model selection

Justin Lo

2023-08-24

In this project, I explore methods of how to determine what models to select. I will be using the dataset of college in the ISLR package. I will be continuing from the nonlinear modeling project. Let's first revisit the models I have built in the nonlinear project.

```r
library(ISLR)
library(modelr)
library(splines)
college<-College

quadratic_regression_model<- lm(Expend ~ poly(PhD, 2), data =college)
summary(quadratic_regression_model)
```

```
##
## Call:
## lm(formula = Expend ~ poly(PhD, 2), data = college)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -12750  -2263   -357   1309  40415
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     9660.2      154.4   62.55   <2e-16 ***
## poly(PhD, 2)1  62950.2     4304.9   14.62   <2e-16 ***
## poly(PhD, 2)2  53405.8     4304.9   12.41   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4305 on 774 degrees of freedom
## Multiple R-squared:  0.3221, Adjusted R-squared:  0.3203
## F-statistic: 183.9 on 2 and 774 DF,  p-value: < 2.2e-16
```

```r
fitted_vals_quadratic <- predict(quadratic_regression_model, newdata = data.frame(PhD= 8:103))

cubic_regression_model<- lm(Expend ~ poly(PhD,3), data=college)
summary(cubic_regression_model)
```

```
##
## Call:
## lm(formula = Expend ~ poly(PhD, 3), data = college)
```

```
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -15884  -2266   -373   1330  39272
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)       9660        152  63.544  < 2e-16 ***
## poly(PhD, 3)1    62950       4238  14.855  < 2e-16 ***
## poly(PhD, 3)2    53406       4238  12.603  < 2e-16 ***
## poly(PhD, 3)3    21518       4238   5.078 4.79e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4238 on 773 degrees of freedom
## Multiple R-squared:  0.344,  Adjusted R-squared:  0.3414
## F-statistic: 135.1 on 3 and 773 DF,  p-value: < 2.2e-16
```

```r
fitted_vals_cubic<- predict(cubic_regression_model, newdata = data.frame(PhD=8:103))

cubic_spline_regression_model <- lm(Expend ~ bs(PhD, df = 4, degree = 3), data = College)
summary(cubic_spline_regression_model)
```

