Polynomial Regression

Veerasak Kritsanapraphan

1/17/2020

### Loading Required R packages

library(tidyverse)  
library(modelr)  
library(broom)  
library(dplyr)  
library(ggplot2)

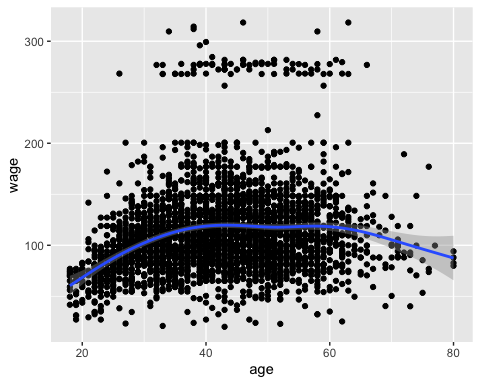
#### Load the data

data("Wage", package = "ISLR")

First, visualize the scatter plot of the medv vs lstat variables as follow:

ggplot(Wage, aes(age, wage) ) +  
 geom\_point() +  
 stat\_smooth()

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



The above scatter plot suggests a non-linear relationship between the two variables

In the following sections, we start by computing linear and non-linear regression models. Next, we’ll compare the different models in order to choose the best one for our data.

## Linear regression

The standard linear regression model equation can be written as wage = b0 + b1\*age.

### Compute linear regression model:

### Build the model

model <- lm(wage ~ age, data = Wage)

### Model performance

linearmodel <- data.frame(  
 name = 'Linear Regression model',  
 R2 = rsquare(model, data=Wage),  
 RMSE = rmse(model, data=Wage),  
 MAE = mae(model, data=Wage)  
)  
linearmodel

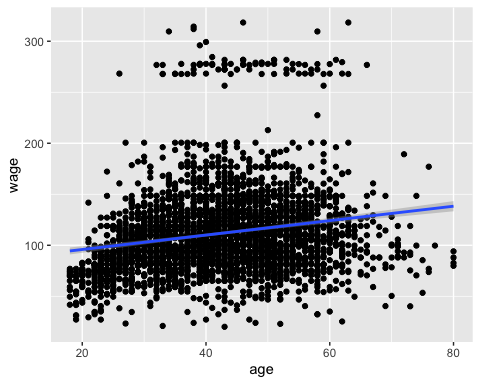
## name R2 RMSE MAE  
## 1 Linear Regression model 0.03827391 40.91543 28.76975

glance(model) %>%  
 dplyr::select(r.squared, adj.r.squared, sigma, AIC, BIC, p.value)

## # A tibble: 1 x 6  
## r.squared adj.r.squared sigma AIC BIC p.value  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 0.0383 0.0380 40.9 30789. 30807. 2.90e-27

### Visualize the data:

ggplot(Wage, aes(age, wage) ) +  
 geom\_point() +  
 stat\_smooth(method = lm, formula = y ~ x)



## Polynomial regression

The polynomial regression adds polynomial or quadratic terms to the regression equation as follow:

wage=b0+b1∗age+b2∗age2

In R, to create a predictor x^2 you should use the function I(), as follow: I(x^2). This raise x to the power 2.

The polynomial regression can be computed in R as follow:

lm(wage ~ age + I(age^2), data = Wage)

##   
## Call:  
## lm(formula = wage ~ age + I(age^2), data = Wage)  
##   
## Coefficients:  
## (Intercept) age I(age^2)   
## -10.42522 5.29403 -0.05301

An alternative simple solution is to use this:

lm(wage ~ poly(age, 2, raw = TRUE), data = Wage)

##   
## Call:  
## lm(formula = wage ~ poly(age, 2, raw = TRUE), data = Wage)  
##   
## Coefficients:  
## (Intercept) poly(age, 2, raw = TRUE)1   
## -10.42522 5.29403   
## poly(age, 2, raw = TRUE)2   
## -0.05301

The following example computes a sixfth-order polynomial fit:

lm(wage ~ poly(age, 6, raw = TRUE), data = Wage) %>%  
 summary()

