Advanced Methods for Regression and Classification

Exercise 8

Bosse Behrens , st.id: 12347333

First we load the packages.

```
library(gclus)
```

```
## Lade nötiges Paket: cluster
```

```
data(ozone)
```

Now we split the data into train and test sets.

```
set.seed(1234)
n <- nrow(ozone)
train <- sample(1:n, round((2/3) * n))
test<-(1:n)[-train]

train_x <- ozone[train, "Temp"]
train_y <- ozone[train, "Ozone"]

test_x <- ozone[test, "Temp"]
test_y <- ozone[test, "Ozone"]
```

We implement the lecturespl function from the lecture.

```
lecturespl<-function(x,nknots=2,M=4){

  n<-length(x)
  X<-matrix(NA,nrow=n,ncol=(M-1)+nknots)

  for(i in 1:(M-1)){X[,i]<-x^i}

  quant<-seq(0,1,1/(nknots+1))[c(2:(nknots+1))]

  qu<-quantile(x,quant)

  for(i in M:(M+nknots-1)){

    X[,i]<-ifelse(x-qu[i-M+1]<0,0,(x-qu[i-M+1])^(M-1))
  }

  list(X=X,quantiles=quant,xquantiles=qu)

}
```

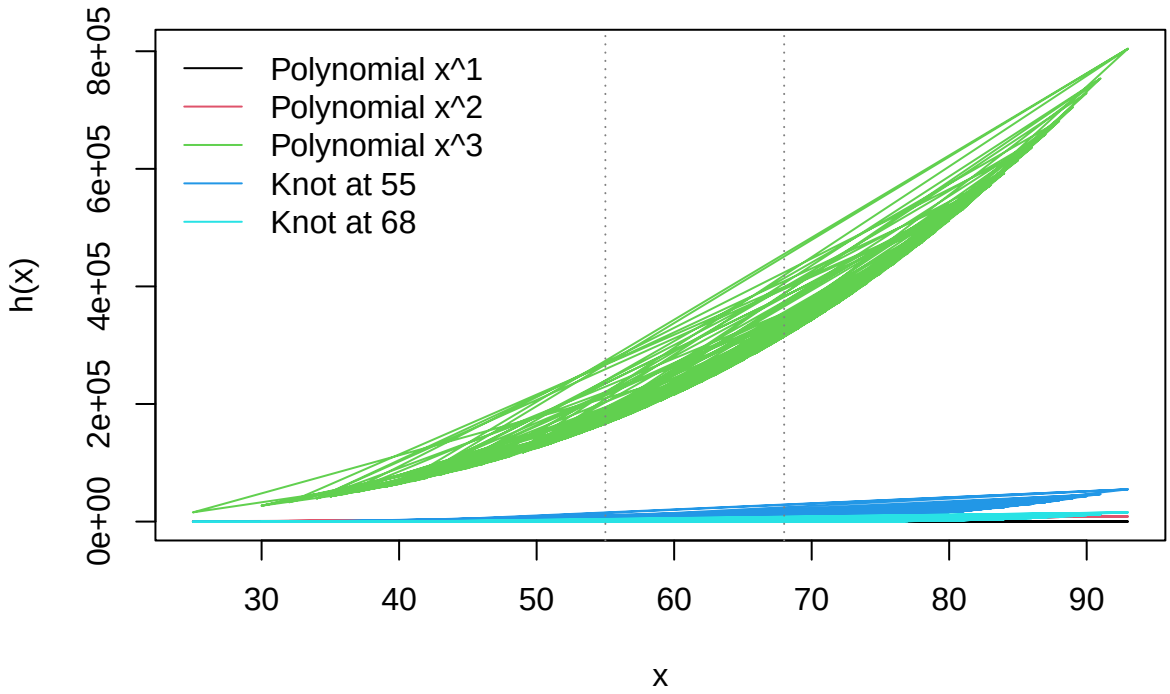
Task 1

