Exercise 6

107.330 - Statistical Simulation and Computerintensive Methods, WS24

11912007 - Yahya Jabary

09.10.2024

```
set.seed(11912007)
```

Task 1

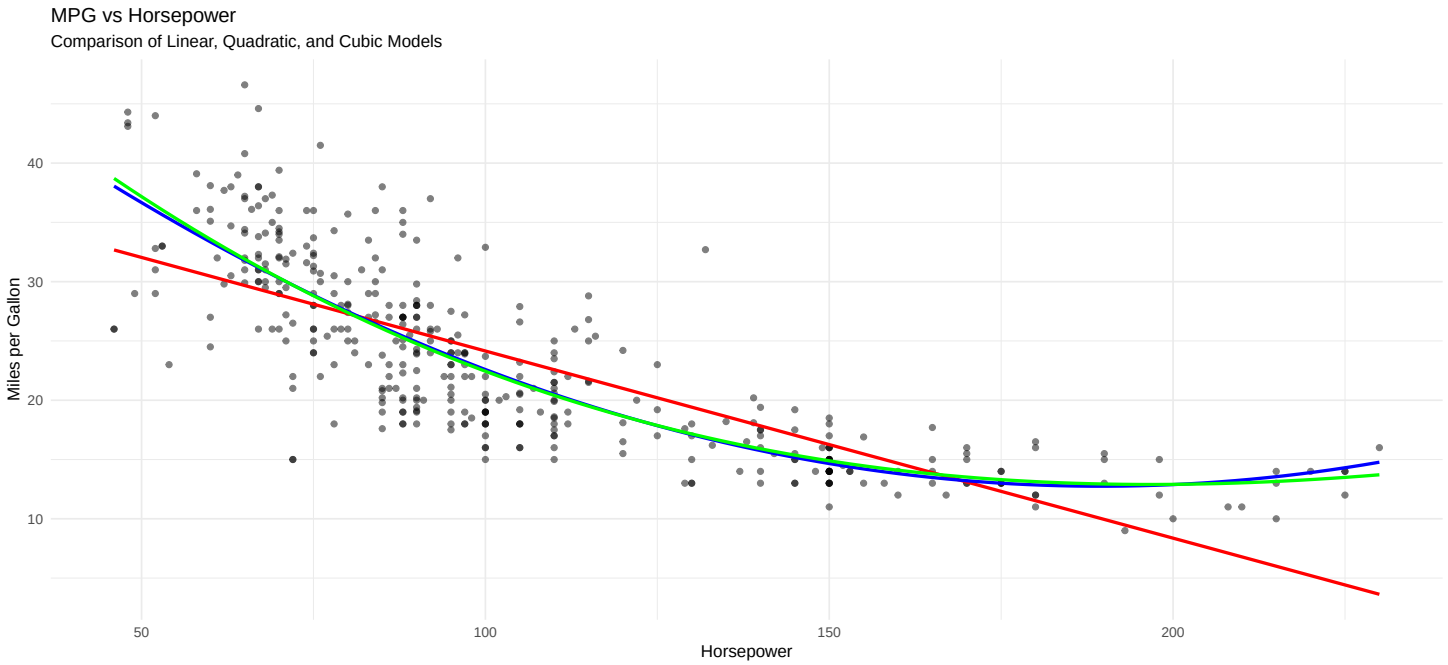
We will work with the dataset Auto in the ISLR package. Obtain information on the data set, its structure and the real world meaning of its variables from the help page.

1.1 Fit the 3 models. Visualise all 3 models in comparison added to a scatterplot of the input data.

The Auto dataset contains information about various car attributes, including mpg (miles per gallon) and horsepower.

```
data(Auto, package = "ISLR")
model1 <- lm(mpg ~ horsepower, data = Auto)
model2 <- lm(mpg ~ poly(horsepower, 2), data = Auto)
model3 <- lm(mpg ~ poly(horsepower, 3), data = Auto)

# scatterplot
ggplot(Auto, aes(x = horsepower, y = mpg)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", formula = y ~ x, color = "red", se = FALSE) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), color = "blue", se = FALSE) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), color = "green", se = FALSE) +
  labs(title = "MPG vs Horsepower",
       subtitle = "Comparison of Linear, Quadratic, and Cubic Models",
       x = "Horsepower", y = "Miles per Gallon") +
  theme_minimal()
```



1.2 Use the validation set approach to compare the models. Use once a train/test split of 50%/50% and once 70%/30%. Choose the best model based on Root Mean Squared Error, Mean Squared Error and Median Absolute Deviation.

