Assignment 6: Cross Validation of Models

Statistical Simulation and Computerintensive Methods

Joan Salvà

November 2022

Task 1

We will work with the dataset Auto in the ISLR package. Obtain information on the data set, its structure and the real world meaning of its variables from the help page.

```r
library(ISLR)

data('Auto')
Auto
df <- Auto
rm(Auto)
```

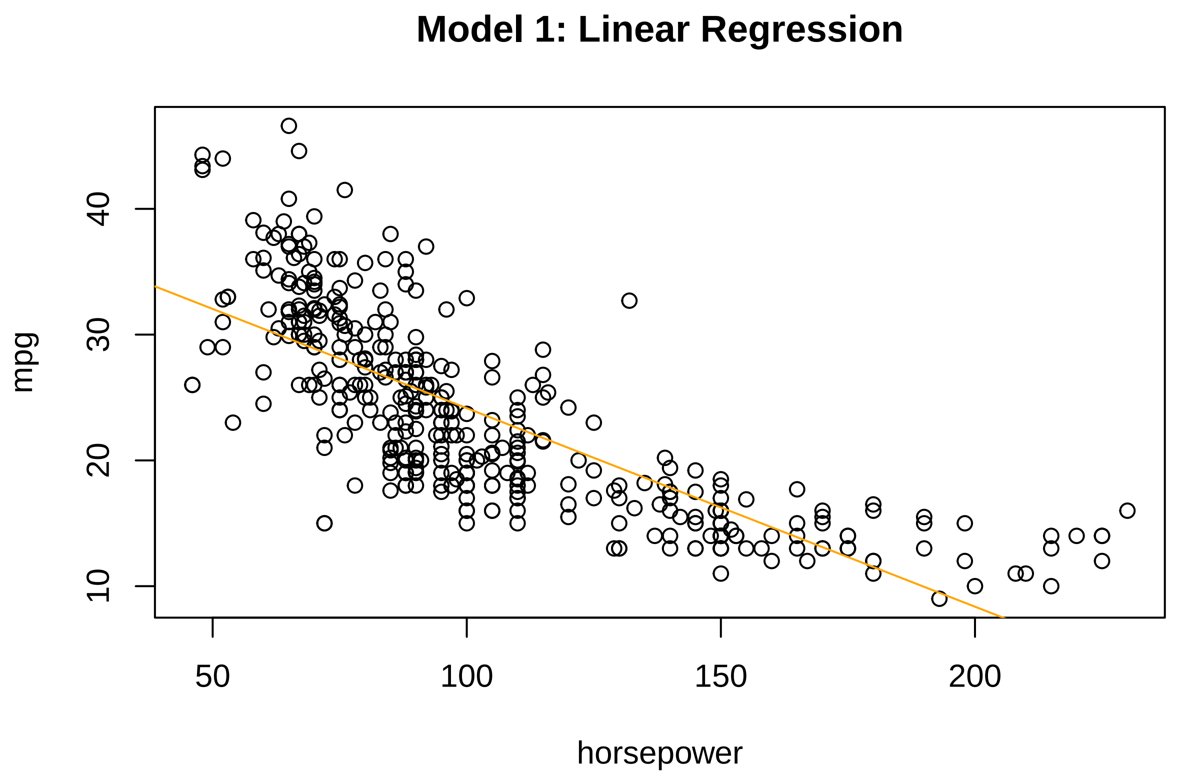
The dataset covers miles per gallon, cylinders, displacement, horsepower, weight, acceleration, year, origin of 397 cars.

The aim of the analysis is to answer the following question: Is miles per gallon (mpg) a function of the remaining variables in the dataset?