We use the lecturespl function with 2 knots and M = 4 to get the splines for the training set. We then use the provided function to plot these basis functions. The plot fdunction is expanded to also include a legend.

```
spline <- lecturespl(train_x, 2, 4)

str(spline)
```

```
## List of 3
##  $ X         : num [1:220, 1:5] 39 47 56 62 46 65 83 48 55 58 ...
##  $ quantiles : num [1:2] 0.333 0.667
##  $ xquantiles: Named num [1:2] 55 68
##   ..- attr(*, "names")= chr [1:2] "33.33333%" "66.66667%"
```

```
plotspl <- function(splobj,x,...){
  matplot(x,splobj$X,type="l",lty=1,
          xlab="x",ylab="h(x)",...)
  abline(v=splobj$xquantiles,lty=3,col=gray(0.5))

  legend("topleft", legend = c(
    paste0("Polynomial x^", 1:(ncol(splobj$X) - length(splobj$xquantiles))),
    paste0("Knot at ", round(splobj$xquantiles, 2))
  ), col = 1:ncol(splobj$X), lty = 1, bty = "n")
}

plotspl(spline, train_x)
```



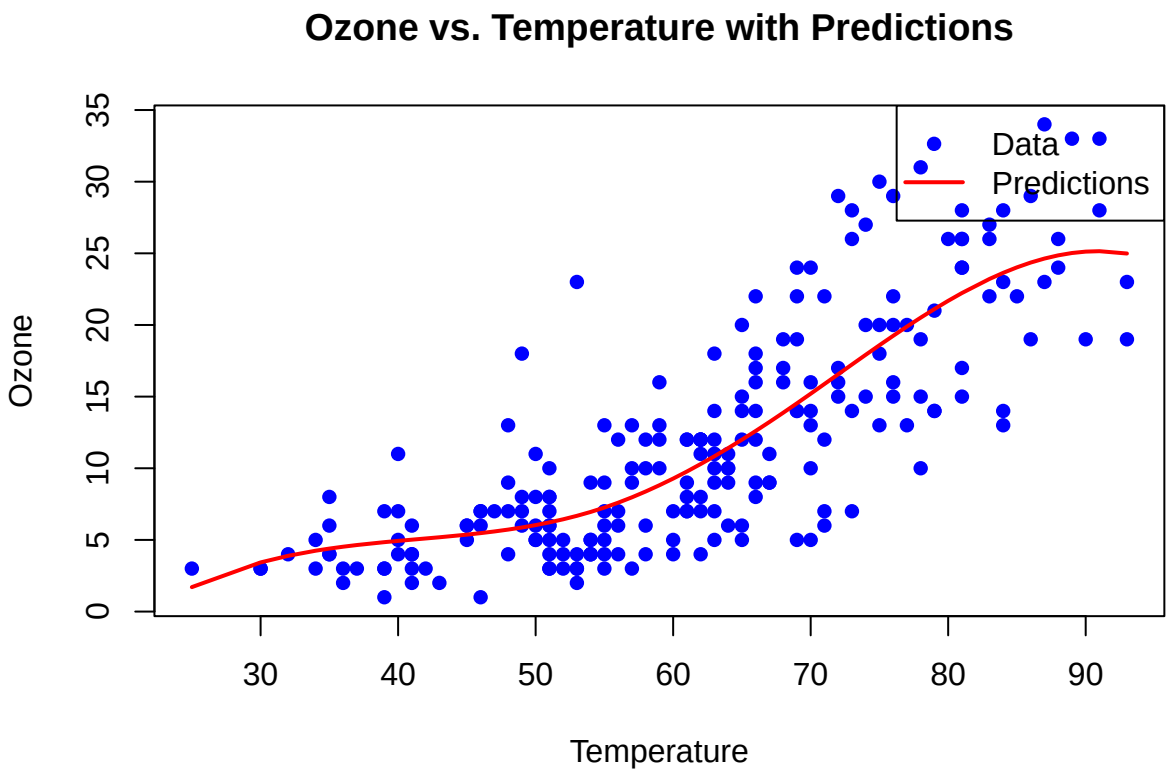
We can observe (with M = 4) the lienar, quadratic and cubic term as we expect them to be. Additionally the

basis terms (also cubic) for the two knots are also plotted.

