Exercise 6 - Cross Validation of Models

Bosse Behrens, st.id 12347333

2024W

Task 1

part 1

First we load the data and inspect it.

```r
library(ISLR)
data("Auto")
head(Auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

```r
summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower        weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
##
##   acceleration        year           origin                      name
##  Min.   : 8.00   Min.   :70.00   Min.   :1.000   amc matador       :  5
##  1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   ford pinto        :  5
##  Median :15.50   Median :76.00   Median :1.000   toyota corolla    :  5
##  Mean   :15.54   Mean   :75.98   Mean   :1.577   amc gremlin       :  4
##  3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000   amc hornet        :  4
##  Max.   :24.80   Max.   :82.00   Max.   :3.000   chevrolet chevette:  4
##                                                  (Other)           :365
```

```r
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241
```

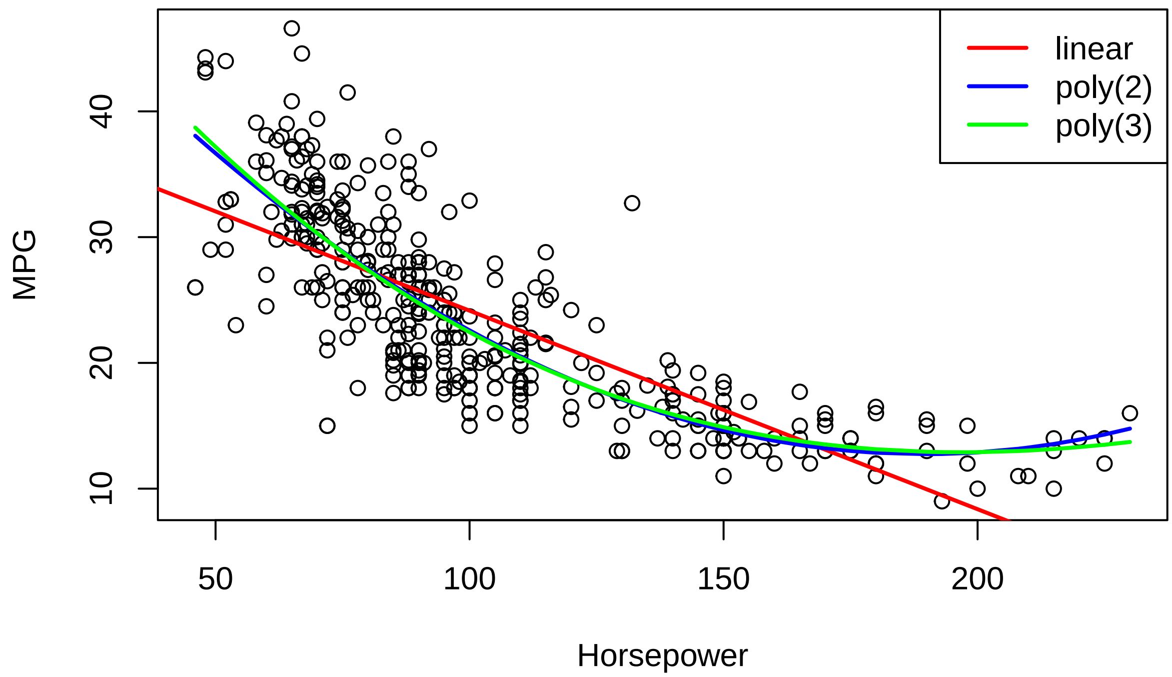
```r
#?Auto
```

Now we create the models using lm() with the specified response and the polynomial degree.

```r
model_1 <- lm(mpg ~ horsepower, data = Auto)
model_2 <- lm(mpg ~ poly(horsepower, 2), data = Auto)
model_3 <- lm(mpg ~ poly(horsepower, 3), data = Auto)
```

We now plot the models by first creating the scatterplot of mpg and horsepower fromm the original data and then sorting the predicted mpg and horsepower by values. We are then plotting a line chart on the same plot for each of the three models.

```r
plot(Auto$horsepower, Auto$mpg, xlab="Horsepower", ylab="MPG")
abline(model_1, col = "red", lwd = 2)
lines(sort(Auto$horsepower),
  predict(model_2, newdata = list(horsepower = sort(Auto$horsepower))),
      col = "blue", lwd = 2)
lines(sort(Auto$horsepower),
  predict(model_3, newdata = list(horsepower = sort(Auto$horsepower))),
      col = "green", lwd = 2)
legend("topright", legend = c("linear", "poly(2)", "poly(3)"),
       col = c("red", "blue", "green"), lty = 1, lwd = 2)
```



part 2

We first create the 50/50 train/validation split by randomly sampling the data for the train set and then taking the valdiation set as the observations that are left.

```
set.seed(12347333)
options(digits = 4)
n <- nrow(Auto)
train_05 <- sample(1:n, round((1/2) * n))
test_05 <-(1:n)[-train_05]

train_data05 <- Auto[train_05, ]
test_data05 <- Auto[test_05, ]
```

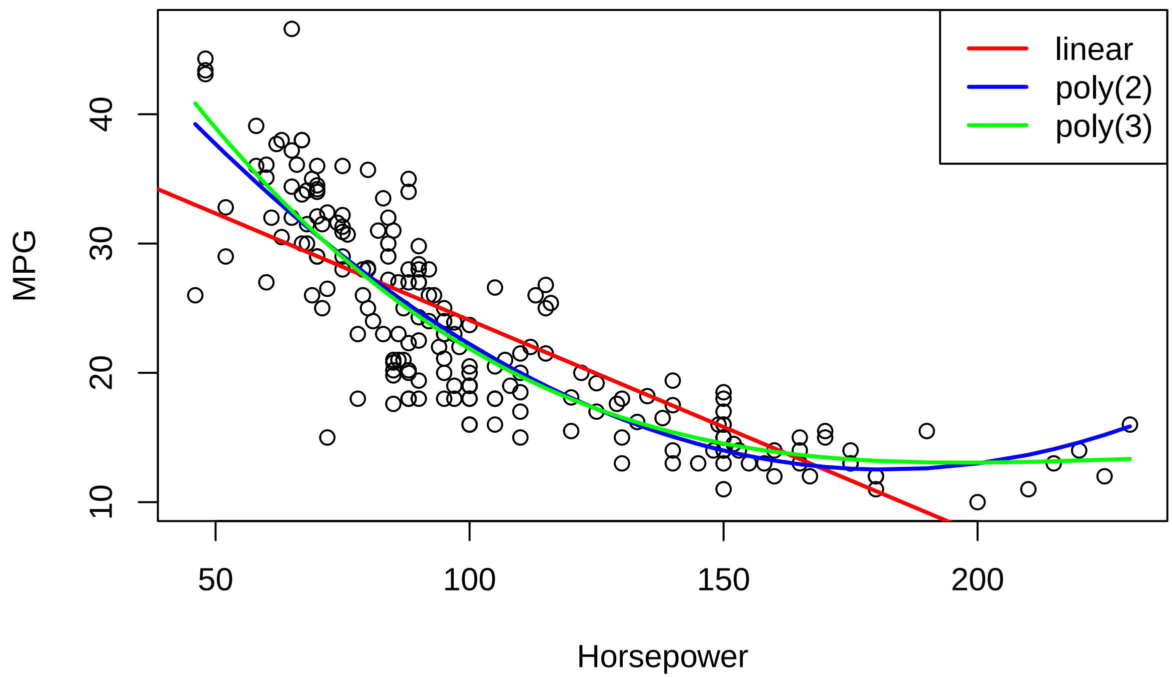
Now we again train the models, this time only on the train data.

