CrossValidationOfModels

Mahtab Nahayati

2024-12-02

Contents

Model Comparison and Validation for Predicting MPG Based on Horsepower

We will work with the dataset Auto in the ISLR package. (Loading the dataset, exploring its structure, and fitting the required models and visualization)

```
# Load necessary packages
library(ISLR)  # For the Auto dataset
library(ggplot2)  # For visualizations
```

```
## Warning: Paket 'ggplot2' wurde unter R Version 4.3.3 erstellt
```

```
# Obtain information about the dataset
?Auto  # Help page to understand the dataset structure and meaning of variables
```

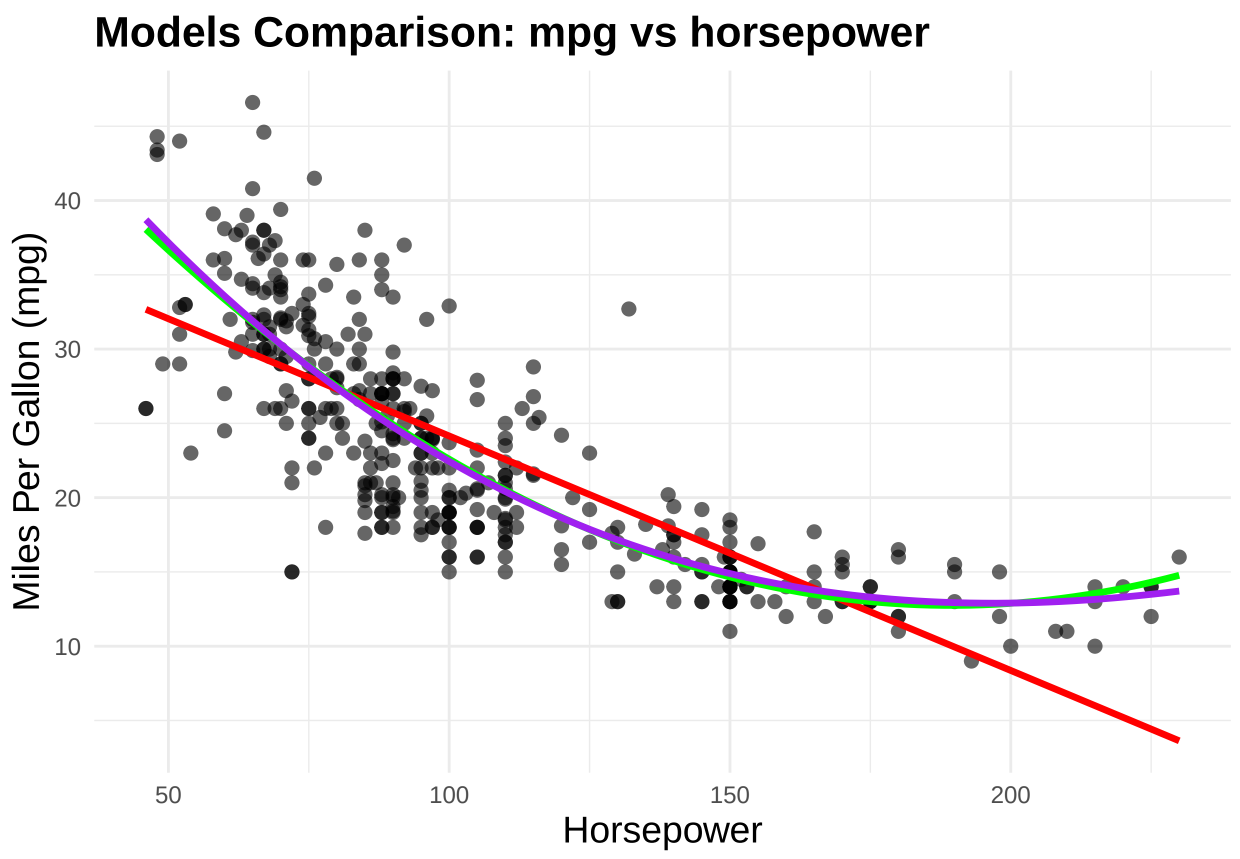
```
## starte den http Server für die Hilfe fertig
```

```
# Load the dataset
data("Auto")

# Check the structure of the dataset
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 
```

```
# Fit the models
model1 <- lm(mpg ~ horsepower, data = Auto)  # Linear model
model2 <- lm(mpg ~ poly(horsepower, 2), data = Auto)  # Polynomial of degree 2
model3 <- lm(mpg ~ poly(horsepower, 3), data = Auto)  # Polynomial of degree 3

# Visualize the models with the data
ggplot(Auto, aes(x = horsepower, y = mpg)) +
  geom_point(color = "black", alpha = 0.6, size = 2) +  # Scatterplot of the data with adjusted point s
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, color = "red", size = 1.2, linetype = "solid"
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), se = FALSE, color = "green", size = 1.2, linetyp
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), se = FALSE, color = "purple", size = 1.2, linety
  labs(title = "Models Comparison: mpg vs horsepower",
       x = "Horsepower",
       y = "Miles Per Gallon (mpg)") +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 16, face = "bold"),
    axis.title = element_text(size = 14),
    legend.position = "bottom"
  )
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



validation set approach

Use once a train/test split 50%/50% and once 70%/30%. Choosing the best model based on Root Mean Squared Error, Mean Squared Error and Median Absolute Deviation.

