CS M148 PS2

Tejas Kamtam

305749402

---

1. Linear Regression

Q1.a

Considering the LAD algorithm and LS algorithms that we discussed in class, which algorithm is more sensitive to outliers and why? Given this, which algorithm will always perform better on the training set and why? Will this also be the case for the validation set?

The LS algorithm is more sensitive to outliers because it minimizes the squared deviations of the residuals while LAD minimizes the absolute deviations. Consider the example of a residual of 10. LAD would calculate a loss of $|10| = 10$ while LS would calculate a loss of $10^2 = 100$. Thus, outliers have a much larger impact on the LS algorithm.

Given this, the LS algorithm will always perform better on the training set because it minimizes the squared deviations and our training data will have been cleaned to remove outliers. Additionally, the LS algo minimizes the Euclidean (L2) distance while LAD minimizes the Manhattan (L1) distance. It is usually the case that the Euclidean distance is smaller than the Manhattan distance, so LS will almost always perform better.

However, this is not necessarily true for validation. The validation set is made to mock real world data so it is usually unclean and could possibly contain outliers. If this is the case, the LAD algorithm may perform better because it is less sensitive to outliers.

Q1.b

Consider the following restaurant data with a column for ambiance, food, service, and overall rating rating in Figure 1. The values for the variables are numeric (and continuous not categorical, but only integers to help with calculations.) You may use calculators as needed,

but do not just call sklearn or statsmodels packages.

restaurant	food	ambience	service	rating
1	8	8	8	8
2	8	9	8	8
3	8	8	7	8
4	7	9	7	7
5	6	8	7	7
6	6	7	5	5
7	5	5	5	5
8	7	9	6	7
9	6	8	7	6
10	8	7	7	8
11	7	9	7	7

i.

Write the multiple regression model to predict overall restaurant rating rating as the response variable using all other variables as the predictor variables. You do not have to fit this model.

Given the data:

- $\vec{x}_1 = [8 \quad 8 \quad 8]$
- $\vec{x}_2 = [8 \quad 9 \quad 8]$
- $\vec{x}_3 = [8 \quad 8 \quad 7]$
- $\vec{x}_4 = [7 \quad 9 \quad 7]$
- $\vec{x}_5 = [6 \quad 8 \quad 7]$
- $\vec{x}_6 = [6 \quad 7 \quad 5]$
- $\vec{x}_7 = [5 \quad 5 \quad 5]$
- $\vec{x}_8 = [7 \quad 9 \quad 6]$
- $\vec{x}_9 = [6 \quad 8 \quad 7]$
- $\vec{x}_{10} = [8 \quad 7 \quad 7]$
- $\vec{x}_{11} = [7 \quad 9 \quad 7]$

We can define the following values:

$$\boldsymbol{X} = \begin{bmatrix} 1 & \vec{x}_1 \\ \vdots & \vdots \\ 1 & \vec{x}_{11} \end{bmatrix}$$

$$\boldsymbol{\theta}^T = [\theta_0 \quad \dots \quad \theta_{11}]$$

Then the multiple regression model is then given by:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\vec{\theta}$$

ii.

Calculate the correlation between restaurant rating rating and each of the predictors by hand

Correlation is given by:

$$r_{x,y} = \sum_{i=1}^{n} \frac{(x_i - \bar{x})(y_i - \bar{y})}{\sigma_x \sigma_y}$$

We can first find the mean and standard deviation of each predictor and response variable:

- food:

$$\bar{x}_{\text{food}} = 6.909$$
$$\sigma_{\text{food}} = 0.996$$

- ambience:

$$\bar{x}_{\text{ambience}} = 7.909$$
$$\sigma_{\text{ambience}} = 1.164$$

- service:

$$\bar{x}_{\text{service}} = 6.727$$
$$\sigma_{\text{service}} = 0.962$$

- rating:

$$\bar{x}_{\text{rating}} = 6.909$$
$$\sigma_{\text{rating}} = 1.083$$

Now we can calculate the correlation between each predictor and response variable as dot products:

$$r_{\text{food,rating}} = \frac{(X_{*,2} - \bar{x}_{\text{food}}) \cdot (y - \bar{y})}{\sigma_{\text{food}} \sigma_{\text{rating}}} = 10.112$$