Task 2

Now we use the basis functions from task 1 within lm() to predict the repsonse for the training data. There have been some difficulties, so to make sure all is done right the random split data is also first sorted again and the column names are added again so everything works as it should. After then predicitng the repsonse for the training data we plot the train data as data point (ozone vs temperature) and add the predictions as a smooth line to the plot.

```
sorted_indices <- order(train_x)
sorted_x <- train_x[sorted_indices]
sorted_y <- train_y[sorted_indices]

sorted_spline <- lecturespl(sorted_x, 2, 4)

train_spline_df <- as.data.frame(sorted_spline$X)
colnames(train_spline_df) <- paste0("X", 1:ncol(train_spline_df))

model_spline <- lm(sorted_y ~ ., data=train_spline_df)

pred_train <- predict(model_spline, newdata = data.frame(sorted_spline$X))

plot(train_x, train_y, main="Ozone vs. Temperature with Predictions",
     xlab="Temperature", ylab="Ozone", pch=16, col="blue")
lines(sorted_x, pred_train, col="red", lwd=2)
legend("topright", legend=c("Data", "Predictions"),
       col=c("blue", "red"), pch=c(16, NA), lty=c(NA, 1), lwd=c(NA, 2))
```



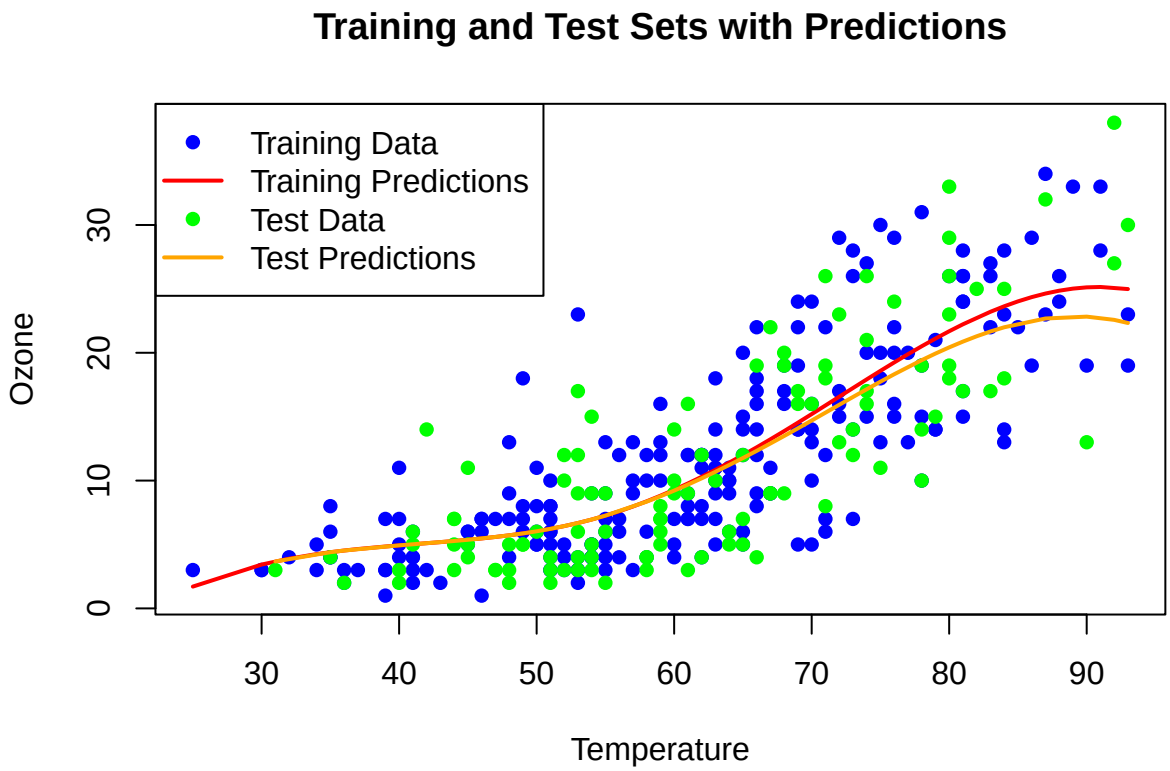
As we can see the predictions follow a smooth line that seems like a good fit for the training data.

