Assignment 3

Group 2

2023-10-23

Task 1

Task 2

Task 3

Task 4

Task 5

Task 6

a)

```
## Warning: Paket 'caret' wurde unter R Version 4.2.3 erstellt

## Warning: Paket 'ggplot2' wurde unter R Version 4.2.3 erstellt

## Warning: Paket 'knitr' wurde unter R Version 4.2.3 erstellt
```

```
# Generate a simulated data set
set.seed(1)
x <- rnorm(100)
y <- x - 2*x^2 + rnorm(100)
```

b)

```
# Data prep
set.seed(100)
df_1 = data.frame(y=y,x=x)
df_2 = data.frame(y=y,x=x,x2=x^2)
df_3 = data.frame(y=y,x=x,x2=x^2,x3=x^3)
df_4 = data.frame(y=y,x=x,x2=x^2,x3=x^3,x4=x^4)
model=list("i", "ii", "iii", "iv")
LOOCV_MSE=list()
kCV_MSE=list()
mse = function(sm)
    mean(sm$results$RMSE^2)
```

```r
# LOOCV cross-validation method
ctrl_loocv <- trainControl(method = "LOOCV")
ctrl_kcv <- trainControl(method = "cv", number = 10)

# First model

## LOOCV
model_LOOCV_1 <- train(y ~ ., data=df_1, method="lm", trControl=ctrl_loocv)
print(model_LOOCV_1)
```

```
## Linear Regression
##
## 100 samples
##   1 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 99, 99, 99, 99, 99, 99, ...
## Resampling results:
##
##   RMSE     Rsquared    MAE
##   2.69966  0.00130345  1.921879
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
## 10 fold CV
model_kCV_1 <- train(y ~ ., data=df_1, method="lm", trControl=ctrl_kcv)
print(model_kCV_1)
```

```
## Linear Regression
##
## 100 samples
##   1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 91, 89, 91, 88, 89, 92, ...
## Resampling results:
##
##   RMSE      Rsquared  MAE
##   2.638627  0.40973   1.925501
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
## Save the results
LOOCV_MSE = append(LOOCV_MSE, mse(model_LOOCV_1))
kCV_MSE = append(kCV_MSE, mse(model_kCV_1))

# Second model

## LOOCV
```

```r
model_LOOCV_2 <- train(y ~ ., data=df_2, method="lm", trControl=ctrl_loocv)
print(model_LOOCV_2)
```

```
## Linear Regression
##
## 100 samples
##   2 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 99, 99, 99, 99, 99, 99, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9682064  0.8663108  0.7829438
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
## 10 fold CV
model_kCV_2 <- train(y ~ ., data=df_2, method="lm", trControl=ctrl_kcv)
print(model_kCV_2)
```

```
## Linear Regression
##
## 100 samples
##   2 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 90, 92, 91, 89, 91, 91, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9459685  0.8724754  0.7751198
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
## Save the results
LOOCV_MSE = append(LOOCV_MSE, mse(model_LOOCV_2))
kCV_MSE = append(kCV_MSE, mse(model_kCV_2))


# Third model

## LOOCV
model_LOOCV_3 <- train(y ~ ., data=df_3, method="lm", trControl=ctrl_loocv)
print(model_LOOCV_3)
```

```
## Linear Regression
##
## 100 samples
##   3 predictor
```

```
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 99, 99, 99, 99, 99, 99, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9780705  0.8637902  0.7924894
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
## 10 fold CV
model_kCV_3 <- train(y ~ ., data=df_3, method="lm", trControl=ctrl_kcv)
print(model_kCV_3)
```

```
## Linear Regression
##
## 100 samples
##   3 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 92, 91, 90, 89, 90, 89, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9520476  0.8334319  0.7787047
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
## Save the results
LOOCV_MSE = append(LOOCV_MSE, mse(model_LOOCV_3))
kCV_MSE = append(kCV_MSE, mse(model_kCV_3))


# Fourth model

## LOOCV
model_LOOCV_4 <- train(y ~ ., data=df_4, method="lm", trControl=ctrl_loocv)
print(model_LOOCV_4)
```

```
## Linear Regression
##
## 100 samples
##   4 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 99, 99, 99, 99, 99, 99, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9766805  0.8639008  0.8019279
```

```
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
## 10 fold CV
model_kCV_4 <- train(y ~ ., data=df_4, method="lm", trControl=ctrl_kcv)
print(model_kCV_4)
```

```
## Linear Regression
##
## 100 samples
##   4 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 89, 91, 91, 91, 90, 88, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9638881  0.7964419  0.8081947
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
## Save the results
LOOCV_MSE = append(LOOCV_MSE, mse(model_LOOCV_4))
kCV_MSE = append(kCV_MSE, mse(model_kCV_4))

# Results
result = data.frame(Model = unlist(model), LOOCV_100 = unlist(LOOCV_MSE),
                    kCV_100 = unlist(kCV_MSE))
kable(result, caption = "Summary")
```

