Cross Validation of Models

04-12-2024

Task 1

The subsection contains the following code:
- Fitting the models (linear, quadratic, cubic) on ISLR::Auto
- Validation set approach (train/test 50/50, 70/30)
- Cross validation (Leave-one0out and k fold 5&10)
- Comparison table and conclusions

```
library(ISLR)
df = ISLR::Auto
print(df[1,])
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504           12   70      1
##                        name
## 1 chevrolet chevelle malibu
```

```
str(df)
```

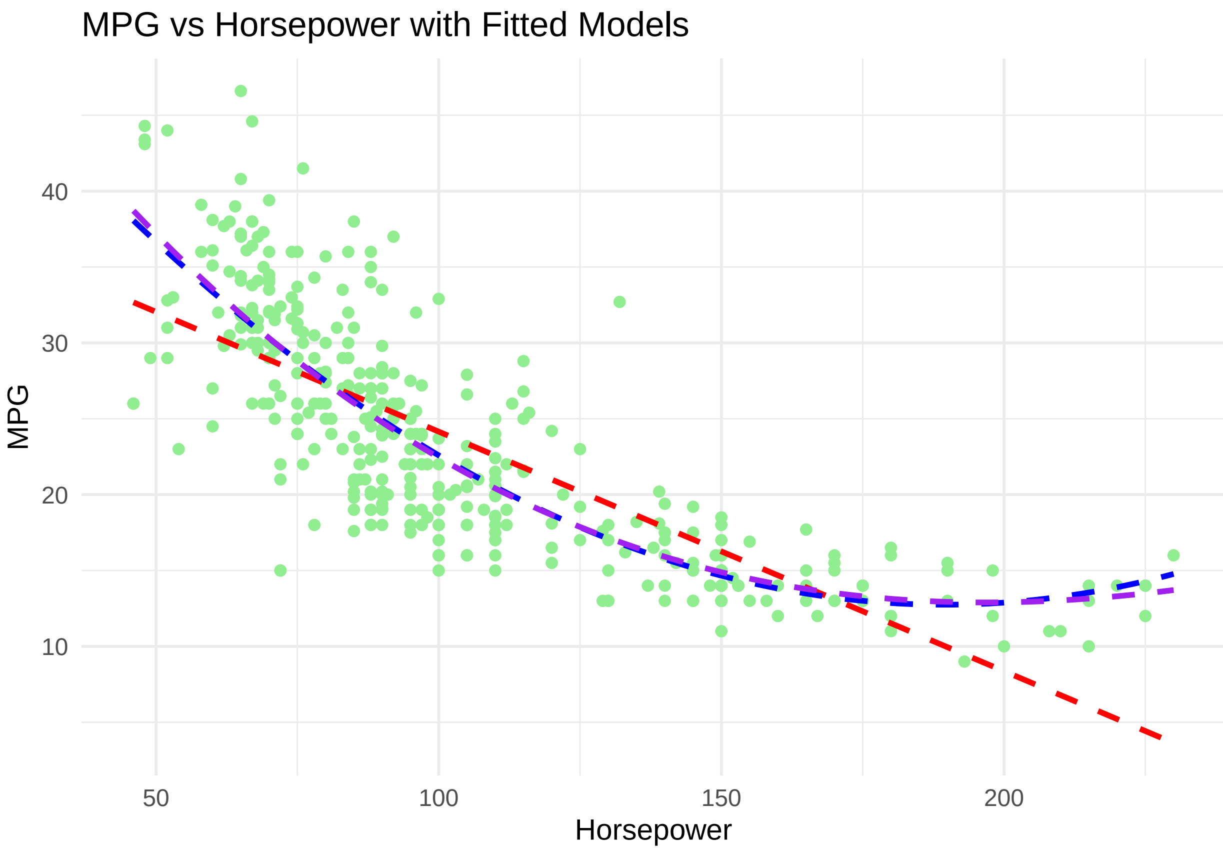
```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ 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 
```

1. Fitting the models

```
library(ggplot2)

m1 <- lm(mpg ~ horsepower, df)
m2 <- lm(mpg ~ poly(horsepower, 2), df)
m3 <- lm(mpg ~ poly(horsepower, 3), df)

ggplot(Auto, aes(x = horsepower, y = mpg)) +
  geom_point(color = "lightgreen") +
  stat_smooth(method = "lm", formula = y ~ x, color = "red", se = FALSE, linetype = "dashed") +
  stat_smooth(method = "lm", formula = y ~ poly(x, 2), color = "blue", se = FALSE, linetype = "dashed")
  stat_smooth(method = "lm", formula = y ~ poly(x, 3), color = "purple", se = FALSE, linetype = "dashed
  labs(title = "MPG vs Horsepower with Fitted Models",
       x = "Horsepower",
       y = "MPG") +
theme_minimal()
```



Observation: The linear model fails to capture the curve of the data and only follows the general trend. Both the quadratic and cubic models better fit the data, with the cubic model slightly more efficient (because it is able to capture subtle variations).