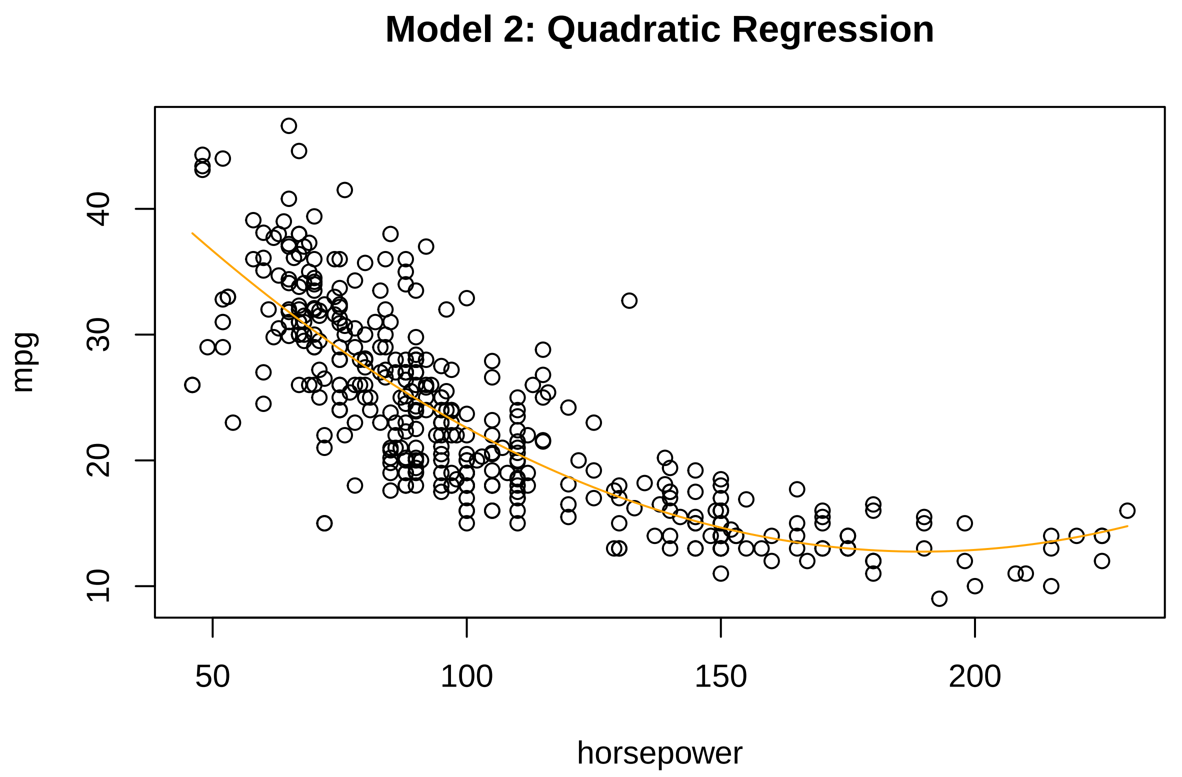
1. Fit the following models:

- mpg ~ horsepower
- mpg ~ poly(horsepower,2)
- mpg ~ poly(horsepower,3)

