DS-6030 Homework Module 4

Tom Lever

06/13/2023

DS 6030 | Spring 2023 | University of Virginia

3. We now review k-fold cross-validation.

   (a) Explain how k-fold cross-validation is implemented.

   According to *An Introduction to Statistical Learning* (Second Edition) by James et al., "k-fold cross validation involves randomly dividing a set of observations int k groups, or 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."

   For models with continuous responses, we consider the error rate to be a linear combination of Mean Squared Errors. "The mean squared error, MSE_1, is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of the test error, $MSE_1, MSE_2, ..., MSE_k$. The k-fold cross validation estimate is computed by averages these values,

$$CV_k = \frac{1}{k} \sum_{i=1}^{k} [MSE_i]$$

   Figure 5.5 illustrates the k-fold CV approach... In practice, one typically performs k-fold CV using $k = 5$ or $k = 10$.

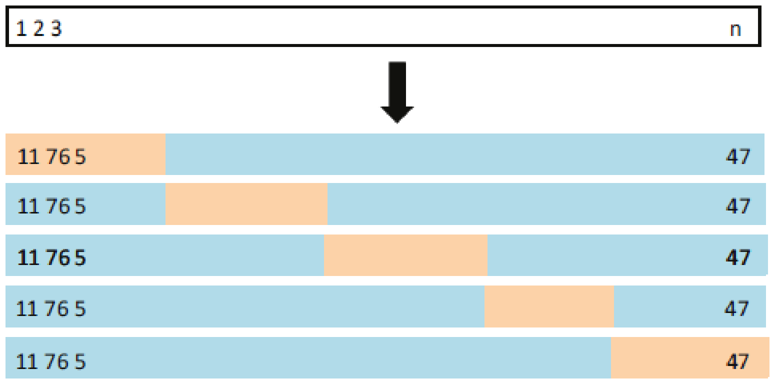


   FIGURE 5.5. *A schematic display of 5-fold CV. A set of n observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting MSE estimates.*

   When we perform cross-validation, our goal might be to determine how well a given statistical learning procedure can be expected to perform on independent data; in this case, the actual estimate of the test MSE is of interest. But at other times we are interested only in the location of the minimum point in the estimated test MSE curve. This is because we might be performing cross-validation on a number of statistical learning methods, or on a single method using different levels of flexibility, in order to identify the method that results in the lowest test error... [D]espite the fact that they sometimes underestimate or [overestimate] the true test MSE, [most] of the CV curves come close to identifying the correct level of flexibility–that is, the flexibility level corresponding to the smallest test MSE."

For models with qualitative responses, we consider the error rate to be a linear combination of numbers of misclassified observations. "For instance, in the classification setting, the LOOCV error rate takes the form

$$Err_i = I\left(y_i \neq \hat{y}_i\right)$$

$$CV_k = \frac{1}{k}\sum_{i=1}^{k}\left[Err_i\right]$$

(b) What are the advantages and disadvantages of k-fold cross-validation relative to:

i. The validation set approach?
The right-hand panel of Figure 5.4 displays nine different 10-fold CV estimates for the `Auto` data set, each resulting from a different random split of the observations into ten folds. As we can see from the figure, there is some variability in the CV estimates as a result of the variability in how the observations are divided into ten folds. But this variability is typically much lower than the variability in the test error estimates that result from the validation set approach (right-hand panel of Figure 5.2)."

