Exercise 6 - Cross Validation of Models

Yannik Gaebel

2022-11-26

Contents

Setup

Set random seed.

```
set.seed(12208157)
```

Import libraries

```
library(ggplot2)
library(ISLR)
library(splines)
library(dplyr)
```

```
##
## Attache Paket: 'dplyr'

## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     filter, lag

## Die folgenden Objekte sind maskiert von 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(boot)
```

Task 1.

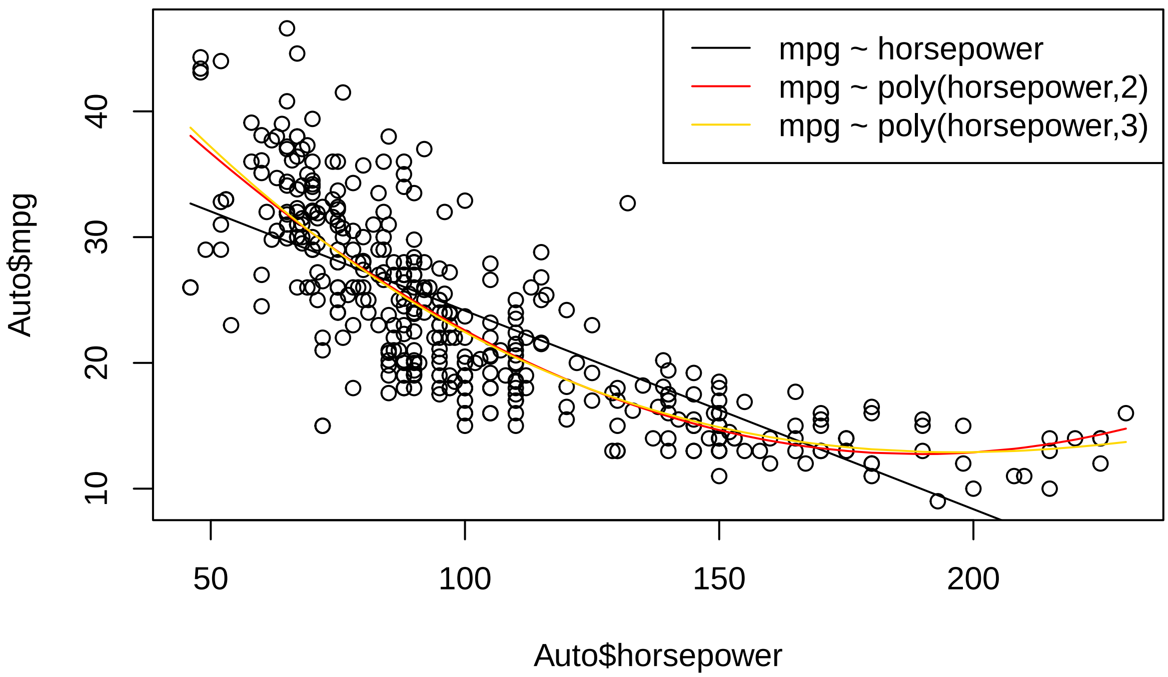
Import data

```
data(Auto)
#?Auto
#pairs(Auto)
```

a)

```
Auto <- arrange(Auto, horsepower)
dfs <- 1:3
FITS <- apply(t(dfs), 2, function(df) lm(mpg ~ poly(horsepower, df=df), data=Auto))

plot(Auto$horsepower,Auto$mpg)
lines(Auto$horsepower, fitted(FITS[[1]]), col="black")
lines(Auto$horsepower, fitted(FITS[[2]]), col="red")
lines(Auto$horsepower, fitted(FITS[[3]]), col="gold")
legend("topright",legend=c("mpg ~ horsepower","mpg ~ poly(horsepower,2)","mpg ~ poly(horsepower,3)"),co
```



b)