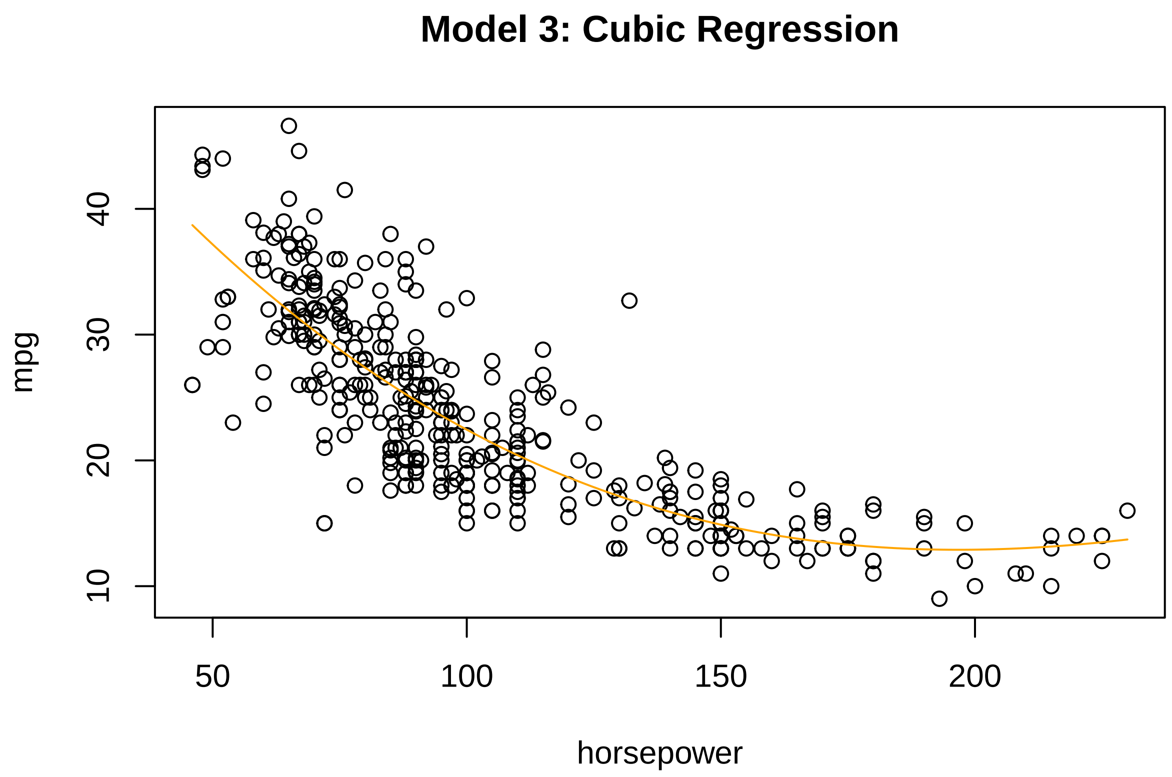
```r
mod.1 <- lm(mpg ~ horsepower, data=df)
mod.2 <- lm(mpg ~ poly(horsepower,2), data=df)
mod.3 <- lm(mpg ~ poly(horsepower,3), data=df)

plot(mpg ~ horsepower, data=df, main='Model 1: Linear Regression')
abline(mod.1, col='orange')
```



```
min_hpw <- min(df$horsepower)
max_hpw <- max(df$horsepower)

x <- data.frame('horsepower' =seq(min_hpw, max_hpw, by=0.1))
y.2 <- predict(mod.2, x)
y.3 <- predict(mod.3, x)

plot(mpg ~ horsepower, data=df, main='Model 2: Quadratic Regression')
lines(x$horsepower, y.2, col='orange', type='l')
```



```
plot(mpg ~ horsepower, data=df, main='Model 3: Cubic Regression')
lines(x$horsepower, y.3, col='orange', type='l')
```



2. 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.

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

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

mad <- function(actual, pred) {
  return(median(abs(actual - pred)))
}

evaluate_model <- function(model, measure, test) {
  test <- df[-train_index,]
  pred <- predict(model, test)

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

```r
n <- nrow(df)
```

```r
# 50% train / 50% test split
train_index <- sample(1:n, size=round(.5 * n))

train <- df[train_index, ]
test <- df[-train_index, ]

mod.1 <- lm(mpg ~ horsepower, data=train)
mod.2 <- lm(mpg ~ poly(horsepower,2), data=train)
mod.3 <- lm(mpg ~ poly(horsepower,3), data=train)

table_50 <- data.frame(
  Model=c('LM', 'Poly 2', 'Poly 3'),
  MSE=c(
    evaluate_model(mod.1, mse, test),
    evaluate_model(mod.2, mse, test),
    evaluate_model(mod.3, mse, test)),
  RMSE=c(
    evaluate_model(mod.1, rmse, test),
    evaluate_model(mod.2, rmse, test),
    evaluate_model(mod.3, rmse, test)),
  MAD=c(
    evaluate_model(mod.1, mad, test),
    evaluate_model(mod.2, mad, test),
    evaluate_model(mod.3, mad, test))
)
knitr::kable(table_50, caption = "Errors on 50% test")
```

