ASSIGNMENT 1

ADVANCED REGRESSION TECHNIQUES (MDTS4313)

Project: Exploring Nonlinear Relationships Using Smoothing Methods

Course: Advanced Regression Techniques (MDTS4313)

Department M.Sc. Data Science

Semester: 3

Name: Sohini Mandal

Roll Number: 444

Dataset description and justification:

Dataset name: Fish Market

Dataset link: https://www.kaggle.com/datasets/vipullrathod/fish-market

Dataset description:

The fish market dataset is a collection of data related to various species of fish and their characteristics. This dataset is designed for polynomial regression analysis and contains several columns with specific information. Here's a description of each column in the dataset:

Species: This column represents the species of the fish. It is a categorical variable that categorizes each fish into one of seven species. The species may include names like "Perch," "Bream," "Roach," "Pike," "Smelt," "Parkki," and "Whitefish." This column is the target variable for the polynomial regression analysis, where we aim to predict the fish's weight based on its other attributes.

Weight: This column represents the weight of the fish. It is a numerical variable that is typically measured in grams. The weight is the dependent variable we want to predict using polynomial regression.

Length1: This column represents the first measurement of the fish's length. It is a numerical variable, typically measured in centimetres.

Length2: This column represents the second measurement of the fish's length. It is another numerical variable, typically measured in centimetres.

Length3: This column represents the third measurement of the fish's length. Similar to the previous two columns, it is a numerical variable, usually measured in centimetres.

Height: This column represents the height of the fish. It is a numerical variable, typically measured in centimetres.

Width: This column represents the width of the fish. Like the other numerical variables, it is also typically measured in centimetres.

The dataset is structured in such a way that each row corresponds to a single fish with its species and various physical measurements (lengths, height, and width). The goal of using polynomial regression on this dataset would be to build a predictive model that can estimate the weight of a fish based on its species and the provided physical measurements. Polynomial regression allows for modelling more complex relationships between the independent variables (lengths, height, and width) and the dependent variable (weight), which may be particularly useful if there are non-linear patterns in the data.

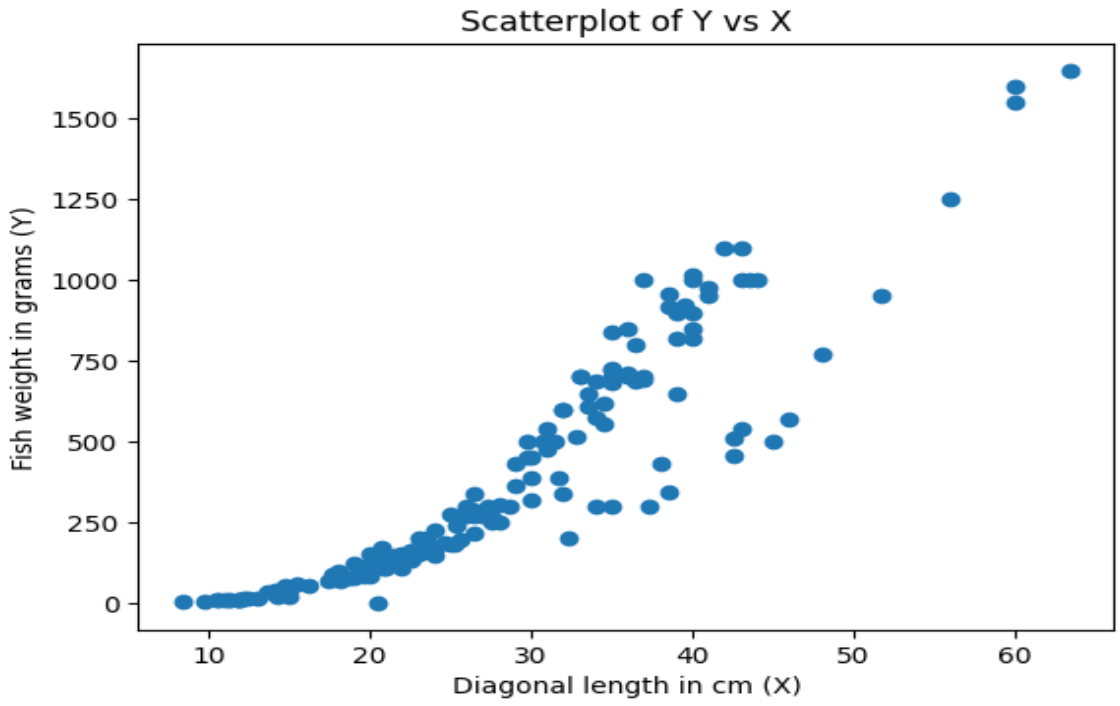
Definitions of X and y:

- **Independent Variable (X):**

  Length2: Diagonal length of fish in cm. It is a continuous variable.