```
model_1 <- lm(mpg ~ horsepower, data = train_data05)
model_2 <- lm(mpg ~ poly(horsepower, 2), data = train_data05)
model_3 <- lm(mpg ~ poly(horsepower, 3), data = train_data05)
```

Againw e plot the scatterplot of the inout data and then then the lines for each model with the predicted mpg valeus based on horsepower.

```
plot(train_data05$horsepower, train_data05$mpg, xlab="Horsepower", ylab="MPG")
abline(model_1, col = "red", lwd = 2)
lines(sort(train_data05$horsepower),
predict(model_2, newdata = list(horsepower = sort(train_data05$horsepower))),
      col = "blue", lwd = 2)
```

```r
lines(sort(train_data05$horsepower),
predict(model_3, newdata = list(horsepower = sort(train_data05$horsepower))),
      col = "green", lwd = 2)
legend("topright", legend = c("linear", "poly(2)", "poly(3)"),
       col = c("red", "blue", "green"), lty = 1, lwd = 2)
```



We create custom functions for calculating the mse, rmse and mad for the predicted values.

```r
mse <- function(y, yhat) mean((y-yhat)^2)
rmse <- function(y, yhat) sqrt(mse(y, yhat))
mad <- function(y, yhat) {
  median(abs(y - yhat))
}
```

Now we first predict the response based on the validation data and then calculate mse, rmse and mad for each of the three models.

```r
pred_1 <- predict(model_1, test_data05)
pred_2 <- predict(model_2, test_data05)
pred_3 <- predict(model_3, test_data05)

results_05 <- cbind(
    Model = c("Linear 50/50 valid split", "Quadratic 50/50 valid split",
              "Cubic 50/50 valid split"),
    mse = sprintf("%.4f", c(mse(test_data05$mpg, pred_1), mse(test_data05$mpg, pred_2),
            mse(test_data05$mpg, pred_3))),
    mad = sprintf("%.4f", c(mad(test_data05$mpg, pred_1), mad(test_data05$mpg, pred_2),
            mad(test_data05$mpg, pred_3))),
```

```
    rmse = sprintf("%.4f", c(rmse(test_data05$mpg, pred_1), rmse(test_data05$mpg, pred_2),
             rmse(test_data05$mpg, pred_3)))
  )

print(results_05)
```

```
##      Model                          mse       mad      rmse
## [1,] "Linear 50/50 valid split"    "24.8436" "2.8093" "4.9843"
## [2,] "Quadratic 50/50 valid split" "22.1273" "2.3946" "4.7040"
## [3,] "Cubic 50/50 valid split"     "22.5050" "2.5476" "4.7439"
```

As we can see the quadratic model (poly(2)) performs best in terms of all 3 evaluation metrics.\ We now do the same steps as before but with a 70/30 train/validation split of the data. First we create the enw split as before.

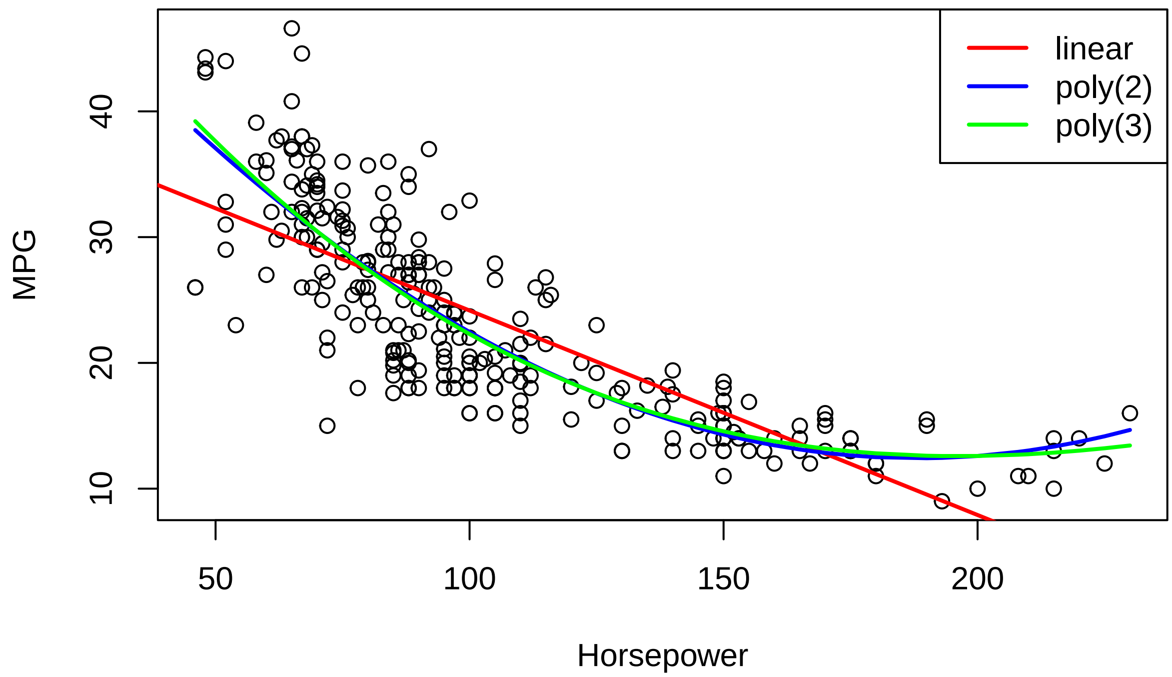
```
set.seed(12347333)
n <- nrow(Auto)
train_07 <- sample(1:n, round((0.7) * n))
test_07 <-(1:n)[-train_07]

train_data07 <- Auto[train_07, ]
test_data07 <- Auto[test_07, ]
```

Then we train the models on the train data.

```
model_1 <- lm(mpg ~ horsepower, data = train_data07)
model_2 <- lm(mpg ~ poly(horsepower, 2), data = train_data07)
model_3 <- lm(mpg ~ poly(horsepower, 3), data = train_data07)
```

Now we plot again the inout data in a scatterplot and add the models as lines.