Here we will use the 50/50 and 70/30 holdout split.

```
calc_metrics <- function(actual, predicted) {
  mse <- mean((actual - predicted)^2)
  rmse <- sqrt(mse)
  mad <- median(abs(actual - predicted))
  return(c(MSE = mse, RMSE = rmse, MAD = mad))
}

# 50/50 split
index_50 <- sample(1:nrow(Auto), 0.5 * nrow(Auto))
train_50 <- Auto[index_50, ]
test_50 <- Auto[-index_50, ]

# 70/30 split
index_70 <- sample(1:nrow(Auto), 0.7 * nrow(Auto))
train_70 <- Auto[index_70, ]
test_70 <- Auto[-index_70, ]

evaluate_models <- function(train, test) {
  models <- list(
    lm(mpg ~ horsepower, data = train),
    lm(mpg ~ poly(horsepower, 2), data = train),
    lm(mpg ~ poly(horsepower, 3), data = train)
  )
  results <- lapply(models, function(model) {
    predictions <- predict(model, newdata = test)
    calc_metrics(test$mpg, predictions)
  })
  return(do.call(rbind, results))
}
results_50 <- evaluate_models(train_50, test_50)
results_70 <- evaluate_models(train_70, test_70)
kable(results_50, caption = "50/50 Split")
```

Table 1: 50/50 Split

MSE	RMSE	MAD
21.16673	4.600732	2.373069
17.57193	4.191889	2.192290
17.98532	4.240910	2.240561

```
kable(results_70, caption = "70/30 Split")
```

Table 2: 70/30 Split

MSE	RMSE	MAD
19.21738	4.383763	2.677396
19.19705	4.381443	2.176978
19.69481	4.437883	2.268290

1.3 Use the `cv.glm` function in the `boot` package for the following steps. Use `cv.glm` for Leave-one-out Cross Validation to compare the models above. Use `cv.glm` for 5-fold and 10-fold Cross Validation to compare the models above.

Here we will use the `cv.glm` function to perform Leave-one-out, 5-fold and 10-fold Cross Validation.

```
cv_models <- function(formula, data, K) {
  model <- glm(formula, data = data)
```

```r
  cv_error <- cv.glm(data, model, K = K)
  return(cv_error$delta[1])  # return mean cross-validated error
}
formulas <- list(
  mpg ~ horsepower,
  mpg ~ poly(horsepower, 2),
  mpg ~ poly(horsepower, 3)
)
cv_results <- lapply(formulas, function(formula) {
  loocv <- cv_models(formula, Auto, nrow(Auto))
  cv5 <- cv_models(formula, Auto, 5)
  cv10 <- cv_models(formula, Auto, 10)
  return(c(LOOCV = loocv, CV5 = cv5, CV10 = cv10))
})
cv_results <- do.call(rbind, cv_results)
rownames(cv_results) <- c("Linear", "Quadratic", "Cubic")
kable(cv_results, caption = "Cross-Validation Results (MSE)")
```

Table 3: Cross-Validation Results (MSE)

	LOOCV	CV5	CV10
Linear	24.23151	24.23725	24.20617
Quadratic	19.24821	19.17884	19.17745
Cubic	19.33498	19.16479	19.37457

1.4 Compare all results from 2 and 3. in a table and draw your conclusions.

Here we will compare the results from the validation set approach and cross-validation.

```r
summary_table_part1 <- data.frame(
  Model = c("Linear", "Quadratic", "Cubic"),
  `50/50_MSE` = results_50[, "MSE"],
  `50/50_RMSE` = results_50[, "RMSE"],
  `50/50_MAD` = results_50[, "MAD"],
  `70/30_MSE` = results_70[, "MSE"],
  `70/30_RMSE` = results_70[, "RMSE"],
  `70/30_MAD` = results_70[, "MAD"]
)

summary_table_part2 <- data.frame(
  Model = c("Linear", "Quadratic", "Cubic"),
  LOOCV = cv_results[, "LOOCV"],
  CV5 = cv_results[, "CV5"],
  CV10 = cv_results[, "CV10"]
)

kable(summary_table_part1, caption = "Summary of Model Comparison (Part 1: Holdout Split)")
```

