Exercise n°6: Cross Validation of Models

Advanced Methods for Regression and Classification

Grégoire de Lambertye

2022-11-28

Task n°1: Import data

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.

```
library(ISLR)

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

This dataset contains information on cars such as their power, year, miles per gallons … The target variable is mpg (miles per gallon).

I.1 Fitting modls

```
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)
```

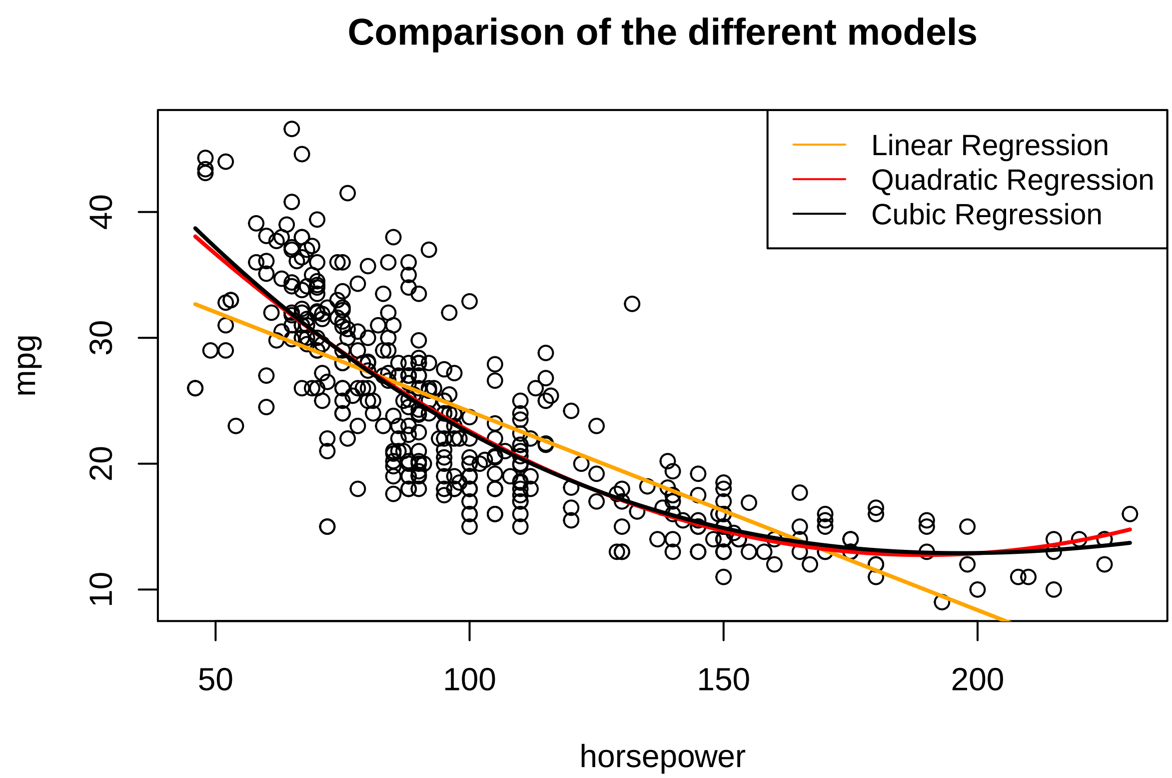
Visualize all 3 models in comparison added to a scatterplot of the input data.

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

x <- data.frame('horsepower' =seq(min_hpw, max_hpw, 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$horsepower, y.1, col='orange', type='l', lwd = 2)
lines(x$horsepower, y.2, col='red', type='l', lwd = 2)
lines(x$horsepower, y.3, col='black', type='l', lwd = 2)
legend('topright', legend=c("Linear Regression", "Quadratic Regression", "Cubic Regression"),
       col=c("orange", "red", "black"), lty=1, cex=.9)
```



I.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.