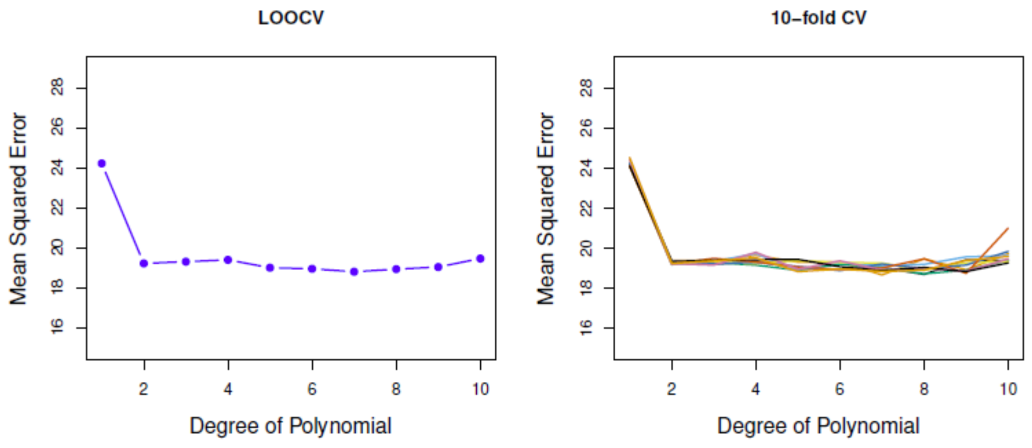


FIGURE 5.4. *Cross-validation was used on the* `Auto` *data set in order to estimate the test error that results from predicting* `mpg` *using polynomial functions of* `horsepower`. Left: *The LOOCV error curve.* Right: 10-*fold CV was run nine separate times, each with a different random split of the data into ten parts. The figure shows the nine slightly different CV error curves.*

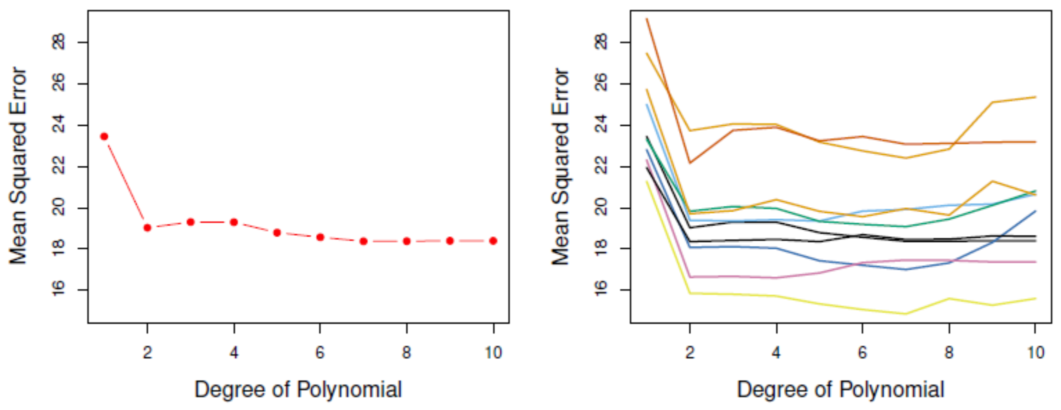


FIGURE 5.2. *The validation set approach was used on the* `Auto` *data set in order to estimate the test error that results from predicting* `mpg` *using polynomial functions of* `horsepower`. Left: *Validation error estimates for a single split into training and validation data sets.* Right: *The validation method was repeated ten times, each time using a different random split of the observations into a training set and a validation set. This illustrates the variability in the estimated test MSE that results from this approach.*

According to What are the advantages and disadvantages of using k-fold cross-validation for predictive analytics?, “K-fold cross-validation has some drawbacks that need to be considered. It increases the computational cost and time, as it requires training and testing the model multiple times. Additionally, it may not be suitable for some types of data, such as time-series or spatial data, due to the order or location of the observations. Furthermore, it may introduce bias if the data is not well distributed or balanced across the folds. Lastly, it may not account for the uncertainty or variability of the model’s performance, as it only gives a point estimate.”

ii. LOOCV?
LOOCV is a special case of k-fold CV in which k is set to equal [number of observations] n... LOOCV requires fitting the statistical learning method n times. This has the potential to be computationally expensive... But cross-validation is a very general approach that can be applied to almost any statistical learning method. Some statistical learning methods have computationally intensive fitting procedures, and so performing LOOCV may pose computational problems, especially in n is extremely large. In contrast, performing 10-fold CV requires fitting the learning procedure only ten times, which may be much more feasible.
k-fold cross validation “often gives more accurate estimates of the test error rate than does LOOCV. This has to do with a bias-variance trade-off... from the perspective of bias reduction, it is clear that LOOCV is to be preferred to k-fold CV. However, we know that bias is not the only source for concern in an estimating procedure; we must also consider the procedure’s variance. It turns out the LOOCV has higher variance than does k-fold CV with $k < n$... When we perform LOOCV, we are in effect averaging the outputs of n fitted models, each of which is trained on an almost identical set of observations; therefore, these outputs are highly (positively) correlated with each other. In contrast, when we perform k-fold CV with $k < n$, we are averaging the outputs of k fitted models that are somewhat less correlated with each other, since the overlap between the training sets in each model is smaller. Since the mean of many highly correlated quantities has higher variance than does the mean of many quantities that are not as highly correlated, the test error estimate resulting from LOOCV tends to have higher variance than does the test error estimate resulting from k-fold CV. To summarize, there is a bias-variance trade-off associated with the choice of k in k-fold cross-validation. Typically, given these considerations, one performs k-fold cross-validation

using $k = 5$ or $k = 10$, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance."