Table 4: Summary of Model Comparison (Part 1: Holdout Split)

Model	X50.50_MSE	X50.50_RMSE	X50.50_MAD	X70.30_MSE	X70.30_RMSE	X70.30_MAD
Linear	21.16673	4.600732	2.373069	19.21738	4.383763	2.677396
Quadratic	17.57193	4.191889	2.192290	19.19705	4.381443	2.176978
Cubic	17.98532	4.240910	2.240561	19.69481	4.437883	2.268290

```r
kable(summary_table_part2, caption = "Summary of Model Comparison (Part 2: Cross-Validation)")
```

Table 5: Summary of Model Comparison (Part 2: Cross-Validation)

	Model	LOOCV	CV5	CV10
Linear	Linear	24.23151	24.23725	24.20617

	Model	LOOCV	CV5	CV10
Quadratic	Quadratic	19.24821	19.17884	19.17745
Cubic	Cubic	19.33498	19.16479	19.37457

Based on all metrics (MSE, RMSE, MAD) and cross-validation results, the quadratic model (`poly(horsepower, 2)`) generally performs the best, followed closely by the cubic model. The linear model consistently shows the poorest performance. We can't compare models across multiple data splits and cross-validation methods but only within each method to ensure consistency.

The scatterplot with fitted lines visually confirms these findings, showing that the quadratic and cubic models capture the non-linear relationship between mpg and horsepower better than the linear model.

The following observations are noteworthy:

- The relative performance of the models is consistent across different validation methods (50/50 split, 70/30 split, LOOCV, 5-fold CV, 10-fold CV), which increases our confidence in the results.
- The cubic model sometimes performs slightly worse than the quadratic model, especially in cross-validation. This suggests that the cubic model might be overfitting the data slightly and that a quadratic model provides a better balance between model complexity and predictive power.
- The 70/30 split generally gives slightly lower error estimates than the 50/50 split, likely due to having more data for training.
- LOOCV, 5-fold CV, and 10-fold CV give very similar results, with LOOCV typically showing slightly higher error estimates.

Task 2

Load the data set 'economics' from the package 'ggplot2'.

2.1 Fit the following models to explain the number of unemployed persons 'unemploy' by the median number of days unemployed 'uempmed' and vice versa: linear model, an appropriate exponential or logarithmic model (which one is appropriate depends on which is the dependent or independent variable), polynomial model of 2nd, 3rd and 10th degree