2. Validation set approach to compare models

```
set.seed(1234)

n <- nrow(df)
t_50 <- sample(1:n, size = 0.5*n)
t_70 <- sample(1:n, size = 0.7*n)


validate <- function(train_i) {
  ## Splitting train and test data
  train <- df[train_i, ]
  test <- df[-train_i, ]

  ## Fitting the models

  m1 <- lm(mpg ~ horsepower, train)
  m2 <- lm(mpg ~ poly(horsepower, 2), train)
  m3 <- lm(mpg ~ poly(horsepower, 3), train)

  ## Predictions
  pred1 <- predict(m1, newdata = test)
  pred2 <- predict(m2, newdata = test)
  pred3 <- predict(m3, newdata = test)
```

```r
  ## Taking metrics
  metrics <- data.frame(
    Model = c("Model_1", "Model_2", "Model_3"),
    RMSE = c(sqrt(mean((test$mpg - pred1)^2)),
             sqrt(mean((test$mpg - pred2)^2)),
             sqrt(mean((test$mpg - pred3)^2))),
    MSE = c(mean((test$mpg - pred1)^2),
            mean((test$mpg - pred2)^2),
            mean((test$mpg - pred3)^2)),
    MAD = c(mean(abs(test$mpg - pred1)),
            mean(abs(test$mpg - pred2)),
            mean(abs(test$mpg - pred3)))
  )
  return(metrics)
}


## Testing 50/50 split
metrics_50 <- validate(t_50)
print(metrics_50)
```

```
##     Model     RMSE      MSE      MAD
## 1 Model_1 4.892444 23.93601 3.888120
## 2 Model_2 4.242747 18.00090 3.198876
## 3 Model_3 4.266185 18.20033 3.204305
```

```r
## Testing 70/30 split
metrics_70 <- validate(t_70)
print(metrics_70)
```

```
##     Model     RMSE      MSE      MAD
## 1 Model_1 4.729490 22.36807 3.640734
## 2 Model_2 4.400080 19.36070 3.269700
## 3 Model_3 4.398326 19.34528 3.264258
```

Observation: For both 50/50 and 70/30 splits, the quadratic (model 2) and cubic (model 3) models show significantly less errors when compared to the linear (model 1) model. Model 2 (quadratic model) almost similar to model 3, meaning that complicating the models doesn't cause significant changes, and might lead to overfitting.

3 cv.glm for Cross Validation
Leave-one-out cross validation function

```r
library(boot)
df <- na.omit(df)
df <- as.data.frame(df)

## Leave one out cross validation
loocv <- function(m, df) {
  n <- nrow(df)
  mse <- numeric(n)
  mad <- numeric(n)

  for (i in 1:n) {
    train <- df[-i, ]
```

```r
    test <- df[i, , drop = FALSE]

    model <- lm(m, data = train)

    pred <- predict(model, newdata = test)
    mse[i] <- (test$mpg - pred)^2
    mad[i] <- abs(test$mpg - pred)
  }

  mse_mean <- mean(mse)
  rmse <- sqrt(mse_mean)
  mad_mean <- mean(mad)

  return(c(MSE = mse_mean, RMSE = rmse, MAD = mad_mean))

}

m1_loocv <- loocv(mpg ~ horsepower, df)
m2_loocv <- loocv(mpg ~ poly(horsepower, 2), df)
m3_loocv <- loocv(mpg ~ poly(horsepower, 3), df)

cat("M1 :\n")
```

```
## M1 :
```

```r
print(m1_loocv)
```

```
##       MSE      RMSE       MAD
## 24.231514  4.922552  3.848748
```

```r
cat("\nM2 :\n")
```

```
##
## M2 :
```

```r
print(m2_loocv)
```

```
##       MSE      RMSE       MAD
## 19.248213  4.387279  3.272041
```

```r
cat("\nM3 :\n")
```

```
##
## M3 :
```

```r
print(m3_loocv)
```

```
##       MSE      RMSE       MAD
## 19.334984  4.397156  3.276807
```