4. In Chapter 4, we used logistic regression to predict the probability of `default` using `income` and `balance` on the `Default` data set.

   We will now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

   (a) Fit a logistic regression model that uses `income` and `balance` to predict `default`.

```
library(ISLR2)
LR_model <- glm(
    formula = default ~ income + balance,
    data = Default,
    family = binomial
)
LR_model
```

```
#
# Call:  glm(formula = default ~ income + balance, family = binomial,
#     data = Default)
#
# Coefficients:
# (Intercept)       income      balance
#  -1.154e+01    2.081e-05    5.647e-03
#
# Degrees of Freedom: 9999 Total (i.e. Null);  9997 Residual
# Null Deviance:        2921
# Residual Deviance: 1579   AIC: 1585
```

   (b) Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

      i. Split the sample set into a training set and a validation set.

```
set.seed(1)
library(TomLeversRPackage)
split_data <- split_data_set_into_training_and_testing_data(
    data_frame = Default,
    proportion_of_training_data = 0.5
)
training_data <- split_data$training_data
validation_data <- split_data$testing_data
nrow(training_data)
```

```
# [1] 5000
```

```
number_of_validation_observations <- nrow(validation_data)
number_of_validation_observations
```

```
# [1] 5000
```

      ii. Fit a multiple logistic regression model using only the training observations.

```
LR_model <- glm(
    formula = default ~ income + balance,
    data = training_data,
    family = binomial
)
LR_model
```

```
#
# Call:  glm(formula = default ~ income + balance, family = binomial,
#     data = training_data)
#
# Coefficients:
# (Intercept)       income      balance
#  -1.194e+01    3.262e-05    5.689e-03
#
# Degrees of Freedom: 4999 Total (i.e. Null);  4997 Residual
# Null Deviance:        1524
# Residual Deviance: 803.3  AIC: 809.3
```

iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the `default` category if the posterior probability is greater than 0.5.

```
vector_of_predicted_probabilities <- predict(
    object = LR_model,
    newdata = validation_data,
    type = "response"
)
vector_of_predicted_default_statuses <- rep(
    x = "No",
    times = number_of_validation_observations
)
condition <- vector_of_predicted_probabilities > 0.5
vector_of_predicted_default_statuses[condition] = "Yes"
vector_of_predicted_default_statuses[1:3]
```

```
# [1] "No" "No" "No"
```

iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

```
mean(vector_of_predicted_default_statuses != training_data$default)
```

```
# [1] 0.048
```

The validation error rate is 4.8 percent.

(c) Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

```
vector_of_error_rates <- double(length = 0)
for (i in 2:4) {
    set.seed(i)
    split_data <- split_data_set_into_training_and_testing_data(
        data_frame = Default,
        proportion_of_training_data = 0.5
    )
    training_data <- split_data$training_data
    validation_data <- split_data$testing_data
    summary_of_performance <- summarize_performance(
        type_of_model = "Logistic Regression",
        formula = default ~ income + balance,
        training_data = training_data,
```

```r
        test_data = validation_data
    )
    error_rate <- summary_of_performance$error_rate
    print(error_rate)
    vector_of_error_rates <- append(vector_of_error_rates, error_rate)
}
```

```
# Error in summarize_performance(type_of_model = "Logistic Regression", : could not find functi
```

```r
mean(vector_of_error_rates)
```

```
# [1] NaN
```

```r
sd(vector_of_error_rates)
```

```
# [1] NA
```

Based on three validation approaches, the mean validation error rate is 2.5 percent with a standard deviation of 0.1 percent.

