##        -2.10e-05        3.05e-05       -3.55e-05        1.48e-05
##             re_w          sale_w           teq_w           roe_w
##        -5.62e-06        6.52e-06       -5.42e-06        7.88e-02
## profit_margin_w asset_turnover_w
##         5.97e-02       -1.73e-02
```

Task 4 (Backward Stepwise Selection)

4. i. Use Backward Stepwise Selection to identify the variables included in the M1, . . . , Mk.

```
#Backward Stepwise Selection
upToOrder = 44
cv.errors = matrix(NA, k, upToOrder, dimnames=list(NULL, paste(1:upToOrder)))

#Using regsubsets from library leaps
regfit_best=regsubsets(roa_w ~., data=USData, nvmax=upToOrder, method="backward")
regfit_summary = summary(regfit_best)

report2 = (1:40)[-report]
#Display variables in a table
#Format table using knitr::kable
kable(t(regfit_summary$outmat[report,]),
             col.names=report,
             caption="Backward Stepwise Selection:
             Variables for 21 of the first models")
```

Table 2: Backward Stepwise Selection: Variables for 21 of the first models

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
pe_w													*	*	*	*	*	*	*	*	*
aco_w																					
ap_w																		*	*	*	*
aqc_w																					
at_w						*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
bkvlps_w				*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
caps_w																					
capx_w																					*
ceq_w																			*	*	*
ch_w					*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
chech_w																					
ci_w								*	*	*	*	*	*	*	*	*	*	*	*	*	*
cidergl_w																					
cogs_w																					
dlc_w																					
dp_w										*	*	*	*	*	*	*	*	*	*	*	*
dvt_w																					
ebitda_w							*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
epspi_w		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
fiao_w																					
fopo_w																					
gdwl_w																					
ib_w																					
icapt_w																					
intano_w									*	*	*	*	*	*	*	*	*	*	*	*	*
invt_w																					

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
ivncf_w																				*	*
ivst_w																	*	*	*	*	*
lco_w																					
lt_w																					
np_w																					
oancf_w																*	*	*	*	*	*
ppent_w											*	*	*	*	*	*	*	*	*	*	*
re_w												*	*	*	*	*	*	*	*	*	*
rect_w																					
sale_w															*	*	*	*	*	*	*
siv_w																					
teq_w																					
tstk_w																					
txt_w																					
roe_w			*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
profit_margin_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
asset_turnover_w														*	*	*	*	*	*	*	*
operating_margin_w																					

```r
kable(t(regfit_summary$outmat[report2,]),
      col.names=report2,
      caption="Backward Stepwise Selection:
      Variables for models 22 to 40")
```

Table 3: Backward Stepwise Selection: Variables for models 22 to 40

	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
pe_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
aco_w				*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
ap_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
aqc_w													*	*	*	*	*	*	*
at_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
bkvlps_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
caps_w									*	*	*	*	*	*	*	*	*	*	*
capx_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
ceq_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
ch_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
chech_w											*	*	*	*	*	*	*	*	*
ci_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
cidergl_w										*	*	*	*	*	*	*	*	*	*
cogs_w		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
dlc_w																			*
dp_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
dvt_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
ebitda_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
epspi_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
fiao_w																*	*	*	*
fopo_w												*	*	*	*	*	*	*	*
gdwl_w																			

	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
ib_w					*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
icapt_w																			
intano_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
invt_w						*	*	*	*	*	*	*	*	*	*	*	*	*	*
ivncf_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
ivst_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
lco_w																			
lt_w																			
np_w																	*	*	*
oancf_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
ppent_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
re_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
rect_w							*	*	*	*	*	*	*	*	*	*	*	*	*
sale_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
siv_w								*	*	*	*	*	*	*	*	*	*	*	*
teq_w																		*	*
tstk_w			*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
txt_w															*	*	*	*	*
roe_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
profit_margin_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
asset_turnover_w	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
operating_margin_w														*	*	*	*	*	*

4. ii. ***Use 10-fold cross-validation to decide which model of the M1, . . . , Mk models is the best model.***