Observation: Quadratic and cubic models consistently outperform the linear model, as seen before. Model 2 shows slightly better metrics compared to model 3, meaning that further complexity does not improve performance.

k-fold cross-validation function

```r
## k-Fold Cross-Validation
kfold <- function(k, m, df) {
  n <- nrow(df)
```

```r
  folds <- sample(rep(1:k, length.out = n))
  mse <- numeric(k)
  mad <- numeric(k)

  for (i in 1:k) {
    train <- df[folds != i, ]
    test <- df[folds == i, ]

    model <- lm(m, data = train)


    pred <- predict(model, newdata = test)
    mse[i] <- mean((test$mpg - pred)^2)
    mad[i] <- mean(abs(test$mpg - pred))
  }

  mse_mean <- mean(mse)
  rmse <- sqrt(mse_mean)
  mad_mean <- mean(mad)

  return(c(MSE = mse_mean, RMSE = rmse, MAD = mad_mean))
}


m1_kfold_5 <- kfold(5, mpg ~ horsepower, df)
m2_kfold_5 <- kfold(5, mpg ~ poly(horsepower, 2), df)
m3_kfold_5 <- kfold(5, mpg ~ poly(horsepower, 3), df)

cat("M1 :\n")
```

```
## M1 :
```

```r
print(m1_kfold_5)
```

```
##       MSE      RMSE       MAD
## 24.348035  4.934373  3.855729
```

```r
cat("\nM2 :\n")
```

```
##
## M2 :
```

```r
print(m2_kfold_5)
```

```
##       MSE      RMSE       MAD
## 19.093742  4.369639  3.263229
```

```r
cat("\nM3 :\n")
```

```
##
## M3 :
```

```r
print(m3_kfold_5)
```

```
##       MSE      RMSE       MAD
## 19.405566  4.405175  3.282339
```

```r
m1_kfold_10 <- kfold(10, mpg ~ horsepower, df)
m2_kfold_10 <- kfold(10, mpg ~ poly(horsepower, 2), df)
m3_kfold_10 <- kfold(10, mpg ~ poly(horsepower, 3), df)

cat("M1 :\n")
```

```
## M1 :
```

```r
print(m1_kfold_10)
```

```
##       MSE      RMSE       MAD
## 24.213027  4.920673  3.846950
```

```r
cat("\nM2 :\n")
```

```
##
## M2 :
```

```r
print(m2_kfold_10)
```

```
##       MSE      RMSE       MAD
## 19.294512  4.392552  3.275592
```

```r
cat("\nM3 :\n")
```

```
##
## M3 :
```

```r
print(m3_kfold_10)
```

```
##       MSE      RMSE       MAD
## 19.257667  4.388356  3.266041
```

Observation: Both 5-fold and 10-fold cross validation show that model 2 and model 3 performance much better than model 1. The differences between model 2 and 3 are less, showing that complicating the model does not improve performance. Also, increasing the fold size also does not significantly change the performance of the models.

