Chapter 5

Notes

Cross-Validation

Cross-validation is the process of splitting the training data into multiple subsets that can be used for evaluating the out-of-sample performance of a statistical model.

There are multiple strategies for this technique.

The Validation Set approach

The most basic strategy for CV is a training/test split of the sample.

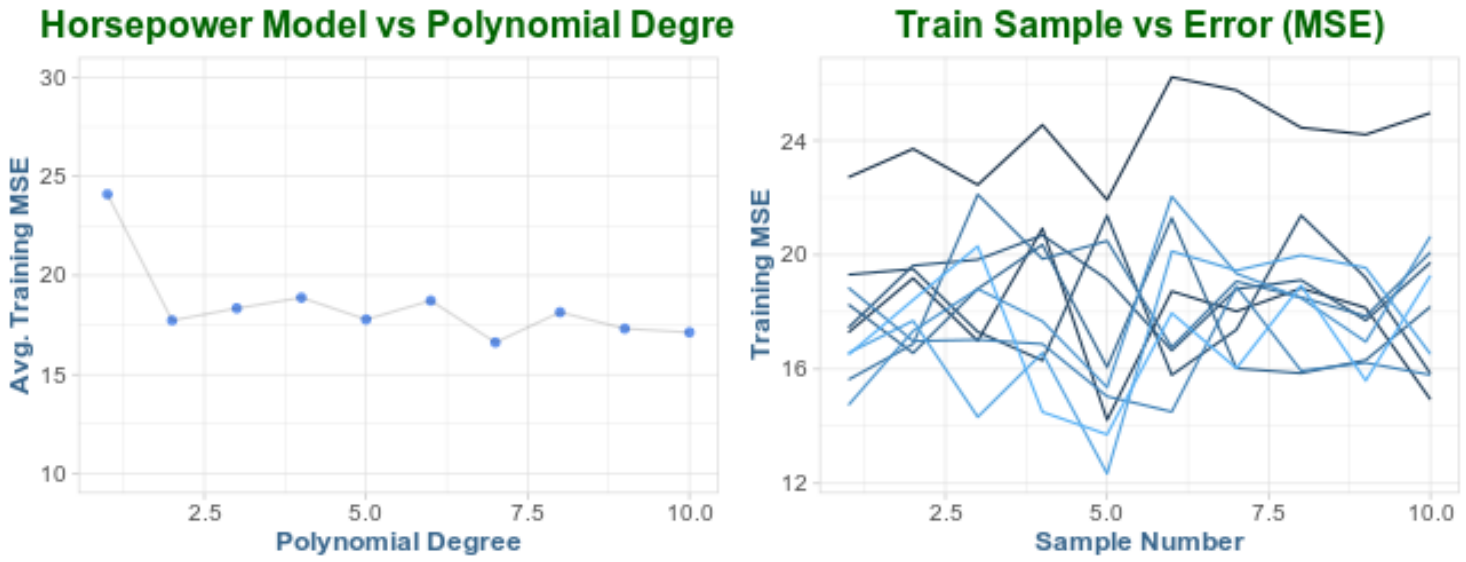
Additionally, stratification can be used to ensure even splitting when the response variable is unevenly distributed in the sample (low response logistic regression, for example).

Using Auto:

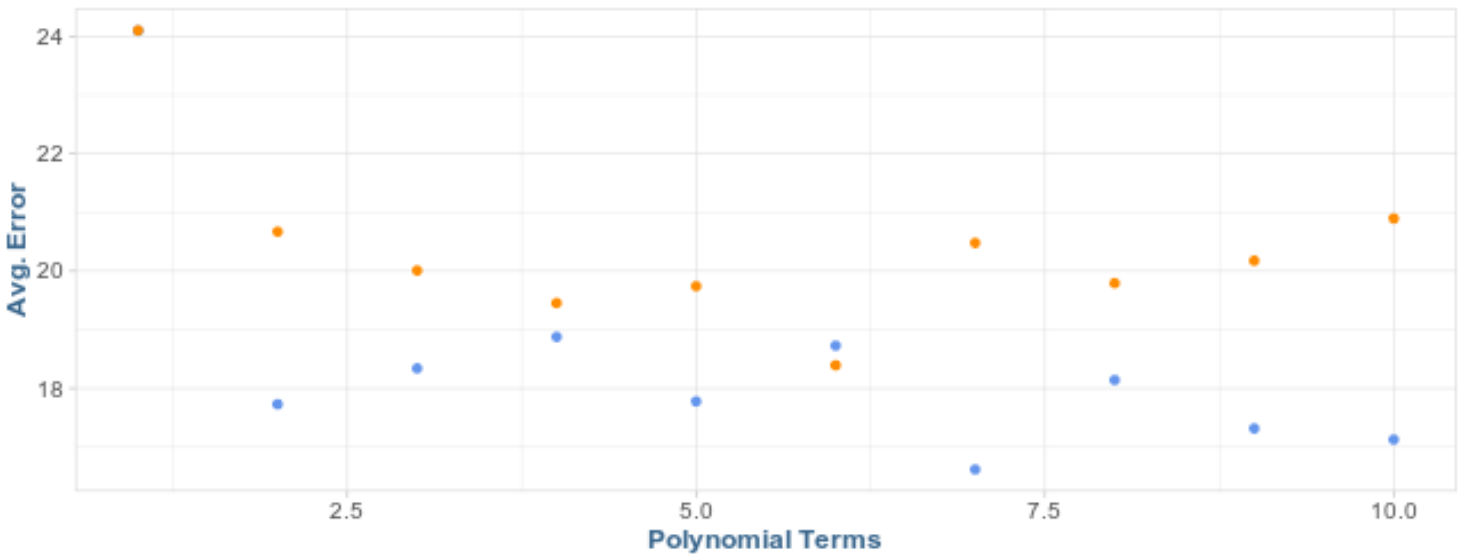
```
auto <- data.table(ISLR::Auto)
```

```
models <- list()

base.model <- "mpg ~ horsepower"

prev <- ""
for(term in 1:10)
{
   cur <- ifelse(term > 1, paste(prev, rep(paste0("+ I(horsepower^", term,")"))), "")
   fmla <- as.formula(paste0(base.model, cur))

   train.error <- numeric(10); test.error <- numeric(10)

   for(iter in 1:10)
   {
      auto.split <- initial_split(auto, prop = .5)

      auto.train <- training(auto.split)
      model <- lm(fmla, data = auto.train)
      train.error[iter] <- mean(model$residuals^2)

      auto.test <- testing(auto.split)
      auto.test$pred <- predict(model, newdata = auto.test)
      test.error[iter] <- with(auto.test, mean( (mpg - pred)^2 ) )
   }
```