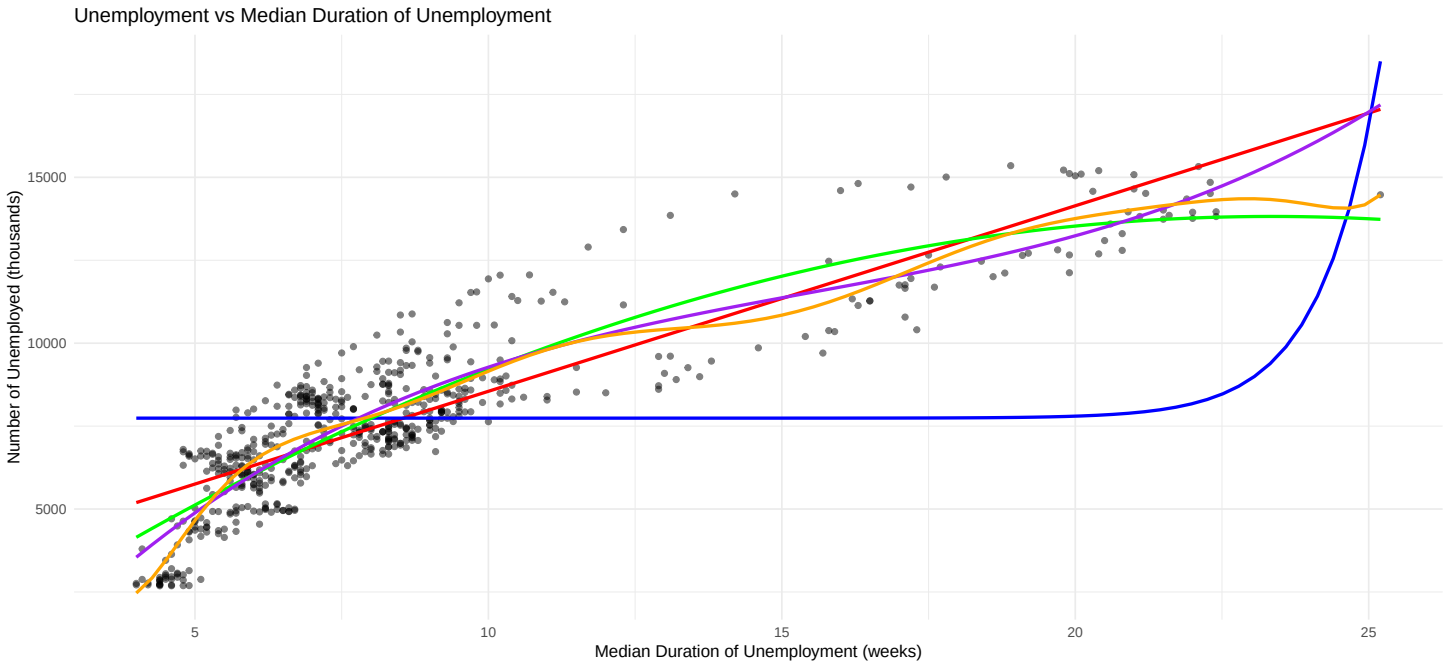
```
data(economics, package = "ggplot2")

# unemply as dependent variable
models_1 <- list(
  lm_1 = lm(unemploy ~ uempmed, data = economics),
  exp_1 = lm(log(unemploy) ~ uempmed, data = economics),
  poly2_1 = lm(unemploy ~ poly(uempmed, 2), data = economics),
  poly3_1 = lm(unemploy ~ poly(uempmed, 3), data = economics),
  poly10_1 = lm(unemploy ~ poly(uempmed, 10), data = economics)
)
# uempmed as dependent variable
models_2 <- list(
  lm_2 = lm(uempmed ~ unemploy, data = economics),
  log_2 = lm(uempmed ~ log(unemploy), data = economics),
  poly2_2 = lm(uempmed ~ poly(unemploy, 2), data = economics),
  poly3_2 = lm(uempmed ~ poly(unemploy, 3), data = economics),
  poly10_2 = lm(uempmed ~ poly(unemploy, 10), data = economics)
)
models <- c(models_1, models_2)
```

2.2 Plot the corresponding data and add all the models for comparison.

```
# unemploy as dependent variable
p1 <- ggplot(economics, aes(x = uempmed, y = unemploy)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", formula = y ~ x, color = "red", se = FALSE) +
  geom_smooth(method = "lm", formula = y ~ exp(x), color = "blue", se = FALSE) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), color = "green", se = FALSE) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), color = "purple", se = FALSE) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 10), color = "orange", se = FALSE) +
  labs(title = "Unemployment vs Median Duration of Unemployment",
```

```
       x = "Median Duration of Unemployment (weeks)",
       y = "Number of Unemployed (thousands)") +
  theme_minimal()
# uempmed as dependent variable
p2 <- ggplot(economics, aes(x = unemploy, y = uempmed)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", formula = y ~ x, color = "red", se = FALSE) +
  geom_smooth(method = "lm", formula = y ~ log(x), color = "blue", se = FALSE) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), color = "green", se = FALSE) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 3), color = "purple", se = FALSE) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 10), color = "orange", se = FALSE) +
  labs(title = "Median Duration of Unemployment vs Unemployment",
       x = "Number of Unemployed (thousands)",
       y = "Median Duration of Unemployment (weeks)") +
  theme_minimal()
print(p1)
```




```
print(p2)
```