```r
# Set seed for reproducibility
set.seed(123)

# Function to calculate performance metrics
calculate_metrics <- function(predictions, actual) {
  mse <- mean((predictions - actual)^2)  # Mean Squared Error
  rmse <- sqrt(mse)  # Root Mean Squared Error
  mad <- median(abs(predictions - actual))  # Median Absolute Deviation
  return(c(MSE = mse, RMSE = rmse, MAD = mad))
}

# Splitting data: 50% training, 50% testing
n <- nrow(Auto)
indices_50 <- sample(seq_len(n), size = n / 2)
train_50 <- Auto[indices_50, ]
test_50 <- Auto[-indices_50, ]

# Splitting data: 70% training, 30% testing
```

```r
indices_70 <- sample(seq_len(n), size = 0.7 * n)
train_70 <- Auto[indices_70, ]
test_70 <- Auto[-indices_70, ]

# Function to fit models and evaluate
evaluate_models <- function(train, test) {
  # Fit models
  model1 <- lm(mpg ~ horsepower, data = train)
  model2 <- lm(mpg ~ poly(horsepower, 2), data = train)
  model3 <- lm(mpg ~ poly(horsepower, 3), data = train)

  # Predictions
  pred1 <- predict(model1, newdata = test)
  pred2 <- predict(model2, newdata = test)
  pred3 <- predict(model3, newdata = test)

  # Calculate metrics
  metrics1 <- calculate_metrics(pred1, test$mpg)
  metrics2 <- calculate_metrics(pred2, test$mpg)
  metrics3 <- calculate_metrics(pred3, test$mpg)

  # Combine results
  metrics <- rbind(Linear = metrics1, Quadratic = metrics2, Cubic = metrics3)
  return(metrics)
}

# Evaluate models for 50%/50% split
metrics_50 <- evaluate_models(train_50, test_50)

# Evaluate models for 70%/30% split
metrics_70 <- evaluate_models(train_70, test_70)

# Print results
cat("Performance Metrics for 50%/50% Train/Test Split:\n")
```

```
## Performance Metrics for 50%/50% Train/Test Split:
```

```r
print(metrics_50)
```

```
##                MSE     RMSE      MAD
## Linear    21.24991 4.609762 3.120399
## Quadratic 16.48112 4.059694 2.611559
## Cubic     16.58276 4.072194 2.595151
```

```r
cat("\nPerformance Metrics for 70%/30% Train/Test Split:\n")
```

```
##
## Performance Metrics for 70%/30% Train/Test Split:
```

```r
print(metrics_70)
```

```
##                MSE     RMSE      MAD
## Linear    24.21268 4.920638 2.987032
## Quadratic 17.40207 4.171579 2.581725
## Cubic     17.40068 4.171412 2.572708
```

The quadratic model performs consistently well across both splits with lower MSE and RMSE compared to the linear model and slightly comparable to the cubic model. The cubic model offers marginal improvement in MAD in the 70%/30% split but has similar MSE and RMSE to the quadratic model.

Leave-One-Out(LOOCV), 5-Fold, and 10-Fold Cross-Validation

Using cv.glm function in the boot package.

```r
# Load necessary package
library(boot)  # For cv.glm function

# Fit the models on the full dataset
model1 <- glm(mpg ~ horsepower, data = Auto)
model2 <- glm(mpg ~ poly(horsepower, 2), data = Auto)
model3 <- glm(mpg ~ poly(horsepower, 3), data = Auto)

# Function to compute cross-validation errors
cv_errors <- function(model, folds) {
  cv.glm(data = Auto, glmfit = model, K = folds)$delta[1]  # Extract cross-validation error
}

# Leave-One-Out Cross-Validation (LOOCV)
loocv1 <- cv_errors(model1, folds = nrow(Auto))  # LOOCV for Linear model
loocv2 <- cv_errors(model2, folds = nrow(Auto))  # LOOCV for Quadratic model
loocv3 <- cv_errors(model3, folds = nrow(Auto))  # LOOCV for Cubic model

# 5-Fold Cross-Validation
cv5_1 <- cv_errors(model1, folds = 5)  # 5-Fold CV for Linear model
cv5_2 <- cv_errors(model2, folds = 5)  # 5-Fold CV for Quadratic model
cv5_3 <- cv_errors(model3, folds = 5)  # 5-Fold CV for Cubic model

# 10-Fold Cross-Validation
cv10_1 <- cv_errors(model1, folds = 10)  # 10-Fold CV for Linear model
cv10_2 <- cv_errors(model2, folds = 10)  # 10-Fold CV for Quadratic model
cv10_3 <- cv_errors(model3, folds = 10)  # 10-Fold CV for Cubic model