4. Comparison of results

```r
m1_loocv <- loocv(mpg ~ horsepower, df)
m2_loocv <- loocv(mpg ~ poly(horsepower, 2), df)
m3_loocv <- loocv(mpg ~ poly(horsepower, 3), df)

m1_kfold_5 <- kfold(5, mpg ~ horsepower, df)
m2_kfold_5 <- kfold(5, mpg ~ poly(horsepower, 2), df)
m3_kfold_5 <- kfold(5, mpg ~ poly(horsepower, 3), df)

m1_kfold_10 <- kfold(10, mpg ~ horsepower, df)
m2_kfold_10 <- kfold(10, mpg ~ poly(horsepower, 2), df)
m3_kfold_10 <- kfold(10, mpg ~ poly(horsepower, 3), df)

results_table <- data.frame(
  Method = c(
    "Train_50_MSE", "Train_50_RMSE", "Train_50_MAD",
    "Train_70_MSE", "Train_70_RMSE", "Train_70_MAD",
    "LOOCV_MSE", "LOOCV_RMSE", "LOOCV_MAD",
    "kFold_5_MSE", "kFold_5_RMSE", "kFold_5_MAD",
    "kFold_10_MSE", "kFold_10_RMSE", "kFold_10_MAD"
```

```r
  ),
  Model_1 = c(
    metrics_50$MSE[1], metrics_50$RMSE[1], metrics_50$MAD[1],
    metrics_70$MSE[1], metrics_70$RMSE[1], metrics_70$MAD[1],
    m1_loocv["MSE"], m1_loocv["RMSE"], m1_loocv["MAD"],
    m1_kfold_5["MSE"], m1_kfold_5["RMSE"], m1_kfold_5["MAD"],
    m1_kfold_10["MSE"], m1_kfold_10["RMSE"], m1_kfold_10["MAD"]
  ),
  Model_2 = c(
    metrics_50$MSE[2], metrics_50$RMSE[2], metrics_50$MAD[2],
    metrics_70$MSE[2], metrics_70$RMSE[2], metrics_70$MAD[2],
    m2_loocv["MSE"], m2_loocv["RMSE"], m2_loocv["MAD"],
    m2_kfold_5["MSE"], m2_kfold_5["RMSE"], m2_kfold_5["MAD"],
    m2_kfold_10["MSE"], m2_kfold_10["RMSE"], m2_kfold_10["MAD"]
  ),
  Model_3 = c(
    metrics_50$MSE[3], metrics_50$RMSE[3], metrics_50$MAD[3],
    metrics_70$MSE[3], metrics_70$RMSE[3], metrics_70$MAD[3],
    m3_loocv["MSE"], m3_loocv["RMSE"], m3_loocv["MAD"],
    m3_kfold_5["MSE"], m3_kfold_5["RMSE"], m3_kfold_5["MAD"],
    m3_kfold_10["MSE"], m3_kfold_10["RMSE"], m3_kfold_10["MAD"]
  )
)

print(results_table)
```

```
##           Method   Model_1   Model_2   Model_3
## 1   Train_50_MSE 23.936012 18.000905 18.200335
## 2  Train_50_RMSE  4.892444  4.242747  4.266185
## 3   Train_50_MAD  3.888120  3.198876  3.204305
## 4   Train_70_MSE 22.368072 19.360702 19.345275
## 5  Train_70_RMSE  4.729490  4.400080  4.398326
## 6   Train_70_MAD  3.640734  3.269700  3.264258
## 7      LOOCV_MSE 24.231514 19.248213 19.334984
## 8     LOOCV_RMSE  4.922552  4.387279  4.397156
## 9      LOOCV_MAD  3.848748  3.272041  3.276807
## 10   kFold_5_MSE 24.040116 19.236382 19.113743
## 11  kFold_5_RMSE  4.903072  4.385930  4.371927
## 12   kFold_5_MAD  3.837464  3.281453  3.266497
## 13  kFold_10_MSE 24.176378 19.212825 19.279811
## 14 kFold_10_RMSE  4.916948  4.383244  4.390878
## 15  kFold_10_MAD  3.845458  3.273027  3.273787
```

Observation:
1. Model 2 constantly performs the best across all the validation methods for all three metrics. This could be because the quadratic model fits the dataset better than cubic and linear models.
2. Model 1 has the highest error metrics across all the methods, meaning that it underfits the given data.
3. Model 3 is only slightly worse than Model 2 across all methods, which is likely due to overfitting. This shows that additional complexity does not mean there will be a significant improvement in the performance.
4. kfold_10 mostly produces better results than kfold_5, because of the larger training set.

For the given dataset, Model 2 seems to be the most accurate model across all validation metrics.

Task 2

The subsection contains the following code:
- Fitting the models (linear, logarithmic, 2nd, 3rd, and 10th degree polynomial) on ggplot2::economics
- Cross validation (Leave-one0out and k fold 5&10)
- Comparison table and conclusions

```
library(ggplot2)

df1 <- ggplot2::economics
print(df1[1,])
```

```
## # A tibble: 1 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
```

```
str(df1)
```