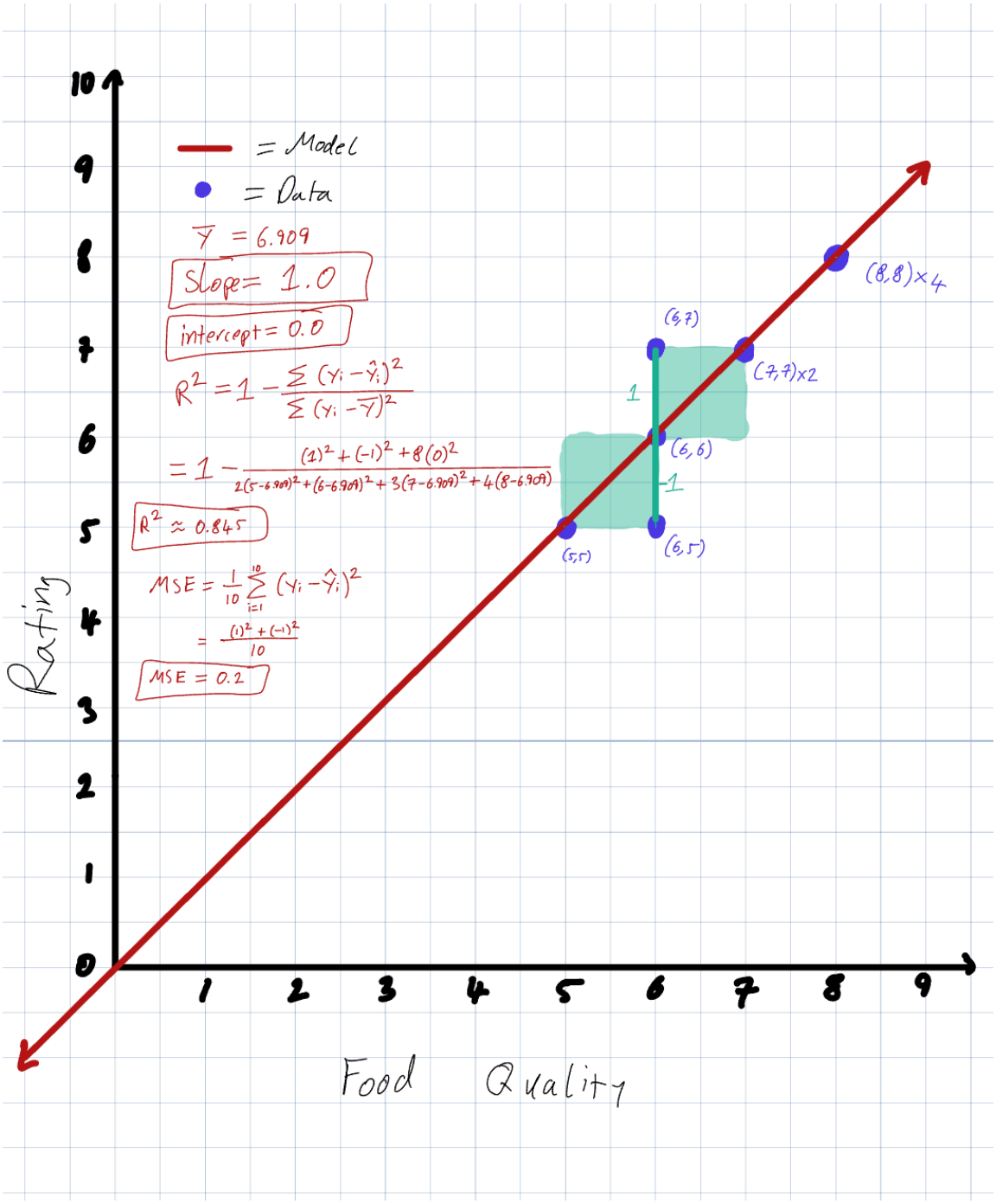
$$r_{\text{ambience,rating}} = \frac{(X_{*,3} - \bar{x}_{\text{ambiance}}) \cdot (y - \bar{y})}{\sigma_{\text{ambiance}} \sigma_{\text{rating}}} = 6.271$$

$$r_{\text{service,rating}} = \frac{(X_{*,4} - \bar{x}_{\text{service}}) \cdot (y - \bar{y})}{\sigma_{\text{service}} \sigma_{\text{rating}}} = 9.333$$

where $X_{*,2}$ is the second column of $\boldsymbol{X}$ and similarly for the other columns; y is the column vector of the response variable: rating.

iii.

