Cross Validation of Models

Teodor Chakarov

2023-11-22

Contents

Task 1

Import the dataset.

```
library(ISLR)
data('Auto')

df <- Auto
head(df)
```

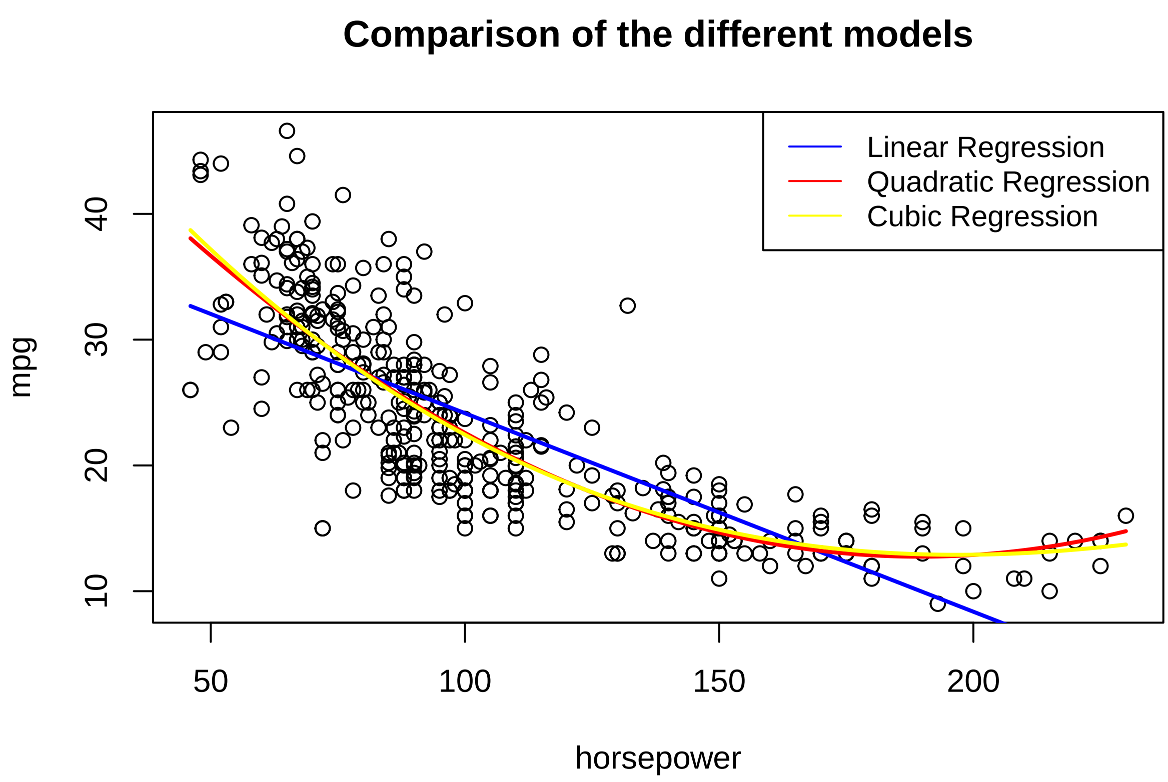
This is pretty standard dataset wich contains information about cars. The target variable is miles per gallon.

Fitting models

```
mod_1 <- lm(mpg ~ horsepower, data=df)
mod_2 <- lm(mpg ~ poly(horsepower, 2), data=df)
mod_3 <- lm(mpg ~ poly(horsepower, 3), data=df)
```

Visualize all 3 models in comparison added to a scatterplot of the input data.

```
x <- data.frame('horsepower'= seq(min(df$horsepower), max(df$horsepower), by=0.1))

y_1 <- predict(mod_1, x)
y_2 <- predict(mod_2, x)
y_3 <- predict(mod_3, x)

plot(mpg ~ horsepower, data=df, main='Comparison of the different models')
lines(x$horsepower, y_1, col='blue', type='l', lwd = 2)
lines(x$horsepower, y_2, col='red', type='l', lwd = 2)
lines(x$horsepower, y_3, col='yellow', type='l', lwd = 2)
legend('topright', legend=c("Linear Regression", "Quadratic Regression", "Cubic Regression"),
       col=c("blue", "red", "yellow"), lty=1, cex=.9)
```



We can see how the Linear regression underfits the data (dont describe the points), while the quadratic and cubic regression does better job.