2.3 Use the `cv.glm` function in the `boot` package for the following steps. Compare the Root Mean Squared Error and Mean Squared Error. Use `cv.glm` for Leave-one-out Cross Validation to compare the models above. Use `cv.glm` for 5-fold and 10-fold Cross Validation to compare the models above.

```r
# LOOCV
results_loo <- data.frame(
  Model = character(),
  MSE = numeric(),
  RMSE = numeric(),
  stringsAsFactors = FALSE
)
for (name in names(models)) {
  glm_model <- glm(models[[name]])
  cv_result <- cv.glm(economics, glm_model, K = nrow(economics))
  mse <- mean(cv_result$delta^2)
  rmse <- sqrt(mean(cv_result$delta^2))
  results_loo <- rbind(results_loo, data.frame(Model = name, MSE = mse, RMSE = rmse))
}

# k-fold CV
perform_k_fold_cv <- function(k) {
  results_kfold <- data.frame(
    Model = character(),
    MSE = numeric(),
    RMSE = numeric(),
    stringsAsFactors = FALSE
  )
  for (name in names(models)) {
    glm_model <- glm(models[[name]])
    cv_result <- cv.glm(economics, glm_model, K = k)
    mse <- mean(cv_result$delta^2)
    rmse <- sqrt(mean(cv_result$delta^2))
    results_kfold <- rbind(results_kfold, data.frame(Model = name, MSE = mse, RMSE = rmse))
  }
  return(results_kfold)
}
results_5fold <- perform_k_fold_cv(5)
results_10fold <- perform_k_fold_cv(10)

results_loo$Type <- "LOOCV"
results_5fold$Type <- "5-Fold"
results_10fold$Type <- "10-Fold"

# combine all
final_results <- rbind(results_loo, results_5fold, results_10fold)
final_table <- pivot_wider(final_results, names_from = Type, values_from = c(MSE, RMSE))
kable(final_table)
```

Model	MSE_LOOCV	MSE_5-Fold	MSE_10-Fold	RMSE_LOOCV	RMSE_5-Fold	RMSE_10-Fold
lm_1	2.941929e+12	3.015010e+12	2.936247e+12	1.715205e+06	1.736378e+06	1.713548e+06
exp_1	2.821600e-03	2.831800e-03	2.806900e-03	5.311900e-02	5.321470e-02	5.298000e-02
poly2_1	2.052124e+12	2.058019e+12	2.080416e+12	1.432524e+06	1.434580e+06	1.442365e+06
poly3_1	1.867016e+12	1.873289e+12	1.874591e+12	1.366388e+06	1.368682e+06	1.369157e+06
poly10_1	2.050159e+13	1.742757e+13	1.697818e+15	4.527868e+06	4.174634e+06	4.120458e+07
lm_2	1.730374e+01	1.704113e+01	1.720738e+01	4.159777e+00	4.128091e+00	4.148177e+00
log_2	4.898349e+01	4.900113e+01	4.896232e+01	6.998820e+00	7.000081e+00	6.997308e+00
poly2_2	9.030577e+00	9.071303e+00	9.029274e+00	3.005092e+00	3.011860e+00	3.004875e+00
poly3_2	9.059541e+00	9.053586e+00	9.102200e+00	3.009907e+00	3.008918e+00	3.016985e+00
poly10_2	8.021511e+00	8.639031e+00	8.233270e+00	2.832227e+00	2.939223e+00	2.869368e+00

2.4 Explain based on the CV and graphical model fits the concepts of Underfitting, Overfitting and how to apply cross-validation to determine the appropriate model fit. Also, describe the different variants of cross validation in this context.

Based on the cross-validation results and graphical model fits, underfitting occurs when a model is too simple to capture the underlying patterns in the data, resulting in high bias and poor performance on both training and test sets. Overfitting happens when a model is overly complex, fitting noise in the training data and performing well on the training set but poorly on unseen data.

Cross-validation helps determine the appropriate model fit by estimating the model's performance on unseen data, allowing us to select a model that balances complexity and generalization. The different variants of cross-validation include k-fold CV, where the data is split into k subsets for multiple rounds of training and testing; leave-one-out CV, which is an extreme case of k-fold CV where k equals the sample size; and repeated k-fold CV, which performs multiple rounds of k-fold CV with different random splits. These methods provide more robust estimates of model performance compared to a single train-test split, helping to identify the optimal model complexity that minimizes both underfitting and overfitting.