- **Dependent Variable (Y):**

  Weight: Weight of fish in grams. It is also continuous.



Why did you choose this dataset?

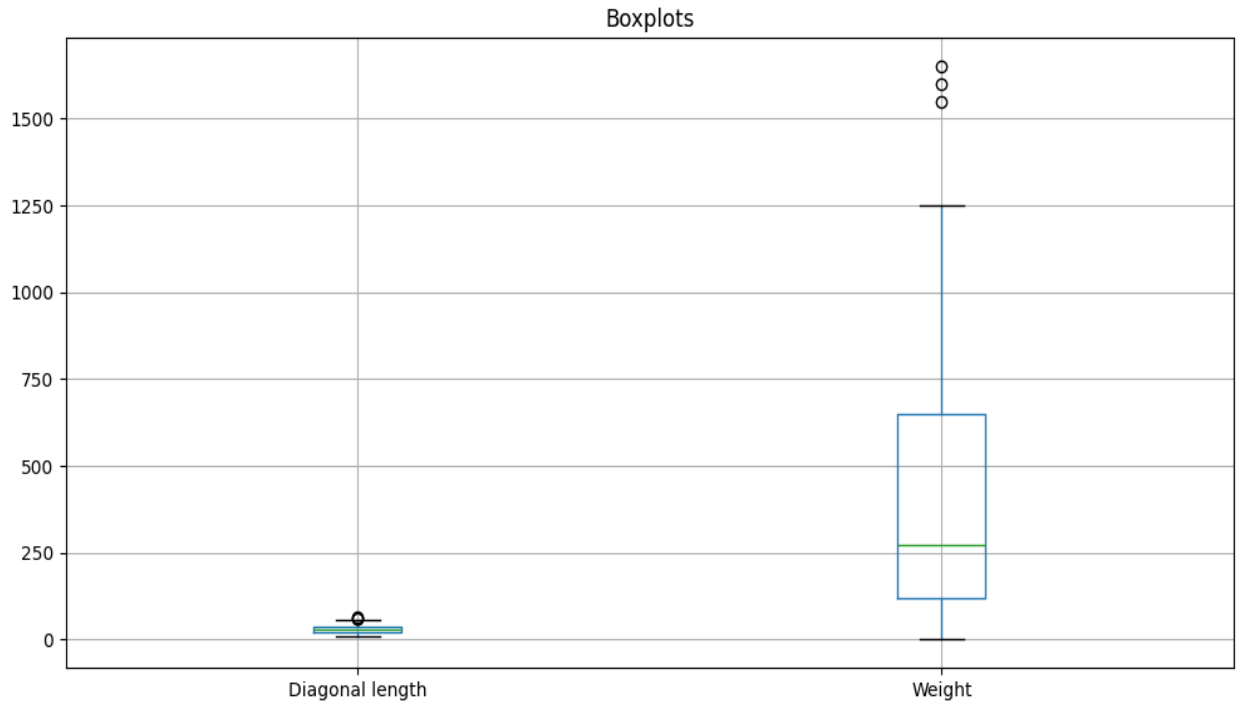
I chose the Fish Market Dataset because it provides clean, real-world continuous variables that describe measurable physical traits of fish. The dataset is small but rich, allowing clear visualization of nonlinear growth patterns between fish dimensions and weight. Moreover, it's publicly available and widely used for demonstrating polynomial and nonlinear regression models.

Why do you expect the relationship between X and Y to be nonlinear?

The relationship between fish length and weight is inherently nonlinear because as a fish grows, its body mass does not increase proportionally to its length. Instead, weight tends to grow with the cube of length (approximately following W∝L^3), since volume—and hence mass—increase more rapidly than linear dimensions. Thus, the data points form a curved pattern instead of a straight line.