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.

```
evaluate <- function(model, measure, position) {
  test <- df[-position,]
  pred <- predict(model, test)

  return(measure(test$mpg, pred))
}
```

```
mse <- function(true, pred) {
  return(mean((true - pred)^2))
}

rmse <- function(true, pred) {
  return(sqrt(mse(true, pred)))
}

median_div <- function(true, pred) {
  return(median(abs(true - pred)))
}
```

Lets do a 50% split of the data

```
n <- nrow(df)
set.seed(12141198)

data_size <- round(0.5 * n)

train_split <- sample(1:n, size=data_size)

reg1 <- lm(mpg ~ horsepower, data=df[train_split,])
reg2 <- lm(mpg ~ poly(horsepower,2), data=df[train_split,])
reg3 <- lm(mpg ~ poly(horsepower,3), data=df[train_split,])

table_50_split <- data.frame(
  Model=c("Linear Reg", "Quadratic Reg", "Cubic Reg"),
  MSE=c(evaluate(reg1, mse, train_split), evaluate(reg2, mse, train_split), evaluate(reg3, mse, train_sp
  RMSE=c(evaluate(reg1, rmse, train_split), evaluate(reg2, rmse, train_split), evaluate(reg3, rmse, trai
  MAD=c(evaluate(reg1, median_div, train_split), evaluate(reg2, median_div, train_split), evaluate(reg3

head(table_50_split)
```

```
##           Model      MSE     RMSE      MAD
## 1    Linear Reg 25.52431 5.052159 3.157644
## 2 Quadratic Reg 21.22420 4.606973 2.205394
## 3     Cubic Reg 21.58753 4.646238 2.166572
```

Lets do the 70% data split now

```r
set.seed(12141198)

split_size <- round(.7 * n)
train_split_70 <- sample(1:n, size=split_size)

train <- df[train_split_70, ]

reg1 <- lm(mpg ~ horsepower, data=train)
reg2 <- lm(mpg ~ poly(horsepower,2), data=train)
reg3 <- lm(mpg ~ poly(horsepower,3), data=train)

table_70_split <- data.frame(
  Model=c("Linear Reg", "Quadratic Reg", "Cubic Reg"),
  MSE=c(evaluate(reg1, mse, train_split_70), evaluate(reg2, mse, train_split_70), evaluate(reg3, mse, t
  RMSE=c(evaluate(reg1, rmse, train_split_70), evaluate(reg2, rmse, train_split_70), evaluate(reg3, rms
  MAD=c(evaluate(reg1, median_div, train_split_70),evaluate(reg2, median_div, train_split_70), evaluate
head(table_70_split)
```

```
##           Model      MSE     RMSE      MAD
## 1    Linear Reg 25.60915 5.060548 2.685296
## 2 Quadratic Reg 22.20382 4.712093 2.108627
## 3     Cubic Reg 22.96237 4.791906 2.224940
```

We see that the Quadratic and Cubic Regression give us similar results, although Quadratic Regression give us the best and the linear regression always perform worse than the two others.

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

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

```r
library(boot)
```

```
## Warning: package 'boot' was built under R version 4.2.3
```

```r
reg1 <- glm(mpg ~ horsepower, data=df)
reg2 <- glm(mpg ~ poly(horsepower,2), data=df)
reg3 <- glm(mpg ~ poly(horsepower,3), data=df)
```

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

```r
# Leave-One-Out
reg1_cv <- cv.glm(glmfit = reg1, data=df)$delta[1]
reg2_cv <- cv.glm(glmfit = reg2, data=df)$delta[1]
reg3_cv <- cv.glm(glmfit = reg3, data=df)$delta[1]

# 5-fold CV
reg_1_5_fold <- cv.glm(glmfit = reg1, data=df, K=5)$delta[1]
reg_2_5_fold <- cv.glm(glmfit = reg2, data=df, K=5)$delta[1]
reg_3_5_fold <- cv.glm(glmfit = reg3, data=df, K=5)$delta[1]
```

```r
# 10-fold CV
res_1_10_fold <- cv.glm(glmfit = reg1, data=df, K=10)$delta[1]
res_2_10_fold <- cv.glm(glmfit = reg2, data=df, K=10)$delta[1]
res_3_10_fold <- cv.glm(glmfit = reg3, data=df, K=10)$delta[1]
```