# Combine results into a table
cv_results <- data.frame(
  Model = c("Linear", "Quadratic", "Cubic"),
  LOOCV = c(loocv1, loocv2, loocv3),
  CV5 = c(cv5_1, cv5_2, cv5_3),
  CV10 = c(cv10_1, cv10_2, cv10_3)
)

# Print the results
cat("Cross-Validation Results:\n")
```

```
## Cross-Validation Results:
```

```r
print(cv_results)
```

```
##       Model    LOOCV      CV5     CV10
## 1    Linear 24.23151 24.33247 24.37496
## 2 Quadratic 19.24821 19.21915 19.22562
## 3     Cubic 19.33498 19.46834 19.28585
```

Compare the result from Validation Set approach and Cross-validation

```r
# Results from validation set approach (Step 2)
validation_results <- data.frame(
  Model = c("Linear", "Quadratic", "Cubic"),
  Split_50_50 = c(21.25, 16.48, 16.58),  # MSE for 50/50 split
  Split_70_30 = c(24.21, 17.40, 17.40)   # MSE for 70/30 split
)

# Results from cross-validation (Step 3)
cv_results <- data.frame(
  Model = c("Linear", "Quadratic", "Cubic"),
  LOOCV = c(24.23, 19.25, 19.33),
  CV5 = c(24.33, 19.22, 19.47),
  CV10 = c(24.37, 19.23, 19.29)
)

# Combine into a single table
combined_results <- merge(validation_results, cv_results, by = "Model")

# Print the combined results
cat("Comparison of Validation and Cross-Validation Results:\n")
```

```
## Comparison of Validation and Cross-Validation Results:
```

```r
print(combined_results)
```

```
##       Model Split_50_50 Split_70_30 LOOCV   CV5  CV10
## 1     Cubic       16.58       17.40 19.33 19.47 19.29
## 2    Linear       21.25       24.21 24.23 24.33 24.37
## 3 Quadratic       16.48       17.40 19.25 19.22 19.23
```

```r
# Conclusions based on the results
cat("\nConclusions:\n")
```

```
##
## Conclusions:
```

```r
cat("- Quadratic model consistently has the lowest errors across all methods.\n")
```

```
## - Quadratic model consistently has the lowest errors across all methods.
```

```r
cat("- Linear model has the highest errors, indicating underfitting.\n")
```

```
## - Linear model has the highest errors, indicating underfitting.
```

```r
cat("- Cubic model performs similarly to the quadratic model but adds unnecessary complexity.\n")
```

```
## - Cubic model performs similarly to the quadratic model but adds unnecessary complexity.
```

Fitting the Specified models to the Economics dataset from ggplot2

Fitting Models

```r
# Load necessary packages
library(ggplot2)
```

```r
# Load the dataset
data("economics")

# View the structure of the dataset
str(economics)
```

```
## spc_tbl_ [574 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ date    : Date[1:574], format: "1967-07-01" "1967-08-01" ...
##  $ pce     : num [1:574] 507 510 516 512 517 ...
##  $ pop     : num [1:574] 198712 198911 199113 199311 199498 ...
##  $ psavert : num [1:574] 12.6 12.6 11.9 12.9 12.8 11.8 11.7 12.3 11.7 12.3 ...
##  $ uempmed : num [1:574] 4.5 4.7 4.6 4.9 4.7 4.8 5.1 4.5 4.1 4.6 ...
##  $ unemploy: num [1:574] 2944 2945 2958 3143 3066 ...
```

```r
# Linear model: unemploy ~ uempmed
linear_model <- lm(unemploy ~ uempmed, data = economics)

# Linear model: uempmed ~ unemploy
linear_model_reverse <- lm(uempmed ~ unemploy, data = economics)

# Exponential model: Assuming 'uempmed' is independent and 'unemploy' is dependent
exp_model <- lm(log(unemploy) ~ uempmed, data = economics)

# Logarithmic model: Assuming 'unemploy' is independent and 'uempmed' is dependent
log_model <- lm(uempmed ~ log(unemploy), data = economics)

# Polynomial models for unemploy ~ uempmed
poly2_model <- lm(unemploy ~ poly(uempmed, 2), data = economics)
poly3_model <- lm(unemploy ~ poly(uempmed, 3), data = economics)
poly10_model <- lm(unemploy ~ poly(uempmed, 10), data = economics)

# Polynomial models for uempmed ~ unemploy
poly2_model_reverse <- lm(uempmed ~ poly(unemploy, 2), data = economics)
poly3_model_reverse <- lm(uempmed ~ poly(unemploy, 3), data = economics)
poly10_model_reverse <- lm(uempmed ~ poly(unemploy, 10), data = economics)