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

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

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

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

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

```r
n <- nrow(df)
Matricul.number = 12202211

# 50% train / 50% test split
set.seed(Matricul.number)
train_index_50 <- sample(1:n, size=round(.5 * n))

mod.1 <- lm(mpg ~ horsepower, data=df[train_index_50,])
mod.2 <- lm(mpg ~ poly(horsepower,2), data=df[train_index_50,])
mod.3 <- lm(mpg ~ poly(horsepower,3), data=df[train_index_50,])

table_50 <- data.frame(
  Model=c("Linear Regression", "Quadratic Regression", "Cubic Regression"),
  MSE=c(
    evaluate(mod.1, mse, train_index_50),
    evaluate(mod.2, mse, train_index_50),
    evaluate(mod.3, mse, train_index_50)),
  RMSE=c(
    evaluate(mod.1, rmse, train_index_50),
    evaluate(mod.2, rmse, train_index_50),
    evaluate(mod.3, rmse, train_index_50)),
  MAD=c(
    evaluate(mod.1, mad, train_index_50),
    evaluate(mod.2, mad, train_index_50),
    evaluate(mod.3, mad, train_index_50))
)
knitr::kable(table_50, caption = "Metrics on 50% test")
```

Table 1: Metrics on 50% test

Model	MSE	RMSE	MAD
Linear Regression	29.04838	5.389655	3.147822
Quadratic Regression	22.65724	4.759963	2.473978
Cubic Regression	22.64266	4.758430	2.553549

```r
# 70% train / 30% test split
set.seed(Matricul.number)
train_index_70 <- sample(1:n, size=round(.7 * n))

train <- df[train_index_70, ]

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("Linear Regression", "Quadratic Regression", "Cubic Regression"),
  MSE=c(
    evaluate(mod.1, mse, train_index_70),
    evaluate(mod.2, mse, train_index_70),
    evaluate(mod.3, mse, train_index_70)),
  RMSE=c(
```

```
    evaluate(mod.1, rmse, train_index_70),
    evaluate(mod.2, rmse, train_index_70),
    evaluate(mod.3, rmse, train_index_70)),
  MAD=c(
    evaluate(mod.1, mad, train_index_70),
    evaluate(mod.2, mad, train_index_70),
    evaluate(mod.3, mad, train_index_70))
)
knitr::kable(table_70, caption = "Metrics on 70% test")
```

Table 2: Metrics on 70% test

Model	MSE	RMSE	MAD
Linear Regression	32.31173	5.684341	3.198054
Quadratic Regression	26.97820	5.194055	2.884166
Cubic Regression	27.00346	5.196486	2.807517

With the metrics we can see that the Quadratic Regression and Cubic Regression have similar result. Cubic Regression performs better for MSE and RMSE with the 50% split and for the MAD with the 70% split. In the other cases Quadratic Regression is the best and the linear regression always perform worse than the two others.

I.3 Use the cv.glm function in the boot package for the following steps.

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

We retrain the models with the complete dataset.

```
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 (K=default)
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]
```

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

```
# 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]
```

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

```
table <- data.frame(
  Model=c("Linear Regression", "Quadratic Regression", "Cubic Regression"),
  Leave_one_out=c(res.1,res.2,res.3),
  Five_CV=c(res.1_5,res.2_5,res.3_5),
  Ten_CV=c(res.1_10,res.2_10,res.3_10),
  Train_test_70=c(
    evaluate(mod.1, mse, train_index_70),
    evaluate(mod.2, mse, train_index_70),
    evaluate(mod.3, mse, train_index_70)
    ),
  Train_test_50=c(
    evaluate(mod.1, mse, train_index_50),
    evaluate(mod.2, mse, train_index_50),
    evaluate(mod.3, mse, train_index_50)
    )
)

knitr::kable(table, caption='Model comparison using mse metric')
```

Table 3: Model comparison using mse metric

Model	Leave_one_out	Five_CV	Ten_CV	Train_test_70	Train_test_50
Linear Regression	24.23151	24.33997	24.11031	30.91362	27.32818
Quadratic Regression	19.24821	19.28747	19.22981	26.08094	21.10301
Cubic Regression	19.33498	19.41610	19.36683	26.10123	21.08110

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 n°2:

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

```
library(ggplot2)

df <- economics
df
```

```
## # A tibble: 574 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
```