Table 1: Summary

Model	LOOCV_100	kCV_100
i	7.2881616	6.9623524
ii	0.9374236	0.8948565
iii	0.9566218	0.9063947
iv	0.9539049	0.9290803

c)

```
# Set seed
set.seed(99)
LOOCV_MSE_2=list()
kCV_MSE_2=list()
# First model

## LOOCV
model_LOOCV_1_2 <- train(y ~ ., data=df_1, method="lm", trControl=ctrl_loocv)
print(model_LOOCV_1_2)
```

```
## Linear Regression
##
## 100 samples
##   1 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 99, 99, 99, 99, 99, 99, ...
## Resampling results:
##
##   RMSE     Rsquared    MAE
##   2.69966  0.00130345  1.921879
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
## 10 fold CV
model_kCV_1_2 <- train(y ~ ., data=df_1, method="lm", trControl=ctrl_kcv)
print(model_kCV_1_2)
```

```
## Linear Regression
##
## 100 samples
##   1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 91, 91, 89, 90, 89, 90, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   2.605796  0.2937923  1.912726
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
## Save the results
LOOCV_MSE_2 = append(LOOCV_MSE_2, mse(model_LOOCV_1_2))
kCV_MSE_2 = append(kCV_MSE_2, mse(model_kCV_1_2))

# Second model

## LOOCV
model_LOOCV_2_2 <- train(y ~ ., data=df_2, method="lm", trControl=ctrl_loocv)
print(model_LOOCV_2_2)
```

```
## Linear Regression
##
## 100 samples
##   2 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 99, 99, 99, 99, 99, 99, ...
## Resampling results:
```

```
##
##   RMSE       Rsquared   MAE
##   0.9682064  0.8663108  0.7829438
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
## 10 fold CV
model_kCV_2_2 <- train(y ~ ., data=df_2, method="lm", trControl=ctrl_kcv)
print(model_kCV_2_2)
```

```
## Linear Regression
##
## 100 samples
##   2 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 91, 90, 88, 91, 91, 91, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9462944  0.8508985  0.7817394
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
## Save the results
LOOCV_MSE_2 = append(LOOCV_MSE_2, mse(model_LOOCV_2_2))
kCV_MSE_2 = append(kCV_MSE_2, mse(model_kCV_2_2))


# Third model

## LOOCV
model_LOOCV_3_2 <- train(y ~ ., data=df_3, method="lm", trControl=ctrl_loocv)
print(model_LOOCV_3_2)
```

```
## Linear Regression
##
## 100 samples
##   3 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 99, 99, 99, 99, 99, 99, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9780705  0.8637902  0.7924894
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
## 10 fold CV
model_kCV_3_2 <- train(y ~ ., data=df_3, method="lm", trControl=ctrl_kcv)
print(model_kCV_3_2)
```

```
## Linear Regression
##
## 100 samples
##   3 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 89, 90, 90, 91, 90, 88, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9697049  0.8468917  0.7957661
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
## Save the results
LOOCV_MSE_2 = append(LOOCV_MSE_2, mse(model_LOOCV_3_2))
kCV_MSE_2 = append(kCV_MSE_2, mse(model_kCV_3_2))


# Fourth model

## LOOCV
model_LOOCV_4_2 <- train(y ~ ., data=df_4, method="lm", trControl=ctrl_loocv)
print(model_LOOCV_4_2)
```

```
## Linear Regression
##
## 100 samples
##   4 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 99, 99, 99, 99, 99, 99, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9766805  0.8639008  0.8019279
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
## 10 fold CV
model_kCV_4_2 <- train(y ~ ., data=df_4, method="lm", trControl=ctrl_kcv)
print(model_kCV_4_2)
```

```
## Linear Regression
##
## 100 samples
```

```
##   4 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 90, 90, 90, 90, 89, 89, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9581952  0.8539694  0.8136199
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
## Save the results
LOOCV_MSE_2 = append(LOOCV_MSE_2, mse(model_LOOCV_4_2))
kCV_MSE_2 = append(kCV_MSE_2, mse(model_kCV_4_2))

# Results
result_2 = data.frame(Model = unlist(model), LOOCV_100 = unlist(LOOCV_MSE),
                      kCV_100 = unlist(kCV_MSE),LOOCV_99=unlist(LOOCV_MSE_2),
                      kCV_99 = unlist(kCV_MSE_2))
kable(result_2, caption = "Comparison of different seeds")
```