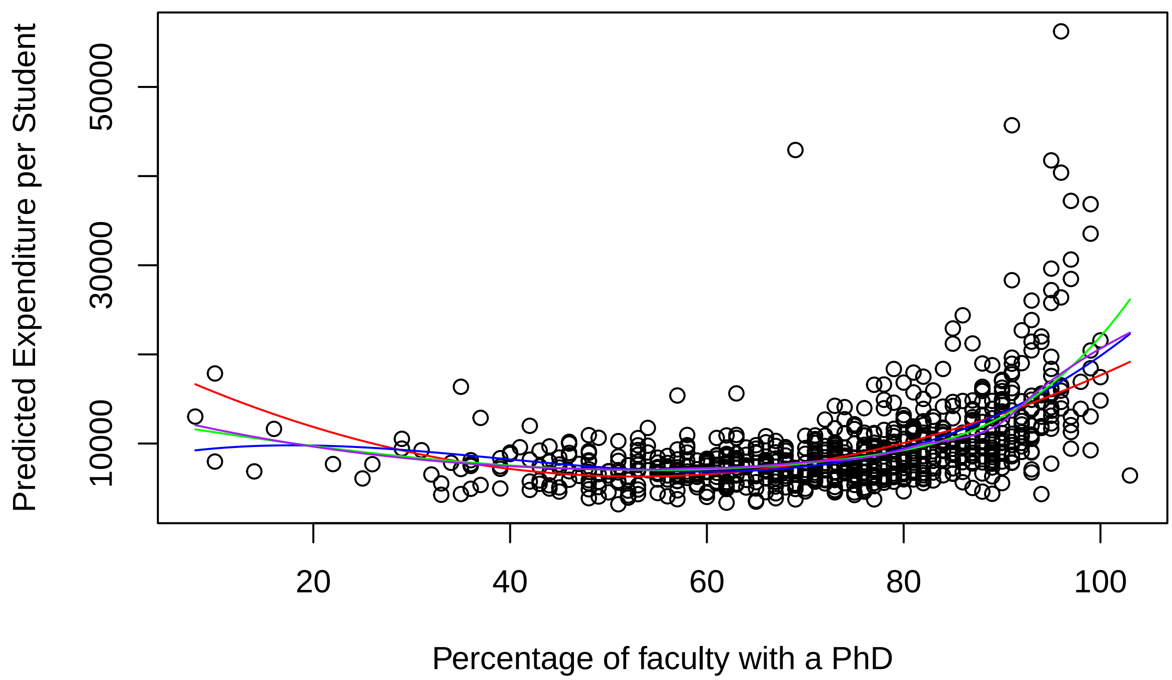
```
##
## Call:
## lm(formula = Expend ~ bs(PhD, df = 4, degree = 3), data = College)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -19758  -2177   -444   1263  38654
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     11590       2157   5.373 1.02e-07 ***
## bs(PhD, df = 4, degree = 3)1    -3747       3469  -1.080 0.280391
## bs(PhD, df = 4, degree = 3)2    -7316       2094  -3.493 0.000504 ***
## bs(PhD, df = 4, degree = 3)3      365       2479   0.147 0.882997
## bs(PhD, df = 4, degree = 3)4    14583       2447   5.959 3.86e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4214 on 772 degrees of freedom
## Multiple R-squared:  0.352,  Adjusted R-squared:  0.3486
## F-statistic: 104.8 on 4 and 772 DF,  p-value: < 2.2e-16
```

```r
fitted_vals_spline <- predict(cubic_spline_regression_model, newdata = data.frame(PhD = 8:103))

loess_regression_model<- loess(Expend ~ PhD, data = College, span = .4)
summary(loess_regression_model)
```

```
## Call:
## loess(formula = Expend ~ PhD, data = College, span = 0.4)
##
```

```
## Number of Observations: 777
## Equivalent Number of Parameters: 8.67
## Residual Standard Error: 4191
## Trace of smoother matrix: 9.57  (exact)
##
## Control settings:
##   span     :  0.4
##   degree   :  2
##   family   :  gaussian
##   surface  :  interpolate      cell = 0.2
##   normalize:  TRUE
##  parametric:  FALSE
## drop.square:  FALSE
```

```
fitted_vals_loess <- predict(loess_regression_model, newdata = data.frame(PhD = 8:103))

plot(College$PhD, College$Expend,
     xlab = "Percentage of faculty with a PhD",
     ylab = "Predicted Expenditure per Student")
lines(8:103, fitted_vals_quadratic, col = "red")
lines(8:103, fitted_vals_cubic, col = "blue")
lines(8:103, fitted_vals_spline, col = "green")
lines(8:103, fitted_vals_loess, col = "purple")
```



Now, to do basic comparison, I will first compare the rmse to see how much of the variability of the dataset is explained by the models.

```r
rmse(quadratic_regression_model, college)
```

```
## [1] 4296.624
```

```r
rmse(cubic_regression_model, college)
```

```
## [1] 4226.709
```

```r
rmse(cubic_spline_regression_model, college)
```

```
## [1] 4200.826
```

```r
rmse(loess_regression_model, college)
```

```
## [1] 4163.094
```

The results show that the loess_regression_model has the lowest rmse of 4163.094, second lowest being the cubic_spline_regression_model with a rmse of 4195.685, third lowest being the cubic_regression_model with a rmse of 4226.709 and the quadratic_regression_model has the highest rmse of 4296.624