```
plot(train_data07$horsepower, train_data07$mpg, xlab="Horsepower", ylab="MPG")
abline(model_1, col = "red", lwd = 2)
lines(sort(train_data07$horsepower),
      predict(model_2,
              newdata = list(horsepower = sort(train_data07$horsepower))),
      col = "blue", lwd = 2)
lines(sort(train_data07$horsepower),
      predict(model_3,
              newdata = list(horsepower = sort(train_data07$horsepower))),
      col = "green", lwd = 2)
legend("topright", legend = c("linear", "poly(2)", "poly(3)"),
       col = c("red", "blue", "green"), lty = 1, lwd = 2)
```



In the next step we again predict the values of mpg using the trained models and then calculate mse, rmse and mad for eahc of the three models.

```
pred_1 <- predict(model_1, test_data07)
pred_2 <- predict(model_2, test_data07)
pred_3 <- predict(model_3, test_data07)

results_07 <- cbind(
    model = c("Linear 70/30 valid split", "Quadratic 70/30 valid split",
              "Cubic 70/30 valid split"),
    mse = sprintf("%.4f", c(mse(test_data07$mpg, pred_1), mse(test_data07$mpg, pred_2),
            mse(test_data07$mpg, pred_3))),
    mad = sprintf("%.4f", c(mad(test_data07$mpg, pred_1), mad(test_data07$mpg, pred_2),
            mad(test_data07$mpg, pred_2))),
      rmse = sprintf("%.4f", c(rmse(test_data07$mpg, pred_1), rmse(test_data07$mpg, pred_2),
              rmse(test_data07$mpg, pred_3))))

print(results_07)
```

```
##      model                         mse       mad      rmse
## [1,] "Linear 70/30 valid split"    "23.5074" "2.7809" "4.8484"
## [2,] "Quadratic 70/30 valid split" "19.9966" "2.2564" "4.4718"
## [3,] "Cubic 70/30 valid split"     "19.9901" "2.2564" "4.4710"
```

As we can see using the 70/30 split the overall metrics are better compared to the 50/50 split. Furthermore both the quadratic as also the cubic models perform best and almost exactly the same in terms of the evaluation metrics, with the cubic performing very slightly better.

part 3

First we load the boot package.

```
library(boot)
```

Now we train the models again, this time using glm(), because in the next step cv.glm only works on models created by glm(), not lm().

```
model_1 <- glm(mpg ~ horsepower, data = Auto)
model_2 <- glm(mpg ~ poly(horsepower, 2), data = Auto)
model_3 <- glm(mpg ~ poly(horsepower, 3), data = Auto)
```

This time instead of using a train/validation split, we trained the model on the whole horsepower data again and will now use Leave-one-out, 5-fold and 10-fold cross-validation to calculate the mse, rmse and mad for each model. The resulting value is the mean of the results of the different cv-split used internally. We start with the Leave-one-out cv.

```
loocv_model_1_mse <- cv.glm(Auto, model_1, K = nrow(Auto),
                            cost = mse)$delta[2]
loocv_model_2_mse <- cv.glm(Auto, model_2, K = nrow(Auto),
                            cost = mse)$delta[2]
loocv_model_3_mse <- cv.glm(Auto, model_3, K = nrow(Auto),
                            cost = mse)$delta[2]
loocv_mse <- sprintf("%.4f", c(loocv_model_1_mse,
                               loocv_model_2_mse ,loocv_model_3_mse))

loocv_model_1_mad <- cv.glm(Auto, model_1, K = nrow(Auto),
                            cost = mad)$delta[2]
loocv_model_2_mad <- cv.glm(Auto, model_2, K = nrow(Auto),
                            cost = mad)$delta[2]
loocv_model_3_mad <- cv.glm(Auto, model_3, K = nrow(Auto),
                            cost = mad)$delta[2]
loocv_mad <- sprintf("%.4f", c(loocv_model_1_mad,
                               loocv_model_2_mad ,loocv_model_3_mad))

loocv_model_1_rmse <- cv.glm(Auto, model_1, K = nrow(Auto),
                             cost = rmse)$delta[2]
loocv_model_2_rmse <- cv.glm(Auto, model_2, K = nrow(Auto),
                             cost = rmse)$delta[2]
loocv_model_3_rmse <- cv.glm(Auto, model_3, K = nrow(Auto),
                             cost = rmse)$delta[2]
loocv_rmse <- sprintf("%.4f", c(loocv_model_1_rmse,
                                loocv_model_2_rmse ,loocv_model_3_rmse))
```

We now aggregate the results into a table to better inspect them.

```
rownames <- c("Leave-one-out-cv linear model",
          "Leave-one-out-cv quadratic model", "Leave-one-out-cv cubic model")

results_loocv <- cbind(rownames, loocv_mse, loocv_mad, loocv_rmse)
colnames(results_loocv) <- c("model", "mse", "mad", "rmse")
print(results_loocv)
```

```
##      model                              mse       mad      rmse
## [1,] "Leave-one-out-cv linear model"    "24.2311" "3.8468" "3.8487"
## [2,] "Leave-one-out-cv quadratic model" "19.2479" "3.2724" "3.2720"
```

```
## [3,] "Leave-one-out-cv cubic model"     "19.3345" "3.2765" "3.2767"
```

We can observe that using Leave-one-out cv the quadratic model performs best for all three evaluation metrics, but with the cubic model being again onyl very slightly worse.\ Now we do the same again but using 5-fold cv.

```
cv5_model_1_mse <- cv.glm(Auto, model_1, K = 5, cost = mse)$delta[2]
cv5_model_2_mse <- cv.glm(Auto, model_2, K = 5, cost = mse)$delta[2]
cv5_model_3_mse <- cv.glm(Auto, model_3, K = 5, cost = mse)$delta[2]
cv5_mse <- sprintf("%.4f", c(cv5_model_1_mse,
                             cv5_model_2_mse ,cv5_model_3_mse))

cv5_model_1_mad <- cv.glm(Auto, model_1, K = 5, cost = mad)$delta[2]
cv5_model_2_mad <- cv.glm(Auto, model_2, K = 5, cost = mad)$delta[2]
cv5_model_3_mad <- cv.glm(Auto, model_3, K = 5, cost = mad)$delta[2]
cv5_mad <- sprintf("%.4f", c(cv5_model_1_mad,
                             cv5_model_2_mad ,cv5_model_3_mad))

cv5_model_1_rmse <- cv.glm(Auto, model_1, K = 5, cost = rmse)$delta[2]
cv5_model_2_rmse <- cv.glm(Auto, model_2, K = 5, cost = rmse)$delta[2]
cv5_model_3_rmse <- cv.glm(Auto, model_3, K = 5, cost = rmse)$delta[2]
cv5_rmse <- sprintf("%.4f", c(cv5_model_1_rmse,
                              cv5_model_2_rmse ,cv5_model_3_rmse))
```

Aggregating the results again to inspect them.