Table 1: Errors on 50% test

Model	MSE	RMSE	MAD
LM	22.20996	4.712745	3.086428
Poly 2	15.89362	3.986681	2.319928
Poly 3	15.90245	3.987788	2.397439

```r
# 70% train / 30% test split
train_index <- sample(1:n, size=round(.7 * n))

train <- df[train_index, ]
test <- df[-train_index, ]

mod.1 <- lm(mpg ~ horsepower, data=train)
mod.2 <- lm(mpg ~ poly(horsepower,2), data=train)
mod.3 <- lm(mpg ~ poly(horsepower,3), data=train)

table_70 <- data.frame(
  Model=c('LM', 'Poly 2', 'Poly 3'),
  MSE=c(
    evaluate_model(mod.1, mse, test),
    evaluate_model(mod.2, mse, test),
    evaluate_model(mod.3, mse, test)),
  RMSE=c(
    evaluate_model(mod.1, rmse, test),
    evaluate_model(mod.2, rmse, test),
    evaluate_model(mod.3, rmse, test)),
  MAD=c(
    evaluate_model(mod.1, mad, test),
    evaluate_model(mod.2, mad, test),
    evaluate_model(mod.3, mad, test))
)
knitr::kable(table_70, caption = "Errors on 70% test")
```

Table 2: Errors on 70% test

Model	MSE	RMSE	MAD
LM	21.28294	4.613343	2.868659
Poly 2	17.33708	4.163781	2.248334
Poly 3	17.26779	4.155453	2.233020

The three measures of error over the two different splits agree on the best model, which is the Quadratic Regressor. The second best model is slightly worse and it is the Cubic Regressor. Due to the distribution that we can see in the scatterplot, it was obvious that a linear model had to perform way worse.

3. 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.
- Use `cv.glm` for 5-fold and 10-fold Cross Validation to compare the models above.

We train again the models with the complete dataset. We know that `glm` is equivalent to `lm` when we do not specify any family of distributions.

```
library(boot)

mod.1 <- glm(mpg ~ horsepower, data=df)
mod.2 <- glm(mpg ~ poly(horsepower,2), data=df)
mod.3 <- glm(mpg ~ poly(horsepower,3), data=df)


# Leave-one-out CV
res.1 <- cv.glm(glmfit = mod.1, data=df)$delta[1]
res.2 <- cv.glm(glmfit = mod.2, data=df)$delta[1]
res.3 <- cv.glm(glmfit = mod.3, data=df)$delta[1]

# 5-fold CV
res.1_5 <- cv.glm(glmfit = mod.1, data=df, K=5)$delta[1]
res.2_5 <- cv.glm(glmfit = mod.2, data=df, K=5)$delta[1]
res.3_5 <- cv.glm(glmfit = mod.3, data=df, K=5)$delta[1]

# 10-fold CV
res.1_10 <- cv.glm(glmfit = mod.1, data=df, K=10)$delta[1]
res.2_10 <- cv.glm(glmfit = mod.2, data=df, K=10)$delta[1]
res.3_10 <- cv.glm(glmfit = mod.3, data=df, K=10)$delta[1]
```

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

```r
table_cv <- data.frame(
  Model=c('LM', 'Poly 2', 'Poly 3'),
  smth=c(res.1, res.2, res.3),
  smth2=c(res.1_5, res.2_5, res.3_5),
  smth3=c(res.1_10, res.2_10, res.3_10)
)
names(table_cv) = c('Model', 'Leave-One-Out CV', '5 fold - CV', '10 fold - CV')

knitr::kable(table_50, caption = '50% test errors')
```

Table 3: 50% test errors

Model	MSE	RMSE	MAD
LM	22.20996	4.712745	3.086428
Poly 2	15.89362	3.986681	2.319928
Poly 3	15.90245	3.987788	2.397439

```r
knitr::kable(table_70, caption = '70% test errors')
```

Table 4: 70% test errors

Model	MSE	RMSE	MAD
LM	21.28294	4.613343	2.868659
Poly 2	17.33708	4.163781	2.248334
Poly 3	17.26779	4.155453	2.233020

```r
knitr::kable(table_cv, caption = "Cross validation errors")
```

Table 5: Cross validation errors

Model	Leave-One-Out CV	5 fold - CV	10 fold - CV
LM	24.23151	24.27432	24.17225
Poly 2	19.24821	19.54096	19.15678
Poly 3	19.33498	19.60439	19.30950

As mentioned in 2, the measures RMSE, MSE and MAD agree on the best model: Model 2. When it comes to the cross-validation error, Leave-One-Out and 10-fold CV suggests that Mod 2 is better. However, 5-fold CV suggests Mod 3 is slighly better.

If we had to choose a model, Model 2 would be the one because it has the best error in almost all measures.