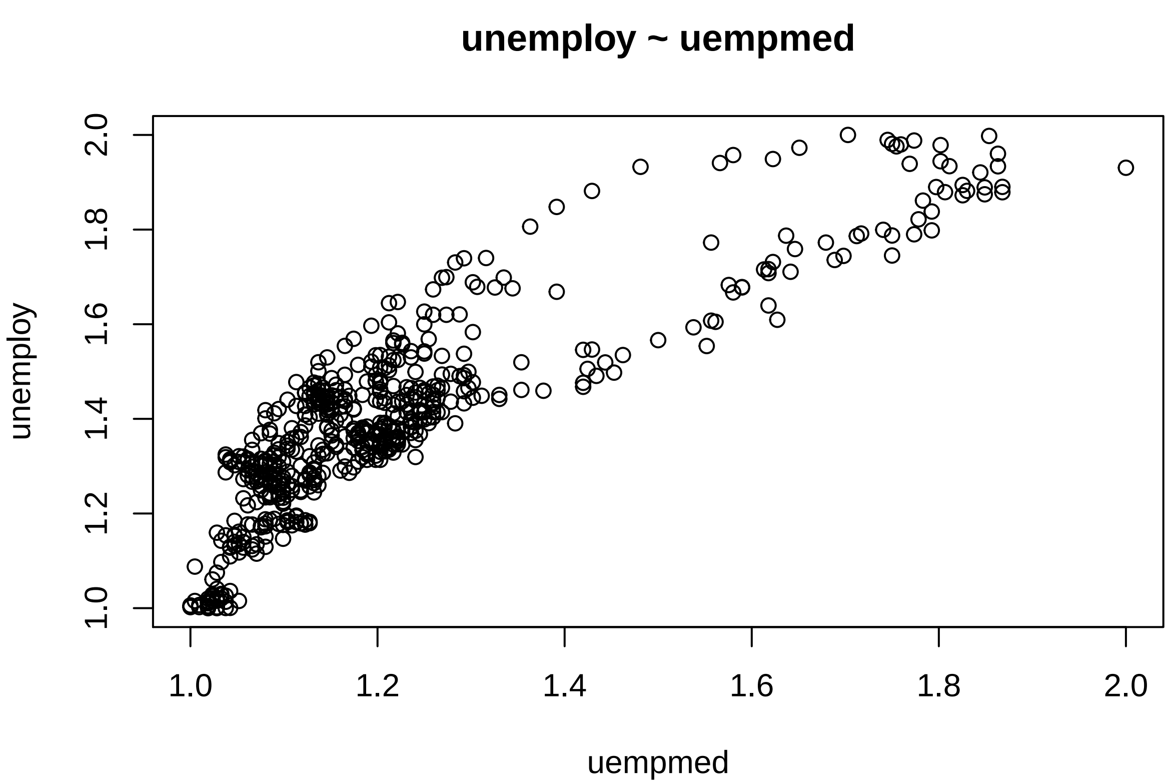
```
##   7 1968-01-01  531. 199808    11.7      5.1      2878
##   8 1968-02-01  534. 199920    12.3      4.5      3001
##   9 1968-03-01  544. 200056    11.7      4.1      2877
##  10 1968-04-01  544  200208    12.3      4.6      2709
## # ... with 564 more rows
```

II.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
- 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