For the predictor with the highest correlation with restaurant quality rating, solve the linear regression model by hand using least squares and the first 10 data points. (This will be a simple linear regression.) Draw by hand a plot of the points and the model. On the plot mark the data points, the intercept, slope, and R2. What is the MSE for the training data? Draw each residual that you used to calculate the MSE and R2 on the graph.



For the final restaurant, $\vec{x}_{11} = [7 \quad 9 \quad 7]$, the model takes the food quality of 7, which when input into the model (because the LR models the function $y = x$) predicts a rating of

$\hat{y}_{11} = 7$.

The true rating is 7, so the error/residual is $7 - 7 = 0$.

2. Classification

Below I use the terms "coefficient" and "weight/bias" interchangeably.

Q2.a

Explain how you could use the lasso algorithm as a feature selection technique.

Lasso regression penalizes the absolute value of the coefficients of the features. If a feature has a coefficient of 0, it is excluded from the model. Thus, by tuning the penalty parameter λ, we can exclude features from the model, selecting for those with the strongest correlation with the response variable.

Q2.b

Describe how ridge regularization simplifies the LS solution when there are many predictors or collinearity

Ridge regression adds a penalty term to the loss function that penalizes large weights by increasing the loss for large coefficients. This simplifies the LS solution by shrinking the weights of the features that are not strongly correlated with the response variable, simplifying the correlation of features to a handful of highly correlated features (but still maintaining the contribution of less strongly correlated features - unlike lasso which completely ignores them).

Q2.c

In either lasso or ridge regularization, what would happen if the λ value were set to 0? What about if it is set to a value < 0?

If the regularization parameter is $\lambda = 0$, the model is equivalent to the unregularized/base model because we effectively remove the regularization term.

If the regularization parameter is $\lambda < 0$, the model rewards large weights causing the model to converge quicker, likely causing the model to overfit the training data.

3. Classification

Q3.a

Suppose there is a logistic regression fit model to predict diabetes risk based on the expression of a gene, g, where high risk is class 1 and low risk is class 0. The fit for the model is

$$p = \frac{1}{1 + e^{-1.75+0.5g}}$$

What are the predicted risk classes for patient 1 who has value of $g = 5$ and patient 2 who has a value of $g = 1$ using $.5$ as the probability threshold?

Using the logistic regression model, we can calculate the predicted risk for each patient:

For patient 1 with $g = 5$:

$$p_1 = \frac{1}{1 + e^{-1.75+5(0.5)}} = \frac{1}{1 + e^{0.75}} \approx 0.321 < 0.5 \implies \text{class } 0$$

For patient 2 with $g = 1$:

$$p_2 = \frac{1}{1 + e^{-1.75+1(0.5)}} = \frac{1}{1 + e^{-1.25}} \approx 0.777 > 0.5 \implies \text{class } 1$$

Q3.b

Consider the following table of fictitious weather conditions data in Figure 2 to determine whether a person should go surfing. Use the 3-KNN algorithm by hand on the first 12 data points to construct a classifier to predict whether to surf or not on the last two data points. (Calculators are fine, but do not use Python.)

outlook	temperature	humidity	windy	surf
sunny	85	85	FALSE	maybe
sunny	80	90	TRUE	no
overcast	83	86	FALSE	yes
rainy	70	96	FALSE	maybe
rainy	68	80	FALSE	yes
rainy	65	70	TRUE	no
overcast	64	65	TRUE	yes
sunny	72	95	FALSE	no
sunny	69	70	FALSE	yes
rainy	75	80	FALSE	maybe
sunny	75	70	TRUE	yes
overcast	72	90	TRUE	maybe
overcast	81	75	FALSE	yes
rainy	71	91	TRUE	no