```
rownames <- c("5-fold-cv linear model", "5-fold-cv quadratic model",
              "5-fold-cv cubic model")

results_cv5 <- cbind(rownames, cv5_mse, cv5_mad, cv5_rmse)
colnames(results_cv5) <- c("model", "mse", "mad", "rmse")
print(results_cv5)
```

```
##      model                       mse       mad      rmse
## [1,] "5-fold-cv linear model"    "24.1697" "3.1899" "4.9046"
## [2,] "5-fold-cv quadratic model" "19.2977" "2.3951" "4.3702"
## [3,] "5-fold-cv cubic model"     "19.4747" "2.4548" "4.3760"
```

As we can see for the 5-fold cv the quadratic mdoel also performs the best in termns of mse, rmse and mad.\ In thge last step we do the same again using 10-fold cv this time.

```
cv10_model_1_mse <- cv.glm(Auto, model_1, K = 10, cost = mse)$delta[2]
cv10_model_2_mse <- cv.glm(Auto, model_2, K = 10, cost = mse)$delta[2]
cv10_model_3_mse <- cv.glm(Auto, model_3, K = 10, cost = mse)$delta[2]
cv10_mse <- sprintf("%.4f", c(cv10_model_1_mse,
                              cv10_model_2_mse ,cv10_model_3_mse))

cv10_model_1_mad <- cv.glm(Auto, model_1, K = 10, cost = mad)$delta[2]
cv10_model_2_mad <- cv.glm(Auto, model_2, K = 10, cost = mad)$delta[2]
cv10_model_3_mad <- cv.glm(Auto, model_3, K = 10, cost = mad)$delta[2]
cv10_mad <- sprintf("%.4f", c(cv10_model_1_mad,
                              cv10_model_2_mad ,cv10_model_3_mad))

cv10_model_1_rmse <- cv.glm(Auto, model_1, K = 10, cost = rmse)$delta[2]
cv10_model_2_rmse <- cv.glm(Auto, model_2, K = 10, cost = rmse)$delta[2]
cv10_model_3_rmse <- cv.glm(Auto, model_3, K = 10, cost = rmse)$delta[2]
cv10_rmse <- sprintf("%.4f", c(cv10_model_1_rmse,
```

```
                                cv10_model_2_rmse ,cv10_model_3_rmse))
```

Aggregating reuslts again.

```r
rownames <- c("10-fold-cv linear model", "10-fold-cv quadratic model",
              "10-fold-cv cubic model")

results_cv10 <- cbind(rownames, cv10_mse, cv10_mad, cv10_rmse)
colnames(results_cv10) <- c("model", "mse", "mad", "rmse")
print(results_cv10)
```

```
##      model                        mse       mad      rmse
## [1,] "10-fold-cv linear model"    "24.1599" "3.1266" "4.8913"
## [2,] "10-fold-cv quadratic model" "19.2632" "2.5774" "4.3689"
## [3,] "10-fold-cv cubic model"     "19.2976" "2.4648" "4.3571"
```

Again for 10-fold cv as well the quadratic model performs best for mse and rmse and the cubic model being slightly better in terms of mad.

part 4

Now we aggregate the results from part 2 and 3 into one big table.

```r
overall_results <- rbind(results_05, results_07,
                         results_loocv, results_cv5, results_cv10)
print(overall_results)
```