We will scale and transformed our data to use the regressor in an easier way. Adding one to both variables will not impact the prediction in any way but avoid the problem of log(0).

```
eps <- 1
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
plot(unemploy ~ uempmed, data = df, main='unemploy ~ uempmed')
```



```
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)

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

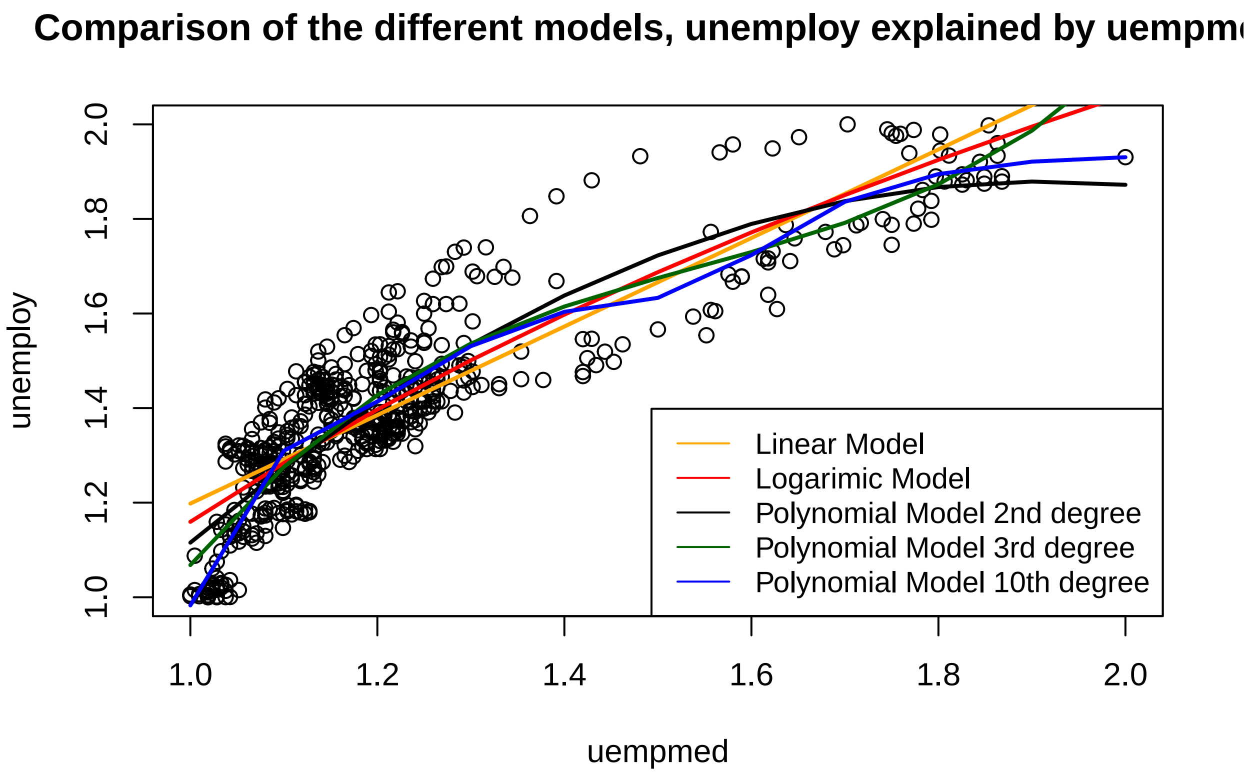
II.2 Plot the corresponding data and add all the models for comparison