Table 2: Comparison of different seeds

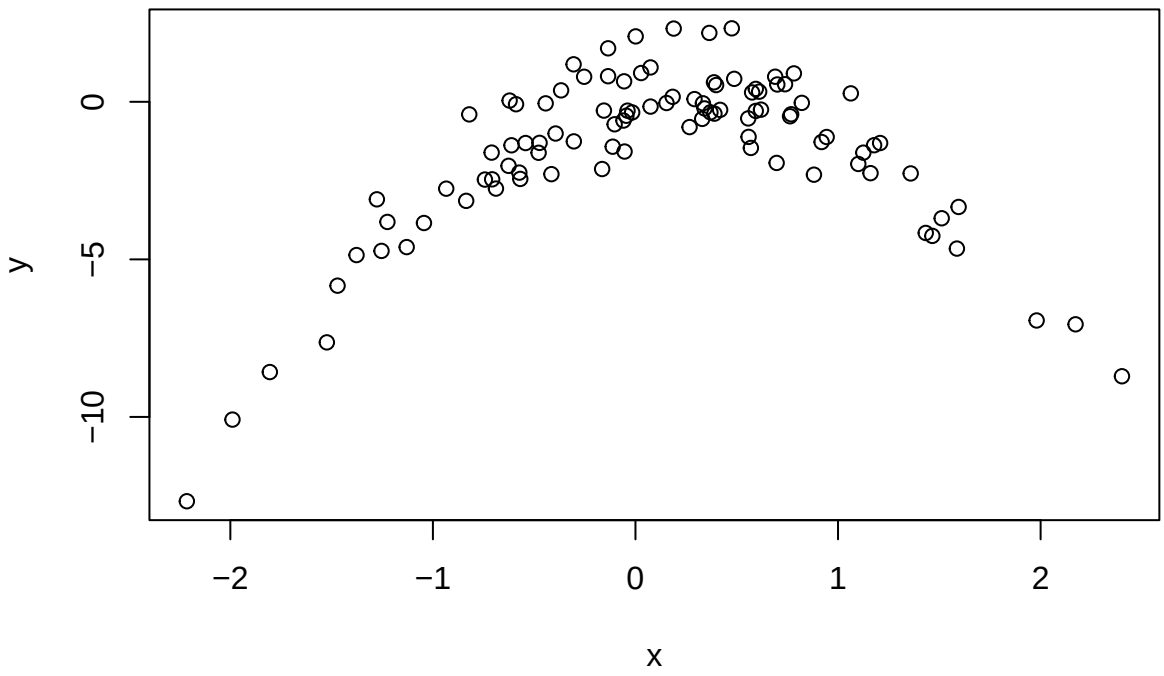
Model	LOOCV_100	kCV_100	LOOCV_99	kCV_99
i	7.2881616	6.9623524	7.2881616	6.7901735
ii	0.9374236	0.8948565	0.9374236	0.8954731
iii	0.9566218	0.9063947	0.9566218	0.9403276
iv	0.9539049	0.9290803	0.9539049	0.9181379