(d) Now consider a logistic regression model that predicts the probability of `default` using `income`, `balance`, and a dummy variable for `student`. Estimate the test error for this model using the validation set approach. Comment on whether or not including a dummy variable for student leads to a reduction in the test error rate.

```r
number_of_observations <- nrow(Default)
vector_of_indicators_of_whether_borrower_is_a_student <- rep(
    x = 0,
    times = number_of_observations
)
condition <- Default$student == "Yes"
vector_of_indicators_of_whether_borrower_is_a_student[condition] <- 1
data_frame <- data.frame(
    Default,
    indicator_of_whether_borrower_is_a_student =
        vector_of_indicators_of_whether_borrower_is_a_student
)
vector_of_error_rates <- double(length = 0)
for (i in 2:4) {
    set.seed(i)
    split_data <- split_data_set_into_training_and_testing_data(
        data_frame = data_frame,
        proportion_of_training_data = 0.5
    )
    training_data <- split_data$training_data
    validation_data <- split_data$testing_data
    summary_of_performance <- summarize_performance(
        type_of_model = "Logistic Regression",
        formula = default
            ~ income + balance + indicator_of_whether_borrower_is_a_student,
        training_data = training_data,
        test_data = validation_data
    )
    error_rate <- summary_of_performance$error_rate
    print(error_rate)
    vector_of_error_rates <- append(vector_of_error_rates, error_rate)
}
```

```
# Error in summarize_performance(type_of_model = "Logistic Regression", : could not find functi
```

```r
mean(vector_of_error_rates)
```

```
# [1] NaN
```

```r
sd(vector_of_error_rates)
```

```
# [1] NA
```

Including a dummy variable for student does not lead to a reduction in mean validation error rate; the mean validation error rate is higher by 0.1 percent and its standard deviation is slightly lower.

5. We continue to consider the use of a logistic regression model to predict the probability of `default` using `income` and `balance` on the `Default` data set.

   In particular, we will now compute estimates for the standard errors of the `income` and `balance` logistic regression coefficients in two different ways: (1) using the bootstrap, and (2) using the standard formula for computing the standard errors in the `glm()` function. Do not forget to set a random seed before beginning your analysis.

   (a) Using the `summary()` and `glm()` functions, determine the estimated standard errors for the coefficients associated with `income` and `balance` in a multiple logistic regression model that uses both predictors.

```r
set.seed(1)
LR_model <- glm(
    formula = default ~ income + balance,
    data = Default,
    family = binomial
)
summary(LR_model)
```

```
#
# Call:
# glm(formula = default ~ income + balance, family = binomial,
#     data = Default)
#
# Coefficients:
#                Estimate Std. Error z value Pr(>|z|)
# (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
# income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
# balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# (Dispersion parameter for binomial family taken to be 1)
#
#     Null deviance: 2920.6  on 9999  degrees of freedom
# Residual deviance: 1579.0  on 9997  degrees of freedom
# AIC: 1585
#
# Number of Fisher Scoring iterations: 8
```

The estimated standard errors for the coefficients associated with `income` and `balance` in a multiple logistic regression model with formula $default \sim income + balance$ are 4.985×10^{-6} and 2.274×10^{-4}, respectively.

(b) Write a function, `boot.fn()`, that takes as input the `Default` data set as well as an index of the observations, and that outputs the coefficient estimates for income and balance in the multiple logistic regression model.

```
boot.fn <- function(Default, vector_of_indices_of_training_observations) {
    LR_model <- glm(
        formula = default ~ income + balance,
        data = Default,
        family = "binomial",
        subset = vector_of_indices_of_training_observations
    )
    vector_of_coefficients <- coef(LR_model)
    return(vector_of_coefficients)
}
set.seed(1)
vector_of_random_indices <- sample(1:number_of_observations)
boot.fn(
    Default = Default,
    vector_of_indices_of_training_observations = vector_of_random_indices
)
```

```
#   (Intercept)        income       balance
# -1.154047e+01  2.080898e-05  5.647103e-03
```

(c) Use the `boot()` function together with your `boot.fn()` function to estimate the standard errors of the logistic regression coefficients for `income` and `balance`.

```
library(boot)
set.seed(1)
# statistic is a function which when applied to data
# returns a vector containing statistic(s) of interest.
# R is the number of bootstrap replicates.
boot(data = Default, statistic = boot.fn, R = 10)
```

```
#
# ORDINARY NONPARAMETRIC BOOTSTRAP
#
#
# Call:
# boot(data = Default, statistic = boot.fn, R = 10)
#
#
# Bootstrap Statistics :
#          original        bias     std. error
# t1* -1.154047e+01 -7.520387e-02 2.811104e-01
# t2*  2.080898e-05  1.589838e-06 3.324258e-06
# t3*  5.647103e-03  1.784084e-05 1.752440e-04
```

Bootstrap estimates of the standard errors of the logistic regression coefficients for `income` and `balance` are 3.324×10^{-6} and 1.752×10^{-4}.

