DSC1107: Formative Assessment 4

Due: March 4, 2025 at 11:59pm

Contents

Instructions

Materials

The allowed materials are as stated on the Syllabus:

> "Students may consult all course materials, including course textbooks, for all assignments and assessments. For programming-based assignments (homeworks and exams), students may also consult the internet (e.g. Stack Overflow) for help with general programming tasks (e.g. how to add a dashed line to a plot). Students may not search the internet for help with specific questions or specific datasets on any homework or exam. In particular, students may not use solutions to problems that may be available online and/or from past iterations of the course."

Collaboration

The collaboration policy is as stated on the Syllabus:

> "Students are permitted to work together on homework assignments, but must write up and submit solutions individually. In particular, students may not copy each others' solutions. Furthermore, students must disclose all classmates with whom they collaborated on a given homework assignment."

In accordance with this policy,

Please list anyone you discussed this homework with:

Writeup

Use this document as a starting point for your writeup, adding your solutions after "**Solution**". Add your R code using code chunks and add your text answers using **bold text**. In particular, if the instructions ask you to "print a table", you should use kable. If the instructions ask you to "print a tibble", you should not use kable and instead print the tibble directly.

Programming

The tidyverse paradigm for data visualization, manipulation, and wrangling is required. No points will be awarded for code written in base R.

We'll need to use the following R packages:

```
library(tidyverse)      # tidyverse
library(readxl)         # for reading Excel files
library(knitr)          # for include_graphics()
library(kableExtra) # for printing tables
library(cowplot)        # for side by side plots
library(FNN)            # for K-nearest-neighbors regression
library(stat471)        # for cross_validate_spline()
```

Grading

The point value for each problem sub-part is indicated. Additionally, the presentation quality of the solution for each problem will be evaluated on a per-problem basis (e.g. in this homework, there are three problems). There are 100 points possible on this homework, 85 of which are for correctness and 15 of which are for presentation. But the total points will be converted to a maximum of 30 points as FA, per policy, has less than number of points than SA (100 points).

Submission

Compile your writeup to PDF and submit to Canvas: FA 4 including your R code in markdown.

1 Case study: Bone mineral density (30 points for correctness; 7 points for presentation)

In this exercise, we will be looking at a data set (given in *bmd-data.xlsx,* see Canvas: FA 4 for the dataset file location) on spinal bone mineral density, a physiological indicator that increases during puberty when a child grows. In this dataset, idnum is an identifier for each child and spnbmd represents the relative change in spinal bone mineral density between consecutive doctor's visits.

The goal is to learn about the typical trends of growth in bone mineral density during puberty for boys and girls.

1.1 Import (2 points)

Since the data are in Excel format, the functions in readr are insufficient to import it. Instead, you must use readxl, another tidyverse package. Familiarize yourself with readxl by referring to the data import cheat sheet or the package website.

1. Using the readxl package, import the data into a tibble called bmd_raw.
2. Print the imported tibble.

```
library(readxl)
library(tidyverse)
library(knitr)
library(kableExtra)
library(cowplot)
library(FNN)
library(stat471)

# Assuming bmd-data.xlsx is in your working directory, otherwise specify the path
bmd_raw <- read_excel("bmd-data.xlsx")

# Print the imported tibble
print(bmd_raw)
```

1.2 Tidy (2 points)

1. Comment on the layout of the data in the tibble. What should be the variables in the data? What operation is necessary to get it into tidy format?
2. Apply this operation to the data, storing the result in a tibble called bmd.

```
bmd <- bmd_raw %>%
  pivot_longer(
    cols = starts_with(c("age", "spnbmd")),
    names_to = c(".value", "visit"),
    names_pattern = "([^\\d]+)(\\d+)"
  ) %>%
  select(-visit) %>% # Remove the visit column as it's no longer needed
  drop_na() # Remove rows with NA values introduced by unequal number of measurements
```

1.3 Explore (6 points)

1. What is the total number of children in this dataset? What are the number of boys and girls? What are the median ages of these boys and girls?
2. Produce plots to compare the distributions of spnbmd and age between boys and girls (display these as two plots side by side, one for spnbmd and one for age). Are there apparent differences in either spnbmd or age between these two groups?
3. Create a scatter plot of spnbmd (y axis) versus age (x axis), faceting by gender. What trends do you see in this data?

```
# Total number of children
n_children <- bmd %>% distinct(idnum) %>% nrow()
cat("Total number of children:", n_children, "\n")

# Number of boys and girls
gender_counts <- bmd %>% distinct(idnum, gender) %>% count(gender)
```

```
print(gender_counts)

# Median ages for boys and girls
median_ages <- bmd %>%
  group_by(gender) %>%
  summarise(median_age = median(age))
print(median_ages)

p_spnbmd <- bmd %>%
  ggplot(aes(x = spnbmd, fill = gender)) +
  geom_density(alpha = 0.5) +
  labs(title = "Distribution of spnbmd by Gender")

p_age <- bmd %>%
  ggplot(aes(x = age, fill = gender)) +
  geom_density(alpha = 0.5) +
  labs(title = "Distribution of Age by Gender")

plot_grid(p_spnbmd, p_age, ncol = 2)

bmd %>%
  ggplot(aes(x = age, y = spnbmd, color = gender)) +
  geom_point(alpha = 0.5) +
  facet_wrap(~ gender) +
  labs(title = "Scatter plot of spnbmd vs Age, Faceted by Gender")
```