# Summarize the models
cat("Summary of Linear Model (unemploy ~ uempmed):\n")
```

```
## Summary of Linear Model (unemploy ~ uempmed):
```

```r
summary(linear_model)
```

```
##
## Call:
## lm(formula = unemploy ~ uempmed, data = economics)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3005.2  -792.9  -109.8   931.1  3600.7
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2956.8      126.8   23.32   <2e-16 ***
## uempmed        559.3       13.3   42.06   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1307 on 572 degrees of freedom
## Multiple R-squared:  0.7557, Adjusted R-squared:  0.7553
## F-statistic:  1769 on 1 and 572 DF,  p-value: < 2.2e-16
```

```
cat("\nSummary of Linear Model (uempmed ~ unemploy):\n")
```

```
##
## Summary of Linear Model (uempmed ~ unemploy):
```

```
summary(linear_model_reverse)
```

```
##
## Call:
## lm(formula = uempmed ~ unemploy, data = economics)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2674 -1.5802  0.0181  1.0254  7.5343
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.892e+00  2.637e-01  -7.177 2.22e-12 ***
## unemploy     1.351e-03  3.212e-05  42.064  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.032 on 572 degrees of freedom
## Multiple R-squared:  0.7557, Adjusted R-squared:  0.7553
## F-statistic:  1769 on 1 and 572 DF,  p-value: < 2.2e-16
```

```
cat("\nSummary of Exponential Model (log(unemploy) ~ uempmed):\n")
```

```
##
## Summary of Exponential Model (log(unemploy) ~ uempmed):
```

```
summary(exp_model)
```

```
##
## Call:
## lm(formula = log(unemploy) ~ uempmed, data = economics)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.75125 -0.08753  0.02885  0.15297  0.40115
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.32073    0.02231  372.90   <2e-16 ***
## uempmed      0.06705    0.00234   28.66   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.23 on 572 degrees of freedom
## Multiple R-squared:  0.5894, Adjusted R-squared:  0.5887
## F-statistic: 821.2 on 1 and 572 DF,  p-value: < 2.2e-16
```

```r
cat("\nSummary of Logarithmic Model (uempmed ~ log(unemploy)):\n")
```

```
##
## Summary of Logarithmic Model (uempmed ~ log(unemploy)):
```

```r
summary(log_model)
```

```
##
## Call:
## lm(formula = uempmed ~ log(unemploy), data = economics)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7856 -1.9608 -0.4871  0.7934 10.5946
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -69.6121     2.7317  -25.48   <2e-16 ***
## log(unemploy)   8.7909     0.3068   28.66   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.634 on 572 degrees of freedom
## Multiple R-squared:  0.5894, Adjusted R-squared:  0.5887
## F-statistic: 821.2 on 1 and 572 DF,  p-value: < 2.2e-16
```

```r
cat("\nPolynomial Models for unemploy ~ uempmed:\n")
```

```
##
## Polynomial Models for unemploy ~ uempmed:
```

```r
cat("Degree 2:\n")
```

```
## Degree 2:
```

```r
summary(poly2_model)
```

```
##
## Call:
## lm(formula = unemploy ~ poly(uempmed, 2), data = economics)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2606.1  -915.0  -181.6   953.3  2842.0
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         7771.31      49.81  156.00   <2e-16 ***
## poly(uempmed, 2)1  54976.61    1193.47   46.06   <2e-16 ***
## poly(uempmed, 2)2 -12796.93    1193.47  -10.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1193 on 571 degrees of freedom
## Multiple R-squared:  0.7966, Adjusted R-squared:  0.7959
## F-statistic:  1118 on 2 and 571 DF,  p-value: < 2.2e-16
```

```r
cat("\nDegree 3:\n")
```

```
##
## Degree 3:
```

```r
summary(poly3_model)
```

```
##
## Call:
## lm(formula = unemploy ~ poly(uempmed, 3), data = economics)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2711.2  -904.5  -265.9   887.7  3396.6
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          7771.31      48.33 160.792  < 2e-16 ***
## poly(uempmed, 3)1  54976.61    1157.94  47.478  < 2e-16 ***
## poly(uempmed, 3)2 -12796.93    1157.94 -11.051  < 2e-16 ***
## poly(uempmed, 3)3   7003.62    1157.94   6.048 2.65e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1158 on 570 degrees of freedom
## Multiple R-squared:  0.8089, Adjusted R-squared:  0.8079
## F-statistic: 804.3 on 3 and 570 DF,  p-value: < 2.2e-16
```

```r
cat("\nDegree 10:\n")
```

```
##
## Degree 10:
```

```r
summary(poly10_model)
```

```
##
## Call:
## lm(formula = unemploy ~ poly(uempmed, 10), data = economics)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2152.6  -820.4  -262.4   798.1  3897.7
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            7771.31      46.36 167.627  < 2e-16 ***
## poly(uempmed, 10)1   54976.61    1110.73  49.496  < 2e-16 ***
## poly(uempmed, 10)2  -12796.93    1110.73 -11.521  < 2e-16 ***
## poly(uempmed, 10)3    7003.62    1110.73   6.305 5.81e-10 ***
## poly(uempmed, 10)4   -6532.03    1110.73  -5.881 7.00e-09 ***
## poly(uempmed, 10)5    2910.36    1110.73   2.620  0.00902 **
## poly(uempmed, 10)6   -2355.61    1110.73  -2.121  0.03438 *
## poly(uempmed, 10)7    2425.47    1110.73   2.184  0.02940 *
## poly(uempmed, 10)8     661.13    1110.73   0.595  0.55193
## poly(uempmed, 10)9   -2363.17    1110.73  -2.128  0.03381 *
## poly(uempmed, 10)10   1047.93    1110.73   0.943  0.34585
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1111 on 563 degrees of freedom
## Multiple R-squared:  0.8263, Adjusted R-squared:  0.8232
## F-statistic: 267.9 on 10 and 563 DF,  p-value: < 2.2e-16
```