```
#unemploy explained by uempmed
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='Comparison of the different models, unemploy explained by uempm
lines(x$uempmed, y.1, col='orange', type='l', lwd = 2)
lines(x$uempmed, y.2, col='red', type='l', lwd = 2)
lines(x$uempmed, y.3, col='black', type='l', lwd = 2)
lines(x$uempmed, y.4, col='darkgreen', type='l', lwd = 2)
lines(x$uempmed, y.5, col='blue', type='l', lwd = 2)
legend('bottomright', legend=c("Linear Model", "Logarimic Model", "Polynomial Model 2nd degree",
                              "Polynomial Model 3rd degree", "Polynomial Model 10th degree"),
       col=c("orange", "red", "black", 'darkgreen', 'blue'),
       lty=1, cex=.9)
```

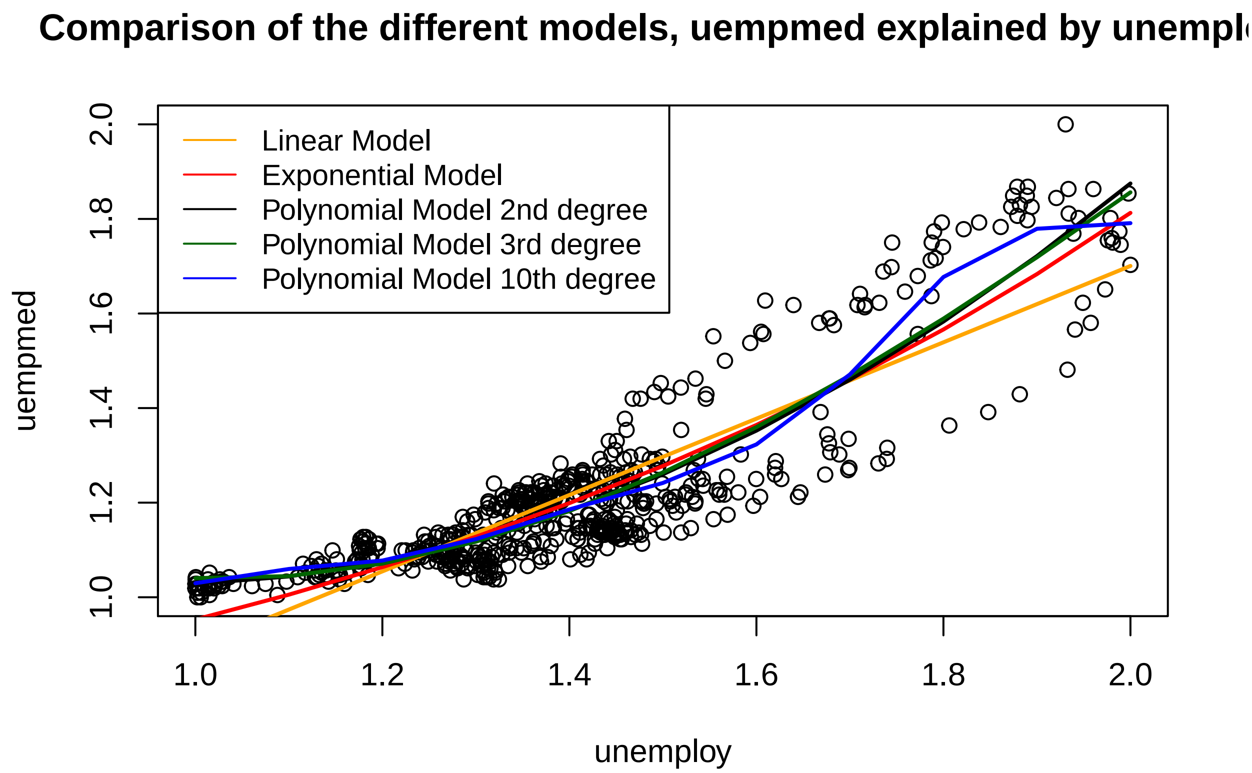



```
#uempmed explained by unemploy
min_unemploy <- min(df$unemploy)
max_unemploy <- max(df$unemploy)

x <- data.frame('unemploy' =seq(min_unemploy, max_unemploy, by=0.1))

y.1.inv <- predict(mod.1.inv, x)
y.2.inv <- predict(mod.2.inv, x)
y.3.inv <- predict(mod.3.inv, x)
y.4.inv <- predict(mod.4.inv, x)
y.5.inv <- predict(mod.5.inv, x)