Use the validation set approach to compare the models. Use once a train/test split of 50%/50% and once 70%/30%. Choose the best model based on Root Mean Squared Error, Mean Squared Error and Median Absolute Deviation.

```
dfs <- 1:3
set.seed(12208157)

n <- nrow(Auto)

train_sample_50 <- sample(1:n,n*0.5)
train_sample_70 <- sample(1:n,n*0.7)

TRAIN_50 <- Auto[train_sample_50,]
TEST_50 <- Auto[-train_sample_50,]
TRAIN_70 <- Auto[train_sample_70,]
TEST_70 <- Auto[-train_sample_70,]

FITS <- apply(t(dfs), 2, function(df) lm(mpg ~ poly(horsepower, df=df), data=TRAIN_50))
FITS2 <- apply(t(dfs), 2, function(df) lm(mpg ~ poly(horsepower, df=df), data=TRAIN_70))

PREDS <- lapply(FITS, predict, TEST_50)
PREDS2 <- lapply(FITS2, predict, TEST_70)
```

Calculate Mean Squared Error values for three models

```
MSE <- function(yhat, y) mean((yhat-y)^2)
MSES <- lapply(PREDS, MSE, y=TEST_50$y)
MSES2 <- lapply(PREDS2, MSE, y=TEST_70$y)

#print("50/50 split:")
#print(paste0("MSE for Model 1: ",round(unlist(MSES)[1],1)))
#print(paste0("MSE for Model 2: ",round(unlist(MSES)[2],1)))
#print(paste0("MSE for Model 3: ",round(unlist(MSES)[3],1)))

#print("70/30 split:")
#print(paste0("MSE for Model 1: ",round(unlist(MSES2)[1],1)))
#print(paste0("MSE for Model 2: ",round(unlist(MSES2)[2],1)))
#print(paste0("MSE for Model 3: ",round(unlist(MSES2)[3],1)))
```

Calculate Root Mean Squared Error values for three models

```
RMSE <- function(yhat, y) sqrt(mean((yhat-y)^2))
RMSES <- lapply(PREDS, RMSE, y=TEST_50$y)
RMSES2 <- lapply(PREDS2, RMSE, y=TEST_70$y)

#print("50/50 split:")
#print(paste0("RMSE for Model 1: ",round(unlist(RMSES)[1],2)))
#print(paste0("RMSE for Model 2: ",round(unlist(RMSES)[2],2)))
#print(paste0("RMSE for Model 3: ",round(unlist(RMSES)[3],2)))

#print("70/30 split:")
#print(paste0("RMSE for Model 1: ",round(unlist(RMSES2)[1],2)))
#print(paste0("RMSE for Model 2: ",round(unlist(RMSES2)[2],2)))
#print(paste0("RMSE for Model 3: ",round(unlist(RMSES2)[3],2)))
```

Calculate Median Absolute Deviation values for three models

```
MAD <- function(yhat, y) median(abs(yhat-y))
MADS <- lapply(PREDS, MAD, y=TEST_50$y)
MADS2 <- lapply(PREDS2, MAD, y=TEST_70$y)

#print("50/50 split:")
#print(paste0("MAD for Model 1: ",round(unlist(MADS)[1],2)))
#print(paste0("MAD for Model 2: ",round(unlist(MADS)[2],2)))
#print(paste0("MAD for Model 3: ",round(unlist(MADS)[3],2)))

#print("70/30 split:")
#print(paste0("MAD for Model 1: ",round(unlist(MADS2)[1],2)))
#print(paste0("MAD for Model 2: ",round(unlist(MADS2)[2],2)))
#print(paste0("MAD for Model 3: ",round(unlist(MADS2)[3],2)))
```

c)

Use the cv.glm function in the boot package for the following steps.

