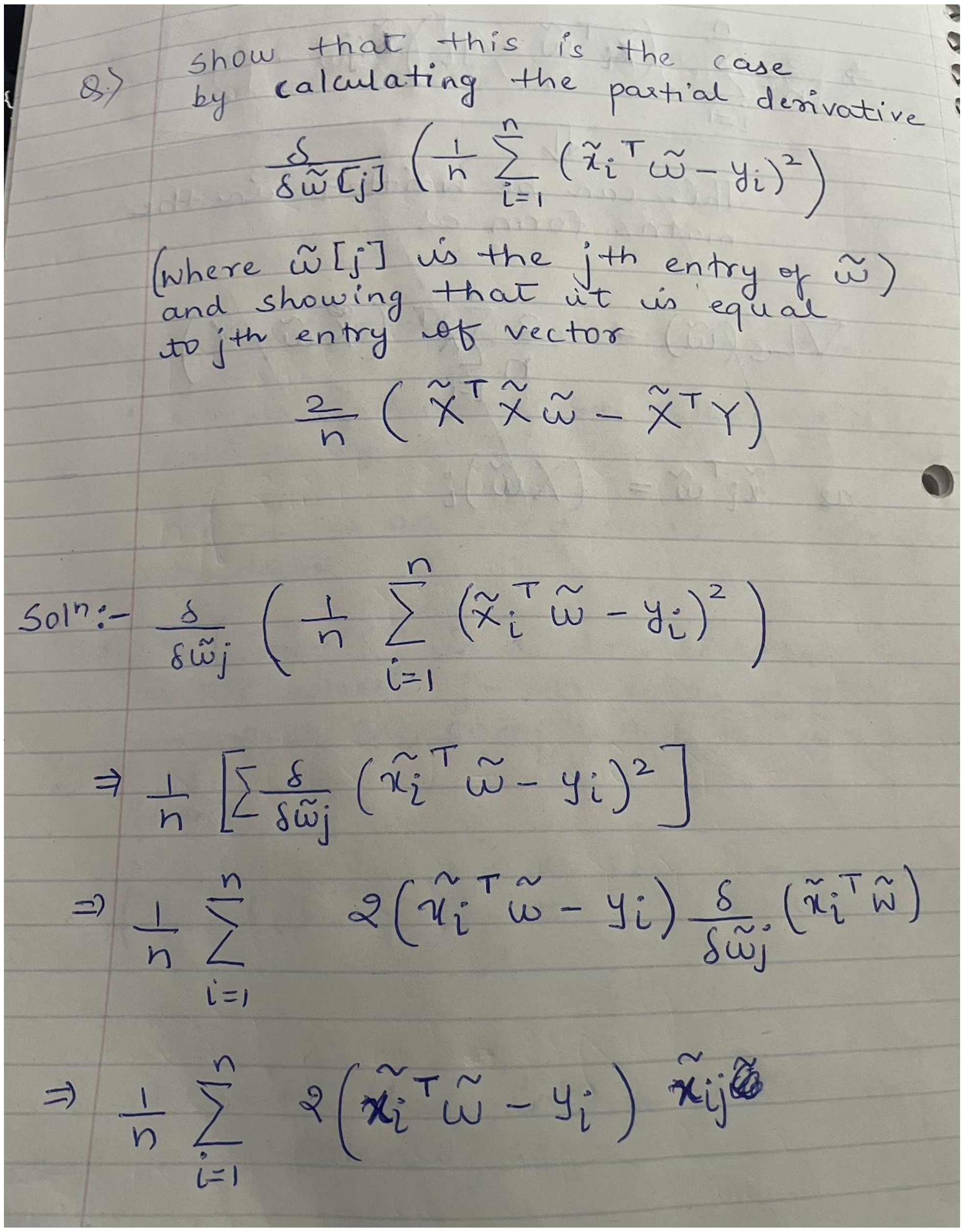
Q1> [10 pts] Show that this is the case by calculating the partial derivative



Q.) Show that this is the case by calculating the partial derivative

$$\frac{\delta}{\delta \tilde{\omega}[j]} \left(\frac{1}{n} \sum_{i=1}^{n} (\tilde{x}_i^T \tilde{\omega} - y_i)^2 \right)$$