Task 3

Now we also predict the response for the test data using the trained model with the new spline matrix using the lecruresplfunction. To do so we first again sort the data and name the columns correctly. Then we plot train and test data in different colors and add smooth lines for both the predictions of the train and test set.

```
sorted_test <- order(test_x)
sort_test_x <- test_x[sorted_test]
sort_test_y <- test_y[sorted_test]

test_spline <- lecturespl(sort_test_x, 2, 4)
test_spline_df <- as.data.frame(test_spline$X)
colnames(test_spline_df) <- colnames(train_spline_df)

test_pred <- predict(model_spline, newdata = test_spline_df)

plot(train_x, train_y, main="Training and Test Sets with Predictions",
     xlab="Temperature", ylab="Ozone", pch=16, col="blue",
     xlim=range(c(train_x, test_x)), ylim=range(c(train_y, test_y)))

lines(sorted_x, pred_train, col="red", lwd=2)

points(sort_test_x, sort_test_y, col="green", pch=16)

lines(sort_test_x, test_pred, col="orange", lwd=2)
```

```
legend("topleft", legend=c("Training Data", "Training Predictions",
                           "Test Data", "Test Predictions"),
       col=c("blue", "red", "green", "orange"), pch=c(16,NA,16,NA),
       lwd=c(NA,2,NA,2))
```



As we can observe, for both train and test data the prediciotns follow a smooth curve that seems like a good fit to its data.

Task 4

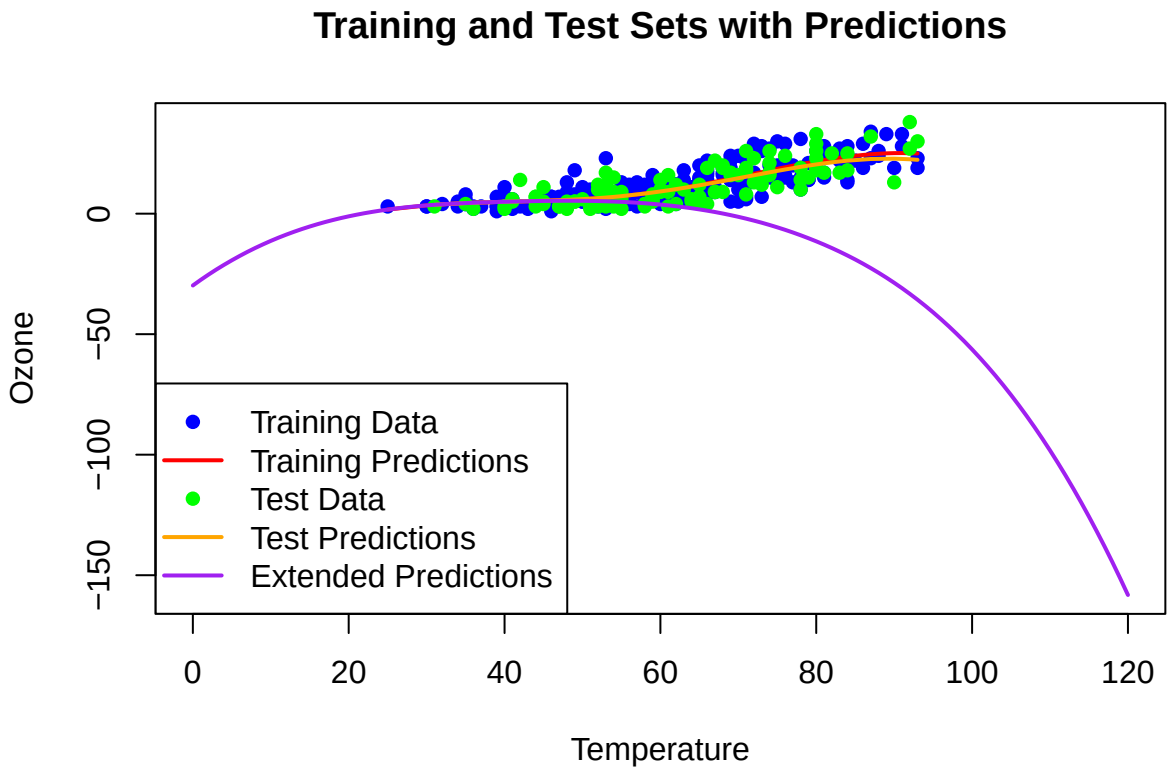
We create the specified sequence and use the model to predict the response by generating the spline matrix again and then usign the model. We then again plot the same plot from task 3 and additionally add a line that represents the predictions for the new sequence.

```
new_temp <- seq(0,120)

new_spline <- lecturespl(new_temp, nknots = 2, M = 4)

new_spline_df <- as.data.frame(new_spline$X)
colnames(new_spline_df) <- colnames(train_spline_df)

new_predictions <- predict(model_spline, newdata = new_spline_df)
```