1.4 Model (12 points)

There are clearly some trends in this data, but they are somewhat hard to see given the substantial amount of variability. This is where splines come in handy.

1.4.1 Split (1 point)

To ensure unbiased assessment of predictive models, let's split the data before we start modeling it.

1. Split `bmd` into training (80%) and test (20%) sets, using the rows in `train_samples` below for training. Store these in tibbles called `bmd_train` and `bmd_test`, respectively.

```
set.seed(5) # seed set for reproducibility (DO NOT CHANGE)
n <- nrow(bmd)
train_samples <- sample(1:n, round(0.8*n))
```

set.seed(5) # seed set for reproducibility (DO NOT CHANGE)
n <- nrow(bmd)
train_samples <- sample(1:n, round(0.8*n))

```
bmd_train <- bmd %>% slice(train_samples)
bmd_test <- bmd %>% slice(-train_samples)
```

1.4.2 Tune (10 points)

1. Since the trends in spnbmd look somewhat different for boys than for girls, we might want to fit separate splines to these two groups. Separate bmd_train into bmd_train_male and bmd_train_female, and likewise for bmd_test.
2. Using cross_validate_spline from the stat471 R package, perform 10-fold cross-validation on bmd_train_male and bmd_train_female, trying degrees of freedom 1,2,. . . ,15. Display the two resulting CV plots side by side.
3. What are the degrees of freedom values minimizing the CV curve for boys and girls, and what are the values obtained from the one standard error rule?
4. For the sake of simplicity, let's use the same degrees of freedom for males as well as females. Define df.min to be the maximum of the two df.min values for males and females, and define df.1se likewise. Add these two spline fits to the scatter plot of spnbmd (y axis) versus age (x axis), faceting by gender.
5. Given your intuition for what growth curves look like, which of these two values of the degrees of freedom makes more sense?

```
degrees_of_freedom <- 1:15

cv_male <- cross_validate_spline(
  data = bmd_train_male,
  x_var = "age",
  y_var = "spnbmd",
  df_values = degrees_of_freedom,
  folds = 10
)

cv_female <- cross_validate_spline(
  data = bmd_train_female,
  x_var = "age",
  y_var = "spnbmd",
  df_values = degrees_of_freedom,
  folds = 10
)

p_cv_male <- plot_cv_spline(cv_male) + labs(title = "CV Plot for Males")
p_cv_female <- plot_cv_spline(cv_female) + labs(title = "CV Plot for Females")

plot_grid(p_cv_male, p_cv_female, ncol = 2)

min_df_male <- cv_male %>% summarise(min_df = df_values[which.min(cv_error)]) %>% pull(min_df)
se_1se_df_male <- cv_male %>% summarise(df_1se = df_values[min(which(cv_error <= min(cv_error) + cv_error_se[which.min(cv_error)]))]) %>% pull(df_1se)

cat("Males: Minimum CV error df =", min_df_male, ", 1-SE rule df =", se_1se_df_male, "\n")

min_df_female <- cv_female %>% summarise(min_df = df_values[which.min(cv_error)]) %>% pull(min_df)
se_1se_df_female <- cv_female %>% summarise(df_1se = df_values[min(which(cv_error <= min(cv_error) + cv_error_se[which.min(cv_error)]))]) %>% pull(df_1se)

cat("Females: Minimum CV error df =", min_df_female, ", 1-SE rule df =", se_1se_df_female, "\n")
```

```
df.min <- max(min_df_male, min_df_female)
df.1se <- max(se_1se_df_male, se_1se_df_female)

spline_fit_min_male <- smooth.spline(bmd_train_male$age, bmd_train_male$spnbmd, df = df.min)
spline_fit_min_female <- smooth.spline(bmd_train_female$age, bmd_train_female$spnbmd, df = df.min)

spline_fit_1se_male <- smooth.spline(bmd_train_male$age, bmd_train_male$spnbmd, df = df.1se)
spline_fit_1se_female <- smooth.spline(bmd_train_female$age, bmd_train_female$spnbmd, df = df.1se)


bmd %>%
  ggplot(aes(x = age, y = spnbmd, color = gender)) +
  geom_point(alpha = 0.3) +
  facet_wrap(~ gender) +
  geom_line(data = augment(spline_fit_min_male) %>% mutate(gender = "male"), aes(x = age, y = .fitted), color = "blue",
linewidth = 1) +
  geom_line(data = augment(spline_fit_min_female) %>% mutate(gender = "female"), aes(x = age, y = .fitted), color =
"red", linewidth = 1) +
  geom_line(data = augment(spline_fit_1se_male) %>% mutate(gender = "male"), aes(x = age, y = .fitted), color = "blue",
linetype = "dashed", linewidth = 1) +
  geom_line(data = augment(spline_fit_1se_female) %>% mutate(gender = "female"), aes(x = age, y = .fitted), color =
"red", linetype = "dashed", linewidth = 1) +
  labs(title = "Scatter plot with Spline Fits (Solid: min CV, Dashed: 1-SE rule)",
       subtitle = paste0("df.min = ", df.min, ", df.1se = ", df.1se))
```