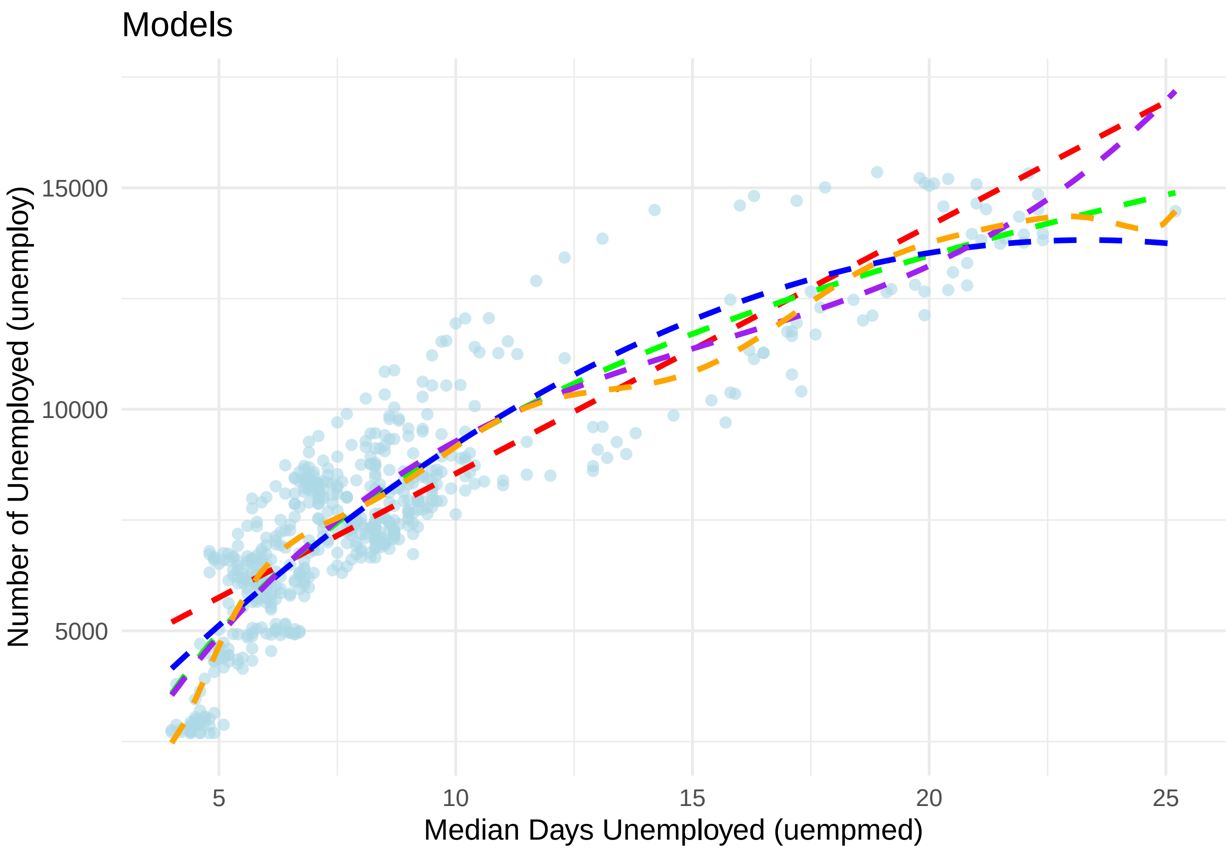
```
## spc_tbl_ [574 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ date    : Date[1:574], format: "1967-07-01" "1967-08-01" ...
##  $ pce     : num [1:574] 507 510 516 512 517 ...
##  $ pop     : num [1:574] 198712 198911 199113 199311 199498 ...
##  $ psavert : num [1:574] 12.6 12.6 11.9 12.9 12.8 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 ...
```

1. Fitting the models

```
m1 <- lm(unemploy ~ uempmed, data = df1)
m2 <- lm(unemploy ~ log(uempmed), data = df1)
m3 <- lm(unemploy ~ poly(uempmed, 2), data = df1)
m4 <- lm(unemploy ~ poly(uempmed, 3), data = df1)
m5 <- lm(unemploy ~ poly(uempmed, 10), data = df1)
```

2. Plotting all models

```
ggplot(df1, aes(x = uempmed, y = unemploy)) +
  geom_point(color = "lightblue", alpha = 0.6) +
  geom_smooth(method = "lm", formula = y ~ x, color = "red", se = FALSE, linetype = "dashed") +
  geom_smooth(method = "lm", formula = y ~ log(x), color = "green", se = FALSE, linetype = "dashed") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), color = "blue", se = FALSE, linetype = "dashed")
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), color = "purple", se = FALSE, linetype = "dashed
  geom_smooth(method = "lm", formula = y ~ poly(x, 10), color = "orange", se = FALSE, linetype = "dashe
  labs(title = "Models",
       x = "Median Days Unemployed (uempmed)",
       y = "Number of Unemployed (unemploy)") +
  theme_minimal()
```



Observation: The linear model is able to show the general trend of the data but fail to capture the curve. Logarithmic and polynomial models of higher degrees provide a better fit, but polynomial models provide a risk of overfitting, which is particularly seen at the edges of 10th degree model.