As can be seen from Table 2, the results for Leave-One-Out CV is similar for different seeds since in LOOCV we average the result of n models, which differ in only one observation, therefore we can say that there is more overlap. However, for 10-fold cross validation we observe an (insignificant) difference. In this case, we use 10 fitted models and average the results. Here the difference can be explained by the lower correlation between the training sets resulting from a smaller overlap.

d)

$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$ has the smallest LOOCV and 10-fold cross validation error in both seeds. This can be explained with the scatterplot below. One can see the curvature on the graph, which can be explained by a second degree polynomial, consistent with our results.

```
plot(x,y)
```



Task 7

Task 8

Task 9

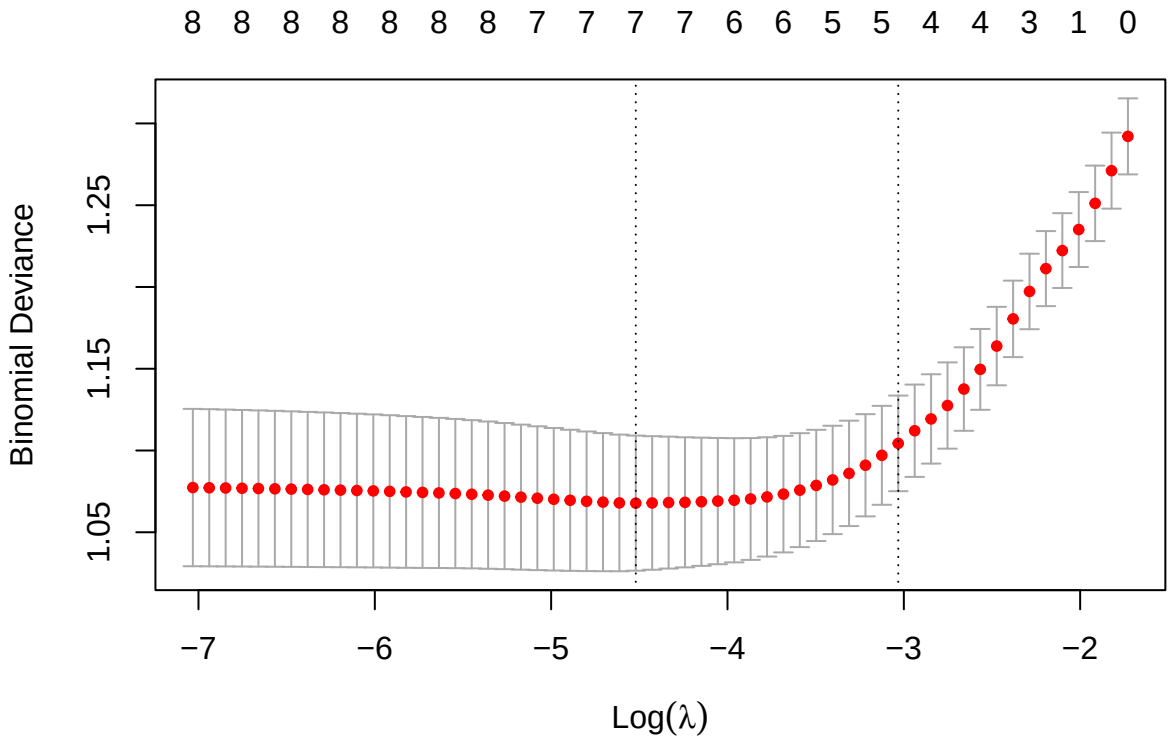
```
## Warning: Paket 'glmnet' wurde unter R Version 4.2.3 erstellt
```

```
## Warning: Paket 'Matrix' wurde unter R Version 4.2.3 erstellt
```

```
# Import dataset
data("SAheart", package="ElemStatLearn")
df = SAheart
X<-cbind(df$sbp,df$tobacco,df$ldl,df$adiposity,factor(df$famhist),
         df$typea,df$obesity,df$alcohol,df$age)
colnames(X) <- c('sbp','tobacco','ldl','adiposity','famhist',
                 'typea','obesity','alcohol','age')

# Fit a logistic regression model with Lasso penalty using only
#linear effects for the covariates.
model<-glmnet(y = df$chd,x=X,family = "binomial",alpha = 1, nfolds=20)
model_cv<-cv.glmnet(y = df$chd,x=X,family = "binomial",alpha = 1, nfolds=20)

# Visualize the results
plot(model_cv)
```




```
# Penalty selection
cat("Penalty value which minimizes the cross-validation loss: ",model_cv$lambda.min)
```

```
## Penalty value which minimizes the cross-validation loss:  0.01088875
```

```
cat("Penalty value according to 1 - SE rule: ",model_cv$lambda.1se)
```

```
## Penalty value according to 1 - SE rule:  0.04824393
```

```
# Model selection
kable(as.matrix(cbind(coef(model_cv$glmnet.fit,s = model_cv$lambda.1se),
                      coef(model_cv$glmnet.fit,s = model_cv$lambda.min))),
      caption="Complexity Assessment", col.names = c("Lambda - 1SE", "Lambda - Min"))
```