1.4.3 Final fit (1 point)

1. Using the degrees of freedom chosen above, fit final spline models to bmd_train_male and bmd_train_female.

```
final_spline_male <- smooth.spline(bmd_train_male$age, bmd_train_male$spnbmd, df = df.min)
final_spline_female <- smooth.spline(bmd_train_female$age, bmd_train_female$spnbmd, df = df.min)
```

1.5 Evaluate (2 points)

1. Using the final models above, answer the following questions for boys and girls separately: What is the training RMSE? What is the test RMSE? Print these metrics in a nice table.
2. How do the training and test errors compare? What does this suggest about the extent of overfitting that has occurred?

```
# Function to calculate RMSE
calculate_rmse <- function(model, data, age_col = "age", spnbmd_col = "spnbmd") {
  predictions <- predict(model, x = data[[age_col]])$y
  rmse <- sqrt(mean((predictions - data[[spnbmd_col]])^2))
  return(rmse)
}

# Training RMSE
train_rmse_male <- calculate_rmse(final_spline_male, bmd_train_male)
train_rmse_female <- calculate_rmse(final_spline_female, bmd_train_female)

# Test RMSE
test_rmse_male <- calculate_rmse(final_spline_male, bmd_test_male)
test_rmse_female <- calculate_rmse(final_spline_female, bmd_test_female)



rmse_table <- tibble(
  Gender = c("Male", "Female"),
  Training_RMSE = c(train_rmse_male, train_rmse_female),
  Test_RMSE = c(test_rmse_male, test_rmse_female)
)

kable(rmse_table, caption = "RMSE for Spline Models") %>%
  kable_styling()
```

1.6 Interpret (6 points)

1. Using the degrees of freedom chosen above, redo the scatter plot with the overlaid spline fits, this time without faceting in order to directly compare the spline fits for boys and girls. Instead of faceting, distinguish the genders by color.

2. The splines help us see the trend in the data much more clearly. Eyeballing these fitted curves, answer the following questions. At what ages (approximately) do boys and girls reach the peaks of their growth spurts? At what ages does growth largely level off for boys and girls? Do these seem in the right ballpark?

```
bmd %>%
  ggplot(aes(x = age, y = spnbmd, color = gender)) +
  geom_point(alpha = 0.3) +
  geom_line(data = augment(final_spline_male) %>% mutate(gender = "male"), aes(x = age, y = .fitted), color = "blue", linewidth = 1) +
  geom_line(data = augment(final_spline_female) %>% mutate(gender = "female"), aes(x = age, y = .fitted), color = "red", linewidth = 1) +
  scale_color_manual(values = c("female" = "red", "male" = "blue")) +
  labs(title = "Scatter plot with Final Spline Fits (df=4)", color = "Gender")
```

2 KNN and bias-variance tradeoff (55 points for correctness; 8 points for presentation)

Setup: Apple farming

You own a square apple orchard, measuring 200 meters on each side. You have planted trees in a grid ten meters apart from each other. Last apple season, you measured the yield of each tree in your orchard (in average apples per week). You noticed that the yield of the different trees seems to be higher in some places of the orchard and lower in others, perhaps due to differences in sunlight and soil fertility across the orchard.

Unbeknownst to you, the yield Y of the tree planted E_1 meters to the right and E_2 meters up from the bottom left-hand corner of the orchard has distribution $Y = f(E) + \epsilon$, where

$$f(E) = 50 + 0.001E_1^2 + 0.001E_2^2, \epsilon \sim N(0, \sigma^2), \sigma = 4.$$

The data you collected are as in Figure 1.

The underlying trend is depicted in Figure 2, with the top right-hand corner of the orchard being more fruitful.

NOTE: Some of your answers for this question will include mathematical expressions. Please see this page for a quick guide on how to write mathematical expressions in R Markdown. Alternatively, you may write any mathematical derivations by hand, take photos of them, and include the images in your writeup via include_graphics().