(where $\tilde{\omega}[j]$ is the jth entry of $\tilde{\omega}$) and showing that it is equal to jth entry of vector

$$\frac{2}{n} (\tilde{X}^T \tilde{X} \tilde{\omega} - \tilde{X}^T Y)$$

Soln:- $\frac{\delta}{\delta \tilde{\omega}_j} \left(\frac{1}{n} \sum_{i=1}^{n} (\tilde{x}_i^T \tilde{\omega} - y_i)^2 \right)$

$$\Rightarrow \frac{1}{n} \left[\sum \frac{\delta}{\delta \tilde{\omega}_j} (\tilde{x}_i^T \tilde{\omega} - y_i)^2 \right]$$

$$\Rightarrow \frac{1}{n} \sum_{i=1}^{n} 2 (\tilde{x}_i^T \tilde{\omega} - y_i) \frac{\delta}{\delta \tilde{\omega}_j} (\tilde{x}_i^T \tilde{\omega})$$

$$\Rightarrow \frac{1}{n} \sum_{i=1}^{n} 2 (\tilde{x}_i^T \tilde{\omega} - y_i) \tilde{x}_{ij}$$

$$\frac{\delta L_s(\tilde{\omega})}{\delta \tilde{\omega}_j} = \frac{2}{n} \sum_{i=1}^{n} (\tilde{x}_i{}^T \tilde{\omega} - y_i) \cdot \tilde{x}_{ij}$$

$$\nabla L_s(\tilde{\omega}) = \left[\frac{\delta L_s(\tilde{\omega})}{\delta \tilde{\omega}_1}, \frac{\delta L_s(\tilde{\omega})}{\delta \tilde{\omega}_2}, \dots\dots, \frac{\delta L_s(\tilde{\omega})}{\delta \tilde{\omega}_d} \right]^T$$

We can rewrite in matrix notation

- $X\tilde{\omega}$ is n-dimensional vector where each entry is $\tilde{x}_i{}^T \tilde{\omega}$
- $X\tilde{\omega} - Y$ is an $\tilde{n}$-dimentional vector of residuals.
- $X^T(X\tilde{\omega} - Y)$ is a d-dimensional vector where j-th entry is the sum over all sample of $(\tilde{x}_i{}^T \tilde{\omega} - y_i)\tilde{x}_{ij}$

Hence,

$$\left(X^T(X\tilde{\omega} - Y)\right)_j = \sum_{i=1}^{n} (\tilde{x}_i{}^T \tilde{\omega} - y_i) \cdot \tilde{x}_{ij}$$

Combining this into a vector for all components j :-

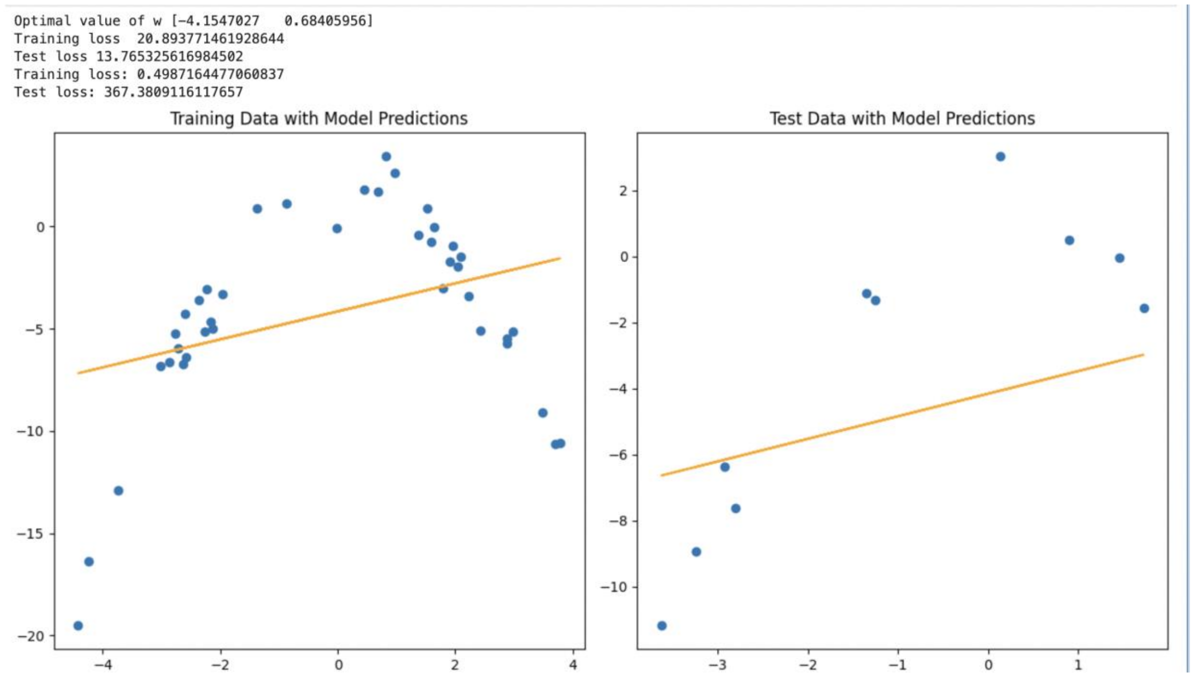
$$\boxed{\nabla L_S(\tilde{\omega}) = \frac{2}{n} X^T (X\tilde{\omega} - Y)}$$