1. Use cv.glm for Leave-one-out Cross Validation to compare the models above.

```r
dfs <- 1:3
FITS <- apply(t(dfs), 2, function(df) lm(mpg ~ poly(horsepower, df=df), data=Auto))

cv.mod1.1 <- cv.glm(Auto,  glm(FITS[[1]]))
cv.mod2.1 <- cv.glm(Auto,  glm(FITS[[2]]))
cv.mod3.1 <- cv.glm(Auto,  glm(FITS[[3]]))
```

2. Use cv.glm for 5-fold and 10-fold Cross Validation to compare the models above.

5-fold cross validation:

```r
cv.mod1.2 <- cv.glm(Auto,  glm(FITS[[1]]), K=5)
cv.mod2.2 <- cv.glm(Auto,  glm(FITS[[2]]), K=5)
cv.mod3.2 <- cv.glm(Auto,  glm(FITS[[3]]), K=5)
```

10-fold cross validation:

```r
cv.mod1.3 <- cv.glm(Auto,  glm(FITS[[1]]), K=10)
cv.mod2.3 <- cv.glm(Auto,  glm(FITS[[2]]), K=10)
cv.mod3.3 <- cv.glm(Auto,  glm(FITS[[3]]), K=10)
```

d)

Compare all results from 2 and 3. in a table and draw your conclusions.

Results: MSE, RMSE, MDA for each model and both train/test splits

```r
data.frame(
  "Split" = c("50/50 split","50/50 split","50/50 split","70/30 split","70/30 split","70/30 split"),
  "Model" = c("model1", "model2", "model3","model1", "model2", "model3"),
  "MSE" = c(round(unlist(MSES)[1],1),round(unlist(MSES)[2],1),round(unlist(MSES)[3],1),round(unlist(MSE
            ,round(unlist(MSES2)[2],1),round(unlist(MSES2)[3],1)),
  "RMSE" = c(round(unlist(RMSES)[1],1),round(unlist(RMSES)[2],1),round(unlist(RMSES)[3],1),round(unlist
            ,round(unlist(RMSES2)[2],1),round(unlist(RMSES2)[3],1)),
  "MAD" = c(round(unlist(MADS)[1],1),round(unlist(MADS)[2],1),round(unlist(MADS)[3],1),round(unlist(MAD
            ,round(unlist(MADS2)[2],1),round(unlist(MADS2)[3],1)))
```

```
##         Split  Model    MSE RMSE  MAD
## 1 50/50 split model1 2700.6 52.0 51.7
## 2 50/50 split model2 2707.9 52.0 51.6
## 3 50/50 split model3 2708.9 52.0 51.7
## 4 70/30 split model1 2768.7 52.6 52.9
## 5 70/30 split model2 2804.3 53.0 54.1
## 6 70/30 split model3 2806.4 53.0 54.4
```

Conclusions:

For the 50/50 train/test split the second model performs best if you look at MAD values but the first model has a lower MSE value. The models perform overall very similar.

For the 70/30 train/test split first model performs better than the other ones for all metrics.

Results: MSE for each model with 5-fold 10-fold and LOO CV

```
data.frame(
  "Model" = c("model1", "model2", "model3"),
  "5-folf CV MSE" = c(round(cv.mod1.2$delta[1],2),round(cv.mod2.2$delta[1],2),round(cv.mod3.2$delta[1],
  "10-folf CV MSE" = c(round(cv.mod1.3$delta[1],2),round(cv.mod2.3$delta[1],2),round(cv.mod3.3$delta[1]
  "LOOCV MSE" = c(round(cv.mod1.1$delta[1],2),round(cv.mod2.1$delta[1],2),round(cv.mod3.1$delta[1],2)))
```

```
##    Model X5.folf.CV.MSE X10.folf.CV.MSE LOOCV.MSE
## 1 model1          24.27           24.26     24.23
## 2 model2          19.21           19.33     19.25
## 3 model3          19.53           19.38     19.33
```

Conclusions:

The second model performs best. It is only slightly better than the third model but much better than the first model.

The results are very similar for 5-fold, 10-fold and LOO CV.