2.1 A simple rule to predict this season's yield (15 points)

This apple season is right around the corner, and you'd like to predict the yield of each tree. You come up with perhaps the simplest possible prediction rule: predict this year's yield for any given tree based on last year's yield from that same tree. Without doing any programming, answer the following questions:

1. What is the training error of such a rule?

If we predict this year's yield for each tree based on last year's yield from the same tree, then for the training data (which is last year's data itself), the prediction for each tree is simply its own observed yield from last year. Therefore, the training error is zero. There is no difference between the prediction and the actual observed value for the data used to build the rule.

2. What is the mean squared bias, mean variance, and expected test error of this prediction rule?

2. What is the mean squared bias, mean variance, and expected test error of this prediction rule?

Let Y_{last} be last year's yield and Y_{this} be this year's yield for a given tree. Our prediction rule is $\hat{Y}_{this} = Y_{last}$. We know that $Y_{last} = f(E) + \epsilon_{last}$ and $Y_{this} = f(E) + \epsilon_{this}$, where $f(E)$ is the true underlying yield trend and $\epsilon_{last}, \epsilon_{this} \sim N(0, \sigma^2)$ are independent error terms with $\sigma^2 = 4^2 = 16$.

- **Bias:** The expected prediction is $E[\hat{Y}_{this}] = E[Y_{last}] = E[f(E) + \epsilon_{last}] = f(E) + E[\epsilon_{last}] = f(E)$. The true yield is $Y_{this} = f(E) + \epsilon_{this}$. The bias is $Bias[\hat{Y}_{this}] = E[\hat{Y}_{this}] - f(E) = f(E) - f(E) = 0$. Therefore, the **mean squared bias is** $Bias[\hat{Y}_{this}]^2 = 0^2 = 0$.
- **Variance:** The variance of the prediction is $Var[\hat{Y}_{this}] = Var[Y_{last}] = Var[f(E) + \epsilon_{last}] = Var[\epsilon_{last}] = \sigma^2 = 16$. Therefore, the **mean variance is** $Var[\hat{Y}_{this}] = 16$.
- **Expected Test Error:** The expected test error (Mean Squared Error, MSE) is given by $MSE = Bias^2 + Variance + \sigma^2_{irreducible}$. Here, irreducible error is also σ^2 as the test data also has noise $\epsilon_{this} \sim N(0, \sigma^2)$. So, $ExpectedTestError = Bias^2 + Variance + \sigma^2 = 0 + 16 + 16 = 32$.

  Alternatively, consider the test error for a single tree: $(\hat{Y}_{this} - Y_{this})^2 = (Y_{last} - Y_{this})^2 = ((f(E) + \epsilon_{last}) - (f(E) + \epsilon_{this}))^2 = (\epsilon_{last} - \epsilon_{this})^2$. The expected test error is $E[(\epsilon_{last} - \epsilon_{this})^2] = E[\epsilon^2_{last} - 2\epsilon_{last}\epsilon_{this} + \epsilon^2_{this}] = E[\epsilon^2_{last}] - 2E[\epsilon_{last}\epsilon_{this}] + E[\epsilon^2_{this}]$. Since ϵ_{last} and ϵ_{this} are independent with mean 0 and variance σ^2, we have $E[\epsilon^2_{last}] = Var[\epsilon_{last}] + E[\epsilon_{last}]^2 = \sigma^2$, $E[\epsilon^2_{this}] = \sigma^2$, and $E[\epsilon_{last}\epsilon_{this}] = E[\epsilon_{last}]E[\epsilon_{this}] = 0$. Thus, $ExpectedTestError = \sigma^2 - 0 + \sigma^2 = 2\sigma^2 = 2 \times 16 = 32$.

In summary:

- **Mean Squared Bias:** 0
- **Mean Variance:** 16
- **Expected Test Error:** 32

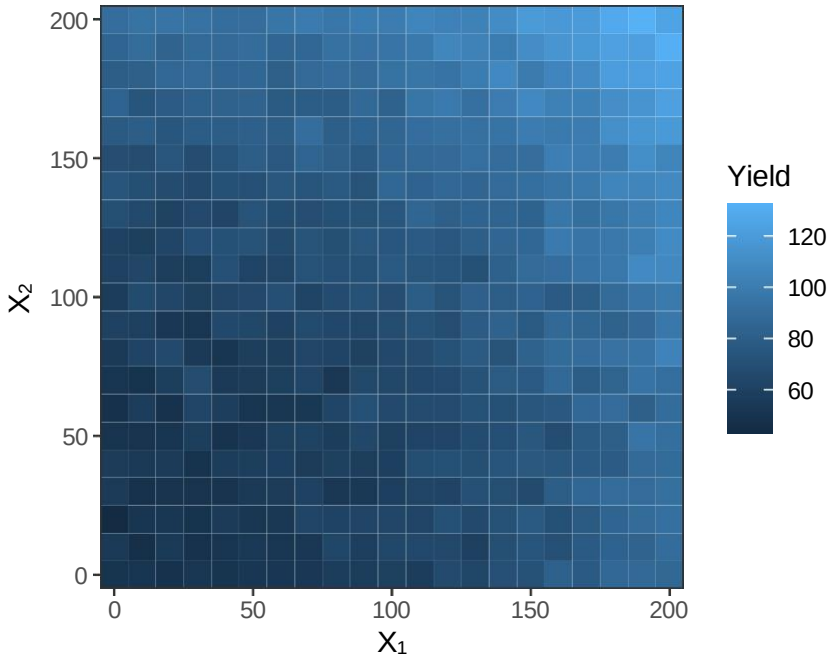

Figure 1: Apple tree yield for each 10m by 10m block of the orchard in a given year.

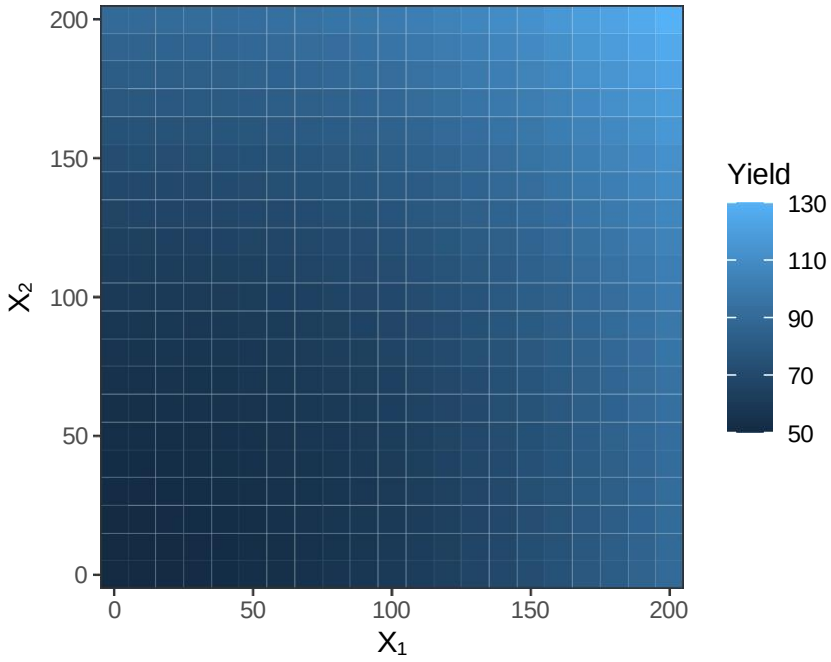

Figure 2: Underlying trend in apple yield for each 10m by 10m block of the orchard.

3. Why is this not the best possible prediction rule?

While this rule has zero training error and zero bias, it has a non-zero variance and a considerable expected test error (32). This rule is essentially memorizing the noise from last year's data. It's not learning the underlying trend

f(E)*f*(*E*)

. A better prediction rule would try to estimate

f(E)*f*(*E*)

and use that to predict this year's yield, rather than just copying last year's noisy observation. It is likely to overfit to the training data (last year's yield) and not generalize well to predict this year's yield, even though it has zero bias. It doesn't take into account the location

E*E*

of the tree, which is related to the underlying trend

f(E)*f*(*E*)

.

2.2 K-nearest neighbors regression (conceptual) (15 points)

As a second attempt to predict a yield for each tree, you average together last year's yields of the K trees closest to it (including itself, and breaking ties randomly if necessary). So if you choose $K = 1$, you get back the simple rule from the previous section. This more general rule is called *K-nearest neighbors (KNN) regression* (see ISLR p. 105).

KNN is not a parametric model like linear or logistic regression, so it is a little harder to pin down its degrees of freedom.

1. What happens to the model complexity as K increases? Why?

As **K increases, the model complexity decreases**. When K is small (e.g., K=1), the prediction for a tree is based on the yield of only the closest tree (itself in the K=1 case, or its nearest neighbor for K=2 and so on). This makes the model very flexible and able to fit complex patterns in the training data, potentially including noise. As K increases, the prediction is based on the average yield of a larger number of neighbors. Averaging over more neighbors smooths out the predictions, making the model less sensitive to local fluctuations and noise in the training data. In the extreme case where K is very large (approaching the total number of trees), the prediction becomes closer to the overall average yield, regardless of the location of the tree, making the model very simple and inflexible.

2. The degrees of freedom for KNN is sometimes considered n/K, where n is the training set size. Why might this be the case? [Hint: consider a situation where the data are clumped in groups of K.]