3. Using cv.glm for leave-one-out cross validation

```
## Leave-one-out cross validation

loocv1 <- function(m, df) {
  n <- nrow(df)
  mse <- numeric(n)
  mad <- numeric(n)

  for (i in 1:n) {
    train <- df[-i, ]
    test <- df[i, , drop = FALSE]

    model <- lm(m, data = train)

    pred <- predict(model, newdata = test)
    mse[i] <- (test$unemploy - pred)^2
    mad[i] <- abs(test$unemploy - pred)
  }

  mse_mean <- mean(mse)
  rmse <- sqrt(mse_mean)
  mad_mean <- mean(mad)
```

```r
  return(c(MSE = mse_mean, RMSE = rmse, MAD = mad_mean))
}

m1_loocv <- loocv1(unemploy ~ uempmed, df1)
m2_loocv <- loocv1(unemploy ~ log(uempmed), df1)
m3_loocv <- loocv1(unemploy ~ poly(uempmed, 2), df1)
m4_loocv <- loocv1(unemploy ~ poly(uempmed, 3), df1)
m5_loocv <- loocv1(unemploy ~ poly(uempmed, 10), df1)

cat("M1 :\n")
```

```
## M1 :
```

```r
print(m1_loocv)
```

```
##         MSE        RMSE         MAD
## 1715210.805    1309.661    1040.017
```

```r
cat("\nM2 :\n")
```

```
##
## M2 :
```

```r
print(m2_loocv)
```

```
##          MSE         RMSE          MAD
## 1333996.6577    1154.9877     980.4186
```

```r
cat("\nM3 :\n")
```

```
##
## M3 :
```

```r
print(m3_loocv)
```

```
##         MSE        RMSE         MAD
## 1432530.614    1196.884    1012.255
```

```r
cat("\nM4 :\n")
```

```
##
## M4 :
```

```r
print(m4_loocv)
```

```
##          MSE         RMSE          MAD
## 1366404.5907    1168.9331     984.5664
```

```r
cat("\nM5 :\n")
```

```
##
## M5 :
```

```r
print(m5_loocv)
```

```
##         MSE        RMSE         MAD
## 4530738.278    2128.553     996.629
```

Observation: The logarithmic model (M2) performance better than all the others, meaning that it is best fit for data under LOOCV. Higher degree polynomials show a decrease n performance, with M5 being the worst due to overfitting.

3. cv.glm for k-fold cross validation

```r
## k-fold cross validation

kfold2 <- function(k, m, df) {
  n <- nrow(df)
  folds <- sample(rep(1:k, length.out = n))
  mse <- numeric(k)
  mad <- numeric(k)

  for (i in 1:k) {
    train <- df[folds != i, ]
    test <- df[folds == i, ]

    model <- lm(m, data = train)

    pred <- predict(model, newdata = test)
    mse[i] <- mean((test$unemploy - pred)^2)
    mad[i] <- mean(abs(test$unemploy - pred))
  }

  mse_mean <- mean(mse)
  rmse <- sqrt(mse_mean)
  mad_mean <- mean(mad)

  return(c(MSE = mse_mean, RMSE = rmse, MAD = mad_mean))
}

m1_kfold_5 <- kfold2(5, unemploy ~ uempmed, df1)
m2_kfold_5 <- kfold2(5, unemploy ~ log(uempmed), df1)
m3_kfold_5 <- kfold2(5, unemploy ~ poly(uempmed, 2), df1)
m4_kfold_5 <- kfold2(5, unemploy ~ poly(uempmed, 3), df1)
m5_kfold_5 <- kfold2(5, unemploy ~ poly(uempmed, 10), df1)

cat("M1 :\n")
```

```
## M1 :
```

```r
print(m1_kfold_5)
```

```
##         MSE        RMSE         MAD
## 1714832.709    1309.516    1039.558
```

```r
cat("\nM2 :\n")
```

```
##
## M2 :
```

```r
print(m2_kfold_5)
```

```
##          MSE         RMSE          MAD
## 1337076.8047    1156.3204     980.5561
```

```r
cat("\nM3 :\n")
```

```
##
## M3 :
```

```r
print(m3_kfold_5)
```

```
##         MSE        RMSE         MAD
## 1428091.558    1195.028    1011.430
```

```r
cat("\nM4 :\n")
```

```
##
## M4 :
```

```r
print(m4_kfold_5)
```

```
##          MSE          RMSE           MAD
## 1368433.0556     1169.8004      984.0774
```

```r
cat("\nM5 :\n")
```

```
##
## M5 :
```

```r
print(m5_kfold_5)
```

```
##          MSE          RMSE           MAD
## 4054716.4984     2013.6327      986.2971
```

```r
m1_kfold_10 <- kfold2(10, unemploy ~ uempmed, df1)
m2_kfold_10 <- kfold2(10, unemploy ~ log(uempmed), df1)
m3_kfold_10 <- kfold2(10, unemploy ~ poly(uempmed, 2), df1)
m4_kfold_10 <- kfold2(10, unemploy ~ poly(uempmed, 3), df1)
m5_kfold_10 <- kfold2(10, unemploy ~ poly(uempmed, 10), df1)

cat("M1 :\n")
```

```
## M1 :
```

```r
print(m1_kfold_10)
```

```
##         MSE        RMSE         MAD
## 1714941.257    1309.558    1040.462
```

```r
cat("\nM2 :\n")
```

```
##
## M2 :
```

```r
print(m2_kfold_10)
```

```
##          MSE          RMSE           MAD
## 1333051.9786     1154.5787      979.3552
```

```r
cat("\nM3 :\n")
```

```
##
## M3 :
```

```r
print(m3_kfold_10)
```

```
##         MSE        RMSE         MAD
## 1428355.268    1195.138    1011.175
```

```r
cat("\nM4 :\n")
```

```
##
## M4 :
```

```r
print(m4_kfold_10)
```

```
##          MSE         RMSE          MAD
## 1391136.2988    1179.4644     987.4054
```

```r
cat("\nM5 :\n")
```

```
##
## M5 :
```

```r
print(m5_kfold_10)
```

```
##         MSE        RMSE         MAD
## 1474671.631    1214.361     937.643
```