```
#Perform cross-validation. Calculate test MSE for all 20x10 models
for(j in 1:k){
  best.fit=regsubsets(roa_w~., data=USData[folds!=j,], nvmax=upToOrder, method="backward")
  for(i in 1:upToOrder){
    pred=predict(best.fit, USData[folds==j,], id=i)
    cv.errors[j,i]= mean((USData$roa_w[folds==j]-pred)^2)
  }
}

#Calculate mean MSE for all orders
mean.cv.errors=apply(cv.errors, 2, mean)
bestOrder = which.min(mean.cv.errors)[[1]]
smallestMSE = min(mean.cv.errors)

#Best model order and its MSE
cat("Order:", bestOrder, "\tMSE:", signif(smallestMSE, 3))
```

```
## Order: 19    MSE: 0.0311
```

4. iii. ***Report the coefficients for the variables included in the final best model that you have chosen using Backward Stepwise Selection.***

```
#Coefficients of the best model. Fitted on full data
best.fit=regsubsets(roa_w~.,data=USData, nvmax=upToOrder, method="backward")
signif(coef(best.fit, bestOrder), 3)
```

```
##      (Intercept)            pe_w            ap_w            at_w
##        -4.75e-02        6.43e-05       -1.50e-05       -5.39e-06
##         bkvlps_w           ceq_w            ch_w            ci_w
##         2.13e-03       -6.13e-06        5.16e-05       -9.24e-05
##             dp_w        ebitda_w         epspi_w        intano_w
##        -1.37e-04        1.03e-04        1.49e-02        2.55e-05
##           ivst_w         oancf_w         ppent_w            re_w
##         3.65e-05       -3.15e-05        1.20e-05       -5.47e-06
##           sale_w           roe_w profit_margin_w asset_turnover_w
##         6.54e-06        7.87e-02        5.97e-02       -1.76e-02
```

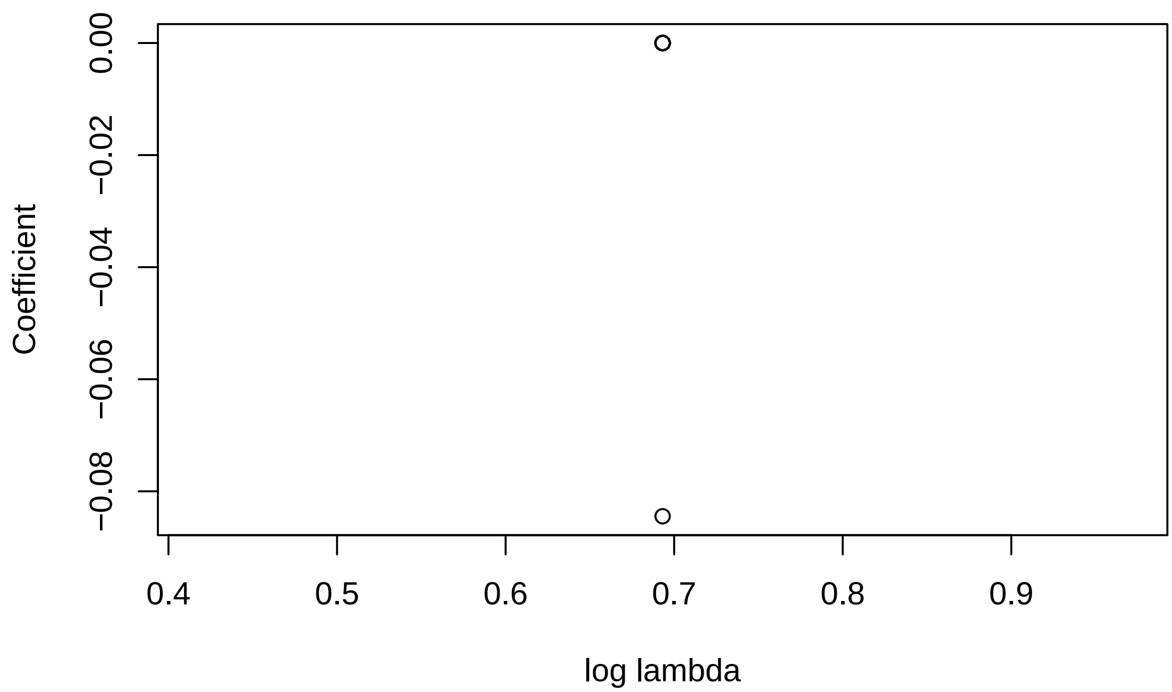
Task 5 (Lasso)