```
cat("\nPolynomial Models for uempmed ~ unemploy:\n")
```

```
##
## Polynomial Models for uempmed ~ unemploy:
```

```
cat("Degree 2:\n")
```

```
## Degree 2:
```

```
summary(poly2_model_reverse)
```

```
##
## Call:
## lm(formula = uempmed ~ poly(unemploy, 2), data = economics)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.1198 -1.0305 -0.0088  0.9332  5.6293
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          8.60871    0.07198  119.59   <2e-16 ***
## poly(unemploy, 2)1 85.45530    1.72461   49.55   <2e-16 ***
## poly(unemploy, 2)2 25.73821    1.72461   14.92   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.725 on 571 degrees of freedom
## Multiple R-squared:  0.8243, Adjusted R-squared:  0.8236
## F-statistic:  1339 on 2 and 571 DF,  p-value: < 2.2e-16
```

```
cat("\nDegree 3:\n")
```

```
##
## Degree 3:
```

```
summary(poly3_model_reverse)
```

```
##
## Call:
## lm(formula = uempmed ~ poly(unemploy, 3), data = economics)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9754 -1.0193 -0.0223  0.9950  5.4680
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          8.60871    0.07192 119.705   <2e-16 ***
## poly(unemploy, 3)1 85.45530    1.72298  49.597   <2e-16 ***
## poly(unemploy, 3)2 25.73821    1.72298  14.938   <2e-16 ***
## poly(unemploy, 3)3 -2.48410    1.72298  -1.442     0.15
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.723 on 570 degrees of freedom
## Multiple R-squared:  0.8249, Adjusted R-squared:  0.824
## F-statistic:   895 on 3 and 570 DF,  p-value: < 2.2e-16
```

```
cat("\nDegree 10:\n")
```

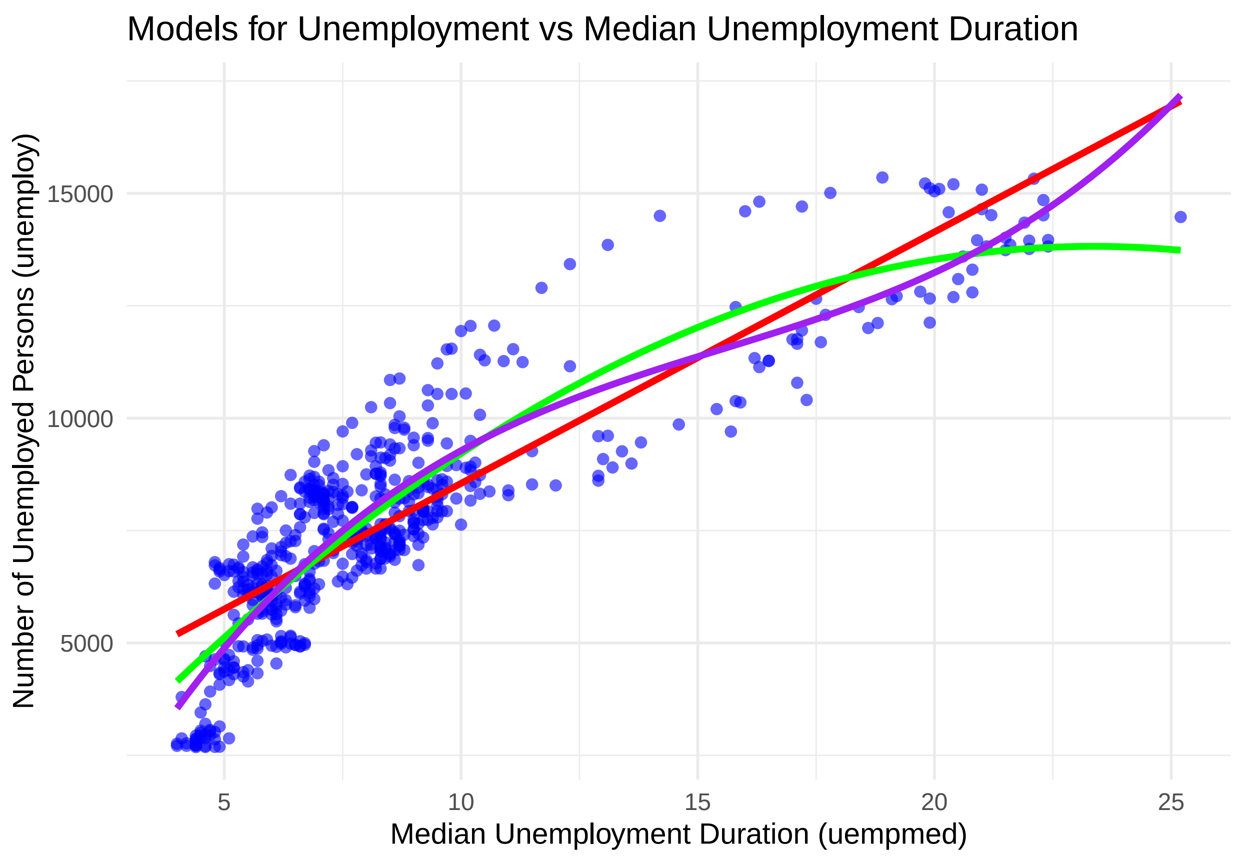
```
##
## Degree 10:
```