Observation: In both 5-fold and 10-fold cross validation, the logarithmic model performance better than the other models. Higher degree polynomials show a decrease n performance, with M5 being the worst in both cases due to overfitting. Also, increasing the fold size also does not significantly change the performance of the models.

```r
m1_loocv <- loocv1(unemploy ~ uempmed, df1)
m2_loocv <- loocv1(unemploy ~ log(uempmed), df1)
m3_loocv <- loocv1(unemploy ~ poly(uempmed, 2), df1)
m4_loocv <- loocv1(unemploy ~ poly(uempmed, 3), df1)
m5_loocv <- loocv1(unemploy ~ poly(uempmed, 10), df1)

m1_kfold_5 <- kfold2(5, unemploy ~ uempmed, df1)
m2_kfold_5 <- kfold2(5, unemploy ~ log(uempmed), df1)
m3_kfold_5 <- kfold2(5, unemploy ~ poly(uempmed, 2), df1)
m4_kfold_5 <- kfold2(5, unemploy ~ poly(uempmed, 3), df1)
m5_kfold_5 <- kfold2(5, unemploy ~ poly(uempmed, 10), df1)

m1_kfold_10 <- kfold2(10, unemploy ~ uempmed, df1)
m2_kfold_10 <- kfold2(10, unemploy ~ log(uempmed), df1)
m3_kfold_10 <- kfold2(10, unemploy ~ poly(uempmed, 2), df1)
m4_kfold_10 <- kfold2(10, unemploy ~ poly(uempmed, 3), df1)
m5_kfold_10 <- kfold2(10, unemploy ~ poly(uempmed, 10), df1)
```

```r
results_table <- data.frame(
  Method = c(
    "LOOCV_MSE", "LOOCV_RMSE", "LOOCV_MAD",
    "kFold_5_MSE", "kFold_5_RMSE", "kFold_5_MAD",
    "kFold_10_MSE", "kFold_10_RMSE", "kFold_10_MAD"
  ),
  Model_1 = c(
    m1_loocv["MSE"], m1_loocv["RMSE"], m1_loocv["MAD"],
    m1_kfold_5["MSE"], m1_kfold_5["RMSE"], m1_kfold_5["MAD"],
    m1_kfold_10["MSE"], m1_kfold_10["RMSE"], m1_kfold_10["MAD"]
  ),
  Model_2 = c(
    m2_loocv["MSE"], m2_loocv["RMSE"], m2_loocv["MAD"],
    m2_kfold_5["MSE"], m2_kfold_5["RMSE"], m2_kfold_5["MAD"],
    m2_kfold_10["MSE"], m2_kfold_10["RMSE"], m2_kfold_10["MAD"]
  ),
```

```r
  Model_3 = c(
    m3_loocv["MSE"], m3_loocv["RMSE"], m3_loocv["MAD"],
    m3_kfold_5["MSE"], m3_kfold_5["RMSE"], m3_kfold_5["MAD"],
    m3_kfold_10["MSE"], m3_kfold_10["RMSE"], m3_kfold_10["MAD"]
  ),
  Model_4 = c(
    m4_loocv["MSE"], m4_loocv["RMSE"], m4_loocv["MAD"],
    m4_kfold_5["MSE"], m4_kfold_5["RMSE"], m4_kfold_5["MAD"],
    m4_kfold_10["MSE"], m4_kfold_10["RMSE"], m4_kfold_10["MAD"]
  ),
  Model_5 = c(
    m5_loocv["MSE"], m5_loocv["RMSE"], m5_loocv["MAD"],
    m5_kfold_5["MSE"], m5_kfold_5["RMSE"], m5_kfold_5["MAD"],
    m5_kfold_10["MSE"], m5_kfold_10["RMSE"], m5_kfold_10["MAD"]
  )
)

print(results_table)
```

```
##          Method     Model_1      Model_2     Model_3      Model_4      Model_5
## 1     LOOCV_MSE 1715210.805 1333996.6577 1432530.614 1366404.5907  4530738.278
## 2    LOOCV_RMSE    1309.661    1154.9877    1196.884    1168.9331     2128.553
## 3     LOOCV_MAD    1040.017     980.4186    1012.255     984.5664      996.629
## 4   kFold_5_MSE 1713108.441 1341791.0552 1428880.356 1374147.6162 14081243.181
## 5  kFold_5_RMSE    1308.858    1158.3570    1195.358    1172.2404     3752.498
## 6   kFold_5_MAD    1039.948     982.0662    1011.632     989.5521     1066.861
## 7  kFold_10_MSE 1719980.138 1331251.6329 1438968.472 1361245.5748 10070108.907
## 8 kFold_10_RMSE    1311.480    1153.7988    1199.570    1166.7243     3173.343
## 9  kFold_10_MAD    1040.639     979.5118    1014.158     982.8523     1045.104
```