Task 2.

```
data(economics)
```

a)

Fit the following models to explain the number of unemployed persons ‘unemploy’ by the median number of days unemployed ‘uempmed’ and vice versa

1. Linear Model

```
lm_unemploy <- glm(unemploy ~ uempmed, data = economics)
summary(lm_unemploy)
```

```
##
## Call:
## glm(formula = unemploy ~ uempmed, data = economics)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3005.2   -792.9   -109.8    931.1   3600.7
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2956.8      126.8   23.32   <2e-16 ***
## uempmed        559.3       13.3   42.06   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1708187)
##
##     Null deviance: 3999510381  on 573  degrees of freedom
```

```
## Residual deviance:  977082779  on 572  degrees of freedom
## AIC: 9870.4
##
## Number of Fisher Scoring iterations: 2
```

```
lm_uempmed <- glm(uempmed ~ unemploy, data = economics)
summary(lm_uempmed)
```

```
##
## Call:
## glm(formula = uempmed ~ unemploy, data = economics)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.2674  -1.5802   0.0181   1.0254   7.5343
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.892e+00  2.637e-01  -7.177 2.22e-12 ***
## unemploy     1.351e-03  3.212e-05  42.064  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 4.127218)
##
##     Null deviance: 9663.4  on 573  degrees of freedom
## Residual deviance: 2360.8  on 572  degrees of freedom
## AIC: 2446.6
##
## Number of Fisher Scoring iterations: 2
```

2. exponential or logarithmic model

```
log_unemploy <- glm(unemploy ~ log(uempmed), data = economics)
summary(log_unemploy)
```

```
##
## Call:
## glm(formula = unemploy ~ log(uempmed), data = economics)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2280.5   -891.0   -252.0    878.5   3133.5
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -4935.7      261.8  -18.85   <2e-16 ***
## log(uempmed)   6143.9      124.4   49.37   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1328910)
```

```
##
##     Null deviance: 3999510381  on 573  degrees of freedom
## Residual deviance:  760136476  on 572  degrees of freedom
## AIC: 9726.3
##
## Number of Fisher Scoring iterations: 2
```

```
exp_uempmed <- glm(uempmed ~ log(unemploy), data = economics)
summary(exp_uempmed)
```

```
##
## Call:
## glm(formula = uempmed ~ log(unemploy), data = economics)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7856  -1.9608  -0.4871   0.7934  10.5946
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -69.6121     2.7317  -25.48   <2e-16 ***
## log(unemploy)   8.7909     0.3068   28.66   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 6.935919)
##
##     Null deviance: 9663.4  on 573  degrees of freedom
## Residual deviance: 3967.3  on 572  degrees of freedom
## AIC: 2744.6
##
## Number of Fisher Scoring iterations: 2
```

3.polynomial model of 2nd, 3rd and 10th degree

```
poly.2_unemploy <- glm(unemploy ~ poly(uempmed,2), data = economics)
poly.3_unemploy <- glm(unemploy ~ poly(uempmed,3), data = economics)
poly.10_unemploy <- glm(unemploy ~ poly(uempmed,10), data = economics)

poly.2_uempmed <- glm(uempmed ~ poly(unemploy,2), data = economics)
poly.3_uempmed <- glm(uempmed ~ poly(unemploy,3), data = economics)
poly.10_uempmed <- glm(uempmed ~ poly(unemploy,10), data = economics)
```