plot(uempmed ~ unemploy, data=df, main='Comparison of the different models, uempmed explained by unempl
lines(x$unemploy, y.1.inv, col='orange', type='l', lwd = 2)
lines(x$unemploy, y.2.inv, col='red', type='l', lwd = 2)
lines(x$unemploy, y.3.inv, col='black', type='l', lwd = 2)
lines(x$unemploy, y.4.inv, col='darkgreen', type='l', lwd = 2)
lines(x$unemploy, y.5.inv, col='blue', type='l', lwd = 2)
legend('topleft', legend=c("Linear Model", "Exponential Model", "Polynomial Model 2nd degree",
                           "Polynomial Model 3rd degree", "Polynomial Model 10th degree"),
       col=c("orange", "red", "black", 'darkgreen', 'blue'),
       lty=1, cex=.9)
```



II.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.

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

table1 <- data.frame(
  Model=c("Linear Model", "Exponential Model", "Polynomial Model 2nd degree",
                              "Polynomial Model 3rd degree", "Polynomial Model 10th degree"),
  cv_loo_mse=c(
    cv_error_mse(mod.1.inv),
    cv_error_mse(mod.2.inv),
    cv_error_mse(mod.3.inv),
    cv_error_mse(mod.4.inv),
```

```
    cv_error_mse(mod.5.inv)
  ),
  cv_5_fold_mse=c(
    cv_error_mse(mod.1.inv, folds=5),
    cv_error_mse(mod.2.inv, folds=5),
    cv_error_mse(mod.3.inv, folds=5),
    cv_error_mse(mod.4.inv, folds=5),
    cv_error_mse(mod.5.inv, folds=5)
  ),
  cv_10_fold_mse=c(
    cv_error_mse(mod.1.inv, folds=10),
    cv_error_mse(mod.2.inv, folds=10),
    cv_error_mse(mod.3.inv, folds=10),
    cv_error_mse(mod.4.inv, folds=10),
    cv_error_mse(mod.5.inv, folds=10)
  )
)

table2 <- data.frame(
  Model=c("Linear Model", "Exponential Model", "Polynomial Model 2nd degree",
                              "Polynomial Model 3rd degree", "Polynomial Model 10th degree"),
   cv_loo_rmse=c(
    cv_error_rmse(mod.1.inv),
    cv_error_rmse(mod.2.inv),
    cv_error_rmse(mod.3.inv),
    cv_error_rmse(mod.4.inv),
    cv_error_rmse(mod.5.inv)
  ),
  cv_5_fold_rmse=c(
    cv_error_rmse(mod.1.inv, folds=5),
    cv_error_rmse(mod.2.inv, folds=5),
    cv_error_rmse(mod.3.inv, folds=5),
    cv_error_rmse(mod.4.inv, folds=5),
    cv_error_rmse(mod.5.inv, folds=5)
  ),
  cv_10_fold_rmse=c(
    cv_error_rmse(mod.1.inv, folds=10),
    cv_error_rmse(mod.2.inv, folds=10),
    cv_error_rmse(mod.3.inv, folds=10),
    cv_error_rmse(mod.4.inv, folds=10),
    cv_error_rmse(mod.5.inv, folds=10)
  )
)


table3 <- data.frame(
  Model=c("Linear Model", "Logaritmic Model", "Polynomial Model 2nd degree",
                              "Polynomial Model 3rd degree", "Polynomial Model 10th degree"),
  cv_loo_mse=c(
    cv_error_mse(mod.1),
    cv_error_mse(mod.2),
    cv_error_mse(mod.3),
    cv_error_mse(mod.4),
```

```
    cv_error_mse(mod.5)
  ),
  cv_5_fold_mse=c(
    cv_error_mse(mod.1, folds=5),
    cv_error_mse(mod.2, folds=5),
    cv_error_mse(mod.3, folds=5),
    cv_error_mse(mod.4, folds=5),
    cv_error_mse(mod.5, folds=5)
  ),
  cv_10_fold_mse=c(
    cv_error_mse(mod.1, folds=10),
    cv_error_mse(mod.2, folds=10),
    cv_error_mse(mod.3, folds=10),
    cv_error_mse(mod.4, folds=10),
    cv_error_mse(mod.5, folds=10)
  )
)

table4 <- data.frame(
  Model=c("Linear Model", "Logaritmic Model", "Polynomial Model 2nd degree",
                            "Polynomial Model 3rd degree", "Polynomial Model 10th degree"),
   cv_loo_rmse=c(
    cv_error_rmse(mod.1),
    cv_error_rmse(mod.2),
    cv_error_rmse(mod.3),
    cv_error_rmse(mod.4),
    cv_error_rmse(mod.5)
  ),
  cv_5_fold_rmse=c(
    cv_error_rmse(mod.1, folds=5),
    cv_error_rmse(mod.2, folds=5),
    cv_error_rmse(mod.3, folds=5),
    cv_error_rmse(mod.4, folds=5),
    cv_error_rmse(mod.5, folds=5)
  ),
  cv_10_fold_rmse=c(
    cv_error_rmse(mod.1, folds=10),
    cv_error_rmse(mod.2, folds=10),
    cv_error_rmse(mod.3, folds=10),
    cv_error_rmse(mod.4, folds=10),
    cv_error_rmse(mod.5, folds=10)
  )
)

knitr::kable(table3, caption='Crossvalidation Errors: unemploy explained by uempmed (mse)')
```