Table 3: Complexity Assessment

	Lambda - 1SE	Lambda - Min
(Intercept)	-3.5109766	-6.4776371
sbp	0.0000000	0.0039272
tobacco	0.0424312	0.0697322
ldl	0.0778817	0.1448104
adiposity	0.0000000	0.0000000
famhist	0.4847274	0.8005463

	Lambda - 1SE	Lambda - Min
typea	0.0044875	0.0288383
obesity	0.0000000	-0.0137939
alcohol	0.0000000	0.0000000
age	0.0314041	0.0435242

In terms of model complexity, we can see that the coefficients with the 1-SE lambda show that we eliminate 4 features whereas in the min lambda coefficients the algortihm eliminates only 2 features. That makes the min lambda model more complex. Which was expected as we increase the lambda, the penalty increases and more coefficients become zero and that removes less efficient features from the model. Choosing between the two models, we can say that the minimum lambda model is more prone to overfitting and the 1-SE model is a more conservative choice with fewer features eliminated.

```
preds_min <- ifelse(predict(model_cv, X, s = "lambda.min", type = "response")>0.5, 1, 0)

preds_1se <- ifelse(predict(model_cv, X, s = "lambda.1se", type = "response")>0.5, 1, 0)

expected=df$chd
temp = table(expected,preds_min)
kable(temp, caption="Confusion Matrix - Min")
```

Table 4: Confusion Matrix - Min

	0	1
0	261	41
1	77	83

```
temp2 = table(expected,preds_1se)
kable(temp2, caption="Confusion Matrix - 1-SE")
```

Table 5: Confusion Matrix - 1-SE

	0	1
0	282	20
1	99	61

From Table 4 and Table 5, we can see that the minimum lambda model did a better job correctly classifying the 1s(Yes) and the 1-SE model did a better job classifying the 0s(No). Minimum lambda model classified more observations as 1s and the 1-SE model classified more observations as 0s.

```
cat("Misclassification error for minimum lambda: ", (temp[2]+temp[3])/sum(temp))
```

```
## Misclassification error for minimum lambda:  0.2554113
```

```
cat("Misclassification error for 1-SE lambda: ", (temp2[2]+temp2[3])/sum(temp2))
```

```
## Misclassification error for 1-SE lambda:  0.2575758
```

In terms of misclassification error, minimum lambda model did a better job by correctly classifying the observations. However, the difference is not significant. Overall, in this case we do not want to miss the true 1s considering the medical context, therefore the choice should be the the model minimizing cross-validation loss.

Task 10