Consider a situation where the data points are clumped into groups of size approximately K. When we perform KNN regression with neighborhood size K, all data points within a group will tend to have very similar predictions because their K-nearest neighbors will largely be within the same group. Effectively, for each group of K points, the KNN model is fitting roughly one parameter (the average value for that group). If there are roughly n/K such independent groups, the model behaves as if it has approximately n/K degrees of freedom. This is a simplified analogy. In reality, degrees of freedom for non-parametric models like KNN are not as straightforward as in linear models, but n/K provides a reasonable intuition for how complexity changes with K. A larger K leads to fewer "effective parameters" or lower degrees of freedom.

3. Conceptually, why might increasing K tend to improve the prediction rule? What does this have to do with the bias-variance tradeoff?

Increasing K in KNN regression generally **reduces variance**. By averaging over more neighbors, we are smoothing out the predictions and making them less sensitive to the noise in individual training data points. This reduction in variance can lead to improved prediction performance, especially if the original model (with small K) was overfitting, i.e., had high variance and low bias. However, increasing K also tends to **increase bias**. As K gets larger, the KNN

prediction becomes a more global average, potentially ignoring local patterns and thus moving away from the true underlying function

f(E)*f*(*E*)

. There is a bias-variance tradeoff. Starting with a small K (high variance, low bias), increasing K initially can reduce variance more than it increases bias, leading to a decrease in the expected test error.

4. Conceptually, why might increasing *K* tend to worsen the prediction rule? What does this have to do with the bias-variance tradeoff?

While increasing K initially can improve predictions by reducing variance, **further increasing K can worsen the prediction rule by excessively increasing bias**. If K becomes too large, the KNN model becomes too smooth and inflexible. It may start to underfit the data, meaning it fails to capture important patterns in the underlying function

f(E)*f*(*E*)

. The increase in bias can eventually outweigh the reduction in variance, causing the expected test error to increase. This again illustrates the bias-variance tradeoff. There is an optimal value of K that balances bias and variance to minimize the expected test error. Too small K leads to high variance (overfitting), and too large K leads to high bias (underfitting).

2.3 K-nearest neighbors regression (simulation) (25 points)

Now, we try KNN for several values of *K*. For each value of *K*, we use a numerical simulation to compute the bias and variance for every tree in the orchard. These results are contained in training_results_summary below.

```
training_results_summary <- readRDS("training_results_summary.rds")
training_results_summary
```

```
## # A tibble: 6,174 x 5
##        K    X1    X2     bias variance
##    <int> <dbl> <dbl>    <dbl>    <dbl>
##  1     1     0     0  -0.250       16.2
##  2     1     0    10   0.140       12.2
##  3     1     0    20  -0.523       20.4
##  4     1     0    30   0.109       15.6
##  5     1     0    40  -0.566       21.4
##  6     1     0    50  -0.336       15.9
##  7     1     0    60  -1.04        12.4
##  8     1     0    70  -0.0213      12.4
##  9     1     0    80  -0.884       13.5
## 10     1     0    90  -0.342       14.6
## # ... with 6,164 more rows
## # i Use `print(n = ...)` to see more rows
```

1. Create a new tibble called overall_results the contains the mean squared bias, mean variance, and expected test error for each value of K. This tibble should have four columns: K, mean_sq_bias, mean_variance, and expected_test_error.

```
overall_results <- training_results_summary %>%
  group_by(K) %>%
  summarise(
    mean_bias = mean(bias),
```

```
  mean_variance = mean(variance)
 ) %>%
 mutate(
  mean_sq_bias = mean_bias^2,
  expected_test_error = mean_sq_bias + mean_variance + 16 # Assuming sigma^2 = 16 is irreducible error
 ) %>%
 select(K, mean_sq_bias, mean_variance, expected_test_error)


print(overall_results)
```

2. Using `overall_results`, plot the mean squared bias, mean variance, and expected test error on the same axes as a function of K. Based on this plot, what is the optimal value of `K`?

```
overall_results_long <- overall_results %>%
 pivot_longer(cols = c(mean_sq_bias, mean_variance, expected_test_error),
          names_to = "metric", values_to = "value")


ggplot(overall_results_long, aes(x = K, y = value, color = metric)) +
 geom_line() +
 geom_point() +
 labs(title = "Bias-Variance Tradeoff for KNN Regression",
    y = "Value") +
 scale_color_discrete(labels = c("Expected Test Error", "Mean Squared Bias", "Mean Variance"))
```

3. We are used to the bias decreasing and the variance increasing when going from left to right in the plot. Here, the trend seems to be reversed. Why is this the case?

In the typical bias-variance tradeoff plots we often see in textbooks or lectures, the x-axis usually represents model complexity, which *decreases* as we move from left to right (e.g., in polynomial regression, the degree of polynomial might be plotted in reverse order on the x-axis). In this case, the x-axis is K for KNN. **As K increases, the model complexity *decreases*** (as discussed in 2.2.1). Therefore, moving from left to right on the x-axis in our plot (increasing K, decreasing complexity) corresponds to moving from a complex model to a simpler model.
For KNN, as we increase K (move right in the plot):

- **Variance decreases:** As expected, simpler models have lower variance.
- **Bias increases:** As expected, simpler models have higher bias (although in this case, the squared bias is already very small and increasing slowly but steadily from left to right).