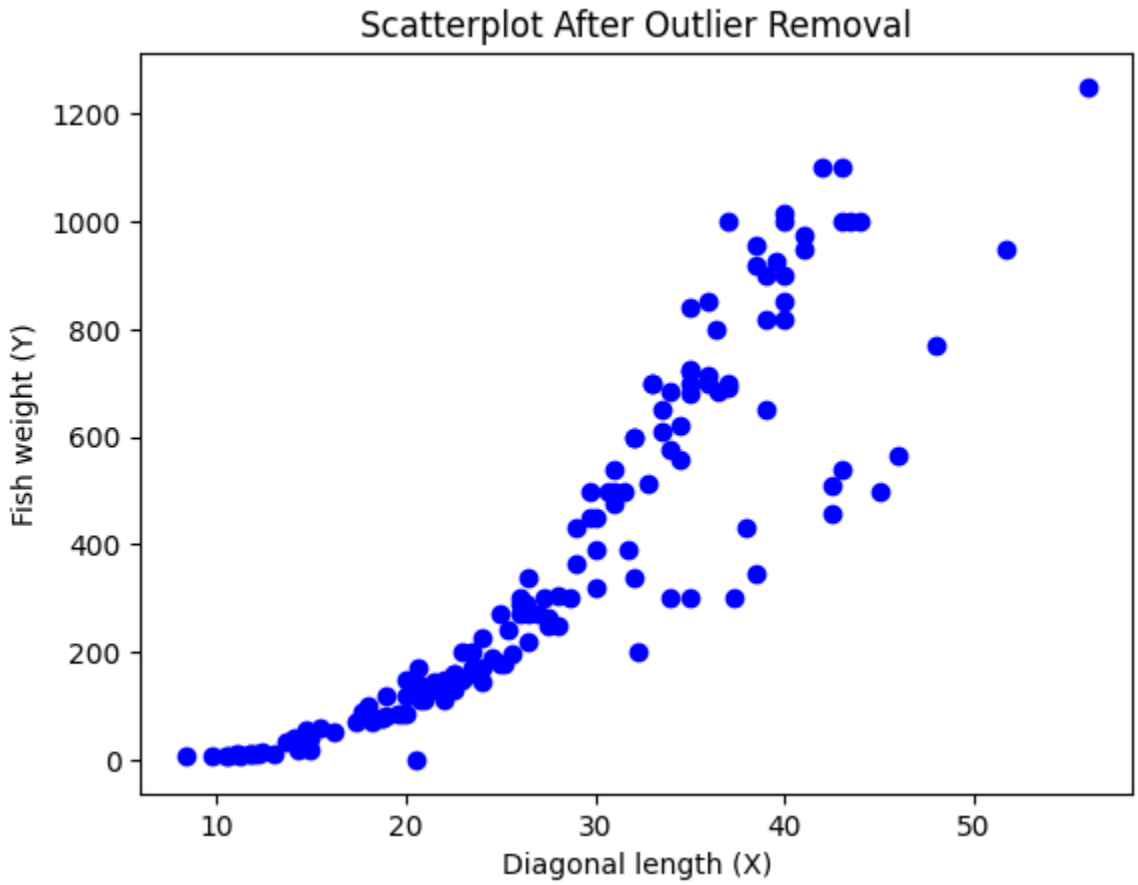
What type of nonlinear relationship do you suspect (e.g., quadratic, exponential, saturating)?

I suspect a quadratic relationship, where fish weight increases with the square (or higher-order polynomial) of length. This is evident from the upward curvature in the scatterplot of Weight vs. Length2. The rate of increase in weight accelerates as length increases, forming a parabolic trend typical of quadratic growth.



The boxplot showed a few high-value points in Weight, which lie far above the upper whisker.

These are potential outliers representing unusually high weighted fish, which may disproportionately influence regression fitting.



How many missing values were found and removed?

There were no missing values.

What criteria did you use to detect outliers?

I used the Interquartile Range (IQR) method to detect outliers.

Criterion Explanation:

The IQR method identifies outliers based on the spread of the middle 50% of the data

IQR=Q3−Q1

where

Q1 = 25th percentile,

Q3 = 75th percentile.

An observation is considered an outlier if it lies beyond 1.5 times the IQR below the first quartile or above the third quartile:

Lower bound=Q1−1.5×IQR

Upper bound=Q3+1.5×IQR

Any data point:

X< Lower bound or X> Upper bound is treated as an outlier.

Why this criterion was chosen:

- **Non-parametric method** — does not assume normality, suitable for skewed or nonlinear data like the Fish Market dataset.

- **Robust to extreme values —** uses medians and quartiles instead of means, which are sensitive to outliers.

- **Visual alignment** — aligns with boxplot visualization, where whiskers represent the same 1.5×IQR threshold.