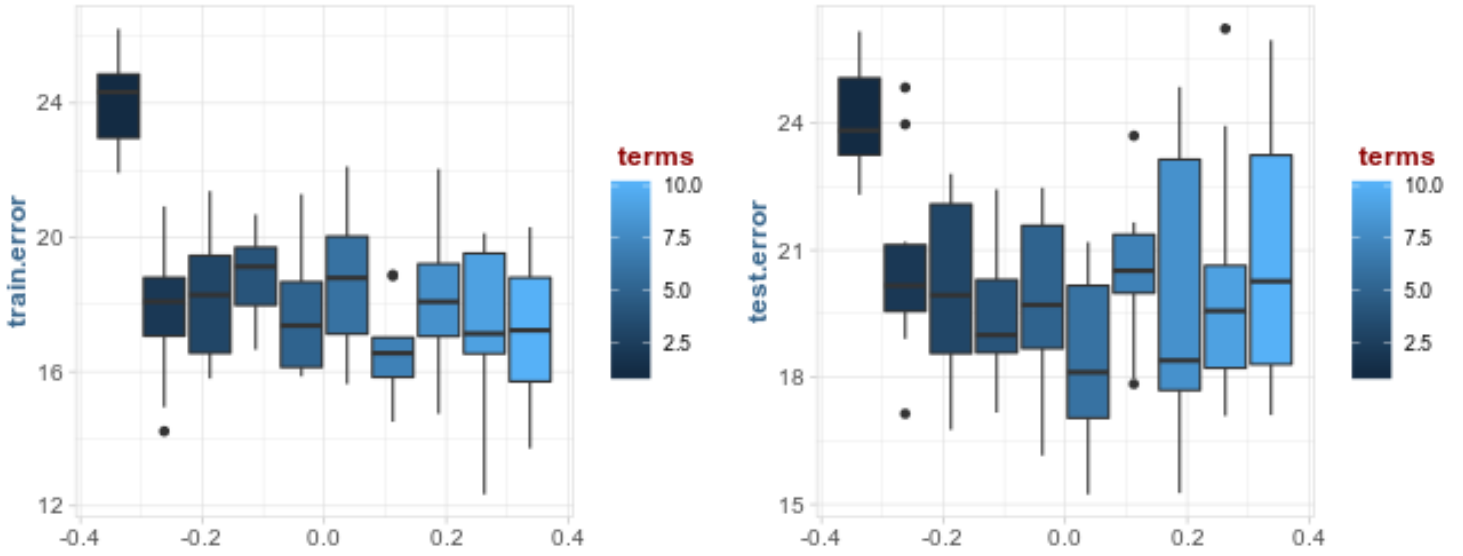
```
   models[[term]] <- list(terms = term, train.error = train.error, test.error = test.error)
   prev <- cur
}

auto.fits <- rbindlist(models, fill = F)

p1 <- auto.fits %>%
   group_by(terms) %>%
   summarise(mean_error = mean(train.error)) %>%
   ggplot(., aes(terms, mean_error)) +
      geom_point(col = "cornflowerblue") +
      geom_line(alpha = .15) +
      labs(title = "Horsepower Model vs Polynomial Degree", x = "Polynomial Degree", y = "Avg.
      scale_y_continuous(limits = c(10, 30))

p2 <- auto.fits[, .(num = 1:.N, train.error), by = list(terms)] %>%
   ggplot() +
   geom_line(aes(num, train.error, group = terms, col = terms)) +
   labs(title = "Train Sample vs Error (MSE)", x = "Sample Number", y = "Training MSE") +
   theme(legend.position = "none")

gridExtra::grid.arrange(p1, p2, nrow = 1)
```




```
ggplot(auto.fits[, .(train = mean(train.error), test = mean(test.error)), by = list(terms)]) +
   geom_point(aes(terms, train), col = "cornflowerblue") +
   geom_point(aes(terms, test), col = "darkorange") +
   labs(x = "Polynomial Terms", y = "Avg. Error")
```




```
p1 <- ggplot(auto.fits) +
   geom_boxplot(aes(y = train.error, group = terms, fill = terms))

p2 <- ggplot(auto.fits) +
   geom_boxplot(aes(y = test.error, group = terms, fill = terms))

gridExtra::grid.arrange(p1, p2, nrow = 1)
```



Drawbacks:

- 1.) The validation estimate of the test error rate can be highly variable, depending on precisely which observations are included in the training set and which observations are in the validation set.
- 2.) In the validation approach, only a subset of the observations - those that are included in the training set rather than the validation set - are used to fit the model. This can lead to overestimation of model performance.

Leave-One-Out Cross-Validation

Leave-one-out cross-validation (LOOCV) is closely related to the validation set approach, but attempts to address some of the drawbacks.

The basic premise of LOOCV is to leave exactly 1 observation out of the data to test against, and use the remaining n-1 observations to train the model.

After each resample, the final test error is produced by:

$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} MSE_i$

This is more of systematic approach to model training/validation.

Advantages:

- Far less bias than the validation/test set.

k-Fold Cross-Validation

An alternative to LOOCV is k-Fold CV. This approach randomly divides the set of observations into *k* groups, or *k*-folds, of approximately equal size.