Task 2

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

```r
library(ggplot2)

df <- economics
```

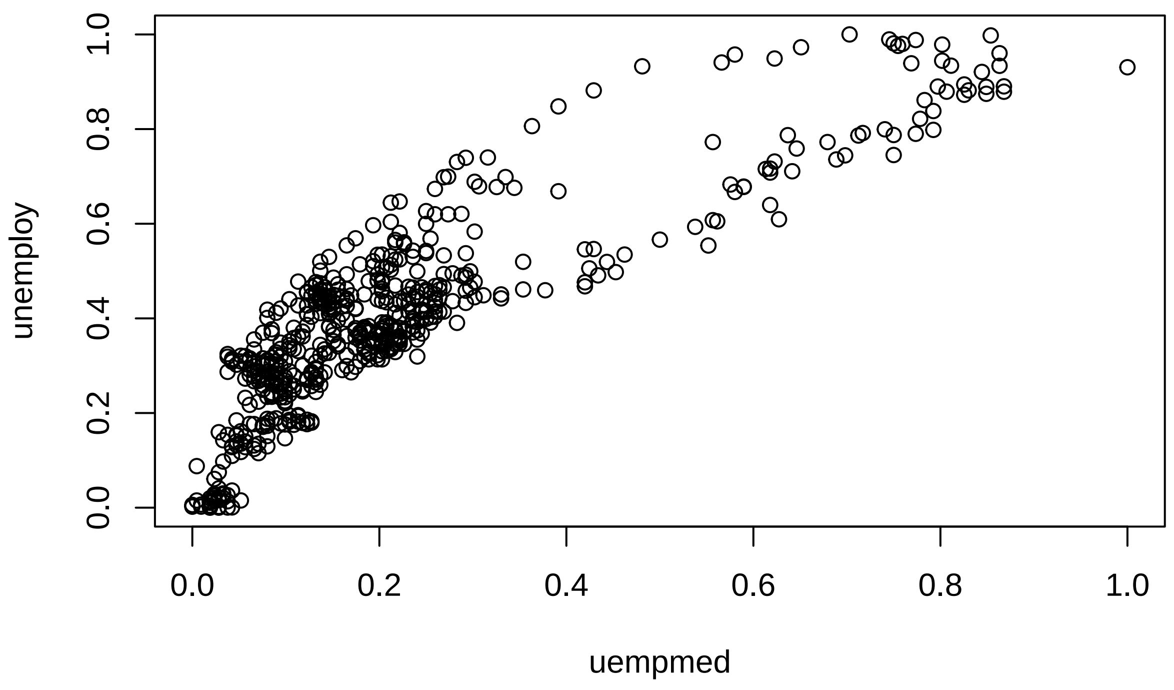
1. Fit the following models to explain the number of unemployed persons 'unemploy' by the median number of days unemployed 'uempmed' and vice versa using:

- Linear model
- An appropriate exponential or logarithmic model (which one is appropriate depends on which is the dependent or independent variable)
- Polynomial model of 2nd, 3rd and 10th degree