2.1 Part A

Q.> [5 pts] Use the provided plot data and model function to plot

(a) the training data with your model's predictions overlaid

(b) the test data with your model's predictions overlaid. Describe what you see and explain why this happened.



```
Optimal value of w [-4.1547027   0.68405956]
Training loss  20.893771461928644
Test loss 13.765325616984502
Training loss: 0.4987164477060837
Test loss: 367.3809116117657
```

Training Data with Model Predictions
Test Data with Model Predictions

Observations:

1. The optimal value of W is [-4.1547027   0.68405956]. The first value is intercept and second value is slope.
2. Training loss is 20.437359511348877. This is the Mean Square Error between predicted and actual values. Lower value means the model fits the training data nicely.
3. Testing loss is 15.57977101584046. This is the MSE, and the value indicates that the model is doing well with new unseen data points.
4. Training Data Plot Observations:

- The orange line indicates the model prediction. Blue points are the training data points.
- As it is visible the model is trying to fit well for most of the points in starting and end. Hence, the line passes more closely to data points.

5. Test Data Plot Observations:
   - The orange line indicates the model prediction. Blue points are the test data points.
   - The model does not fit the points as closely as it fits the training data.

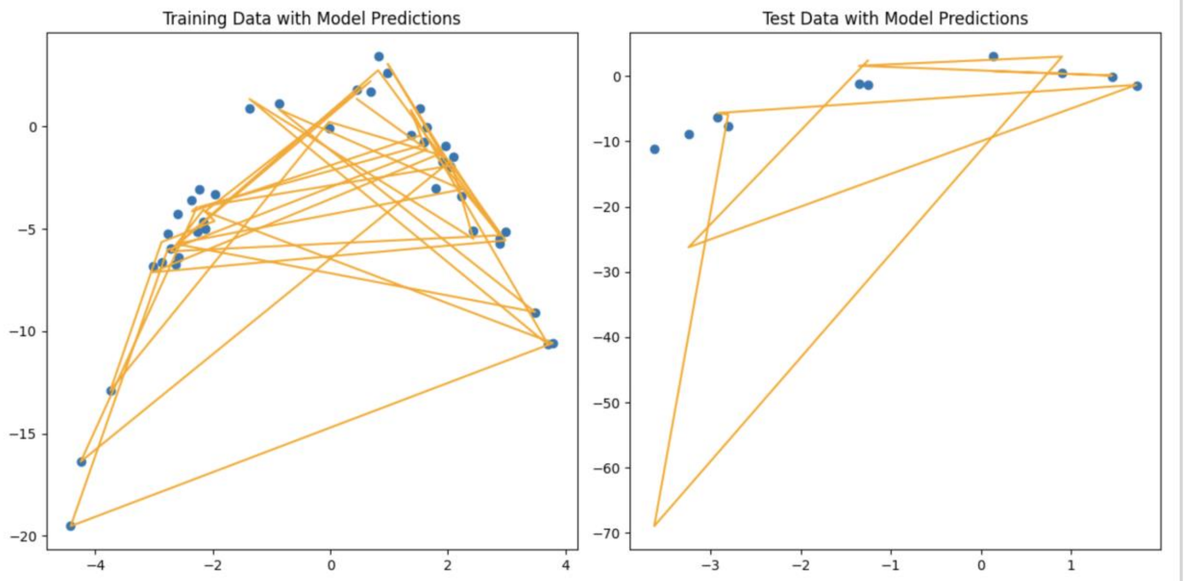
Why might this have occurred?

1. Since the test loss is lower than the training loss, it suggests that the model may be underfitting. The model is too simple (linear) to capture the underlying patterns in the data, which may require a more complex (nonlinear) model.
2. Underfitting occurs when a model is too simple to capture the underlying patterns in the data, resulting in a relatively good fit on both training and test data but not capturing the complexity of the data.

2.2 PART B:

1. [5 pts] If we learn a linear model based on the polynomial features with k = 3, describe what functions the linear model is capable of representing.
-Ans: If we learn a linear model based on polynomial features with k=3 , the model can represent any function that is a **cubic polynomial.**

2. [10 pts] Create the design matrix for degree-20 polynomial features; compute the parameters that minimize the training loss with these features; report the training and test loss; and use plot data and model to visualize the model's predictions on the train and test data. [5 pts] Describe what you see and explain why this happened.