The first fold is treated as a validation set, and the method is fit on the remaining *k-1* folds. LOOCV is a special case of k-Fold, (k = n).

However, there is a bias/variance trade-off associated with the choice of k. If you have a large number of k (example, n), then each model output will be highly correlated to each other.

Typically, a good choice of k is 5 or 10.

Cross-Validation on Classification

Instead of MSE for error metric, we will use:

$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} Err_i$

The Bootstrap

Bootstrapping is the process of resampling a data set (with replacement) to obtain confidence intervals (SE) for unknown population parameters.

R lab

The Validation Set Approach

```
set.seed(1)

train <- sample(392, 196)

lm.fit <- lm(mpg ~ horsepower, data = auto, subset = train)

mean(lm.fit$residuals^2) # train MSE
```

```
[1] 25.05959
```

```
test <- auto[!train]
test$pred <- predict(lm.fit, newdata = test)
with(test, mean((mpg - pred)^2)) # test MSE
```

```
[1] 23.26601
```

```
lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data = auto, subset = train)
test$pred <- predict(lm.fit2, newdata = test) # update predictions
with(test, mean((mpg - pred)^2)) # model 2 MSE
```

```
[1] 18.71646
```

```
lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = auto, subset = train)
test$pred <- predict(lm.fit3, newdata = test) # update predictions
with(test, mean((mpg - pred)^2)) # model 2 MSE
```

```
[1] 18.79401
```

Leave-One-Out CV

```
glm.fit <- glm(mpg ~ horsepower, data = auto)
coef(glm.fit)
```

```
(Intercept)  horsepower
 39.9358610  -0.1578447
```

```
glm.fit <- glm(mpg ~ horsepower, data = auto)
cv.err <- cv.glm(auto, glm.fit)

cv.err$delta
```

```
[1] 24.23151 24.23114
```

```
cv.error <- rep(0, 5)
for( i in 1:5 )
{
   glm.fit <- glm(mpg ~ poly(horsepower, i), data = auto)
   cv.error[i] <- cv.glm(auto, glm.fit)$delta[1]
}

cv.error
```

```
[1] 24.23151 19.24821 19.33498 19.42443 19.03321
```

k-Fold Cross-Validation

```
set.seed(7)

cv.error.10 <- rep(0, 10)
for( i in 1:10 )
{
   glm.fit <- glm(mpg ~ poly(horsepower, i), data = auto)
   cv.error.10[i] <- cv.glm(auto, glm.fit)$delta[1]
}

cv.error.10
```

```
 [1] 24.23151 19.24821 19.33498 19.42443 19.03321 18.97864 18.83305 18.96115
 [9] 19.06863 19.49093
```

The Bootstrap

```
portfolio <- data.table(ISLR::Portfolio)

alpha.fn <- function(data, index ) {
   X <- data$X; Y <- data$Y
   return ((var(Y)-cov(X, Y)) / (var(X) + var(Y) - 2*cov(X, Y)))
}

alpha.fn(portfolio, 1:100)
```

```
[1] 0.5758321
```

```
set.seed(1)

alpha.fn(portfolio, sample(100, 100, replace = T))
```

```
[1] 0.5758321
```

```
boot(data = portfolio, statistic = alpha.fn, R = 1000)
```

```

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = portfolio, statistic = alpha.fn, R = 1000)


Bootstrap Statistics :
     original  bias    std. error
t1* 0.5758321       0           0
```

Estimating the Accuracy of a Linear Regression Model

```
boot.fn <- function(data, index) {
   return (coef(lm(mpg ~ horsepower, data = data, subset = index)))
}

boot.fn(auto, 1:396)
```

```
(Intercept)  horsepower
 39.9358610  -0.1578447
```

```
set.seed(1)

boot.fn(auto, sample(392, 392, replace = T))
```

```
(Intercept)  horsepower
 40.3404517  -0.1634868
```

```
boot(auto, boot.fn, 1000)
```

```

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = auto, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
      original        bias    std. error
```