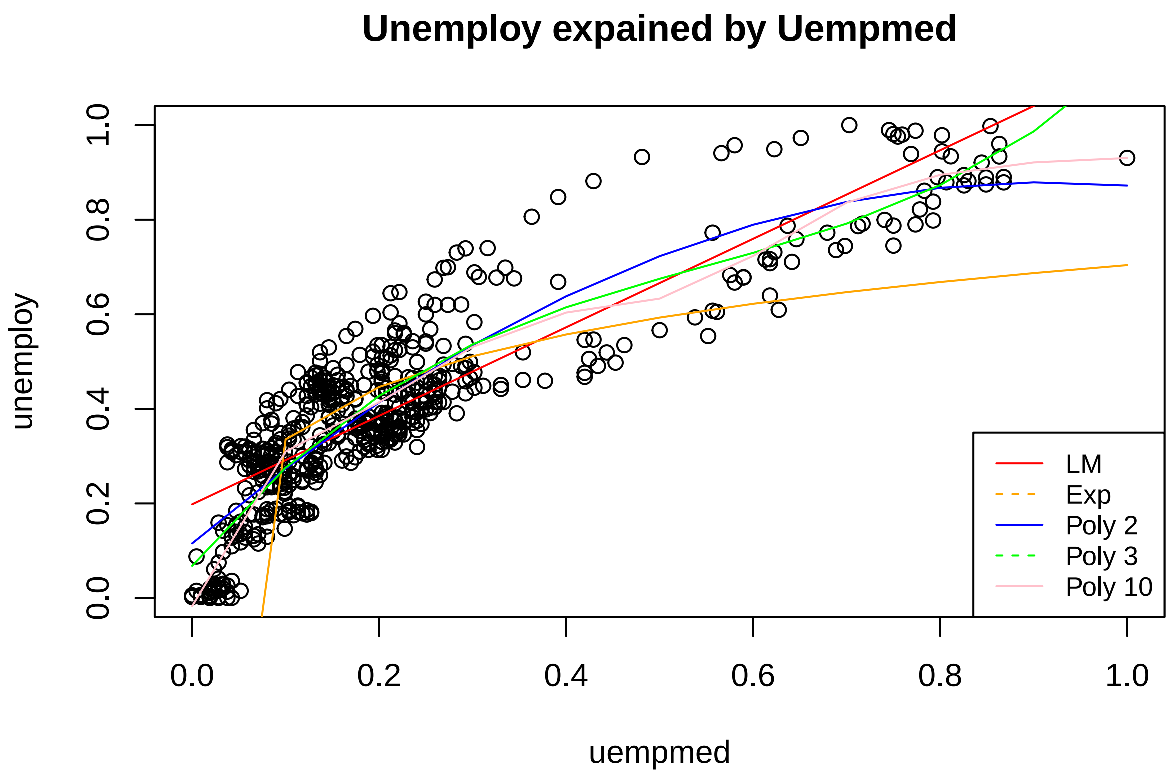
2. Plot the corresponding data and add all the models for comparison.

We will first Min/Max scale the two variables so the errors are in an easier to interpret scale.