Compare all "mean squared error" results from 2 and 3. in a table and draw your conclusions.

```r
table <- data.frame(
  "model"=c("Linear Regression", "Quadratic Regression", "Cubic Regression"),
  "leave_one_out"=c(reg1_cv,reg2_cv,reg3_cv),
  "cv_5" = c(reg_1_5_fold,reg_2_5_fold,reg_3_5_fold),
  "cv_10" = c(res_1_10_fold,res_2_10_fold,res_3_10_fold)
)

head(table)
```

```
##                  model leave_one_out     cv_5    cv_10
## 1    Linear Regression      24.23151 24.43413 24.17281
## 2 Quadratic Regression      19.24821 19.27839 19.23121
## 3     Cubic Regression      19.33498 19.12190 19.49525
```

With the cross validation we can see that the Quadratic Regression performs slightly better than the other models. Cross validation also provides better result than the classical train test splitting techniques.

Task 2:

Load the data set 'df2' from the package 'ggplot2'.

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```r
df2 <- economics
head(df2)
```

```
## # A tibble: 6 x 6
##   date         pce    pop psavert uempmed unemploy
##   <date>     <dbl>  <dbl>   <dbl>   <dbl>    <dbl>
## 1 1967-07-01  507. 198712    12.6     4.5     2944
## 2 1967-08-01  510. 198911    12.6     4.7     2945
## 3 1967-09-01  516. 199113    11.9     4.6     2958
## 4 1967-10-01  512. 199311    12.9     4.9     3143
## 5 1967-11-01  517. 199498    12.8     4.7     3066
## 6 1967-12-01  525. 199657    11.8     4.8     3018
```

1 Fit the following models to explain the number of unemployed persons 'unemploy' by the median number of days unemployed 'uempmed' and vice versa:

- linear model

```r
lm_unemploy <- glm(unemploy ~ uempmed, data = df2)
summary(lm_unemploy)
```

```
##
## Call:
## glm(formula = unemploy ~ uempmed, data = df2)
##
## 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
```

Reversed:

```r
lm_uempmed <- glm(uempmed ~ unemploy, data = df2)
summary(lm_uempmed)
```

```
##
## Call:
## glm(formula = uempmed ~ unemploy, data = df2)
##
## 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
```

- an appropriate exponential or logarithmic model (which one is appropriate depends on which is the dependent or independent variable)

```
log_unemploy <- glm(unemploy ~ log(uempmed), data = df2)
summary(log_unemploy)
```

```
##
## Call:
## glm(formula = unemploy ~ log(uempmed), data = df2)
##
## 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
```

Reversed:

```
log_uempmed <- glm(uempmed ~ log(unemploy), data = df2)
summary(log_uempmed)
```