After cleaning, how would you describe the visual pattern between X and Y?

After removing the outliers, the scatterplot between Diagonal Length (X) and Fish Weight (Y) shows a smooth, upward curving pattern. The relationship is nonlinear — as the diagonal length increases, the fish weight increases at an accelerating rate.

This curvature suggests a quadratic or cubic relationship, since larger fish gain weight more rapidly with slight increases in body length. The removal of extreme points made the pattern more continuous and clearly visible, confirming that the underlying growth pattern is systematic and nonlinear, not random.

Do you think a parametric regression would fit this data? Why or why not?

A simple linear parametric regression would not fit this data well, because the relationship between fish length and weight is nonlinear. As fish grow, their weight increases roughly with the cube of their length ($W \propto L^3$), meaning the rate of weight gain accelerates as length increases. This causes the data points to form a curved, upward-bending pattern rather than a straight line.

However, a nonlinear parametric model, such as a polynomial regression (e.g., quadratic or cubic), could fit the data effectively. Such models still assume a specific functional form but can capture curvature by including higher-order terms of the predictor.

You may choose k = 5, 8, or 10 folds for cross-validation. Justify your choice of number of folds — consider data size, computation cost, and variance-bias tradeoff.

For this study, **10-fold cross-validation** was selected. The training dataset contains **124 observations**, which is relatively small. Using 10 folds allows approximately **90% of the data** to be used for training in each iteration, ensuring that the model learns effectively from as much data as possible while maintaining a sufficient test portion for reliable error estimation.

From the **bias–variance tradeoff** perspective, increasing the number of folds reduces bias since each model is trained on a larger subset of the data. Although variance may increase slightly due to smaller test folds, the effect is minimal for this sample size, as each test fold still contains about 12–13 observations. The reduction in bias obtained with 10 folds outweighs the minor increase in variance.

In terms of **computational cost**, running 10 folds instead of 5 or 8 is only marginally more expensive. Given the modest dataset size, the additional computation is negligible.

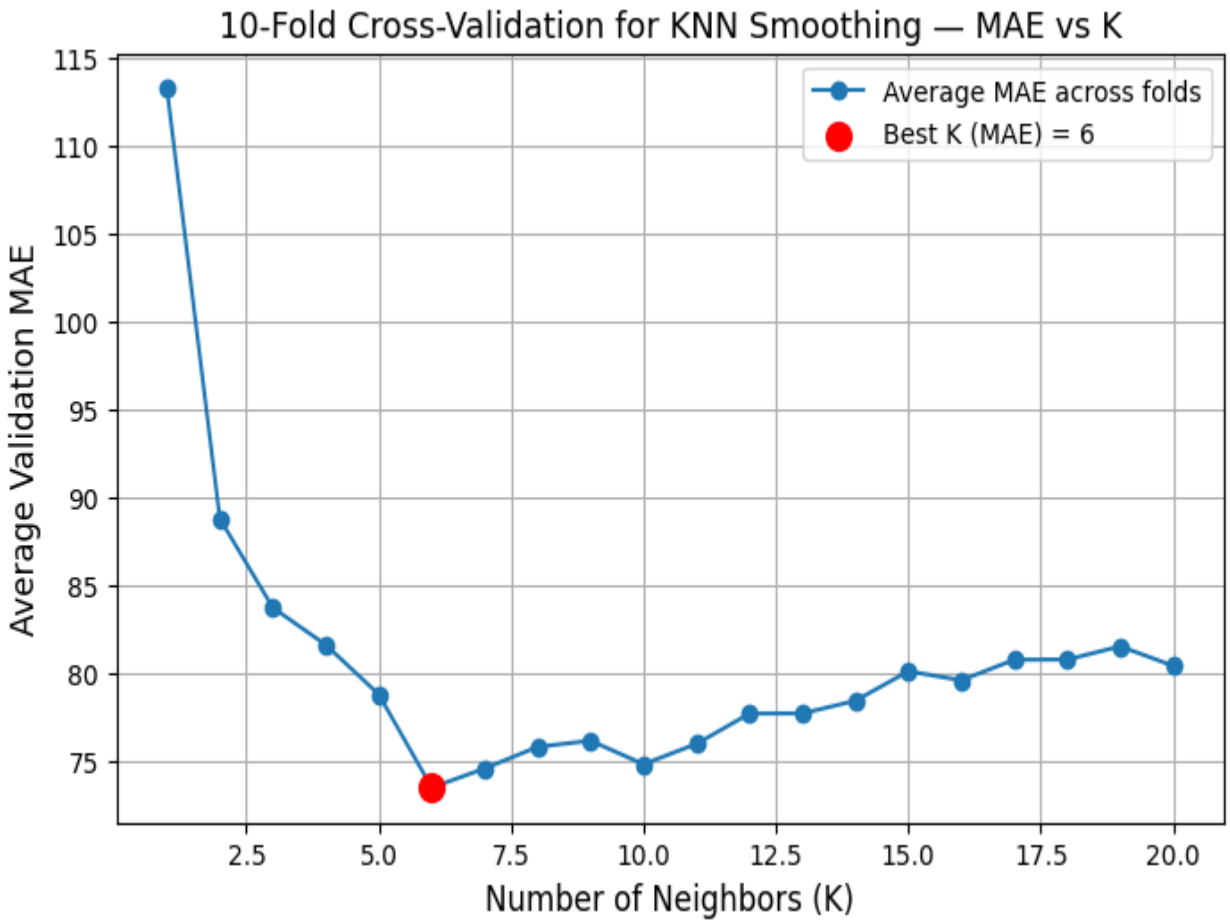
Overall, **10-fold cross-validation** provides a well-balanced approach, offering **low bias, acceptable variance, and efficient computation**, making it the most appropriate choice for this analysis.