(d) Comment on the estimated standard errors obtained using the `glm()` function and using your bootstrap function.

The rate of difference between the bootstrap estimates of the standard errors of the coefficients of the logistic regression model with formula $default \sim income + balance$ and the single-model estimates are $\left[\left(3.324 \times 10^{-6}\right) - \left(4.985 \times 10^{-6}\right)\right] / \left(4.985 \times 10^{-6}\right) = -0.333$ and $\left[\left(1.752 \times 10^{-4}\right) - \left(2.274 \times 10^{-4}\right)\right] / \left(2.274 \times 10^{-4}\right) = -0.230$.

6. In Sections 5.3.2 and 5.3.3, we saw that the `cv.glm()` function can be used in order to compute the LOOCV test error estimate.

   Alternatively, one could compute those quantities using just the `glm()` and `predict.glm()` functions, and a for loop. You will now take this approach in order to compute the LOOCV error for a simple logistic regression model on the Weekly data set. Recall that in the context of classification problems, the LOOCV error is given in (5.4).

   (a) Fit a logistic regression model that predicts `Direction` using `Lag1` and `Lag2`.

```
set.seed(1)
LR_model <- glm(
    formula = Direction ~ Lag1 + Lag2,
    data = Weekly,
    family = binomial
)
LR_model
```

```
#
# Call:  glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = Weekly)
#
# Coefficients:
# (Intercept)         Lag1         Lag2
#     0.22122     -0.03872      0.06025
#
# Degrees of Freedom: 1088 Total (i.e. Null);  1086 Residual
# Null Deviance:        1496
# Residual Deviance: 1488   AIC: 1494
```

   (b) Fit a logistic regression model that predicts `Direction` using `Lag1` and `Lag2` using all but the first observation.

```
set.seed(1)
training_data <- Weekly[-1, ]
testing_data <- Weekly[1, ]
LR_model <- glm(
    formula = Direction ~ Lag1 + Lag2,
    data = training_data,
    family = binomial
)
LR_model
```

```
#
# Call:  glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = training_data)
#
# Coefficients:
# (Intercept)         Lag1         Lag2
#     0.22324     -0.03843      0.06085
#
# Degrees of Freedom: 1087 Total (i.e. Null);  1085 Residual
# Null Deviance:        1495
# Residual Deviance: 1487   AIC: 1493
```

(c) Use the model from (b) to predict the direction of the first observation. You can do this by predicting that the first observation will go up if $P(\text{Direction=“Up”}|\text{Lag1, Lag2}) > 0.5$. Was this observation correctly classified?

```
number_of_test_observations <- nrow(testing_data)
vector_of_predicted_probabilities <- predict(
    object = LR_model,
    newdata = testing_data,
    type = "response"
)
vector_of_predicted_directions <- rep("Down", number_of_test_observations)
condition <- vector_of_predicted_probabilities > 0.5
vector_of_predicted_directions[condition] = "Up"
vector_of_predicted_directions
```

```
# [1] "Up"
```

```
testing_data
```

```
#   Year  Lag1  Lag2   Lag3   Lag4   Lag5   Volume Today Direction
# 1 1990 0.816 1.572 -3.936 -0.229 -3.484 0.154976 -0.27      Down
```

The first observation was incorrectly classified.