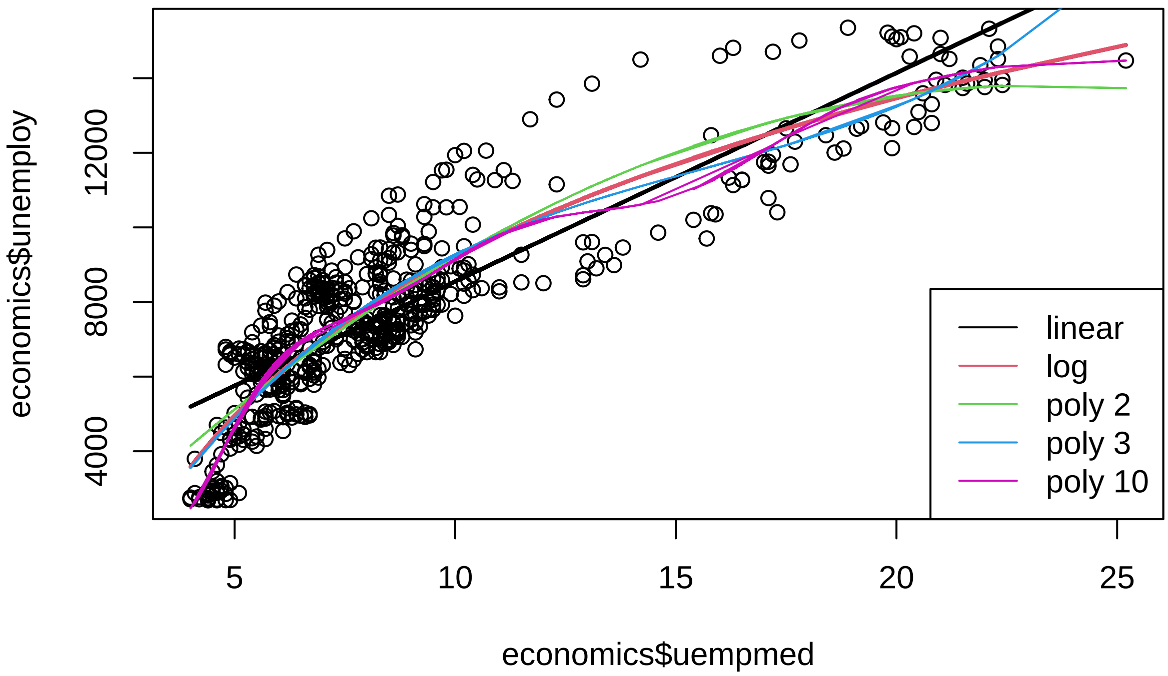
b)

Plot the corresponding data and add all the models for comparison.

Plot models to predict umemploy:

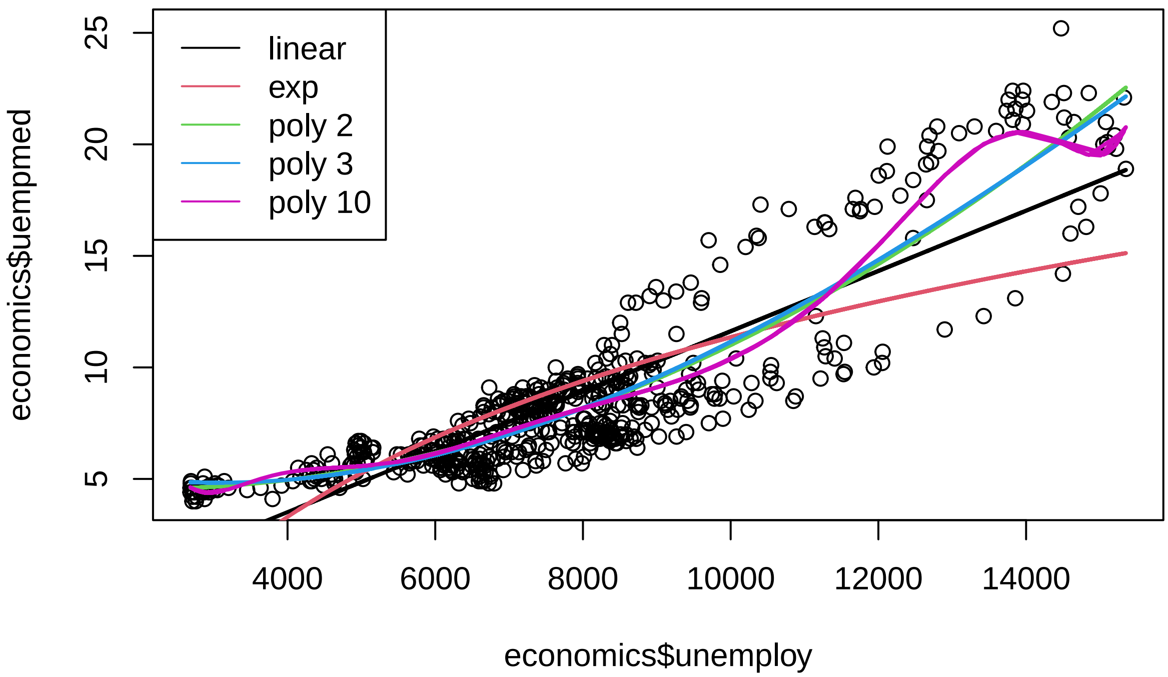
```
plot(economics$uempmed, economics$unemploy)
lines(economics$uempmed, fitted(lm_unemploy), col=1, lwd=2)
lines(economics$uempmed, fitted(log_unemploy), col=2, lwd=2)
```

```
lines(economics$uempmed, fitted(poly.2_unemploy), col=3, lwd=1)
lines(economics$uempmed, fitted(poly.3_unemploy), col=4, lwd=1)
lines(economics$uempmed, fitted(poly.10_unemploy), col=6, lwd=1)
legend("bottomright",legend=c("linear","log","poly 2","poly 3","poly 10"),col=c(1,2,3,4,6), lty=1)
```