For each of the last 2 data points, we can add their Euclidean distances from each of the first 12 data points (we can map boolean variables to 1 for true and 0 for false; and sunny = 1, overcast = 2, rain = 3; and yes = 1, no = 0, maybe = 2). This gives us the following table:

p	outlook	temperature	humidity	windy	surf	dist to p13	dist to p14
1	1	85	85	0	2	10.817	15.395
2	1	80	90	1	0	15.100	9.274
3	2	83	86	0	1	11.180	13.077
4	3	70	96	0	2	23.728	5.196
5	3	68	80	0	1	13.964	11.446
6	3	65	70	1	0	16.823	21.840
7	2	64	65	1	1	19.748	26.944
8	1	72	95	0	0	21.955	4.690
9	1	69	70	0	1	13.038	21.213
10	3	75	80	0	2	7.874	11.747
11	1	75	70	1	1	7.937	21.471
12	2	72	90	1	2	17.521	1.732
13	2	81	75	0	1	0	-

p	outlook	temperature	humidity	windy	surf	dist to p13	dist to p14
14	3	71	91	1	0	-	0

Based on the table above, we can classify the last two points using the 3-KNN algorithm (not counting the point itself, assuming classification of one point is independent of the other, and taking the majority class of the three closest points):

- p13: $\text{surf} = 2 = \text{maybe}$
- p14: $\text{surf} = 2 = \text{maybe}$

4. Deriving Bias-Variance Tradeoff in Regression

In this problem, you will work step-by-step through the derivation of the bias-variance tradeoff starting from the Mean Squared Error (MSE). The MSE is a measure of the average squared difference between predicted values and true values, similar to how we've discussed the Least Squares Error Loss in linear regression. Follow each part carefully and show your work.

Given: Let $f(x)$ represent the true function you are trying to estimate. Let $\hat{f}(x)$ be an estimator of $f(x)$ built from your model using training data. Let y be the observed data such that $y = f(x) + \epsilon$, where ϵ is random noise with zero mean and variance σ^2 (i.e., $\mathbb{E}[\epsilon] = 0$ and $\text{Var}(\epsilon) = \sigma^2$).

Objective: To derive the bias-variance decomposition of the MSE for a regression estimator.

Q4.a

Given the following definitions: trained on different samples from the same data. High variance means the model is sensitive to

1. The Mean Squared Error (MSE) of the estimator $\hat{f}(x)$ is the expected squared difference between the predicted and the true value.
2. The bias of the estimator $\hat{f}(x)$ is the expected difference between the predicted and the true value (Note: not the expected squared difference).
3. The variance of the estimator $\hat{f}(x)$ measures how much the model's predictions vary if

changes in the training data, while low variance indicates more stable predictions. Concretely, it is defined as $\text{Var}(\hat{f}(x)) = \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]$.

Decompose the formula for the MSE into three components: the **bias** of the estimator, the **variance** of the estimator, and the **irreducible noise** σ^2. Specifically, you should express:

$$\text{MSE}(\hat{f}(x)) = \text{Bias}^2(\hat{f}(x)) + \text{Var}(\hat{f}(x)) + \sigma^2$$

(Hint: use the identity $(a-b)^2 = a^2 - 2ab + b^2$, perhaps even several times)

(Hint2: Note that ϵ and $\hat{f}(x)$ are independent variables, meaning that $\mathbb{E}[\epsilon\hat{f}(x)] = \mathbb{E}[\epsilon]\mathbb{E}[\hat{f}(x)]$)

We must show that:

$$\text{MSE}(\hat{f}(x)) = \text{Bias}^2(\hat{f}(x)) + \text{Var}(\hat{f}(x)) + \sigma^2$$

We can begin with the definition of the MSE using property 1:

$$\text{MSE}(\hat{f}(x)) = \mathbb{E}[(\hat{f}(x) - y)^2]$$

Here, we can observe that to decompose bias and variance, we require an **expectation of the estimator** term to be within the squared difference. The most straightforward way to do this while maintaining equality is inserting the term $\mathbb{E}[\hat{f}(x)] - \mathbb{E}[f(x)] = 0$ into the squared difference:

$$\text{MSE}(\hat{f}(x)) = \mathbb{E}\left[\left(\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right) + \left(\mathbb{E}[\hat{f}(x)] - y\right)\right)^2\right]$$

Now, we can expand the square:

$$\text{MSE}(\hat{f}(x)) = \mathbb{E}\Bigg[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)^2 + 2\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)\left(\mathbb{E}[\hat{f}(x)] - y\right) + \left(\mathbb{E}[\hat{f}(x)] - y\right)^2\Bigg]$$

Sums are distributive over expectations, so we can split the expectation into three separate expectations:

$$\text{MSE}(\hat{f}(x)) = \mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)^2\right] + 2\mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)\left(\mathbb{E}[\hat{f}(x)] - y\right)\right] + \mathbb{E}\left[\left(\mathbb{E}[\hat{f}(x)] - y\right)^2\right]$$