Observation:
1. The linear model (model 1) constantly underperformans in all scenarios, meaning that it cannot fully capture the non-linear relationships between the data. 2. Model 2 (logarithmic) constantly performs best across all methods, meaning that it provides the best fit for the data.
3. Models 3 and 4 (2nd and 3rd degree polynomial) doesn't show significant differences from Model 2, meaning that the additional complexity didn't help the dataset. 4. Model 5 (10th degree polynomial) has a very high MSE, especially for LOOcv and kfold-10, meaning that the model is severely overfitting.

Model 2 is the best performing model for the given dataset.

4. Concepts of underfitting, overfitting

- Underfitting

Underfitting occurs when the model cannot capture the underlying relationship between data in the dataset. In this context, the linear model (M1) provides an example for underfitting since it failed to capture the non-linear relationship between 'unemploy' and 'uempmed', resulting in very high error metrics.

- Overfitting

Overfitting occurs when the model is extremely complex that it starts to take in the noise from the dataset, going beyond the underlying relationship. In this context, the 10-degree polynomial model provides an example for overfitting. While it may seem that the model fit the train dataset well, the metrics show that it worsens on the test data, providing high errors.

Applying Cross-Validation for appropriate model fit

Cross validations helps identify the best model fit by splitting the given dataset into train and test and evaluating the model on untrained data. Cross-validation ensures a balance between underfitting (linear model) and overfitting (higher degree models) by providing unbiased estimates.

- **Leave-One-Out Cross-Validation**: In LOOCV, each observation is added to the test set once while the remaining observations are a part of the training set. Though the method is intensive, it does provide a thorough assessment. In this context, the LOOCV confirmed that the logarithmic model performs better while highlight the disadvantages of higher degree models. However, one major problem with using LOOCV is that it is expensive to compute, especially if the dataset is large.
- **K-Fold Cross-Validation**: In this case, the dataset is divided into k subsets (called a fold). Each fold is used as the test data while the rest is used as the training set. In this case, we use both 5 and 10 folds. Both approaches show that logarithmic model performs better but increasing the number of folds did not increase the performance. K-fold is also less expensive then LOOCV.

When unbiased estimates of the test MSE are required, LOOCV is better. However, K-fold CV provides lower variance and is computationally more efficient. The computational burden of k-fold CV depends heavily on the value of k (i.e., the number of folds). When comparing competing models, the primary focus is on relative performance than estimating the exact test MSE, so the bias in MSE estimation is less important. From extended experimental studies, it was also concluded that k = 5 or k = 10 provides a good balance since they avoid excessive bias while keeping the variance low.