```
##       Model                               mse       mad      rmse
##  [1,] "Linear 50/50 valid split"          "24.8436" "2.8093" "4.9843"
##  [2,] "Quadratic 50/50 valid split"       "22.1273" "2.3946" "4.7040"
##  [3,] "Cubic 50/50 valid split"           "22.5050" "2.5476" "4.7439"
##  [4,] "Linear 70/30 valid split"          "23.5074" "2.7809" "4.8484"
##  [5,] "Quadratic 70/30 valid split"       "19.9966" "2.2564" "4.4718"
##  [6,] "Cubic 70/30 valid split"           "19.9901" "2.2564" "4.4710"
##  [7,] "Leave-one-out-cv linear model"     "24.2311" "3.8468" "3.8487"
##  [8,] "Leave-one-out-cv quadratic model"  "19.2479" "3.2724" "3.2720"
##  [9,] "Leave-one-out-cv cubic model"      "19.3345" "3.2765" "3.2767"
## [10,] "5-fold-cv linear model"            "24.1697" "3.1899" "4.9046"
## [11,] "5-fold-cv quadratic model"         "19.2977" "2.3951" "4.3702"
## [12,] "5-fold-cv cubic model"             "19.4747" "2.4548" "4.3760"
## [13,] "10-fold-cv linear model"           "24.1599" "3.1266" "4.8913"
## [14,] "10-fold-cv quadratic model"        "19.2632" "2.5774" "4.3689"
## [15,] "10-fold-cv cubic model"            "19.2976" "2.4648" "4.3571"
```

As we can observe, the most often the quadratic model performs best for all three metrics. Only for the 70/30 train/valid split and the mad for 10-fold cv the cubic model performs better very slightly. The linear model performs way worse overall then both higher polynomial variants. The quadartic and cubic model are also very similar in performance every time. In conclusion the quadratic would be chosen here, because while the cubic model does not perform significantly worse it is more complex and therefore an unnecessary increase in complexity. The lienar model on the other hand seems to not be able to capture the data quite as well as the other two models and should therefore be dicarded in favor of one of higher polynomial degree. As we could also see in the previous scatterpltos with the models as lines added, there seems to be a trend in the data that follows some curve. Therefore a linear model is not able to catch this underlying structure, while a quadratic or cubic model are able to produce smooth curves that can capture this structure.

Task 2

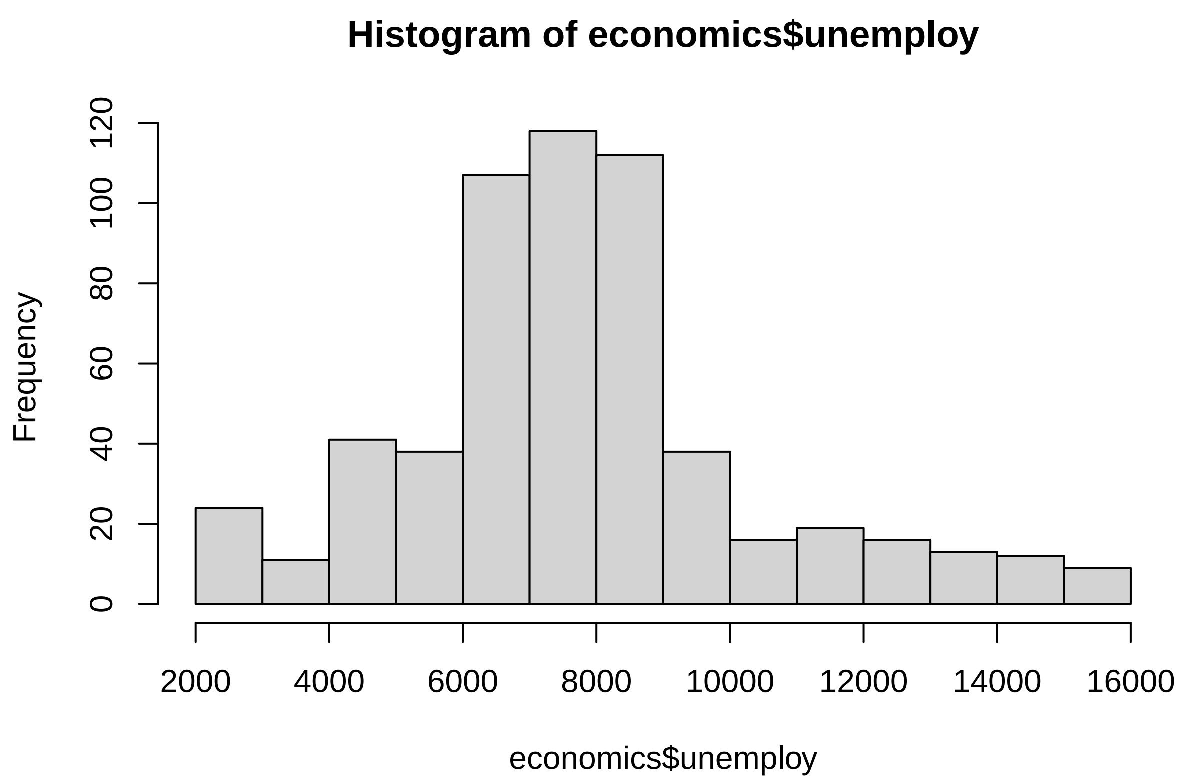
part 1

First we load the data "economics" from ggplot2.

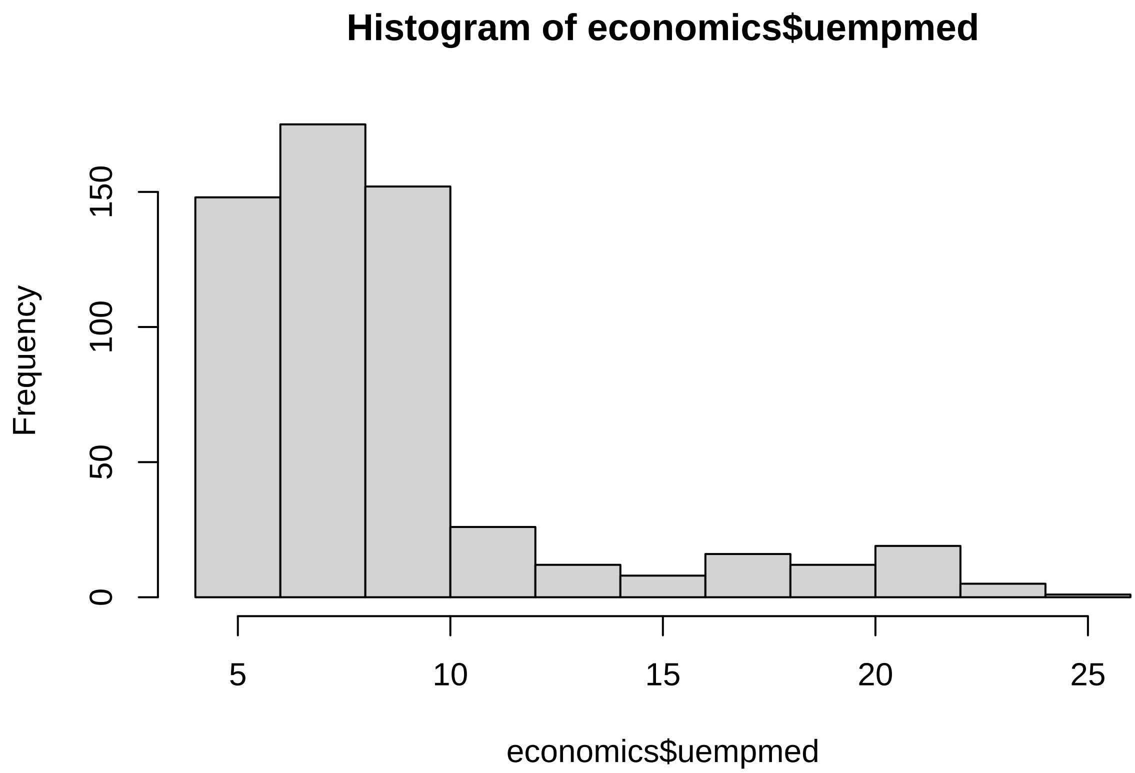
```
library(ggplot2)
data(economics)
```

We have to fit several models and of those one should be logarithmic and one exponential, depending on which is the dependent or independent variable of unemploy and uempmed. We therefore take a look at the distributions of both.

```
hist(economics$unemploy)
```




```
hist(economics$uempmed)
```



As we can see, unemploy looks mostly normally distributed, while uempmed is very right-skewed. This means the relationship in this case should be $unemploy \sim log(uempmed) \iff exp(unemploy) \sim uempmed \quad (= exp(log(uempmed)))$. This raised the problem, that unemploy has very high values, ranging (3000 - 15000) which R cannot handle as argument for exp() (returning inf). I tried different appraoches here. Scaling the original data for unemploy, so the range would be $\approx [-10, 10]$, using instead $log(uempmed) \sim unemploy$, meaning tranforming the repsonse instead of the predictor. Scaling made the values very small which led to inconsistencies later in calculating the MSE and RMSE due to machine precision. For the other approach the prediciotns need to be transformed back to be able to compare them with the results of the other models. FOr the plotting thatworked well, but for the Cross validation that would means using custom cost functions especially for the exponential model. I tried that and it again led to problems with precision and inconsistency. In the end I decided on approximating the MSE and RMSE for the exponential model by normally calculating both values for the log-transformed response and then using $MSE/RMSE \approx exp(logMSE/logRMSE)$. This approach is not very accurate and resulted in lower values than they should have been, but it is the best I could ocme up with. furthermore, the plots are correct and there one can better observe how well the exponential model performs in comparison to the others.\ First we train all the models on the data