##   
## Call:  
## lm(formula = wage ~ poly(age, 6, raw = TRUE), data = Wage)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -98.521 -24.536 -4.848 15.471 202.108   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 5.778e+02 4.307e+02 1.341 0.1799   
## poly(age, 6, raw = TRUE)1 -9.315e+01 6.500e+01 -1.433 0.1520   
## poly(age, 6, raw = TRUE)2 6.257e+00 3.925e+00 1.594 0.1110   
## poly(age, 6, raw = TRUE)3 -2.006e-01 1.216e-01 -1.650 0.0991 .  
## poly(age, 6, raw = TRUE)4 3.374e-03 2.045e-03 1.650 0.0990 .  
## poly(age, 6, raw = TRUE)5 -2.872e-05 1.773e-05 -1.620 0.1052   
## poly(age, 6, raw = TRUE)6 9.752e-08 6.206e-08 1.571 0.1162   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 39.91 on 2993 degrees of freedom  
## Multiple R-squared: 0.08726, Adjusted R-squared: 0.08543   
## F-statistic: 47.69 on 6 and 2993 DF, p-value: < 2.2e-16

### Build the fifth polynomial model

model <- lm(wage ~ poly(age, 5, raw = TRUE), data = Wage)

### Model performance

polymodel <- data.frame(  
 name = 'Polynomial Regression model',  
 R2 = rsquare(model, data=Wage),  
 RMSE = rmse(model, data=Wage),  
 MAE = mae(model, data=Wage)  
)  
polymodel

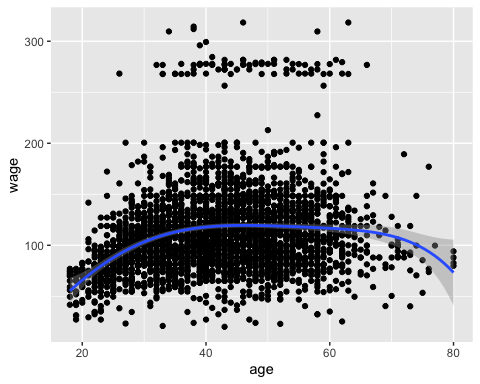
## name R2 RMSE MAE  
## 1 Polynomial Regression model 0.08651028 39.87615 27.71125

glance(model) %>%  
 dplyr::select(r.squared, adj.r.squared, sigma, AIC, BIC, p.value)

## # A tibble: 1 x 6  
## r.squared adj.r.squared sigma AIC BIC p.value  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 0.0865 0.0850 39.9 30642. 30684. 1.67e-56

Visualize the fith polynomial regression line as follow:

ggplot(Wage, aes(age, wage) ) +  
 geom\_point() +  
 stat\_smooth(method = lm, formula = y ~ poly(x, 5, raw = TRUE))



## Log transformation

When you have a non-linear relationship, you can also try a logarithm transformation of the predictor variables:

### Build the model

model <- lm(wage ~ log(age), data = Wage)

### Model performance

logmodel <- data.frame(  
 name = 'Log Transform Regression model',  
 R2 = rsquare(model, data=Wage),  
 RMSE = rmse(model, data=Wage),  
 MAE = mae(model, data=Wage)  
)  
logmodel

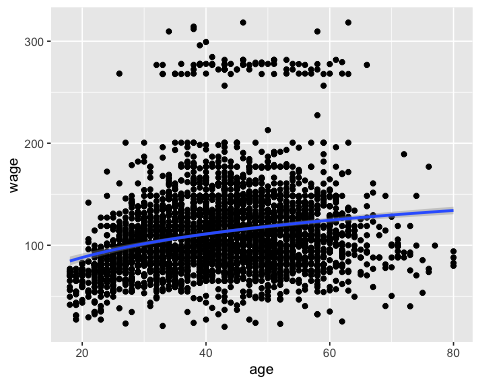
## name R2 RMSE MAE  
## 1 Log Transform Regression model 0.05284604 40.60427 28.45308

glance(model) %>%  
 dplyr::select(r.squared, adj.r.squared, sigma, AIC, BIC, p.value)

## # A tibble: 1 x 6  
## r.squared adj.r.squared sigma AIC BIC p.value  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 0.0528 0.0525 40.6 30743. 30761. 2.84e-37

Visualize the data:

ggplot(Wage, aes(age, wage) ) +  
 geom\_point() +  
 stat\_smooth(method = lm, formula = y ~ log(x))



## Spline regression

knots <- quantile(Wage$age, p = c(0.25, 0.5, 0.75))