The first term is the definition of variance of the estimator $\hat{f}(x)$. So we can substitute the definition of variance for the first term and expand the last term:

$$\text{MSE}(\hat{f}(x)) = \text{Var}(\hat{f}(x)) + 2\mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)\left(\mathbb{E}[\hat{f}(x)] - y\right)\right] + \mathbb{E}\left[\mathbb{E}[\hat{f}(x)]^2 + y^2 - 2y\mathbb{E}[\hat{f}(x)]\right]$$

We can substitute $y = f(x) + \epsilon$ into the last term:

$$\text{MSE}(\hat{f}(x)) = \text{Var}(\hat{f}(x)) + 2\mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)\left(\mathbb{E}[\hat{f}(x)] - y\right)\right] + \mathbb{E}\left[\mathbb{E}[\hat{f}(x)]^2 + f(x)^2 + \epsilon^2 - 2f(x)\epsilon - 2f(x)\mathbb{E}[\hat{f}(x)] - 2\epsilon\mathbb{E}[\hat{f}(x)]\right]$$

Now, distributing the expectation over the linear combination in the 3rd term, we get (note we use the property that the expectation of the expectation is the inner expectation, i.e., $\mathbb{E}[\mathbb{E}[\hat{f}(x)]] = \mathbb{E}[\hat{f}(x)]$, a corrolary to this is that the expectation of a random variable is a constant so expectations including an inner expectation as a product can be brought out of the expectation as a multiplicative constant):

Observe that we also use the fact that the estimator and noise are independent, so $\mathbb{E}[\hat{f}(x)\epsilon] = \mathbb{E}[\hat{f}(x)]\mathbb{E}[\epsilon]$:

$$\text{MSE}(\hat{f}(x)) = \text{Var}(\hat{f}(x)) + 2\mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)\left(\mathbb{E}[\hat{f}(x)] - y\right)\right] + \mathbb{E}[\hat{f}(x)]^2 - 2\mathbb{E}[\hat{f}(x)]\mathbb{E}[f(x)] - 2\mathbb{E}[\hat{f}(x)]\mathbb{E}[\epsilon] + \mathbb{E}[f(x)^2] + 2\mathbb{E}[f(x)]\mathbb{E}[\epsilon] + \mathbb{E}[\epsilon^2]$$

We know $\mathbb{E}[\epsilon] = 0$, so we can simplify to:

$$\text{MSE}(\hat{f}(x)) = \text{Var}(\hat{f}(x)) + 2\mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)\left(\mathbb{E}[\hat{f}(x)] - y\right)\right] + \mathbb{E}[\hat{f}(x)]^2 - 2\mathbb{E}[\hat{f}(x)]\mathbb{E}[f(x)] + \mathbb{E}[f(x)^2] + \mathbb{E}[\epsilon^2]$$

Observe that:

$$\text{Var}(\epsilon) = \mathbb{E}[\epsilon^2] - \mathbb{E}[\epsilon]^2$$

Because the noise is zero mean, $\mathbb{E}[\epsilon] = 0$, so $\mathbb{E}[\epsilon^2] = \sigma^2$. Thus:

$$\text{MSE}(\hat{f}(x)) = \text{Var}(\hat{f}(x)) +$$
$$2\mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)\left(\mathbb{E}[\hat{f}(x)] - y\right)\right] +$$
$$\mathbb{E}[\hat{f}(x)]^2 - 2\mathbb{E}[\hat{f}(x)]\mathbb{E}[f(x)] + \mathbb{E}[f(x)^2] + \sigma^2$$

Now we can expand the 2nd term:

$$\text{MSE}(\hat{f}(x)) = \text{Var}(\hat{f}(x)) +$$
$$2\mathbb{E}\left[\hat{f}(x)\mathbb{E}[\hat{f}(x)] - \hat{f}(x)y - \mathbb{E}[\hat{f}(x)]^2 + \mathbb{E}[\hat{f}(x)]y\right] +$$
$$\mathbb{E}[\hat{f}(x)]^2 - 2\mathbb{E}[\hat{f}(x)]\mathbb{E}[f(x)] + \mathbb{E}[f(x)^2] + \sigma^2$$