```
##
## Call:
## glm(formula = uempmed ~ log(unemploy), data = df2)
##
## 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
```

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

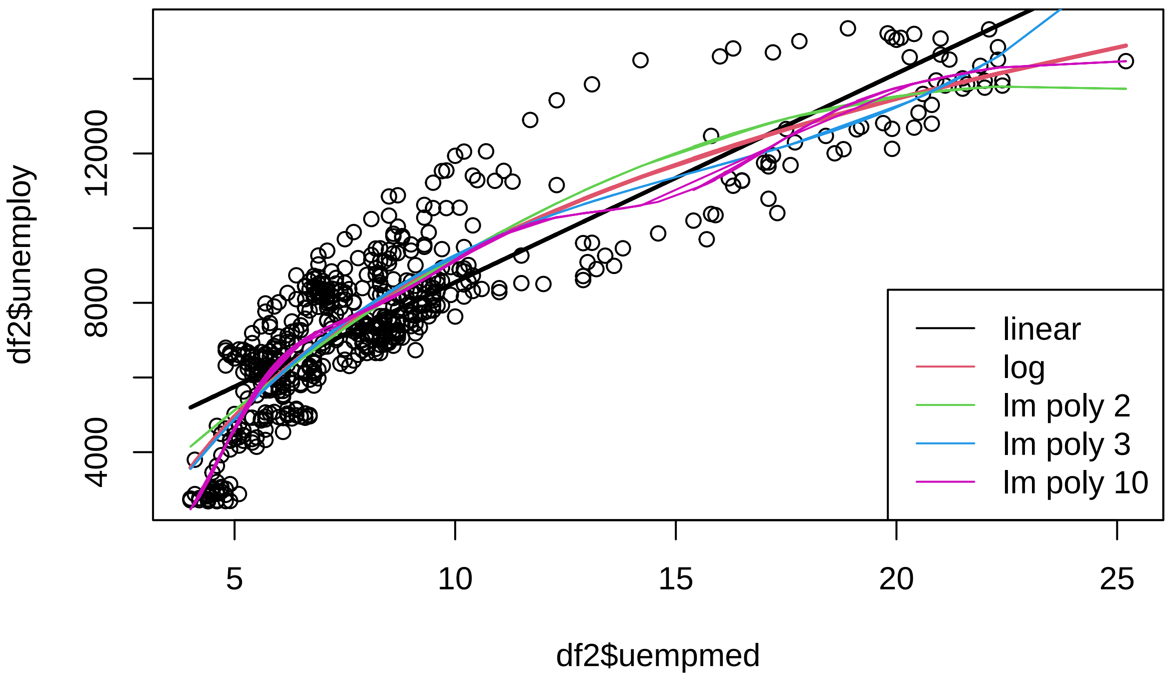
```
lr_poly_2_unemploy <- glm(unemploy ~ poly(uempmed,2), data = df2)
lr_poly_3_unemploy <- glm(unemploy ~ poly(uempmed,3), data = df2)
lr_poly_10_unemploy <- glm(unemploy ~ poly(uempmed,10), data = df2)
```

```
# Reversed:
lr_poly_2_uempmed <- glm(uempmed ~ poly(unemploy,2), data = df2)
lr_poly_3_uempmed <- glm(uempmed ~ poly(unemploy,3), data = df2)
lr_poly_10_uempmed <- glm(uempmed ~ poly(unemploy,10), data = df2)
```

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

Plot models to predict umemploy:

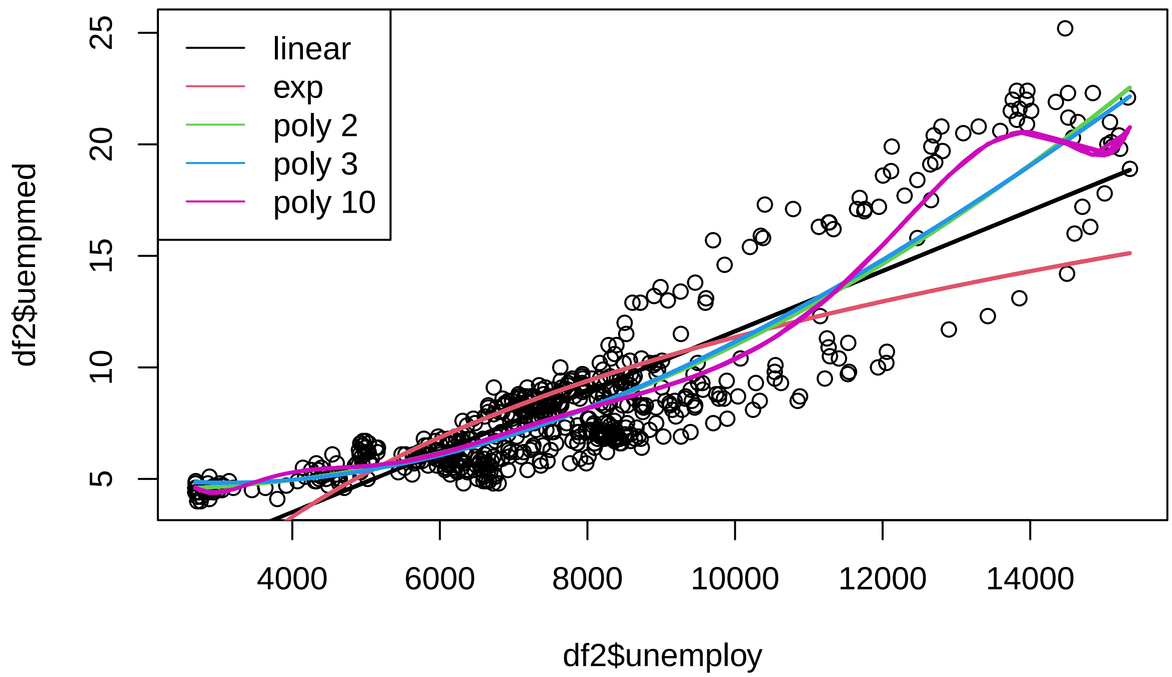
```
plot(df2$uempmed, df2$unemploy)
lines(df2$uempmed, fitted(lm_unemploy), col=1, lwd=2)
lines(df2$uempmed, fitted(log_unemploy), col=2, lwd=2)