```
summary(poly10_model_reverse)
```

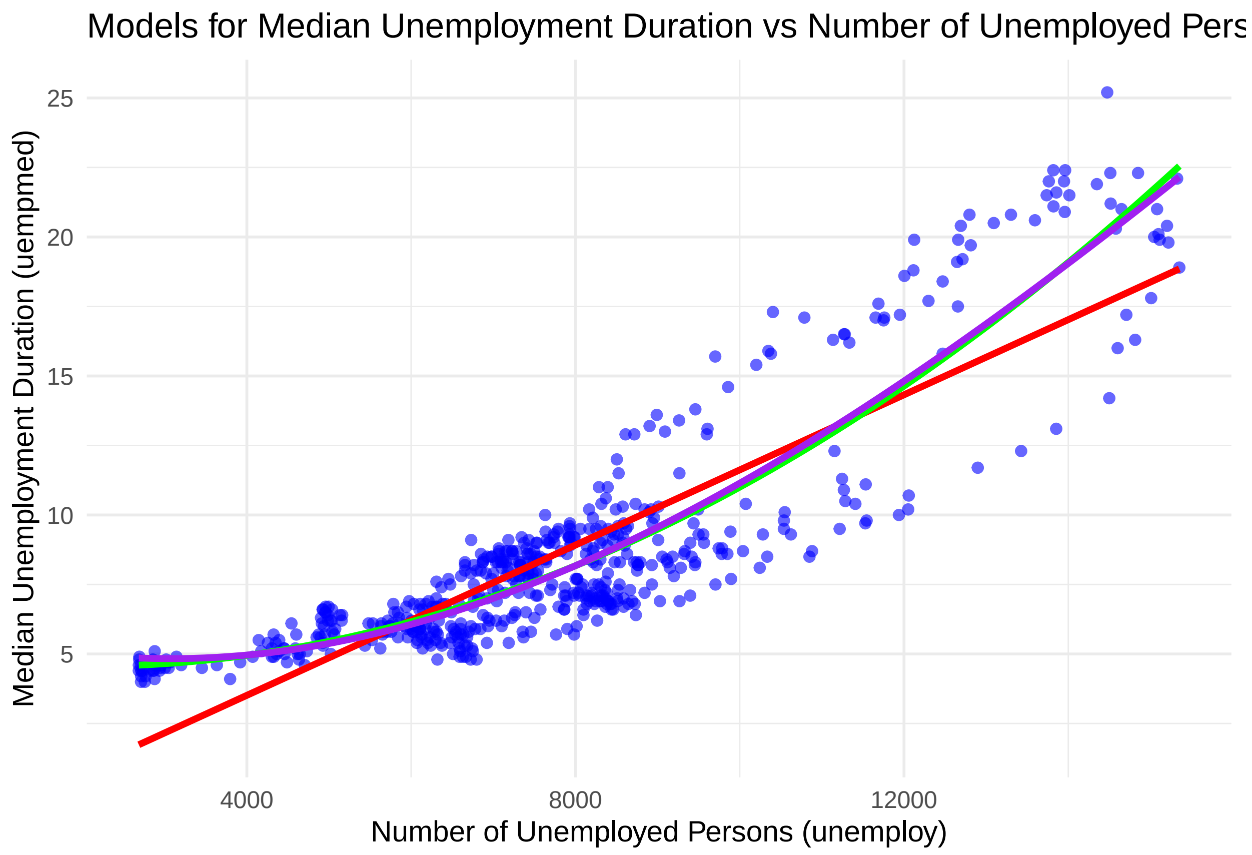
```
##
## Call:
## lm(formula = uempmed ~ poly(unemploy, 10), data = economics)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.7220 -0.9584  0.0484  0.9590  6.2248
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            8.60871    0.06881 125.114  < 2e-16 ***
## poly(unemploy, 10)1  85.45530    1.64850  51.838  < 2e-16 ***
## poly(unemploy, 10)2  25.73821    1.64850  15.613  < 2e-16 ***
## poly(unemploy, 10)3  -2.48410    1.64850  -1.507   0.1324
## poly(unemploy, 10)4  -8.81973    1.64850  -5.350 1.28e-07 ***
## poly(unemploy, 10)5  -7.05050    1.64850  -4.277 2.23e-05 ***
## poly(unemploy, 10)6  -2.00523    1.64850  -1.216   0.2243
## poly(unemploy, 10)7   3.77674    1.64850   2.291   0.0223 *
## poly(unemploy, 10)8   3.75639    1.64850   2.279   0.0231 *
## poly(unemploy, 10)9   0.28411    1.64850   0.172   0.8632
## poly(unemploy, 10)10  1.48176    1.64850   0.899   0.3691
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.648 on 563 degrees of freedom
## Multiple R-squared:  0.8417, Adjusted R-squared:  0.8389
## F-statistic: 299.3 on 10 and 563 DF,  p-value: < 2.2e-16
```

Plot the data