Plot models to predict uempmed:

```
plot(economics$unemploy, economics$uempmed)
lines(economics$unemploy, fitted(lm_uempmed), col=1, lwd=2)
lines(economics$unemploy, fitted(exp_uempmed), col=2, lwd=2)
lines(economics$unemploy, fitted(poly.2_uempmed), col=3, lwd=2)
lines(economics$unemploy, fitted(poly.3_uempmed), col=4, lwd=2)
lines(economics$unemploy, fitted(poly.10_uempmed), col=6, lwd=2)
legend("topleft",legend=c("linear","exp","poly 2","poly 3","poly 10"),col=c(1,2,3,4,6), lty=1)
```



c)

Use the cv.glm function in the boot package for the following steps. Compare the Root Mean Squared Error and Mean Squared Error.

1. Use cv.glm for Leave-one-out Cross Validation to compare the models above

```
# cost.MSE <- function(r, pi = 0) mean((r-pi)^2)

cv.lm_unemploy <- cv.glm(economics, glm(lm_unemploy))
cv.log_unemploy <- cv.glm(economics, glm(log_unemploy))
cv.poly.2_unemploy <- cv.glm(economics, glm(poly.2_unemploy))
cv.poly.3_unemploy <- cv.glm(economics, glm(poly.3_unemploy))
cv.poly.10_unemploy <- cv.glm(economics, glm(poly.10_unemploy))
```

```
data.frame(
  "Model" = c("linear", "exponential", "poly 2", "poly 3", "poly 10"),
  "LOOCV MSE" = c(cv.lm_unemploy$delta[1],cv.log_unemploy$delta[1],cv.poly.2_unemploy$delta[1],
                cv.poly.3_unemploy$delta[1],cv.poly.10_unemploy$delta[1]),
  "LOOCV RMSE" = c(sqrt(cv.lm_unemploy$delta[1]),sqrt(cv.log_unemploy$delta[1]),sqrt(cv.poly.2_unemploy$
                sqrt(cv.poly.3_unemploy$delta[1]),sqrt(cv.poly.10_unemploy$delta[1])) )
```

```
##         Model LOOCV.MSE LOOCV.RMSE
## 1      linear   1715211   1309.661
```

```
## 2 exponential   1333997   1154.988
## 3      poly 2   1432531   1196.884
## 4      poly 3   1366405   1168.933
## 5     poly 10   4530738   2128.553
```

2. Use cv.glm for 5-fold and 10-fold Cross Validation to compare the models above.

```
cv5.lm_unemploy <- cv.glm(economics, glm(lm_unemploy), K=5)
cv5.log_unemploy <- cv.glm(economics, glm(log_unemploy), K=5)
cv5.poly.2_unemploy <- cv.glm(economics, glm(poly.2_unemploy), K=5)
cv5.poly.3_unemploy <- cv.glm(economics, glm(poly.3_unemploy), K=5)
cv5.poly.10_unemploy <- cv.glm(economics, glm(poly.10_unemploy), K=5)
```