KNN Smoother:

Hyperparameters and corresponding cross-validation MAE:

k	Average MAE
1	113.244615
2	88.793173
3	83.741090
4	81.627292
5	78.814538
6	73.515865
7	74.602564
8	75.830986
9	76.180848

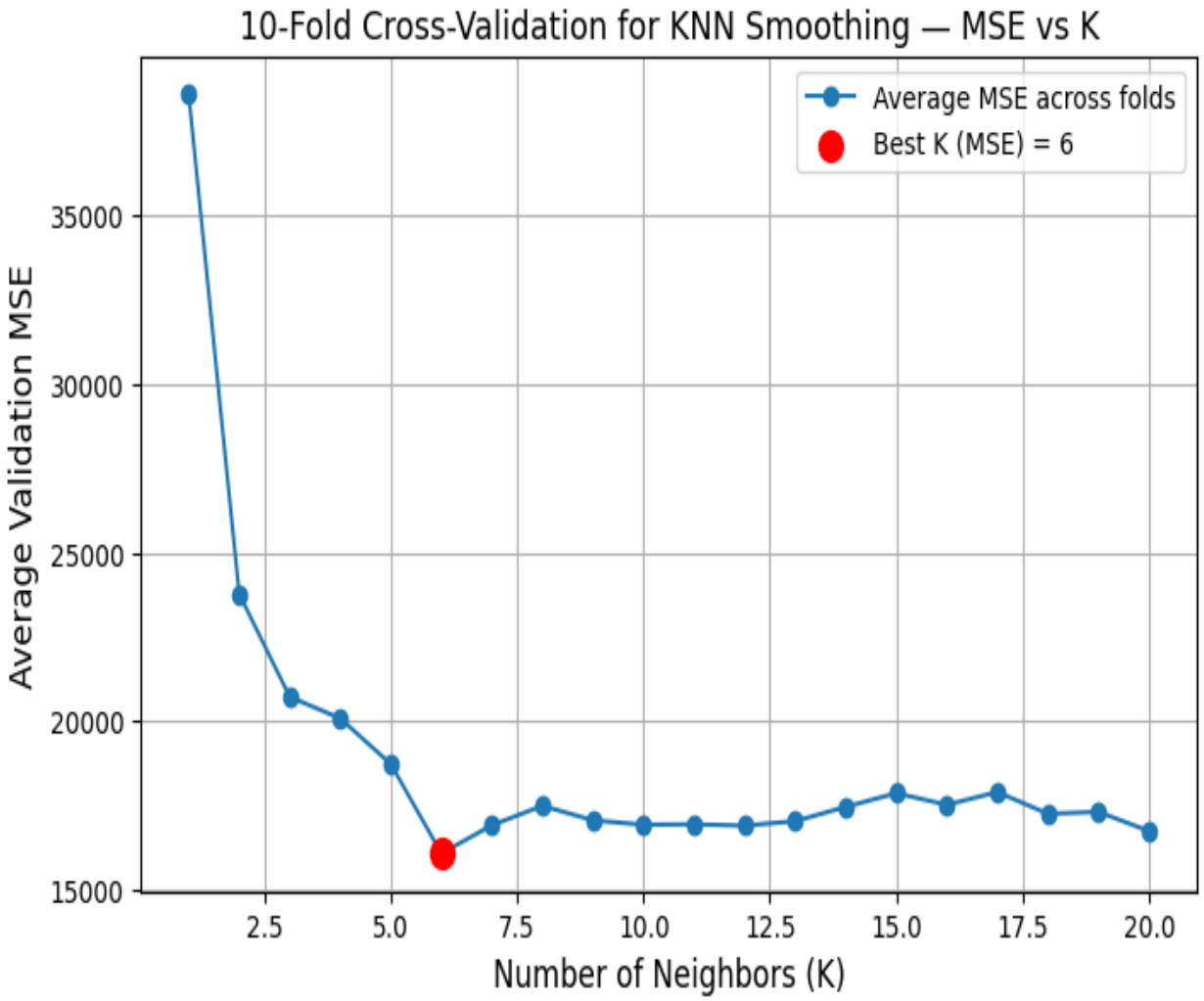
10	74.844923
11	76.002214
12	77.722436
13	77.731381
14	78.450943
15	80.126684
16	79.612736
17	80.781897
18	80.793162
19	81.541312
20	80.435811