(d) Write a `for` loop from $i = 1$ to $i = n$, where n is the number of observations in the data set, that performs each of the following steps:

i. Fit a logistic regression model using all but the i-th observation to predict `Direction` using `Lag1` and `Lag2`.
ii. Compute the posterior probability of the market moving up for the i-th observation.
iii. Use the posterior probability for the i-th observation in order to predict whether or not the market moves up.
iv. Determine whether or not an error was made in predicting the direction for the i-th observation. If an error was made, then indicate this as a 1, and otherwise indicate it as a 0.

```
number_of_observations <- nrow(Weekly)
formula <- Direction ~ Lag1 + Lag2
vector_of_indicators_of_error <- integer(0)
for (i in 1:number_of_observations) {
    training_data <- Weekly[-i, ]
    testing_data <- Weekly[i, ]
    LR_model <- glm(
        formula = formula,
        data = training_data,
        family = binomial
    )
    vector_of_predicted_probabilities <- predict(
        object = LR_model,
        newdata = testing_data,
        type = "response"
    )
    vector_of_predicted_directions <- rep("Down", number_of_test_observations)
    condition <- vector_of_predicted_probabilities > 0.5
    vector_of_predicted_directions[condition] = "Up"
    if (vector_of_predicted_directions != testing_data$Direction) {
        vector_of_indicators_of_error <-
            append(vector_of_indicators_of_error, 1)
    } else {
        vector_of_indicators_of_error <-
            append(vector_of_indicators_of_error, 0)
    }
}
```

(e) Take the average of the n numbers obtained in (d) in order to obtain the LOOCV estimate for the test error. Comment on the results.

```
mean(vector_of_indicators_of_error)
```

```
# [1] 0.4499541
```

7. We will now perform cross-validation on a simulated data set.

(a) Generate a simulated data set as follows:

```
set.seed(1)
y <- rnorm(100)
x <- rnorm(100)
y <- x - 2*x^2 + rnorm(100)
```

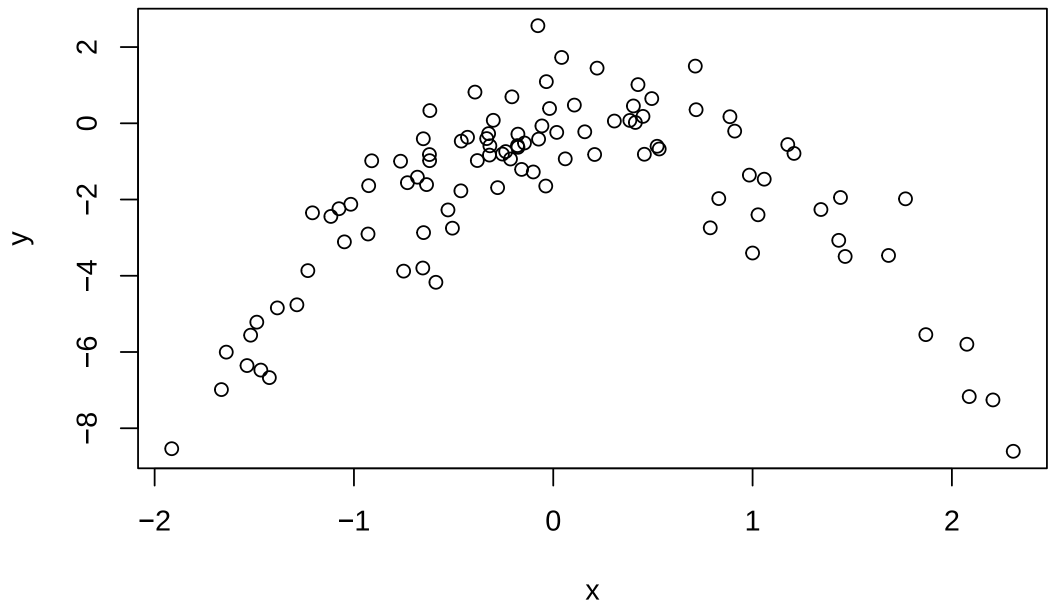
In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

The number of observations $n = 100$. The number of predictors $p = 2$. Here, a predictor is considered to be a predictive term as opposed single variable x. The model used to generate the data

$$\boldsymbol{y} = rnorm(100) + \boldsymbol{x} - 2\boldsymbol{x}^2$$
$$y = rnorm(1) + x - 2x^2$$

(b) Create a scatterplot of `x` against `y`. Comment on what you find.

```
plot(x = x, y = y)
```



This scatterplot suggests that y relates to x quadratically.