```
#unemploy ~ uempmed
lin_model1 <- glm(unemploy ~ uempmed, data = economics)
log_model <- glm(unemploy ~ log(uempmed), data = economics)
poly_model1_2 <- glm(unemploy ~ poly(uempmed, 2), data = economics)
poly_model1_3 <- glm(unemploy ~ poly(uempmed, 3), data = economics)
poly_model1_10 <- glm(unemploy ~ poly(uempmed, 10), data = economics)
#uempmed ~ unemploy
lin_model2 <-glm(uempmed ~ unemploy, data = economics)
exp_model <- glm(log(uempmed) ~ unemploy, data = economics)
poly_model2_2 <- glm(uempmed ~ poly(unemploy, 2), data = economics)
poly_model2_3 <- glm(uempmed ~ poly(unemploy, 3), data = economics)
```

```
poly_model2_10 <- glm(uempmed ~ poly(unemploy, 10), data = economics)
```

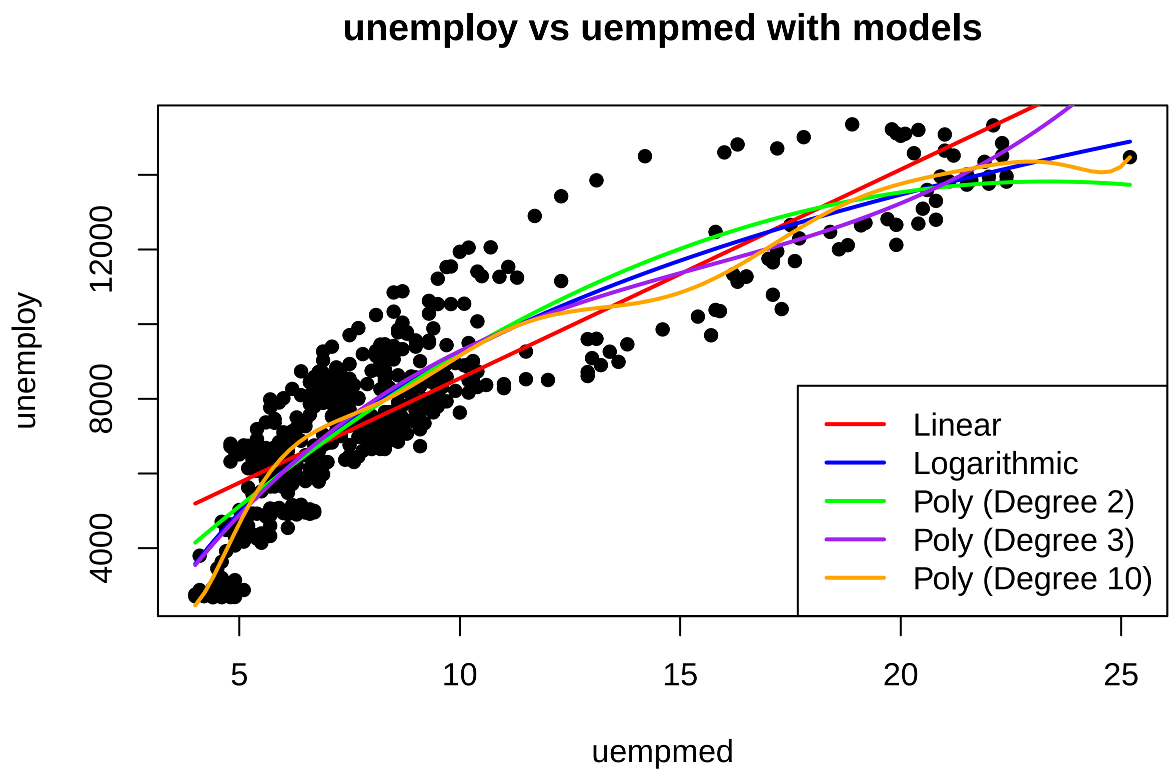
part 2

We create sequences to better plot smooth lines for the different models and then predict the values of the dependant variable using th etrained models respectively.

```
uempmed_seq <- seq(min(economics$uempmed), max(economics$uempmed), length.out = 100)
unemploy_seq <- seq(min(economics$unemploy), max(economics$unemploy), length.out = 100)

#unemploy ~ uempmed
pred_lin_model1 <-
  predict(lin_model1, newdata = data.frame(uempmed = uempmed_seq))
pred_log_model <-
  predict(log_model, newdata = data.frame(uempmed = uempmed_seq))
pred_poly1_2 <-
  predict(poly_model1_2, newdata = data.frame(uempmed = uempmed_seq))
pred_poly1_3 <-
  predict(poly_model1_3, newdata = data.frame(uempmed = uempmed_seq))
pred_poly1_10 <-
  predict(poly_model1_10, newdata = data.frame(uempmed = uempmed_seq))

#uempmed ~ unemploy
pred_lin_model2 <-
  predict(lin_model2, newdata = data.frame(unemploy = unemploy_seq))
pred_exp_model <-
  exp(predict(exp_model, newdata = data.frame(unemploy = unemploy_seq)))
#using exp() to get the back.transformed prediciotns for plotting
pred_poly2_2 <-
  predict(poly_model2_2, newdata = data.frame(unemploy = unemploy_seq))
pred_poly2_3 <-
  predict(poly_model2_3, newdata = data.frame(unemploy = unemploy_seq))
pred_poly2_10 <-
  predict(poly_model2_10, newdata = data.frame(unemploy = unemploy_seq))
```

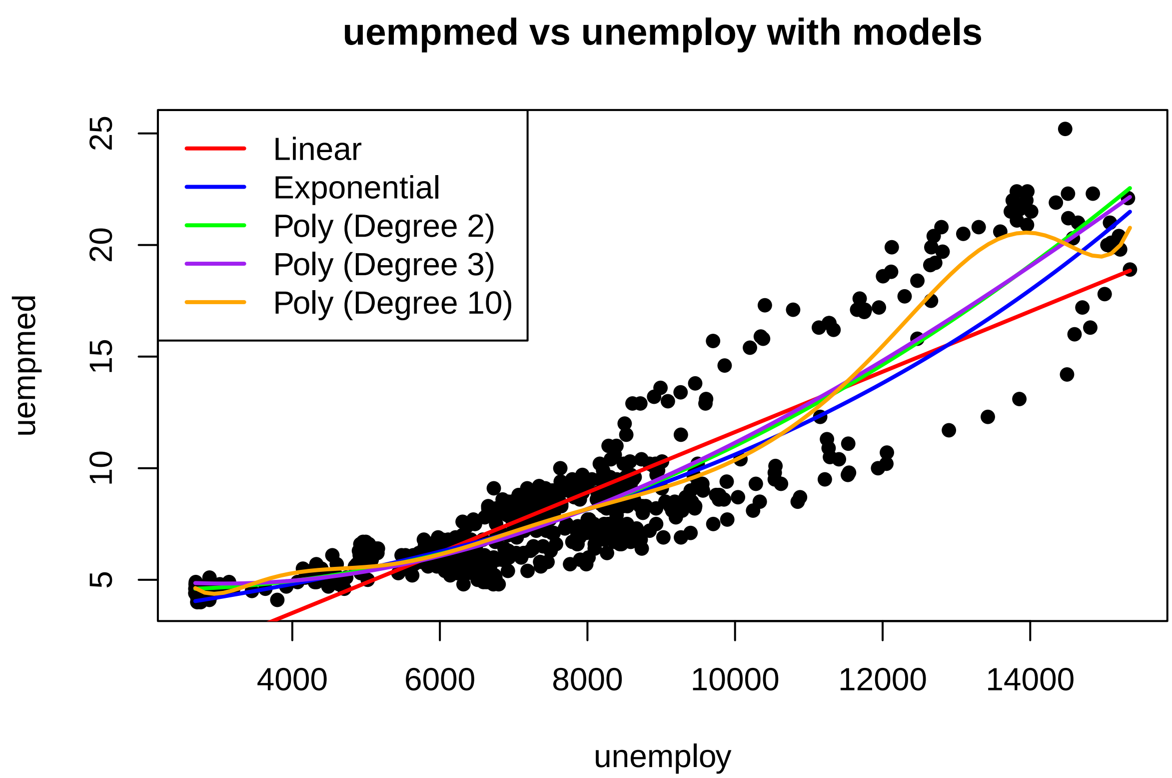
Now using the sequence for uempmed and predicted values for unemploy we plot the lines of the models into the scatterplot of the original data points.