Now, I will split the data into training and testing set to perform leave-one-out cross validation(LOOCV)

```r
for(i in 1:nrow(college)){

  loocv_results <- data.frame(
  Quadratic = numeric(nrow(college)),
  Cubic = numeric(nrow(college)),
  CubicSpline = numeric(nrow(college)),
  Loess = numeric(nrow(college))
)
  #splitting the dataset into training and testing data
  test_data<- college[i,]
  train_data<-college[-i,]

  #fitting the models
  quadratic_regression_model <- lm(Expend ~ poly(PhD, 2), data = train_data)
  cubic_regression_model <- lm(Expend ~ poly(PhD, 3), data = train_data)
  cubic_spline_model <- lm(Expend ~ bs(PhD, df = 4, degree = 3), data = train_data)
  loess_model <- loess(Expend ~ PhD, data = train_data, span = .4)

  #predicting the test data
  pred_quadratic <- predict(quadratic_regression_model, newdata=test_data[c("PhD")])
  pred_cubic <- predict(cubic_regression_model, newdata=test_data[c("PhD")])
  pred_cubic_spline <- predict(cubic_spline_model, newdata=test_data[c("PhD")])
  pred_loess <- predict(loess_model, newdata=test_data[c("PhD")])

  #calculating the square error and storing it in dataframe
  loocv_results[i, "Quadratic"] <- (test_data$Expend - pred_quadratic)^2
  loocv_results[i, "Cubic"] <- (test_data$Expend - pred_cubic)^2
  loocv_results[i, "CubicSpline"] <- (test_data$Expend - pred_cubic_spline)^2
```

```
  loocv_results[i, "Loess"] <- (test_data$Expend - pred_loess)^2
}

#calculate the mean of squared errors for each model
mean_quadratic <- mean(loocv_results$Quadratic)
mean_cubic <- mean(loocv_results$Cubic)
mean_cubic_spline <- mean(loocv_results$CubicSpline)
mean_loess <- mean(loocv_results$Loess)

mean_quadratic
```

```
## [1] 23736.14
```

```
mean_cubic
```

```
## [1] 17890.8
```

```
mean_cubic_spline
```

```
## [1] 19630.52
```

```
mean_loess
```

```
## [1] 18510.3
```

This gives the mean squared error. The model with the lowest mean squared error is the cubic model with a mse of 17890.8, then the loess with a mse of 18510.3, then the cubic spline of 19630.52 and lastly with the highest mse, the quadratic with a mse of 23736.14

Now, let's do K-fold valuation

```
# Split data into k folds
k <- 10
folds <- cut(x=seq(1,nrow(college)),breaks=k,labels=FALSE)

# Function to compute RMSE
rmse <- function(actual, predicted){
  sqrt(mean((actual - predicted)^2))
}

# Hold RMSE for each model
rmse_vals <- matrix(NA, ncol=4, nrow=k)
colnames(rmse_vals) <- c("Quadratic", "Cubic", "Cubic Spline", "Loess")

# Cross validation loop
for(i in 1:k){
  # Create test and train sets
  test_idx <- which(folds==i, arr.ind=TRUE)
  test <- college[test_idx, ]
  train <- college[-test_idx, ]
```

```r
  # Fit models on train
  quadratic_model <- lm(Expend ~ poly(PhD, 2), data = train)
  cubic_model <- lm(Expend ~ poly(PhD,3), data = train)
  spline_model <- lm(Expend ~ bs(PhD, df = 5, degree = 3), data = train)
  loess_model <- loess(Expend ~ PhD, data = train, span = .4)

  # Make predictions on test
  quadratic_pred <- predict(quadratic_model, newdata = test)
  cubic_pred <- predict(cubic_model, newdata = test)
  spline_pred <- predict(spline_model, newdata = test)
  loess_pred <- predict(loess_model, newdata = test)

  # Compute RMSE on test set
  rmse_vals[i,1] <- rmse(test$Expend, quadratic_pred)
  rmse_vals[i,2] <- rmse(test$Expend, cubic_pred)
  rmse_vals[i,3] <- rmse(test$Expend, spline_pred)
  rmse_vals[i,4] <- rmse(test$Expend, loess_pred)
}

# Average RMSE across folds
colMeans(rmse_vals)
```

```
##    Quadratic        Cubic Cubic Spline        Loess
##     4197.195     4123.031     4132.689           NA
```

The K-fold cross validation result shows that the cubic model has the lowest rmse of 4123.031 and then cublic spline with a rmse of 4132.689, then quadratic with 4197.195. the loop gives NA to the loess model, it could be due to a few different reasons: it could be the fact that there isn't enough data to feed the loop, or it could be some extreme outliers.

In my own judgement, it seems like the cubic model fits the data the best and would be of best use for interpolation.