lines(df2$uempmed, fitted(lr_poly_2_unemploy), col=3, lwd=1)
lines(df2$uempmed, fitted(lr_poly_3_unemploy), col=4, lwd=1)
lines(df2$uempmed, fitted(lr_poly_10_unemploy), col=6, lwd=1)
legend("bottomright",legend=c("linear","log","lm poly 2","lm poly 3","lm poly 10"),col=c(1,2,3,4,6), lty
```



We can see how those models try to describe the data on the most opitmal way. Linear regression just gives us one line which under fits the data. While others are almost similar.

Plot models to predict uempmed:

```
plot(df2$unemploy, df2$uempmed)
lines(df2$unemploy, fitted(lm_uempmed), col=1, lwd=2)
lines(df2$unemploy, fitted(log_uempmed), col=2, lwd=2)
lines(df2$unemploy, fitted(lr_poly_2_uempmed), col=3, lwd=2)
lines(df2$unemploy, fitted(lr_poly_3_uempmed), col=4, lwd=2)
lines(df2$unemploy, fitted(lr_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)
```



Here we can see how the Ploynomial of 10th describes the data ecpacially at the end. The curve represents overfitting and learning the noise of the data in order to represent it perfeclty, which we don't want, because the test data wont be so representative.

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

Predict unemploy variable:

```
get_comparison_unemploy <- function(data, folds){
  cv_lm_unemploy <- cv.glm(data, glm(lm_unemploy), K=folds)
  cv_log_unemploy <- cv.glm(data, glm(log_unemploy), K=folds)
  cv_poly_2_unemploy <- cv.glm(data, glm(lr_poly_2_unemploy), K=folds)
  cv_poly_3_unemploy <- cv.glm(data, glm(lr_poly_3_unemploy), K=folds)
  cv_poly_10_unemploy <- cv.glm(data, glm(lr_poly_10_unemploy), K=folds)

  data.frame(
  "Model" = c("linear", "exponential", "poly 2", "poly 3", "poly 10"),
  "MSE" = c(cv_lm_unemploy$delta[1],cv_log_unemploy$delta[1],cv_poly_2_unemploy$delta[1], cv_poly_3_unem
  "RMSE" = c(sqrt(cv_lm_unemploy$delta[1]),sqrt(cv_log_unemploy$delta[1]),sqrt(cv_poly_2_unemploy$delta
}
```

1. Leave-one-out Cross Validation

```r
get_comparison_unemploy(df2, nrow(df2))
```

```
##         Model     MSE     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. 5-fold and 10-fold Cross Validation

5-fold CV

```r
get_comparison_unemploy(df2, 5)
```

```
##         Model     MSE     RMSE
## 1      linear 1719603 1311.336
## 2 exponential 1344336 1159.455
## 3      poly 2 1426608 1194.407
## 4      poly 3 1361551 1166.855
## 5     poly 10 1278393 1130.660
```