Best k= 6

Hyperparameters and corresponding cross-validation MSE:

k	Average MSE
1	38608.135628
2	23776.185627
3	20725.324613
4	20085.564218
5	18731.591766
6	16080.546838
7	16918.546197
8	17493.320601
9	17062.042671
10	16932.155476
11	16944.986069
12	16904.645953
13	17037.890031
14	17461.232693
15	17867.774308
16	17522.902037
17	17901.615631
18	17257.137676
19	17312.432821
20	16744.839959



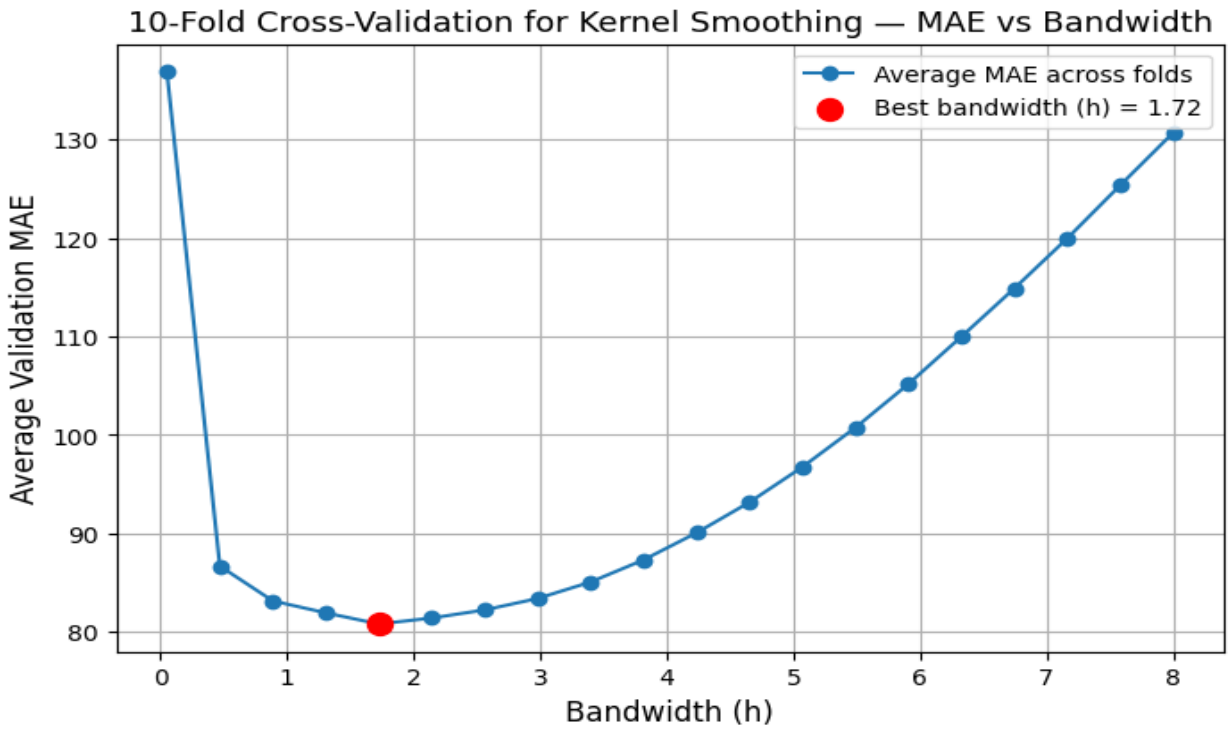
Best k= 6

Final Test MSE= 15396.135711805558

Kernel Smoother:

Hyperparameters and corresponding cross-validation MAE:

h	Average MAE
0.050000	136.898179
0.468421	86.676528
0.886842	83.137680
1.305263	81.925679
1.723684	80.793968
2.142105	81.400575
2.560526	82.220939
2.978947	83.385796
3.397368	85.058552
3.815789	87.303782
4.234211	90.052832
4.652632	93.161982
5.071053	96.744405
5.489474	100.745370
5.907895	105.169349
6.326316	109.980218
6.744737	114.881532
7.163158	119.968054
7.581579	125.362502
8.000000	130.702201

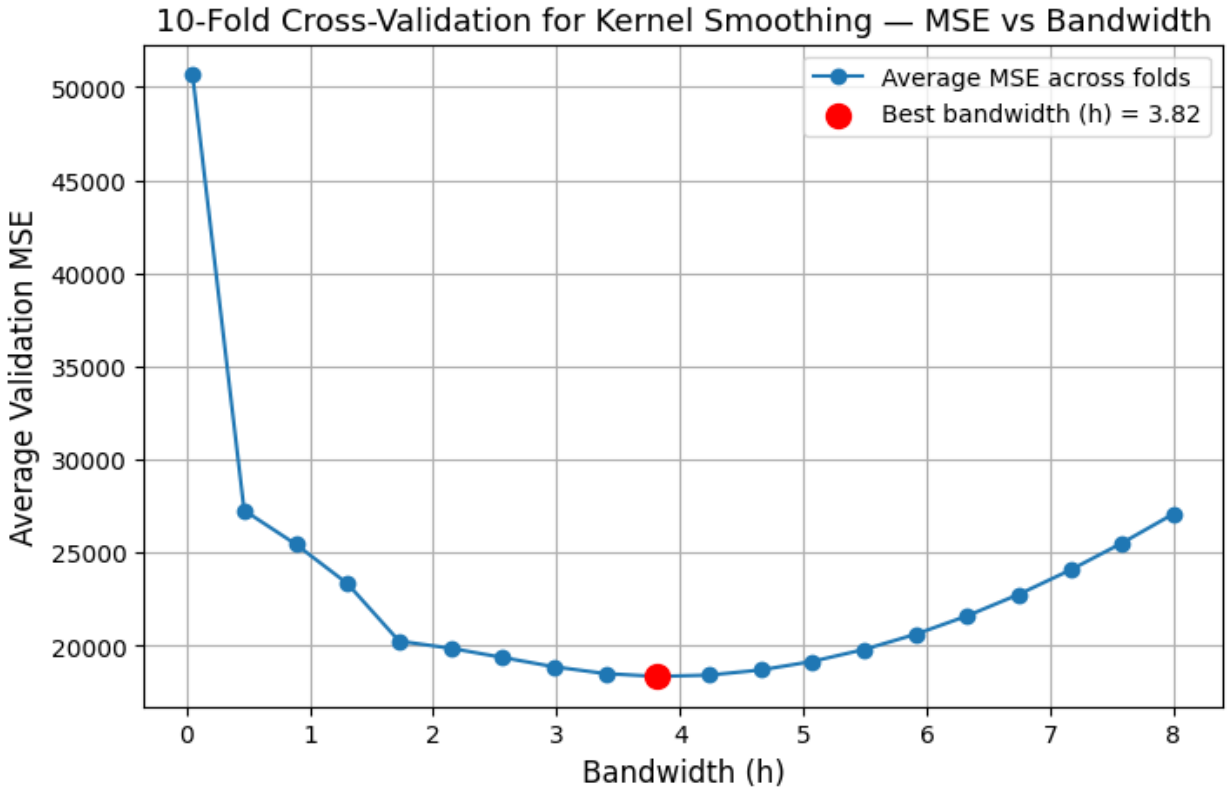


Best h= 1.723684210526316

Hyperparameters and corresponding cross-validation MSE:

h	Average MSE
0.050000	50665.493103
0.468421	27270.639113
0.886842	25453.908360
1.305263	23340.940825
1.723684	20252.573776
2.142105	19869.653207
2.560526	19377.566533
2.978947	18875.591680
3.397368	18507.678417
3.815789	18357.236018