We can again distribute the expectation over the linear combination in the 2nd term:

$$\text{MSE}(\hat{f}(x)) = \text{Var}(\hat{f}(x)) +$$
$$2\mathbb{E}[\hat{f}(x)]^2 - 2\mathbb{E}[\hat{f}(x)y] - 2\mathbb{E}[\hat{f}(x)]^2 + 2\mathbb{E}[\hat{f}(x)]\mathbb{E}[y] +$$
$$\mathbb{E}[\hat{f}(x)]^2 - 2\mathbb{E}[\hat{f}(x)]\mathbb{E}[f(x)] + \mathbb{E}[f(x)^2] + \sigma^2$$

And simplifying, we get:

$$\text{MSE}(\hat{f}(x)) = \text{Var}(\hat{f}(x)) -$$
$$2\mathbb{E}[\hat{f}(x)y] + 2\mathbb{E}[\hat{f}(x)]\mathbb{E}[y] +$$
$$\mathbb{E}[\hat{f}(x)]^2 - 2\mathbb{E}[\hat{f}(x)]\mathbb{E}[f(x)] + \mathbb{E}[f(x)^2] + \sigma^2$$

Substituting $y = f(x) + \epsilon$ into the 2nd term and simplifying via $\mathbb{E}[\epsilon] = 0$, we get:

$$\text{MSE}(\hat{f}(x)) = \text{Var}(\hat{f}(x)) -$$
$$2\mathbb{E}[\hat{f}(x)f(x)] + 2\mathbb{E}[\hat{f}(x)]\mathbb{E}[f(x)] +$$
$$\mathbb{E}[\hat{f}(x)]^2 - 2\mathbb{E}[\hat{f}(x)]\mathbb{E}[f(x)] + \mathbb{E}[f(x)^2] + \sigma^2$$

$$\text{MSE}(\hat{f}(x)) = \text{Var}(\hat{f}(x)) + \mathbb{E}[\hat{f}(x)]^2 - 2\mathbb{E}[\hat{f}(x)f(x)] + \mathbb{E}[f(x)^2] + \sigma^2$$

Note here $f(x)$ is a constant (independent of random variables, i.e. 0 variance), so $\mathbb{E}[f(x)] = f(x)$ and $\mathbb{E}[f(x)^2] = f(x)^2 = \mathbb{E}[\hat{f}(x)]^2$.

Now observe the expansion of the Bias term we are looking for (using the fact that $\mathbb{E}[\epsilon] = 0$):

$$\text{Bias}^2(\hat{f}(x)) = \left(\mathbb{E}[\hat{f}(x)] - \mathbb{E}[y]\right)^2 = \mathbb{E}[\hat{f}(x)]^2 - 2\mathbb{E}[\hat{f}(x)(f(x) + \epsilon)] + \mathbb{E}[f(x) + \epsilon]^2$$

$$= \mathbb{E}[\hat{f}(x)]^2 - 2\mathbb{E}[\hat{f}(x)f(x)] + \mathbb{E}[f(x)]^2$$

We can now combine the bias, variance, and irreducible noise terms to get:

$$\text{MSE}(\hat{f}(x)) = \text{Bias}^2(\hat{f}(x)) + \text{Var}(\hat{f}(x)) + \sigma^2$$

QED

Q4.b

The bias-variance tradeoff is a fundamental concept in statistical modeling. Why is it important to manage the tradeoff between bias and variance when building predictive models, and how does this tradeoff impact the overall error? Make sure to use the terms "overfitting" or "underfitting" in your answer.

It's important to manage bias and varianve when building/training models as high bias can cause underfitting and high variance can cause overfitting. The bias-variance tradeoff can also help inform us about the choice of model complexity. High bias models are simple and have low variance resulting in high error on training and test data. High variance models tend to be more complex and resulting in low error on training data but high error on test data.

Q4.c

Give an example of a model with high bias and low variance, and another with low bias and high variance.

A model with high bias and low variance could be a simple linear regression model. We can see this as the model does not vary as it's features scale (because the model is linear). However, the model has high error on training and test data as it is usually unable to model the true complexity of the data.

A model with low bias and high variance could be a high degree polynomial regression model. We can see this as the model varies a lot between datapoints across feature axes (usually thios would be plotted as having many bends and turns in the model). However, the model has low error on training data as it almost perfectly models the training data, but high error on test data as it overfits to the training data.