The plot.

```
plot(train_x, train_y, main="Training and Test Sets with Predictions",
     xlab="Temperature", ylab="Ozone", pch=16, col="blue", xlim=c(0, 120),
```

```
     ylim = range(c(sorted_y, sort_test_y, new_predictions)))

lines(sorted_x, pred_train, col="red", lwd=2)

points(sort_test_x, sort_test_y, col="green", pch=16)

lines(sort_test_x, test_pred, col="orange", lwd=2)

lines(new_temp, new_predictions, col = "purple", lwd = 2)


legend("bottomleft", legend = c("Training Data", "Training Predictions",
                                "Test Data", "Test Predictions",
                                "Extended Predictions"),
       col = c("blue", "red", "green", "orange", "purple"),
       pch = c(16, NA, 16, NA, NA), lty = c(NA, 1, NA, 1, 1), lwd = c(NA, 2, NA, 2, 2))
```



As we can see, the predictions for the new sequence looks very off and does not make sense at all. That is because lecturespl() constructs the knots on the quantiles of the input variable, but we should use the quantiles of the input variable. We will explroe this in the next task.

Task 5

We modify the lectruespl() function so it can also take knots as input argument to create teh spline matrix using the specified knots. As default it is not using knots and still creates new ones using teh provided data as before.

```r
lecturespl_1 <- function(x, nknots = 2, M = 4, knots = NULL) {
  n <- length(x)
  X <- matrix(NA, nrow = n, ncol = (M - 1) + nknots)

  for (i in 1:(M - 1)) {
    X[, i] <- x^i
  }

  if (is.null(knots)) {
    quant <- seq(0, 1, 1 / (nknots + 1))[c(2:(nknots + 1))]
    knots <- quantile(x, quant)
  }

  for (i in M:(M + nknots - 1)) {
    X[, i] <- ifelse(x - knots[i - M + 1] < 0, 0, (x - knots[i - M + 1])^(M - 1))
  }

  list(X = X, knots = knots)
}
```