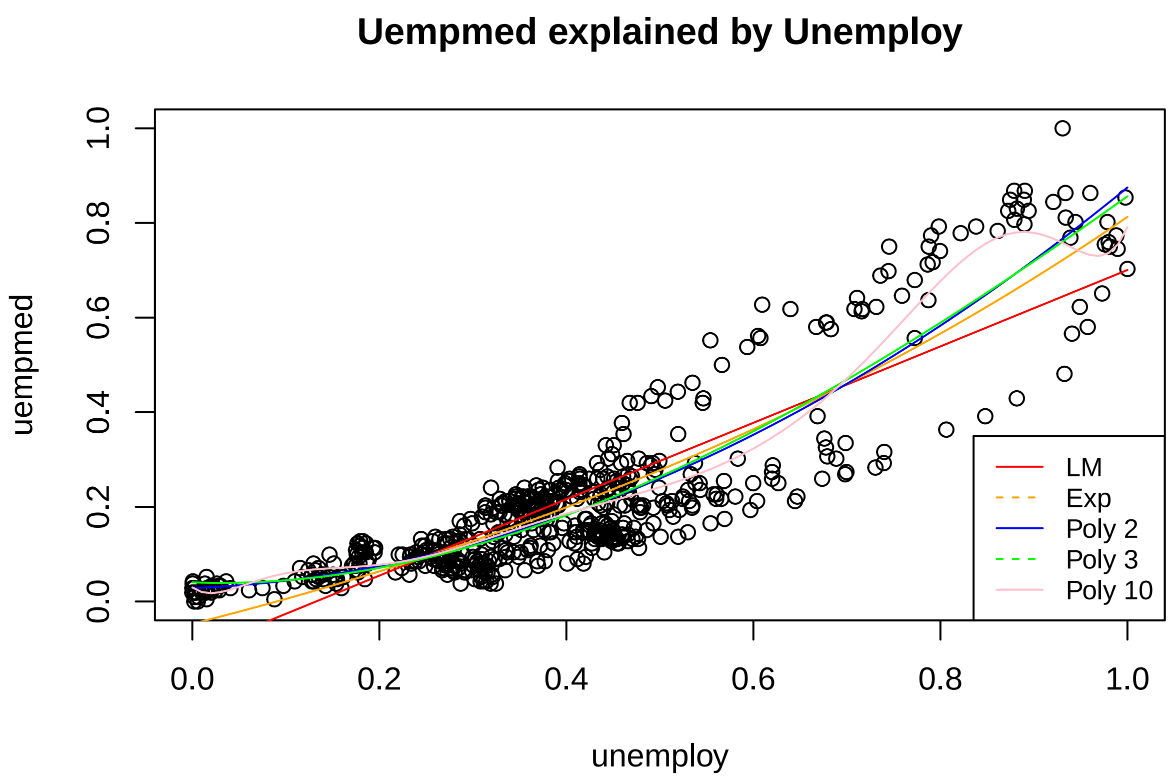
```r
eps <- 0.00001
df$unemploy <- (df$unemploy - min(df$unemploy)) / (max(df$unemploy) - min(df$unemploy)) + eps
df$uempmed <- (df$uempmed - min(df$uempmed)) / (max(df$uempmed) - min(df$uempmed)) + eps

# Unemploy explained by uempmed
plot(unemploy ~ uempmed, data = df)
```



```r
mod.1 <- glm(unemploy ~ uempmed, data = df)
mod.2 <- glm(unemploy ~ log(uempmed), data = df)
mod.3 <- glm(unemploy ~ poly(uempmed, 2), data = df)
mod.4 <- glm(unemploy ~ poly(uempmed, 3), data = df)
mod.5 <- glm(unemploy ~ poly(uempmed, 10), data = df)

min_uempmed <- min(df$uempmed)
max_uempmed <- max(df$uempmed)

x <- data.frame('uempmed' = seq(min_uempmed, max_uempmed, by=0.1))
y.1 <- predict(mod.1, x)
y.2 <- predict(mod.2, x)
y.3 <- predict(mod.3, x)
y.4 <- predict(mod.4, x)
y.5 <- predict(mod.5, x)

plot(unemploy ~ uempmed, data = df, main = 'Unemploy expained by Uempmed')
lines(x$uempmed, y.1, col = 'red')
lines(x$uempmed, y.2, col = 'orange')
lines(x$uempmed, y.3, col = 'blue')
lines(x$uempmed, y.4, col = 'green')
lines(x$uempmed, y.5, col = 'pink')
legend("bottomright", legend=c("LM", "Exp", 'Poly 2', 'Poly 3', 'Poly 10'),
       col=c("red", "orange", 'blue', 'green', 'pink'), lty=1:2, cex=0.8)
```



```r
# Uempmed explained by unemploy
modinv.1 <- glm(uempmed ~ unemploy, data = df)
modinv.2 <- glm(uempmed ~ exp(unemploy), data = df)
modinv.3 <- glm(uempmed ~ poly(unemploy, 2), data = df)
modinv.4 <- glm(uempmed ~ poly(unemploy, 3), data = df)
modinv.5 <- glm(uempmed ~ poly(unemploy, 10), data = df)

min_unemploy <- min(df$unemploy)
max_unemploy <- max(df$unemploy)

x <- data.frame('unemploy' = seq(min_unemploy, max_unemploy, by=0.01))
y.1 <- predict(modinv.1, x)
y.2 <- predict(modinv.2, x)
y.3 <- predict(modinv.3, x)
y.4 <- predict(modinv.4, x)
y.5 <- predict(modinv.5, x)

plot(uempmed ~ unemploy, data = df, main = 'Uempmed explained by Unemploy')
lines(x$unemploy, y.1, col = 'red')
lines(x$unemploy, y.2, col = 'orange')
lines(x$unemploy, y.3, col = 'blue')
lines(x$unemploy, y.4, col = 'green')
lines(x$unemploy, y.5, col = 'pink')
legend("bottomright", legend=c("LM", "Exp", 'Poly 2', 'Poly 3', 'Poly 10'),
       col=c("red", "orange", 'blue', 'green', 'pink'), lty=1:2, cex=0.8)
```