```
#Assign independent variables to x and dependent variable to y.
#This makes using the glmnet package easier
x = model.matrix(roa_w~., USData)[,-1]
y = USData$roa_w
```

5. i. ***Use Lasso regression with tuning parameter = 2. Show a plot of the coefficients (y-axis) and log lambdas (x -axis), or of the coefficients (y-axis) and L1 norm (x-axis), or similar.***

```
#Use function glmnet (from library glmnet) to perform Lasso regression
lasso.mod = glmnet(x, y, alpha=1, lambda=2, standardize=TRUE)

#Plot the coefficients against log lambda
plot(y=coef(lasso.mod),
     x=rep(log(lasso.mod$lambda), nrow(coef(lasso.mod))),
     ylab="Coefficient", xlab="log lambda")
```



5. ii. ***Use the validation method (divide sample into 50% test set and 50% training set) to calculate the test MSEs (mean squared errors) for the models with tuning parameters = 2 and = 10, respectively. Which model performs better? Motivate.***

```
#Validation 50-50
train=sample(c(TRUE, FALSE), nrow(USData), rep=TRUE)
test = !train

#Fit model on training data
lasso.mod = glmnet(x[train,], y[train], alpha=1, lambda=c(2,10), standardize=TRUE)

#Test model on test data
lasso.pred2=predict(lasso.mod, s=2, newx=x[test,])
lambda2MSE = mean((lasso.pred2 - y[test])^2)
cat("MSE for lambda2: \t", signif(lambda2MSE, 3), "\n")
```

```
## MSE for lambda2:      0.0698
```

```
lasso.pred10=predict(lasso.mod, s=10, newx=x[test,])
lambda10MSE = mean((lasso.pred10 - y[test])^2)
cat("MSE for lambda10: \t", signif(lambda10MSE, 3), "\n\n")
```

```
## MSE for lambda10:     0.0698
```

```
out=glmnet(x, y, alpha=1)
lasso.coef=predict(out, type="coefficients", s=10)[1:20,]
signif(lasso.coef[lasso.coef!=0], 3)
```

```
## (Intercept)
##     -0.0845
```

The models perform equally well because their MSEs are exactly the same. The lambdas (2 & 10) are relatively big, which has turned all coefficients to zero. They have in principal become the same model (intercept-model). This is why the MSEs are exactly the same.

5. iii. ***Use 10-fold cross-validation to find the tuning parameter that yields the model with the lowest test MSE. Report this "best" tuning parameter. Show a plot of how the MSE depends on the value of the tuning parameter.***

```
#10-fold cross-validation
set.seed(707)
cv.out=cv.glmnet(x, y, alpha=1, standardize=TRUE, nfolds=10)

#Best tuning parameter
bestlam = cv.out$lambda.min
cat("Best tuning parameter\n",
    "Lambda:", format(bestlam, scientific=T, digits=3),
    "\n log lambda:", signif(log(bestlam), 3))
```

```
## Best tuning parameter
##  Lambda: 3.08e-04
##  log lambda: -8.08
```

```
plot(cv.out)
```