Table 4: Crossvalidation Errors: unemploy explained by uempmed (mse)

Model	cv_loo_mse	cv_5_fold_mse	cv_10_fold_mse
Linear Model	0.0106898	0.0107078	0.0106619
Logaritmic Model	0.0095006	0.0095301	0.0095128

Model	cv_loo_mse	cv_5_fold_mse	cv_10_fold_mse
Polynomial Model 2nd degree	0.0089280	0.0089282	0.0089211
Polynomial Model 3rd degree	0.0085159	0.0085102	0.0085004
Polynomial Model 10th degree	0.0282372	0.1192151	0.0109756

```
knitr::kable(table4, caption='Crossvalidation Errors: unemploy explained by uempmed (rmse)')
```

Table 5: Crossvalidation Errors: unemploy explained by uempmed (rmse)

Model	cv_loo_rmse	cv_5_fold_rmse	cv_10_fold_rmse
Linear Model	0.1033915	0.1032546	0.1033385
Logaritmic Model	0.0974710	0.0974622	0.0975003
Polynomial Model 2nd degree	0.0944883	0.0944542	0.0944511
Polynomial Model 3rd degree	0.0922818	0.0918037	0.0920179
Polynomial Model 10th degree	0.1680392	0.1714266	0.0888495

```
knitr::kable(table1, caption='Crossvalidation Errors: uempmed explained by unemploy (mse)')
```

Table 6: Crossvalidation Errors: uempmed explained by unemploy (mse)

Model	cv_loo_mse	cv_5_fold_mse	cv_10_fold_mse
Linear Model	0.0092555	0.0093564	0.0091800
Exponential Model	0.0072736	0.0072638	0.0072717
Polynomial Model 2nd degree	0.0066863	0.0066413	0.0066354
Polynomial Model 3rd degree	0.0066971	0.0066626	0.0068446
Polynomial Model 10th degree	0.0063018	0.0061933	0.0062847

```
knitr::kable(table2, caption='Crossvalidation Errors: uempmed explained by unemploy (rmse)')
```

Table 7: Crossvalidation Errors: uempmed explained by unemploy (rmse)

Model	cv_loo_rmse	cv_5_fold_rmse	cv_10_fold_rmse
Linear Model	0.0962056	0.0962822	0.0962787
Exponential Model	0.0852855	0.0850076	0.0850831
Polynomial Model 2nd degree	0.0817701	0.0823604	0.0822358
Polynomial Model 3rd degree	0.0818357	0.0819228	0.0817946
Polynomial Model 10th degree	0.0793842	0.0800100	0.0797495

Most of the time, the best model to predict unemploy is the Polynomial Model 3rd degree. Except for the 10 fold cross validation where the Polynomial Model 10th degree is better with the metrics root mean squared error. To predict uempmed the Polynomial Model 10th degree is always the best.

II.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.

Underfitting and overfitting are characteristics that can be associated with a model. As we know, a model is trained on a data set to learn to predict a target. Underfitting happens when the model did not learn enough and just simplifies the insight contained in the data. Overfitting happend when the model learned too much and is too complex for the prediction task. Both will lead to bad predictions. In the very first plot, the linear model underfit the data when the quadratic model overfit them a little bit.

How can cross-validation help? Cross-validation is a splitting technique that consist of dividing the data set in k folders and then to use k-1 folder to train a model and 1 folder as a validation set. The model will be trained k times and will give a better view on the global performance of the model. If k is equal to the number of observations, the cross validation is called 'Leave one out' cross validation. It should be the best possible indicator but requier lot's of time and computation. Cross vlidation is used to reduce overfitting and most of the time to determine the best hyperparameters of a machine learning algorithm.