We’ll create a model using a cubic spline (degree = 3):

library(splines)

#### Build the model

knots <- quantile(Wage$age, p = c(0.25, 0.5, 0.75))  
model <- lm (wage ~ bs(age, knots = knots), data = Wage)

#### Model performance

splinemodel <- data.frame(  
 name = 'Splines model',  
 R2 = rsquare(model, data=Wage),  
 RMSE = rmse(model, data=Wage),  
 MAE = mae(model, data=Wage)  
)  
splinemodel

## name R2 RMSE MAE  
## 1 Splines model 0.08729104 39.8591 27.707

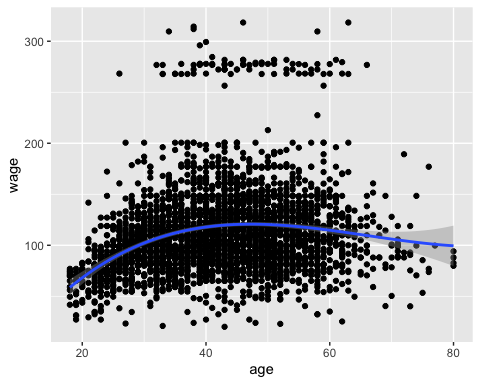
glance(model) %>%  
 dplyr::select(r.squared, adj.r.squared, sigma, AIC, BIC, p.value)

## # A tibble: 1 x 6  
## r.squared adj.r.squared sigma AIC BIC p.value  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 0.0873 0.0855 39.9 30642. 30690. 3.76e-56

Note that, the coefficients for a spline term are not interpretable.

Visualize the cubic spline as follow:

ggplot(Wage, aes(age, wage) ) +  
 geom\_point() +  
 stat\_smooth(method = lm, formula = y ~ splines::bs(x, df = 3))



## Generalized additive models

Once you have detected a non-linear relationship in your data, the polynomial terms may not be flexible enough to capture the relationship, and spline terms require specifying the knots.

Generalized additive models, or GAM, are a technique to automatically fit a spline regression. This can be done using the mgcv R package:

library(mgcv)

#### Build the model

model <- gam(wage ~ s(age), data = Wage)

#### Model performance

GAMmodel <- data.frame(  
 name = 'Generalized additive model',  
 R2 = rsquare(model, data=Wage),  
 RMSE = rmse(model, data=Wage),  
 MAE = mae(model, data=Wage)  
)  
GAMmodel

## name R2 RMSE MAE  
## 1 Generalized additive model 0.08798955 39.84385 27.66433

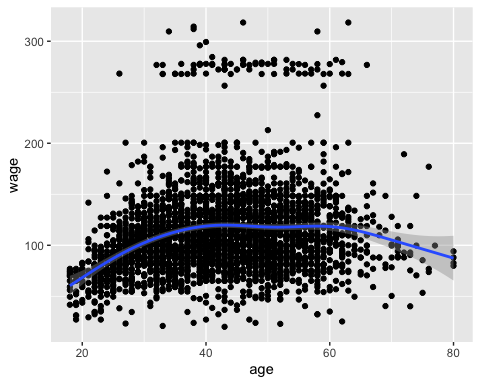
glance(model)

## # A tibble: 1 x 6  
## df logLik AIC BIC deviance df.residual  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 6.30 -15312. 30638. 30682. 4762597. 2994.

The term s(lstat) tells the gam() function to find the best knots for a spline term.

Visualize the data:

ggplot(Wage, aes(age, wage) ) +  
 geom\_point() +  
 stat\_smooth(method = gam, formula = y ~ s(x))



## Comparing the models

From analyzing the RMSE and the R2 metrics of the different models, it can be seen that the polynomial regression, the spline regression and the generalized additive models outperform the linear regression model and the log transformation approaches.

totalmodel <- do.call('rbind', list(linearmodel, polymodel, splinemodel, GAMmodel))  
totalmodel

## name R2 RMSE MAE  
## 1 Linear Regression model 0.03827391 40.91543 28.76975  
## 2 Polynomial Regression model 0.08651028 39.87615 27.71125  
## 3 Splines model 0.08729104 39.85910 27.70700  
## 4 Generalized additive model 0.08798955 39.84385 27.66433