So, the trends are actually consistent with the usual bias-variance tradeoff when we consider that the x-axis is representing K, which is inversely related to model complexity. It's just that here, the x-axis is directly K, not a measure of complexity that decreases with x.

4. The mean squared bias has a strange bump between K = 1 and K = 5, increasing from K = 1 to K = 2 but then decreasing from K = 2 to K = 5. Why does this bump occur? [Hint: Think about the rectangular grid configuration of the trees. So for a given tree, the closest tree is itself, and then the next closest four trees are the ones that are one tree up, down, left, and right from it.]

The bump in mean squared bias between K=1 and K=5 likely arises due to the **grid structure of the orchard**.

- **K=1:** For K=1, the prediction for a tree is just its own yield from last year. This is highly localized.
- **K=2:** For K=2, we are averaging the yield of the tree itself and its nearest neighbor. In a grid, the nearest neighbor for a tree (excluding diagonals) is likely one of its immediate neighbors (up, down, left, right) if they exist. When K=2, we are including a neighbor, which might introduce some bias if the yield trend is not perfectly smooth and varies even between adjacent trees.
- **K=3, K=4, K=5:** As we increase K to 3, 4, and 5, we are now averaging over the tree itself and its closest neighbors (which in a grid, for K=5, will typically include the tree itself and its four immediate neighbors: up, down, left, right). By averaging over these immediate neighbors, we are likely smoothing out some of the local noise and getting closer to the underlying trend
  f(E)$f(E)$
  . This averaging process appears to reduce the bias more effectively as K increases from 2 to 5, compared to K=1 to K=2.
  The increase in bias from K=1 to K=2 could be due to a slight mismatch between the very localized K=1 prediction and the slightly less localized K=2 prediction in relation to the underlying trend
  f(E)$f(E)$
  and the grid structure. It's a subtle effect related to how the neighborhood expands in a grid and how it interacts with the true function and noise.

5. Based on the information in training_results_summary, which tree and which value of K gives the overall highest absolute bias? Does the sign of the bias make sense? Why do this particular tree and this particular value of K give us the largest absolute bias?

```
max_bias_tree <- training_results_summary %>%
  mutate(abs_bias = abs(bias)) %>%
  filter(abs_bias == max(abs_bias))


print(max_bias_tree)
```

The bias is negative. Bias =

E[Y^]−f(E)$E[Y^]-f(E)$

. A negative bias means that on average, the KNN prediction

Y^$Y^$

for this tree and K=7 is *lower* than the true underlying yield

f(E)$f(E)$

for this location.

Why this tree and K? Let's look at the location (X1 ≈ 0, X2 = 60). X2 = 60 is in the middle range of X2 values (0 to 200), but X1 ≈ 0 is at the edge of the orchard in the X1 direction. For trees near the edges or corners of the orchard, the neighborhood might be asymmetrically distributed. For K=7, the neighborhood might be skewed, perhaps missing neighbors in one direction (e.g., if the tree is on the boundary). If the yield trend

f(E)$f(E)$

is changing in a particular direction (e.g., increasing with X1 or X2), an asymmetrically sampled neighborhood could

lead to a biased average. K=7 might be a value where this asymmetry is pronounced for this particular tree's location, leading to a larger bias. For different K values or different tree locations, the neighborhood configuration and resulting bias might change.

The tree with coordinates (X1 ≈ 0.037, X2 = 60) and K=7 has the highest absolute bias, with a bias of approximately -2.92.

6. Redo the bias-variance plot from part 2, this time putting `df = n/K` on the x-axis. What do we notice about the variance as a function of `df`?

```
overall_results_df <- overall_results %>%
  mutate(df = 400/K) # Assuming n = 400 trees (20x20 grid)

overall_results_long_df <- overall_results_df %>%
  pivot_longer(cols = c(mean_sq_bias, mean_variance, expected_test_error),
               names_to = "metric", values_to = "value")

ggplot(overall_results_long_df, aes(x = df, y = value, color = metric)) +
  geom_line() +
  geom_point() +
  scale_x_reverse() + # Reverse x-axis to show df decreasing from left to right (complexity decreasing)
  labs(title = "Bias-Variance Tradeoff vs. Degrees of Freedom (df = n/K)",
       x = "Degrees of Freedom (df = n/K, reversed axis)",
       y = "Value") +
  scale_color_discrete(labels = c("Expected Test Error", "Mean Squared Bias", "Mean Variance"))
```

When we plot against df = n/K (reversed x-axis, so complexity decreases from left to right), we notice that:

- **Variance increases as df increases (complexity increases):** Moving from right to left in the plot (increasing df, increasing complexity), the mean variance curve generally increases. This is consistent with the idea that more complex models have higher variance.
- **Bias (squared bias) decreases as df increases (complexity increases):** The mean squared bias curve generally decreases as we move from right to left. More complex models tend to have lower bias.
- **Expected test error shows a U-shape:** It initially decreases as df increases (complexity increases, variance increases but bias decreases more), reaches a minimum, and then starts to increase again as df becomes very large (complexity too high, variance dominates).