```
plot(economics$uempmed, economics$unemploy,
     main = "unemploy vs uempmed with models",
     xlab = "uempmed",
     ylab = "unemploy",
     pch = 16, col = "black")

lines(uempmed_seq, pred_lin_model1, col = "red", lwd = 2, lty = 1)
lines(uempmed_seq, pred_log_model, col = "blue", lwd = 2, lty = 1)
lines(uempmed_seq, pred_poly1_2, col = "green", lwd = 2, lty = 1)
lines(uempmed_seq, pred_poly1_3, col = "purple", lwd = 2, lty = 1)
lines(uempmed_seq, pred_poly1_10, col = "orange", lwd = 2, lty = 1)

legend("bottomright", legend = c("Linear", "Logarithmic", "Poly (Degree 2)",
                                "Poly (Degree 3)", "Poly (Degree 10)"),
       col = c("red", "blue", "green", "purple", "orange"),
       lty = 1, lwd = 2)
```



We do the same plot but reversed for uempmed as predictions and the sequence of unemploy.

```
plot(economics$unemploy, economics$uempmed,
     main = "uempmed vs unemploy with models",
     xlab = "unemploy",
     ylab = "uempmed",
     pch = 16, col = "black")

lines(unemploy_seq, pred_lin_model2, col = "red", lwd = 2, lty = 1)
lines(unemploy_seq, pred_exp_model, col = "blue", lwd = 2, lty = 1)
lines(unemploy_seq, pred_poly2_2, col = "green", lwd = 2, lty = 1)
lines(unemploy_seq, pred_poly2_3, col = "purple", lwd = 2, lty = 1)
lines(unemploy_seq, pred_poly2_10, col = "orange", lwd = 2, lty = 1)

legend("topleft", legend = c("Linear", "Exponential", "Poly (Degree 2)",
                              "Poly (Degree 3)", "Poly (Degree 10)"),
       col = c("red", "blue", "green", "purple", "orange"),
       lty = 1, lwd = 2)
```



part 3

Now we start with the Leave-one-out Cross Validation for the created models. We iterate over all models for the Leave-one-out cv using cv.glm with the mse as cost function. The rmse then calculates as the squareroot of that value. For the exponential model we transform the resulting value for mse and rmse as previously explained using the approximation $MSE \approx exp(MSE_{log})$. We then obtain the results in a table.

```
models <- list(
  "linear model (unemploy ~ uempmed)" = lin_model1,
  "logarithmic model (unemploy ~ uempmed)" = log_model,
  "quadratic model (unemploy ~ uempmed)" = poly_model1_2,
  "cubic model (unemploy ~ uempmed)" = poly_model1_3,
  "poly(10)-model (unemploy ~ uempmed)" = poly_model1_10,
  "linear model (uempmed ~ unemploy)" = lin_model2,
  "approx. exp model (uempmed ~ unemploy)" = exp_model,
  "quadratic model (uempmed ~ unemploy)" = poly_model2_2,
  "cubic model (uempmed ~ unemploy)" = poly_model2_3,
  "poly(10)-model (uempmed ~ unemploy)" = poly_model2_10
)

results_loocv <- data.frame(Model = names(models), MSE = NA, RMSE = NA)

#loocv
for (i in seq_along(models)) {
  model <- models[[i]]
  mse_value <- cv.glm(economics, model,
```

```r
                      K = nrow(economics), cost = mse)$delta[2]
  rmse_value <- sqrt(mse_value)
  results_loocv[i, "MSE"] <- round(mse_value, 4)
  results_loocv[i, "RMSE"] <- round(rmse_value, 4)
}



#using the approximation
results_loocv[results_loocv$Model ==
                "approx. exp model (uempmed ~ unemploy)", "MSE"] <-
  exp(results_loocv[results_loocv$Model ==
                      "approx. exp model (uempmed ~ unemploy)", "MSE"])

results_loocv[results_loocv$Model ==
                "approx. exp model (uempmed ~ unemploy)", "RMSE"] <-
  exp(results_loocv[results_loocv$Model ==
                      "approx. exp model (uempmed ~ unemploy)", "RMSE"])

print(results_loocv)
```

```
##                                     Model       MSE     RMSE
## 1        linear model (unemploy ~ uempmed) 1.715e+06 1309.656
## 2   logarithmic model (unemploy ~ uempmed) 1.334e+06 1154.984
## 3     quadratic model (unemploy ~ uempmed) 1.433e+06 1196.878
## 4         cubic model (unemploy ~ uempmed) 1.366e+06 1168.919
## 5     poly(10)-model (unemploy ~ uempmed) 4.525e+06 2127.204
## 6        linear model (uempmed ~ unemploy) 4.160e+00    2.039
## 7   approx. exp model (uempmed ~ unemploy) 1.029e+00    1.184
## 8     quadratic model (uempmed ~ unemploy) 3.005e+00    1.734
## 9         cubic model (uempmed ~ unemploy) 3.010e+00    1.735
## 10     poly(10)-model (uempmed ~ unemploy) 2.832e+00    1.683
```

We can observe that for *unemploy* $\sim$ *uempmed* the logarithmic model has the smallest MSE/RMSE, with the quadratic and cubic models being only slightly worse. The linear model has a significantly higher MSE/RMSE and the poly(10)-model has way larger values.\ For *uempmed* $\sim$ *unemploy* the lowest values are for the approximated exponentil model, but these are most probably lower than they should be. Otherwise the poly(10)-model has the lowest RMSE/MSE values, with the quadratic and cubic model only being slightly worse. The linear model is again significanlty worse in terms of RMSE/MSE.\ Now we do the same again for 5-fold and 10-fold Cross Validation.

