Lab Solutions 01

1. Describe what the spaces Ω and $\mathcal{Y}$ would be for a predictive modeling task that estimates whether an email message is either spam or not spam.

The space Ω is all the possible values we would have for the number of capital letters in the text, and is given by the positive integers:

$$\Omega = \{0, 1, 2, 3, \ldots\} = \mathbb{Z}_{\geq 0}.$$

The space $\mathcal{Y}$ are all of the possible values that are trying to predict as outcomes of the spam detection. You could answer with either the categories themselves:

$$\mathcal{Y} = \{\text{ham}, \text{spam}\}.$$

Or their respective numeric codes as using my R example:

$$\mathcal{Y} = \{0, 1\}.$$

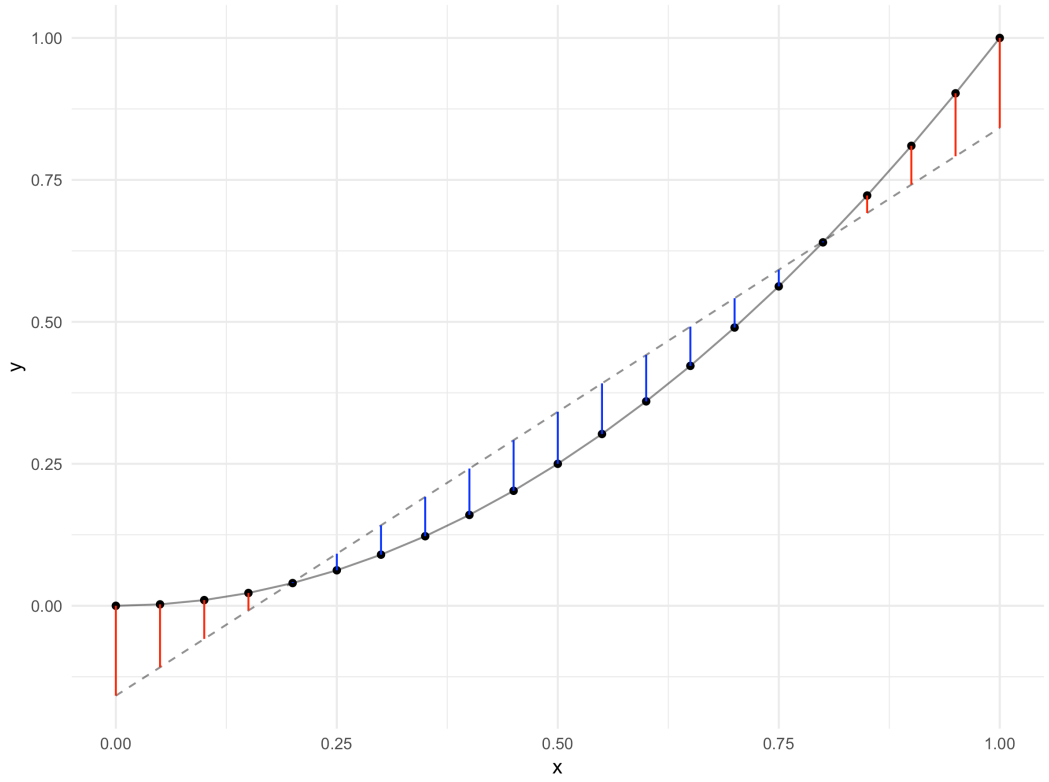
Either is equally correct.

2. Describe what might be some possible columns of the dataset X in a task trying to predict the sale price for a used car.

The columns are the variables that we would use to predict the price of a house. Possible answers include: (i) the square footage of the house, (ii) the lot size, (iii) number of bedrooms, (iv) year build, (v) and the house's zip code.

3. Assume that the feature matrix X has only one column, with values evenly distributed between 0 and 1, and the true function f is deterministic and defined such that to $y_i = x_i^2$. If our modeling algorithm is only allowed to produce a $\widehat{f}$ equal to a linear function of x_i, illustrate using a hand drawn plot how (in this case) the low model complexity directly leads to model bias.

Rather than hand-draw, I will give you an example using an R plot:



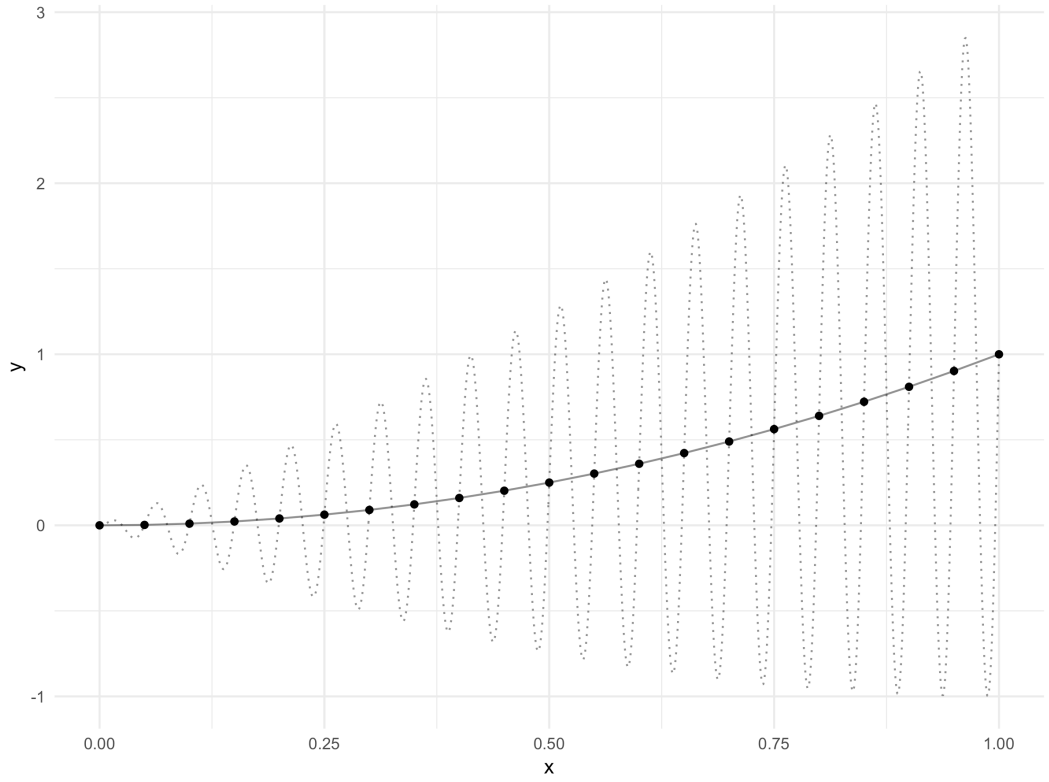
The 'true' function $f(x)$ is given by the solid black line and our prediction $\widehat{f}(x)$ is shown with the dashed line. The errors are given by the colored vertical dashed bars. Points are consistently underpredicted where the colors are red and overpredicted where the colors are blue. This shows how the predicts are biased because the curve we fit is not complex enough to capture the function $f(x)$.

4. Assume that the feature matrix X has only one column, with values evenly distributed between 0 and 1, and the true function f is deterministic and defined such that to $y_i = x_i^2$. If our modeling algorithm is only allowed to produce a $\widehat{f}$ equal to a linear function of x_i, illustrate using a hand drawn plot how (in this case) the low model complexity directly leads to model bias.

I accidentally repeated this question. See answer above.

5. Using the same setup as in the previous question, but allowing $\widehat{f}$ to take on any form, illustrate an example of a curve that has been overfit to the data.

An overfit function $\widehat{f}$ is erroneously overly complex to fit the observed data in a way that will not generalize to new data points.



Notice that the predicted function $\widehat{f}$ perfectly fits the observed training data, but would not do a very good job for new data observed at different values of x with the 'true' function $f(x) = x^2$.

6. Consider the following small dataset:

x	y
1	0
2	1
3	0
4	0
5	1
6	1
7	0
8	1
9	1
10	1

Answer the following questions (you should be able to do this by hand):

What is the best split value derived from this data? What value does your model predict will be the value of y if $x = 1$? How about $x = 10$ and $x = 5$?

The best split value is to use $x = 5$ as the cut-off, with messages containing five or more capital letters being classified as spam. With this, we would classify $x = 1$ as 'ham' and $x = 10$ and $x = 5$ as spam.

Using mis-classification loss, what is the loss of your model on this dataset?

The mis-classification loss (or rate) is the proportion of training examples that we missed. Here, we would *falsely* classify the message with 2 capital letters as spam and *falsely* classify the message with 7 capital letters as spam. All of the others are correct. This yields to a loss of 20%.

7. Take the following data as a *test set* for the model you built in the prior question:

x	y
4	0
4	1
4	1
8	0
8	1
8	0
8	1

What is the misclassification rate on the test data? Is this value better or worse than the loss function applied to the training data?

On this new data, we falsely classify two of the messages with $x = 4$ as ham and two of the $x = 8$ messages as spam. This leads to a loss of 57.1% ($\frac{4}{7}$), not nearly as good as with the original data.

8. If you have not yet installed R, do that now. Using code similar to that in the notes, redo the best split prediction using R. Does it give the same output that you had when computing it by hand?

Here is the code to make this work (minus the function itself, which you can find in the notes):

```
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

```
y <- c(0, 1, 0, 0, 1, 1, 0, 1, 1, 1)

casl_utils_best_split(x, y)
```

```
[1] 5
```

Exactly as we had by hand.

9. (optional) Modify the function `cnlp_utils_best_split` so that it also checks whether the best split would classify every value as a 1 if it is less than some cutoff value. Make sure that the output makes it clear whether the cutoff is a less than or greater than value.

```
casl_utils_best_split_improved <-
function(x, y)
{
  unique_values <- unique(x)
  class_rate_ge <- rep(0, length(unique_values))
  class_rate_le <- rep(0, length(unique_values))
  for (i in seq_along(unique_values))
  {
    class_rate_ge[i] <- sum(y == (x >= unique_values[i]))
    class_rate_le[i] <- sum(y == (x <= unique_values[i]))
  }
  if (max(class_rate_ge) >= max(class_rate_le))
  {
    print("Classify y=1 greater than or equal to cut-off(s)")
    class_rate <- class_rate_ge
  } else {
    print("Classify y=1 less than or equal to cut-off(s)")
    class_rate <- class_rate_le
  }
  unique_values[class_rate == max(class_rate)]
}
```