5-fold CV

```
data.frame(
  "Model" = c("linear", "exponential", "poly 2", "poly 3", "poly 10"),
  "5-fold CV MSE" = c(cv5.lm_unemploy$delta[1],cv5.log_unemploy$delta[1],cv5.poly.2_unemploy$delta[1],
                  cv5.poly.3_unemploy$delta[1],cv5.poly.10_unemploy$delta[1]),
  "5-fold CV RMSE" = c(sqrt(cv5.lm_unemploy$delta[1]),sqrt(cv5.log_unemploy$delta[1]),sqrt(cv5.poly.2_u
                  sqrt(cv5.poly.3_unemploy$delta[1]),sqrt(cv5.poly.10_unemploy$delta[1])) )
```

```
##         Model X5.fold.CV.MSE X5.fold.CV.RMSE
## 1      linear        1718388        1310.873
## 2 exponential        1333038        1154.573
## 3      poly 2        1442848        1201.186
## 4      poly 3        1355067        1164.073
## 5     poly 10        1774928        1332.264
```

```
cv10.lm_unemploy <- cv.glm(economics, glm(lm_unemploy), K=10)
cv10.log_unemploy <- cv.glm(economics, glm(log_unemploy), K=10)
cv10.poly.2_unemploy <- cv.glm(economics, glm(poly.2_unemploy), K=10)
cv10.poly.3_unemploy <- cv.glm(economics, glm(poly.3_unemploy), K=10)
cv10.poly.10_unemploy <- cv.glm(economics, glm(poly.10_unemploy), K=10)
```

10-fold CV

```
data.frame(
  "Model" = c("linear", "exponential", "poly 2", "poly 3", "poly 10"),
  "10-fold CV MSE" = c(cv10.lm_unemploy$delta[1],cv10.log_unemploy$delta[1],cv10.poly.2_unemploy$delta[
                  cv10.poly.3_unemploy$delta[1],cv10.poly.10_unemploy$delta[1]),
  "10-fold CV RMSE" = c(sqrt(cv10.lm_unemploy$delta[1]),sqrt(cv10.log_unemploy$delta[1]),sqrt(cv10.poly
                  sqrt(cv10.poly.3_unemploy$delta[1]),sqrt(cv10.poly.10_unemploy$delta[1])) )
```

```
##         Model X10.fold.CV.MSE X10.fold.CV.RMSE
## 1      linear         1713789         1309.118
## 2 exponential         1335553         1155.661
## 3      poly 2         1430499         1196.035
## 4      poly 3         1363996         1167.902
## 5     poly 10        16079526         4009.928
```

d)

Explain based on the CV and graphical model fits the concepts of Underfitting, Overfitting and how to apply cross-validation to determine the appropriate model fit. Also, describe the different variants of cross validation in this context.

Answer:

The concepts of underfitting and overfitting describe how a model fits a set of data. In unterfitting the model fits the data to rough which results in a prediction bias. An example for underfitting would be the linear model for the economica dataset. The complexity of the linear model is not enough to learn the curved structure of the data. Overfitting means that the model fits the training data too strongly that it also learn noise resulting in a high variance. An example of overfitting is the polynomial model of degree 10 for the economics data. You can also see in the MSE values from the CV that underfittet and overfittet models perform much worse that optimally fittet models. It is very important to find the right balance to achieve the best possible performance

Computing a cross validation score can give you a good idea of when the model starts to under- or overfit the data. Cross validation withholds a portion of the data for model building and uses this data to evaluate the model. Every data point is used for evaluation one time. The evaluation of the models can be done with a percentage 1/k of the data which is called k-fold cross validation. The dataset is split in k folds in this case. Or it can be done with just one datapoint for evaluation in each iteration which is called leave one out cross validation.

This technique gives a good idea of how the model would perform when confronted with unseen data. To find the appropriate model fit it is important to find a balance between bias and variance. The best model fit can be achieved by selecting the model with the lowest crocc validation error.