10-fold CV

```r
get_comparison_unemploy(df2, 10)
```

```
##         Model     MSE     RMSE
## 1      linear 1711176 1308.119
## 2 exponential 1336331 1155.998
## 3      poly 2 1429521 1195.626
## 4      poly 3 1372846 1171.685
## 5     poly 10 1271673 1127.685
```

Conclusions

Based on the results we have, we can see that the polynomial of 10th regresson has the small amount or error, but in leave-one-out we have surprisingly high score for this model. But in general, the errors are pretty similar in all of the cross-validation techniques. We have big RMSE error, in order to tackle this problem, maybe we can aslo scale our predictor variables in order to gett more stable results.

Predict uempmed varable:

```r
get_comparison_uempmed <- function(data, folds){
  cv_lm_uempmed <- cv.glm(data, glm(lm_uempmed), K=folds)
  cv_log_uempmed <- cv.glm(data, glm(log_uempmed), K=folds)
  cv_poly_2_uempmed <- cv.glm(data, glm(lr_poly_2_uempmed), K=folds)
  cv_poly_3_uempmed <- cv.glm(data, glm(lr_poly_3_uempmed), K=folds)
  cv_poly_10_uempmed <- cv.glm(data, glm(lr_poly_10_uempmed), K=folds)
```

```r
  data.frame(
  "Model" = c("linear", "exponential", "poly 2", "poly 3", "poly 10"),
  "MSE" = c(cv_lm_uempmed$delta[1],cv_log_uempmed$delta[1],cv_poly_2_uempmed$delta[1], cv_poly_3_uempmed
  "RMSE" = c(sqrt(cv_lm_uempmed$delta[1]),sqrt(cv_log_uempmed$delta[1]),sqrt(cv_poly_2_uempmed$delta[1]
}
```

1. Leave-one-out Cross Validation

```r
get_comparison_uempmed(df2, nrow(df2))
```

```
##         Model      MSE     RMSE
## 1      linear 4.159797 2.039558
## 2 exponential 6.998858 2.645536
## 3      poly 2 3.005112 1.733526
## 4      poly 3 3.009934 1.734916
## 5     poly 10 2.832303 1.682945
```

2. 5-fold and 10-fold Cross Validation ### 5-fold CV

```r
get_comparison_uempmed(df2, 5)
```

```
##         Model      MSE     RMSE
## 1      linear 4.148894 2.036884
## 2 exponential 6.966429 2.639399
## 3      poly 2 3.036947 1.742684
## 4      poly 3 3.009897 1.734905
## 5     poly 10 2.950388 1.717669
```

10-fold CV

```r
get_comparison_uempmed(df2, 10)
```

```
##         Model      MSE     RMSE
## 1      linear 4.155698 2.038553
## 2 exponential 6.968887 2.639865
## 3      poly 2 3.000170 1.732100
## 4      poly 3 3.012904 1.735772
## 5     poly 10 2.850736 1.688412
```

Here we have smaller error and as we can see again polynomial regressin 10 does the best job, even though th eoders perform also pretty well. The leave-one-out and the 10-fold Cv gives pretty similar results.

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.

When a model i being train on specific data, the model "setts up" based on the data points. When the model is set not in a optimal state we have under fitting, which gives us a lot of bias. In the second task we can see it for a **unemploy** variable, where we have a hudge error. That means the model is not in its optimal form and not describes the data as much. To concuere this problem we will need either more complex model or more attributes which describes well the points. Overfitting in other hand is when the model describes too well the data in it fits perfectly to a lot of the points.In this case we have more variance which comes from the learned noice on the training data set. It usually gives use good results on RMSE or MSE. The plynomial regression of 10th gives us the best results and this is because the model is coplex and will describe the data better. To overcome this problem we can use more data or cross-validation. This method will give use knowleadge when the models starts to overfit. The methods for CV are following: Every data point is used for evaluation one time. The evaluation of the models can be done with a percentage of the data which is called k-fold cross validation. The data is split in k folds in this case. It can be done with just one datapoint for evaluation in each iteration which is called leave one out cross validation.