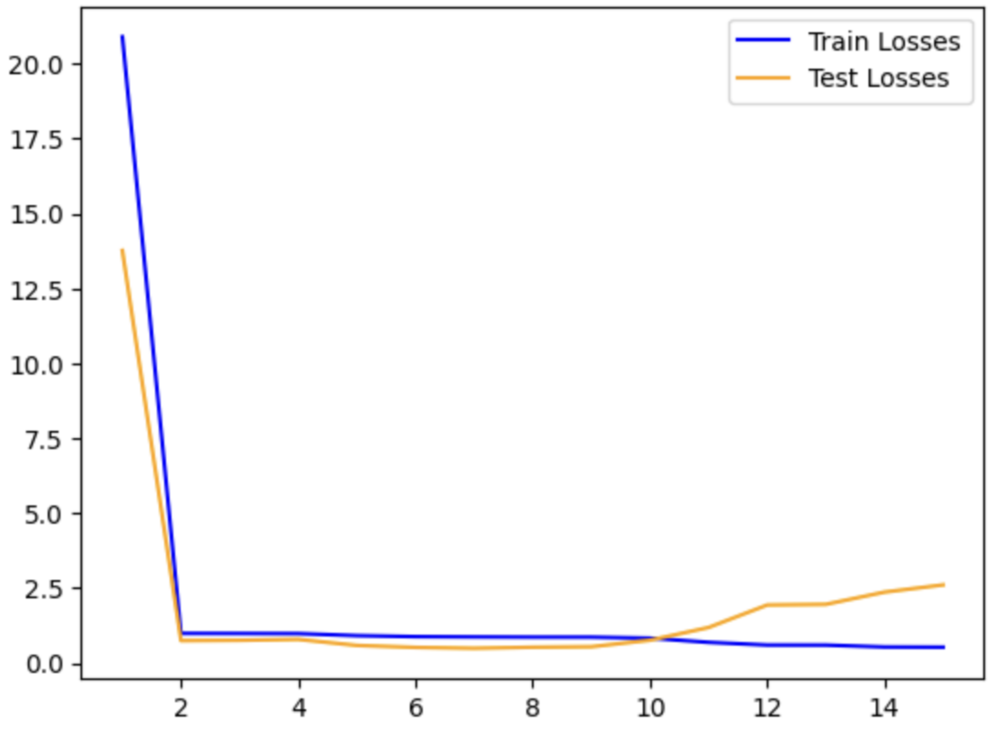
Observations:

1. For k=20, we can see that while training, the model is trying to fit each point as much as possible, but the test model is not working good.
2. There is a possibility of Overfitting where the model fits the training data very well but performs poorly on unseen test data.
3. This is also evident if the training loss is very low while the testing loss is high. In our case, Training loss for k=20 is 0.4987164477060837 and Test loss: 367.3809116117657. This clearly shows the big difference between test and training loss.

Why might this have occurred?

1. The high degree(k=20) allows the polynomial to pass through nearly all the training data points, including noise and outliers. Hence, the model becomes excessively complex, capturing not only the true underlying pattern but also random fluctuations and noise in the training data.
2. This means that while the model might achieve a very low training error, it performs poorly on new, unseen data because the noise does not generalize which has been shown in the plot.

3. Repeat part 3 for each k ∈ {1, 2, . . . , 15}in order to create two length-15 lists train losses and train losses with train losses[k-1] equal to the train loss for the model that used degree-k polynomial features (and similarly for test losses). Plot the train and test losses against k using the code provided at the bottom of hw1.py (currently commented out).
6. [5 pts] Describe what you see. Explain why increasing k has the effect that it does.



Observations:

1. Train losses (blue color): For low k values, training loss decreases very sharply and after k=2 or 3, it stabilizes and remains stable. For high k values, training loss is less.

2. Test losses (Orange color): Like the training loss, the test loss also decreases initially, indicating that the model is performing well on unseen data with simple polynomial degrees (1 to 2). After some point, the test loss starts increasing as the polynomial degree increases.

Why?

- When the degree is too low (1 or 2), the model is too simple and cannot capture the underlying pattern in the data well, resulting in both high training and test losses, resulting in underfitting.
- Somewhere between degrees 2 to 5, the test loss is minimized. This is the point where the model generalizes best.
- At higher degrees (e.g., 10 and above), the test loss increases significantly while the training loss stays low. This indicates the model is too complex and fits the training data too closely, failing to generalize to new data.
- Hence, This happens because the model becomes overly complex, fitting the training data (including noise) very well but performing poorly on unseen data. This is the sign of **overfitting** the model has low bias but high variance, meaning it has learned the training data too well and does not generalize to new data.

6. [5 pts] Having seen what you've seen, if you wanted one final linear model based on polynomial features for this regression problem, which k would you use? Explain.

Ans: I would have chosen k=2 based on my results, because at k=2, the training and test loss maintains a very low difference. Here, the model complexity is just right, balancing bias and variance. The model has enough flexibility to learn the data patterns but not so much that it overfits.