```
library(ggplot2)

# Plot for unemploy ~ uempmed
plot1 <- ggplot(economics, aes(x = uempmed, y = unemploy)) +
  geom_point(color = "blue", alpha = 0.6) +  # Scatterplot
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, color = "red", size = 1.2) +  # Linear model
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), se = FALSE, color = "green", size = 1.2, linetyp
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), se = FALSE, color = "purple", size = 1.2, linety
  labs(
    title = "Models for Unemployment vs Median Unemployment Duration",
    x = "Median Unemployment Duration (uempmed)",
    y = "Number of Unemployed Persons (unemploy)"
  ) +
  theme_minimal()

# Plot for uempmed ~ unemploy
plot2 <- ggplot(economics, aes(x = unemploy, y = uempmed)) +
  geom_point(color = "blue", alpha = 0.6) +  # Scatterplot
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, color = "red", size = 1.2) +  # Linear model
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), se = FALSE, color = "green", size = 1.2, linetype
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), se = FALSE, color = "purple", size = 1.2, linety
  labs(
    title = "Models for Median Unemployment Duration vs Number of Unemployed Persons",
    x = "Number of Unemployed Persons (unemploy)",
    y = "Median Unemployment Duration (uempmed)"
  ) +
  theme_minimal()

# Display the plots
print(plot1)
```



```
print(plot2)
```



##L eave-One-Out(LOOCV), 5-Fold, and 10-Fold Cross-Validation Using cv.glm function in the boot package.

```
# Load necessary library
library(boot)

# Function to calculate RMSE and MSE using cv.glm
calculate_cv_errors <- function(model, data, folds) {
  cv_result <- cv.glm(data = data, glmfit = model, K = folds)
  mse <- cv_result$delta[1]  # Mean Squared Error
  rmse <- sqrt(mse)  # Root Mean Squared Error
  return(c(MSE = mse, RMSE = rmse))
}

# Linear models
linear_model1 <- glm(unemploy ~ uempmed, data = economics)
linear_model2 <- glm(uempmed ~ unemploy, data = economics)

# Polynomial models for unemploy ~ uempmed
poly2_model1 <- glm(unemploy ~ poly(uempmed, 2), data = economics)
poly3_model1 <- glm(unemploy ~ poly(uempmed, 3), data = economics)

# Polynomial models for uempmed ~ unemploy
poly2_model2 <- glm(uempmed ~ poly(unemploy, 2), data = economics)
poly3_model2 <- glm(uempmed ~ poly(unemploy, 3), data = economics)

# Cross-validation for Leave-One-Out (LOOCV)
```

```r
loocv_results <- data.frame(
  Model = c("Linear (unemploy ~ uempmed)", "Linear (uempmed ~ unemploy)",
            "Poly2 (unemploy ~ uempmed)", "Poly3 (unemploy ~ uempmed)",
            "Poly2 (uempmed ~ unemploy)", "Poly3 (uempmed ~ unemploy)"),
  LOOCV_MSE = c(
    calculate_cv_errors(linear_model1, economics, nrow(economics))[1],
    calculate_cv_errors(linear_model2, economics, nrow(economics))[1],
    calculate_cv_errors(poly2_model1, economics, nrow(economics))[1],
    calculate_cv_errors(poly3_model1, economics, nrow(economics))[1],
    calculate_cv_errors(poly2_model2, economics, nrow(economics))[1],
    calculate_cv_errors(poly3_model2, economics, nrow(economics))[1]
  ),
  LOOCV_RMSE = c(
    calculate_cv_errors(linear_model1, economics, nrow(economics))[2],
    calculate_cv_errors(linear_model2, economics, nrow(economics))[2],
    calculate_cv_errors(poly2_model1, economics, nrow(economics))[2],
    calculate_cv_errors(poly3_model1, economics, nrow(economics))[2],
    calculate_cv_errors(poly2_model2, economics, nrow(economics))[2],
    calculate_cv_errors(poly3_model2, economics, nrow(economics))[2]
  )
)