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

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

```r
cv_error <- function(mod, folds) {
  res <- cv.glm(data = df, glmfit = mod, K = folds)$delta[1]
  return(res)
}

table <- data.frame(
  Model=c('Mod 1', 'Mod 2', 'Mod 3', 'Mod 4', 'Mod 5',
          'ModInv 1', 'ModInv 2', 'ModInv 3', 'ModInv 4',
          'ModInv 5'),
  cv_leave=c(
    cv_error(mod.1, folds=nrow(df)),
    cv_error(mod.2, folds=nrow(df)),
    cv_error(mod.3, folds=nrow(df)),
    cv_error(mod.4, folds=nrow(df)),
    cv_error(mod.5, folds=nrow(df)),
    cv_error(modinv.1, folds=nrow(df)),
    cv_error(modinv.2, folds=nrow(df)),
    cv_error(modinv.3, folds=nrow(df)),
    cv_error(modinv.4, folds=nrow(df)),
    cv_error(modinv.5, folds=nrow(df))
  ),
  cv_5=c(
    cv_error(mod.1, folds=5),
    cv_error(mod.2, folds=5),
    cv_error(mod.3, folds=5),
    cv_error(mod.4, folds=5),
    cv_error(mod.5, folds=5),
    cv_error(modinv.1, folds=5),
    cv_error(modinv.2, folds=5),
    cv_error(modinv.3, folds=5),
    cv_error(modinv.4, folds=5),
    cv_error(modinv.5, folds=5)
  ),
  cv_10=c(
    cv_error(mod.1, folds=10),
    cv_error(mod.2, folds=10),
    cv_error(mod.3, folds=10),
    cv_error(mod.4, folds=10),
    cv_error(mod.5, folds=10),
    cv_error(modinv.1, folds=10),
    cv_error(modinv.2, folds=10),
    cv_error(modinv.3, folds=10),
    cv_error(modinv.4, folds=10),
    cv_error(modinv.5, folds=10)
  )
)
names(table) <- c('Model', 'Leave-one-out CV', '5-Fold CV', '10-fold CV')
knitr::kable(table, caption='Crossvalidation Errors')
```

Table 6: Crossvalidation Errors

Model	Leave-one-out CV	5-Fold CV	10-fold CV
Mod 1	0.0106898	0.0107904	0.0106735
Mod 2	0.0186571	0.0182603	0.0228126
Mod 3	0.0089280	0.0089062	0.0089234
Mod 4	0.0085159	0.0086703	0.0084911
Mod 5	0.0282372	0.1716716	0.0132104
ModInv 1	0.0092555	0.0092622	0.0092555
ModInv 2	0.0072736	0.0072912	0.0072183
ModInv 3	0.0066863	0.0066513	0.0066712
ModInv 4	0.0066971	0.0066827	0.0067053
ModInv 5	0.0063018	0.0062646	0.0061753

Mod_i corresponds to uempmed ~ unemploy and *ModInv_i* corresponds to unemploy.

We look now at the cross-validation errors of the table above. *Mod 4* is the one among *Mod_i* with the least error in the three different CV techniques. It makes sense that it's a cubic model when we look at the scatterplot of the data. A quadratic model would also fit the data well and it's actually the second best.

Among the *ModInv_i*, the best one is *ModInv 3*, corresponding to the quadratic model. As mentioned above, that goes with the shape of the distribution that we visualized in the scatterplot.

4. 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: Underfitting happens when a model is too simple and is not ablte to capture most of the information of the training data. In this case, the test error would be large because the model 'has not learned enough'. On the other hand, overfitting happens when the model has captured too much information of the training data, and struggles to generalize on the test data. In this case, the test error would also be large, but because the model 'has learned to much'.

In essence, we would like the model to learn enough, but not too much. There are many ways in which one could try to avoid overfitting. - Remove outliers so the model does not learn noisy data - Stop training when generalization (test error) increases again. This makes sense for algorithms like Neural Networks. - Tune the hyperparameters only looking at the training data.

How can cross-validation help? Cross-validation reduces the 'randomness' of the train / test split by resampling method and using different portions of the data to test and train a model on different iterations.

The number of folds is the number of parts in which we divide the dataset, and at each iteration, the model will be trained in k-1 folds and tested on the other. Sometimes, cross-validation is done multiple times to reduce even more the random effect of the k-folds split.

The cross-validation score is widely used as a measure to determine the best hyperparameters of a ML model.

Specially in classification problems, one is sometimes interested in keepeng the distribution of some variables similar over different folds. For instance, it would make sense to do it on the target class when there's an important class umbalance. That is called stratified k-fold cross-validation.