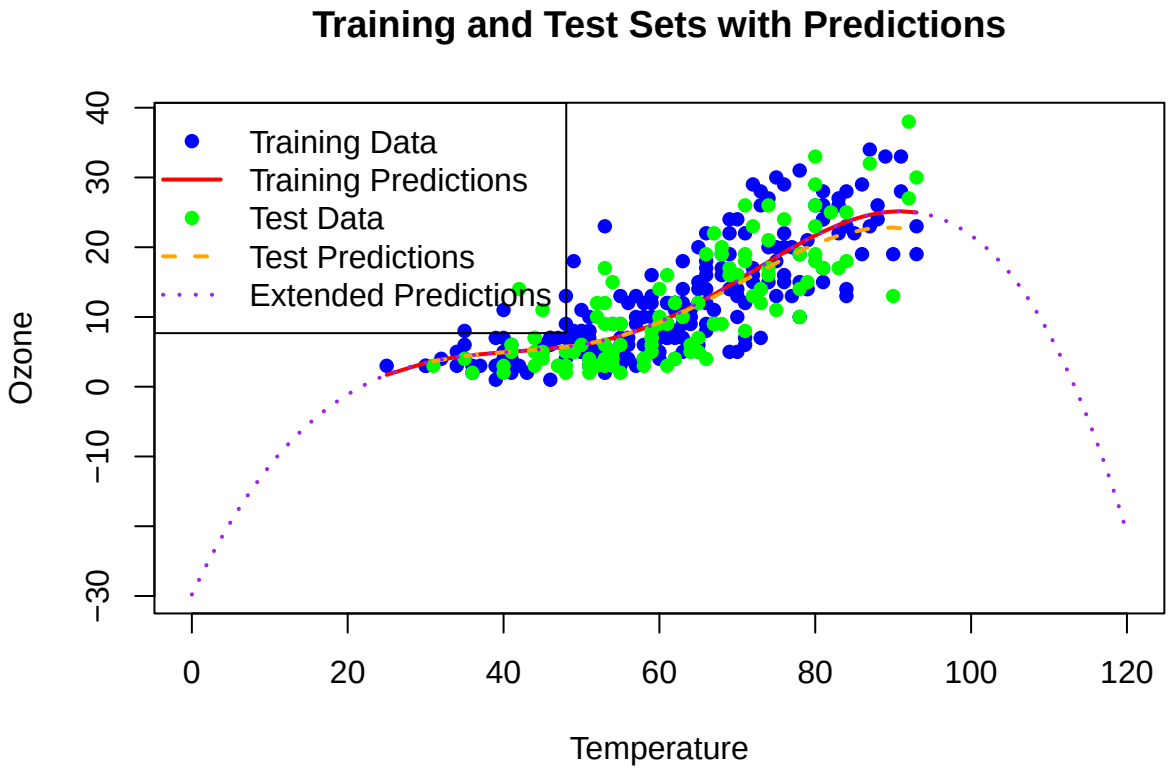
No we again create the basis splines for the train data to get the knots. We then call the function again on the new data but use the knots from the training set. thenw emake prediciotns with the already trrained model using the generated spline matrix for the new data.

```r
train_spline <- lecturespl_1(sorted_x, nknots = 2, M = 4)
train_knots <- train_spline$knots

new_spline <- lecturespl_1(new_temp, nknots = 2, M = 4, knots = train_knots)

new_spline_df <- as.data.frame(new_spline$X)
colnames(new_spline_df) <- colnames(train_spline_df)

new_predictions <- predict(model_spline, newdata = new_spline_df)
```

Now we again plot all the previous train/test data and its prediciotns and now add the prdiciotns for the new data sequence as a line to the plot.

```r
plot(train_x, train_y, main="Training and Test Sets with Predictions",
     xlab="Temperature", ylab="Ozone", pch=16, col="blue", xlim=c(0, 120),
     ylim = range(c(sorted_y, sort_test_y, new_predictions)))

lines(sorted_x, pred_train, col="red", lwd=2)

points(sort_test_x, sort_test_y, col="green", pch=16)

lines(sort_test_x, test_pred, col="orange", lwd=2, lty=2)

lines(new_temp, new_predictions, col = "purple", lwd = 2, lty=3)


legend("topleft", legend = c("Training Data", "Training Predictions",
                             "Test Data", "Test Predictions",
                             "Extended Predictions"),
       col = c("blue", "red", "green", "orange", "purple"),
       pch = c(16, NA, 16, NA, NA), lty = c(NA, 1, NA, 2, 3),
```

```
       lwd = c(NA, 2, NA, 2, 2))
```



As expected, the prediciotns for the new data sequence follow the same as the train data predictions, which makes sense since it is using the same knots. Only for low and large values that we only have now by the extended range with the new data sequence, the predictions behave strangely. This is also not unusual since the model has not been trained on the new data in this range.