These trends are now in the "expected" direction based on the typical bias-variance tradeoff understanding, where complexity is increasing from right to left on the x-axis.

7. Derive a formula for the KNN mean variance. [Hint: First, write down an expression for the KNN prediction for a given tree. Then, compute the variance of this quantity using the fact that the variance of the average of N independent random variables each with variance s^2 is s^2/N. Finally, compute the mean variance by averaging over trees.]

For a given tree location E, let $Y_{(1)}, Y_{(2)}, \ldots, Y_{(K)}$ be the yields of the K-nearest neighbors of this tree (including itself). The KNN prediction is:

$$\hat{Y}_{KNN}(E) = \frac{1}{K}\sum_{i=1}^{K} Y_{(i)}$$

We know that $Y_{(i)} = f(E_{(i)}) + \epsilon_{(i)}$, where $E_{(i)}$ is the location of the i-th nearest neighbor and $\epsilon_{(i)} \sim N(0, \sigma^2)$ are independent error terms. For simplicity, we approximate $f(E_{(i)}) \approx f(E)$ for all neighbors. Then, $Y_{(i)} \approx f(E) + \epsilon_{(i)}$.

So,

$$\hat{Y}_{KNN}(E) \approx \frac{1}{K}\sum_{i=1}^{K}(f(E) + \epsilon_{(i)}) = f(E) + \frac{1}{K}\sum_{i=1}^{K}\epsilon_{(i)}$$

The variance of the KNN prediction is:

$$Var[\hat{Y}_{KNN}(E)] = Var\left[f(E) + \frac{1}{K}\sum_{i=1}^{K}\epsilon_{(i)}\right] = Var\left[\frac{1}{K}\sum_{i=1}^{K}\epsilon_{(i)}\right]$$

Since $\epsilon_{(i)}$ are independent and identically distributed with variance σ^2, and using the property that $Var(\sum X_i) = \sum Var(X_i)$ for independent variables, we have:

$$Var\left[\sum_{i=1}^{K}\epsilon_{(i)}\right] = \sum_{i=1}^{K} Var[\epsilon_{(i)}] = K\sigma^2$$

Therefore,

$$Var[\hat{Y}_{KNN}(E)] = Var\left[\frac{1}{K}\sum_{i=1}^{K}\epsilon_{(i)}\right] = \left(\frac{1}{K}\right)^2 Var\left[\sum_{i=1}^{K}\epsilon_{(i)}\right] = \left(\frac{1}{K}\right)^2 (K\sigma^2) = \frac{\sigma^2}{K}$$

Thus, the variance of the KNN prediction for a given tree location E is approximately $\frac{\sigma^2}{K}$.

Thus, the variance of the KNN prediction for a given tree location E is approximately $\frac{\sigma^2}{K}$.

The mean variance, averaged over all trees, is also approximately $\frac{\sigma^2}{K}$, as this formula does not depend on the specific tree location E. Given $\sigma^2 = 16$, the mean variance formula is $\frac{16}{K}$.

8. Create a plot like that in part 6, but with the mean variance formula from part 7 superimposed as a dashed curve. Do these two variance curves match?

```
variance_formula <- function(K, sigma_squared = 16) {
  sigma_squared / K
}


overall_results_df_with_formula <- overall_results_df %>%
  mutate(formula_variance = variance_formula(K))


ggplot(overall_results_long_df %>% filter(metric == "mean_variance"), aes(x = df, y = value, color = metric)) +
  geom_line() +
  geom_point() +
  geom_line(data = overall_results_df_with_formula, aes(x = df, y = formula_variance, color = "Formula Variance"), linetype = "dashed") +
  scale_x_reverse() +
  labs(title = "Mean Variance: Simulated vs. Formula (df = n/K)",
       x = "Degrees of Freedom (df = n/K, reversed axis)",
       y = "Variance") +
  scale_color_manual(values = c("mean_variance" = "blue", "Formula Variance" = "red"),
```

```
       labels = c("Mean Variance (Simulated)", "Mean Variance (Formula)"))
```

Do these two variance curves match?

Yes, the simulated mean variance curve and the curve derived from the formula

σ2K*Kσ*2

(dashed red line) match quite closely, especially for larger values of df (smaller K). There is a slight deviation for smaller df (larger K), but overall, the formula provides a good approximation for the mean variance observed in the simulation. This confirms our derivation and understanding of how KNN variance scales with K (or degrees of freedom).