(c) Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares:

i. $Y = \beta_0 + \beta_1 X + \epsilon$

```r
set.seed(1)
data_frame <- data.frame(x = x, y = y)
glm_1 <- glm(formula = y ~ x)
boot::cv.glm(data = data_frame, glmfit = glm_1)$delta[1]
```

```
# [1] 5.890979
```

ii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$

```r
glm_2 <- glm(formula = y ~ poly(x, 2))
boot::cv.glm(data = data_frame, glmfit = glm_2)$delta[1]
```

```
# [1] 1.086596
```

iii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$

```r
glm_3 <- glm(formula = y ~ poly(x, 3))
boot::cv.glm(data = data_frame, glmfit = glm_3)$delta[1]
```

```
# [1] 1.102585
```

iv. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon$.

```r
glm_4 <- glm(formula = y ~ poly(x, 4))
boot::cv.glm(data = data_frame, glmfit = glm_4)$delta[1]
```

```
# [1] 1.114772
```

Note you may find it helpful to use the `data.frame()` function to create a single data set containing both `X` and `Y`.

(d) Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

```r
set.seed(2)
glm_1 <- glm(formula = y ~ x)
boot::cv.glm(data = data_frame, glmfit = the_glm)$delta[1]
```

```
# Error in eval(expr, envir, enclos): object 'the_glm' not found
```

```r
glm_2 <- glm(formula = y ~ poly(x, 2))
boot::cv.glm(data = data_frame, glmfit = glm_2)$delta[1]
```

```
# [1] 1.086596
```

```r
glm_3 <- glm(formula = y ~ poly(x, 3))
boot::cv.glm(data = data_frame, glmfit = glm_3)$delta[1]
```

```
# [1] 1.102585
```

```r
glm_4 <- glm(formula = y ~ poly(x, 4))
boot::cv.glm(data = data_frame, glmfit = glm_4)$delta[1]
```

```
# [1] 1.114772
```

Given that each LOOCV error rate that is averaged is based on the same training and testing data sets regardless of seed, our results are the same as what we got in (c).

(e) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

The quartic model in (c) has the smallest LOOCV error. The average mean squared error for the quartic model is slightly less than the average mean squared error for the quadratic model. We expected the average mean squared error for the quadratic model to be less than the average mean squared error for the other models, though it seems reasonable that polynomial of degree greater than 2 would fit noisy data from a quadratic function better than the quadratic function.

(f) Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawn based on the cross-validation results?

```r
summary(glm_1)$coefficients
```

```
#                Estimate Std. Error    t value     Pr(>|t|)
# (Intercept) -1.8185184  0.2364064 -7.6923393 1.143164e-11
# x            0.2430443  0.2478503  0.9806091 3.292002e-01
```

```r
summary(glm_2)$coefficients
```

```
#                Estimate Std. Error   t value     Pr(>|t|)
# (Intercept)   -1.827707  0.1032351 -17.70431 3.804657e-32
# poly(x, 2)1    2.316401  1.0323515   2.24381 2.711854e-02
# poly(x, 2)2 -21.058587  1.0323515 -20.39866 7.333860e-37
```

```r
summary(glm_3)$coefficients
```

```
#                 Estimate Std. Error     t value     Pr(>|t|)
# (Intercept)   -1.8277074  0.1037248 -17.6207390 7.610579e-32
# poly(x, 3)1    2.3164010  1.0372479   2.2332183 2.785714e-02
# poly(x, 3)2 -21.0585869  1.0372479 -20.3023667 1.636959e-36
# poly(x, 3)3   -0.3048398  1.0372479  -0.2938929 7.694742e-01
```

```
summary(glm_4)$coefficients
```

```
#                 Estimate Std. Error     t value     Pr(>|t|)
# (Intercept)  -1.8277074  0.1041467 -17.5493533 1.444977e-31
# poly(x, 4)1   2.3164010  1.0414671   2.2241711 2.850549e-02
# poly(x, 4)2 -21.0585869  1.0414671 -20.2201171 3.457023e-36
# poly(x, 4)3  -0.3048398  1.0414671  -0.2927023 7.703881e-01
# poly(x, 4)4  -0.4926249  1.0414671  -0.4730105 6.372907e-01
```

x is significant in the context of the first-degree polynomial model. x and x^2 are significant in the context of the second-degree polynomial model. x and x^2 but not x^3 are significant in the context of the third-degree polynomial model. x and x^2 but not x^3 and x^4 are significant in the context of the four-degree polynomial model.

These results seem to suggest that a quadratic model might be the best fit. Given that the average mean squared errors for our nonlinear models are similar, these results agree with the conclusions drawn based on the cross-validation results.