Task 6

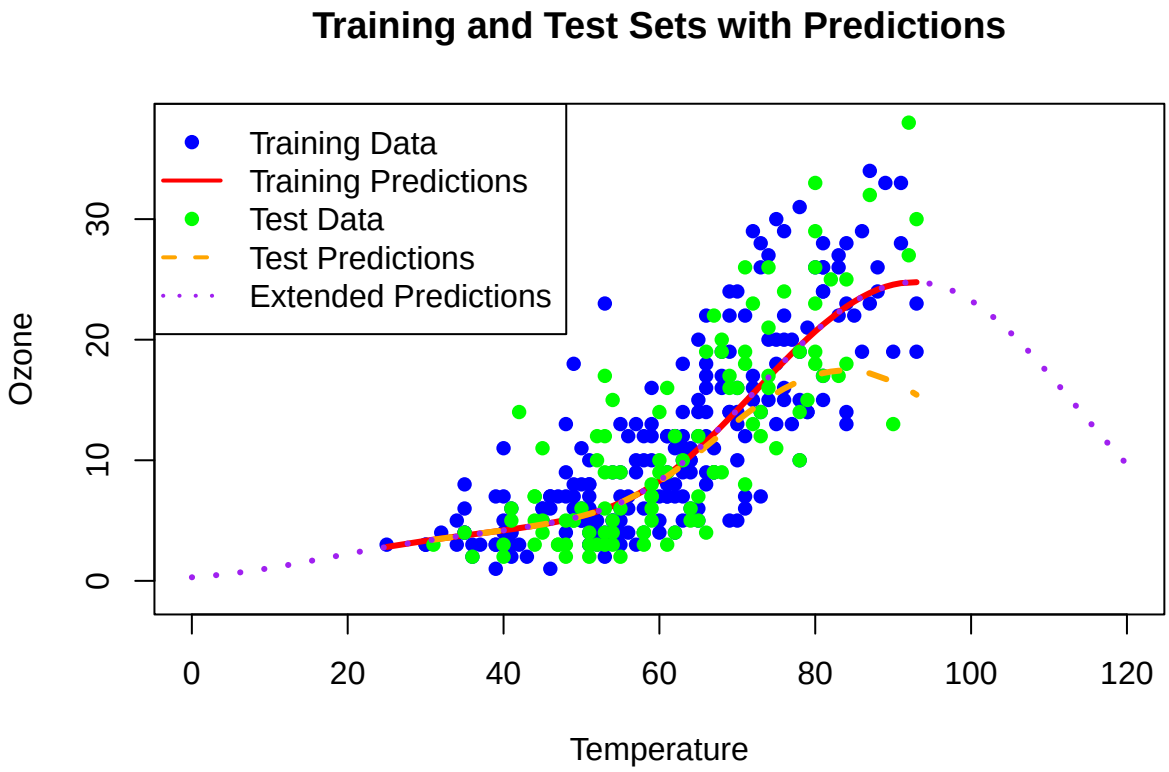
To solve the problem of negative predicted ozone levels for low values of temperature, we use the log-transformed response and then exp() on the predictions. Thus we train the model again with the log-transformed response and then make predictions on the train, test and new data.

```
model_spline <- lm(log(sorted_y) ~ ., data=train_spline_df)

pred_train <- predict(model_spline, newdata = data.frame(sorted_spline$X))
test_pred <- predict(model_spline, newdata = test_spline_df)
new_predictions <- predict(model_spline, newdata = new_spline_df)
```

Now as before we again plot train/test data and add the predicitons for train/test/new data response as lines. To do so we need to exponentiate the rpedicitons since we used the log-transformed resposne inside the model.

```
plot(train_x, train_y, main="Training and Test Sets with Predictions",
     xlab="Temperature", ylab="Ozone", pch=16, col="blue", xlim=c(0, 120),
     ylim = range(c(sorted_y, sort_test_y, new_predictions)))

lines(sorted_x, exp(pred_train), col="red", lwd=3)
```

```r
points(sort_test_x, sort_test_y, col="green", pch=16)

lines(sort_test_x, exp(test_pred), col="orange", lwd=3, lty=2)

lines(new_temp, exp(new_predictions), col = "purple", lwd = 3,lty=3)


legend("topleft", legend = c("Training Data", "Training Predictions",
                             "Test Data", "Test Predictions",
                             "Extended Predictions"),
       col = c("blue", "red", "green", "orange", "purple"),
       pch = c(16, NA, 16, NA, NA), lty = c(NA, 1, NA, 2, 3),
       lwd = c(NA, 2, NA, 2, 2))
```



As we can observe now the predictions for small values of ozone make more sense, but the ones for higher values in the test data got worse. Also the predictions for the new data for values in the range that is not included in the train data anymore the values again decrease which probably doe snot make sense. This still might not be a problem since these high values of temperature are not ver realistic anyway.