# Cross-validation for 5-Fold and 10-Fold
folds_results <- data.frame(
  Model = loocv_results$Model,
  CV5_MSE = c(
    calculate_cv_errors(linear_model1, economics, 5)[1],
    calculate_cv_errors(linear_model2, economics, 5)[1],
    calculate_cv_errors(poly2_model1, economics, 5)[1],
    calculate_cv_errors(poly3_model1, economics, 5)[1],
    calculate_cv_errors(poly2_model2, economics, 5)[1],
    calculate_cv_errors(poly3_model2, economics, 5)[1]
  ),
  CV5_RMSE = c(
    calculate_cv_errors(linear_model1, economics, 5)[2],
    calculate_cv_errors(linear_model2, economics, 5)[2],
    calculate_cv_errors(poly2_model1, economics, 5)[2],
    calculate_cv_errors(poly3_model1, economics, 5)[2],
    calculate_cv_errors(poly2_model2, economics, 5)[2],
    calculate_cv_errors(poly3_model2, economics, 5)[2]
  ),
  CV10_MSE = c(
    calculate_cv_errors(linear_model1, economics, 10)[1],
    calculate_cv_errors(linear_model2, economics, 10)[1],
    calculate_cv_errors(poly2_model1, economics, 10)[1],
    calculate_cv_errors(poly3_model1, economics, 10)[1],
    calculate_cv_errors(poly2_model2, economics, 10)[1],
    calculate_cv_errors(poly3_model2, economics, 10)[1]
  ),
  CV10_RMSE = c(
    calculate_cv_errors(linear_model1, economics, 10)[2],
    calculate_cv_errors(linear_model2, economics, 10)[2],
    calculate_cv_errors(poly2_model1, economics, 10)[2],
    calculate_cv_errors(poly3_model1, economics, 10)[2],
    calculate_cv_errors(poly2_model2, economics, 10)[2],
    calculate_cv_errors(poly3_model2, economics, 10)[2]
  )
)

# Combine results
cv_results <- merge(loocv_results, folds_results, by = "Model")

# Print results
cat("Cross-Validation Results:\n")
```

```
## Cross-Validation Results:
```

```r
print(cv_results)
```

```
##                          Model    LOOCV_MSE  LOOCV_RMSE      CV5_MSE    CV5_RMSE
## 1 Linear (uempmed ~ unemploy) 4.159797e+00    2.039558 4.167349e+00    2.044519
## 2 Linear (unemploy ~ uempmed) 1.715211e+06 1309.660569 1.716834e+06 1308.072695
## 3  Poly2 (uempmed ~ unemploy) 3.005112e+00    1.733526 2.976779e+00    1.738061
## 4  Poly2 (unemploy ~ uempmed) 1.432531e+06 1196.883710 1.438461e+06 1192.033400
## 5  Poly3 (uempmed ~ unemploy) 3.009934e+00    1.734916 3.001550e+00    1.728629
## 6  Poly3 (unemploy ~ uempmed) 1.366405e+06 1168.933099 1.393148e+06 1169.190992
##       CV10_MSE   CV10_RMSE
## 1 4.139937e+00    2.039467
## 2 1.715213e+06 1310.585158
## 3 2.990761e+00    1.738491
## 4 1.433056e+06 1198.085516
## 5 3.011938e+00    1.735990
## 6 1.375421e+06 1166.010352
```

The concepts of Underfitting and Overfitting

Underfitting: Happens when the model is too simple to capture the underlying relationship in the data. - **Example**: Linear models (`unemploy ~ uempmed` and `uempmed ~ unemploy`) underfit the data, as seen in higher residual errors and less flexibility in the graphical fits. - **Detection**: High errors in both training and validation datasets during cross-validation.

Overfitting: Happens when the model is too complex, fitting noise rather than the true pattern. - **Example**: High-degree polynomial models (e.g., degree 10) show reduced training errors but generalize poorly, resulting in increased validation errors. - **Detection**: Large discrepancy between training and cross-validation errors.

Cross-Validation to Determine Fit: - **Purpose**: Balances underfitting and overfitting by evaluating model performance on unseen data. - **Steps**: - Apply cross-validation (e.g., LOOCV, k-fold) to compare validation errors. - Select the model with the lowest validation error while avoiding unnecessary complexity.

Variants of Cross-Validation

1. **Leave-One-Out Cross-Validation (LOOCV)**:
    - Splits data such that each observation is used once as the test set.
    - **Strength**: Provides an unbiased estimate of error.
    - **Weakness**: Computationally expensive.
2. **k-Fold Cross-Validation**:
    - Splits data into `k` folds; each fold is used once as the test set.
    - **Strength**: Reduces computational cost and variance compared to LOOCV.
    - **Common Choices**: 5-fold and 10-fold.

3. **Repeated k-Fold**:
    - Repeats k-fold CV multiple times with different splits.
    - **Strength**: Further reduces variance in error estimates.
4. **Monte Carlo Cross-Validation**:
    - Repeatedly splits data into random train-test sets.
    - **Strength**: Suitable for large datasets.