```r
results_cv <- data.frame(Model = names(models),
                         MSE_5fold = NA, RMSE_5fold = NA,
                         MSE_10fold = NA, RMSE_10fold = NA)

for (i in seq_along(models)) {
  model <- models[[i]]

  # 5-Fold cv
  mse_5fold <- cv.glm(economics, model, cost = mse, K = 5)$delta[2]
  rmse_5fold <- cv.glm(economics, model, cost = rmse, K = 5)$delta[2]

  #10-Fold cv
  mse_10fold <- cv.glm(economics, model, cost = mse, K = 10)$delta[2]
  rmse_10fold <- cv.glm(economics, model, cost = rmse, K = 10)$delta[2]
```

```r
  results_cv[i, "MSE_5fold"] <- round(mse_5fold, 4)
  results_cv[i, "RMSE_5fold"] <- round(rmse_5fold, 4)
  results_cv[i, "MSE_10fold"] <- round(mse_10fold, 4)
  results_cv[i, "RMSE_10fold"] <- round(rmse_10fold, 4)
}

#approximating the MSE/RMSE for the exponential model again
results_cv[results_cv$Model ==
             "approx. exp model (uempmed ~ unemploy)", "MSE_5fold"] <-
  exp(results_cv[results_cv$Model ==
             "approx. exp model (uempmed ~ unemploy)", "MSE_5fold"])
results_cv[results_cv$Model ==
             "approx. exp model (uempmed ~ unemploy)", "MSE_10fold"] <-
  exp(results_cv[results_cv$Model ==
             "approx. exp model (uempmed ~ unemploy)", "MSE_10fold"])

results_cv[results_cv$Model ==
             "approx. exp model (uempmed ~ unemploy)", "RMSE_5fold"] <-
  exp(results_cv[results_cv$Model ==
             "approx. exp model (uempmed ~ unemploy)", "RMSE_5fold"])
results_cv[results_cv$Model ==
             "approx. exp model (uempmed ~ unemploy)", "RMSE_10fold"] <-
  exp(results_cv[results_cv$Model ==
             "approx. exp model (uempmed ~ unemploy)", "RMSE_10fold"])

print(results_cv)
```

```
##                                     Model MSE_5fold RMSE_5fold MSE_10fold
## 1       linear model (unemploy ~ uempmed) 1.714e+06   1311.600  1.715e+06
## 2  logarithmic model (unemploy ~ uempmed) 1.330e+06   1152.584  1.334e+06
## 3    quadratic model (unemploy ~ uempmed) 1.427e+06   1193.606  1.429e+06
## 4        cubic model (unemploy ~ uempmed) 1.351e+06   1170.535  1.365e+06
## 5     poly(10)-model (unemploy ~ uempmed) 2.241e+06   2338.133  2.349e+06
## 6       linear model (uempmed ~ unemploy) 4.159e+00      2.033  4.153e+00
## 7  approx. exp model (uempmed ~ unemploy) 1.029e+00      1.184  1.029e+00
## 8    quadratic model (uempmed ~ unemploy) 3.005e+00      1.724  2.997e+00
## 9        cubic model (uempmed ~ unemploy) 3.001e+00      1.738  3.029e+00
## 10    poly(10)-model (uempmed ~ unemploy) 2.824e+00      1.674  2.844e+00
##    RMSE_10fold
## 1     1300.362
## 2     1153.135
## 3     1192.777
## 4     1164.857
## 5     1113.095
## 6        2.029
## 7        1.184
## 8        1.723
## 9        1.715
## 10       1.653
```

We can observe the same as for the leave-one-out cross validation: for *unemploy* $\sim$ *uempmed* the logarithmic model performs best, with the quadratic and cubic model both having only every slightly higher MSE/RMSE values. For *uempmed* $\sim$ *unemploy* the poly(10)-model has the lowest MSE/RMSE values, with the quadratic and cubic models being slightly worse. The approximated values of the exponential model are most probably too low, but also looking at the plot from part 2 the actual values should be very similar to those of the quadratic and cubic models. The linear model performs worse than all other models in terms of MSE/RMSE values.

part 4

All three Cross-validation methods, Leave-One_out, 5-fold and 10-fold result in similar insights and MSE/RMSE values, especially in regard to comparing the models. The only difference is that Leave-One-Out cv took noticeably longer, which makes sense sicne it has to test on $n = datasize$ different train/valid splits by always "leaving one observation out", training on the rest and predicting that one. For 5/10-fold cv there are only 5/10 train/valid splits. K-Fold cv is a robust technique for evaluation that balances bias and variance, with Leave-one-out being the highest possible and therefore giving the best result with no loss of data in training. The disadvantge though is that it is computationally very expensive, with the difference already being noticeable on this very small (574 x 6) dataset. Usually, as here was also the case, 5- or 10-fold cv give similar qualitative results and are therefore preferred.\ To explain under- and overfitting on this example, it is best to first look at the plots of the models with the datapoints from part 2. The points of data very obviously have some underlying non-linear trend that follows some curve for both response/predictor and its inverted version. The linear model performs worse than most other models in both cases. In the plots the linear models are obviously the straight lines. This strongly suggests underfitting, which means the model is too simple to capture the underlying trends and structures in the data since it can only ever fit a straight line between the data, that has a more complex structure in this case. This results in evaluation measures for both training and validation sets in general to not give good results. Next we look at the polynomial model of degree 10. For $uempmed \sim unemploy$ in actually performed best in terms of MSE/RMSE, but for $unemploy \sim uempmed$ it is by far the worst. Looking at the plots we also see while the data mostly follows some smooth curve, the line for the poly(10)-model is very wiggly and tries to fit close to some data points, others not. This model clearly overfits the data, which means that it is in opposite to the linear model too complex for the data and starts to capture noise or random fluctuations by trying to very closely fit the data. This results in a model very well-suited to the training data, but one that performs bad on new data because it fails to generalize to it and is too specific. As mentioned, for $uempmed \sim unemploy$ it still performed best. This can happen, because it was trained on the whole data and also cross-validated on the whole data, so there always is a chance that it also performs well in cv. That is why in general also the plots should be looked at, because as in this case we can immediately say even though MSE/RMSE are low, that the poly(10)-model is way too excessively complex and definitely overfits the data. In the plot for $unemploy \sim uempmed$ we can actually see that the cubic model also already overfits the data as on the margins it behaves strangely. Therefore the logarithmic or quadratic model should be chosen, preferably the logarithmic one. For $uempmed \sim unemploy$ all three models, the logarithmic, quadratic and cubic, seem a good choice in regards to the plot and the MSE/RMSE. Still when having a choice the most simple well-suited model should be chosen, so again it would be either the exponential or quadratic one.