4.234211	18428.791961
4.652632	18701.499165
5.071053	19160.624987
5.489474	19800.029587
5.907895	20616.670453
6.326316	21606.171098
6.744737	22760.828785
7.163158	24069.359158
7.581579	25517.516550
8.000000	27089.005004



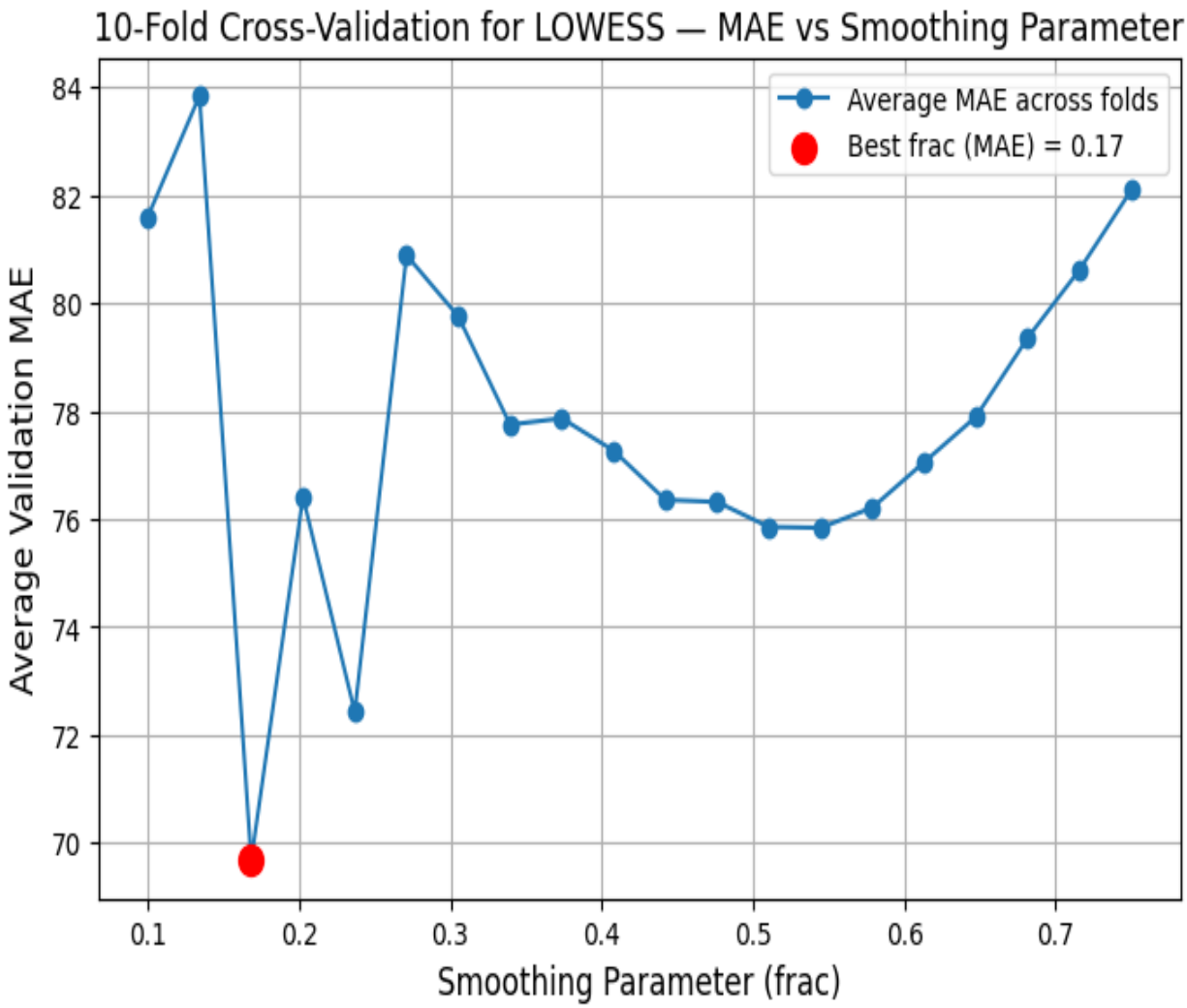
Best h= 3.8157894736842106

Final Test MSE= 13510.36271209925

LOWESS:

Hyperparameters and corresponding cross-validation MAE:

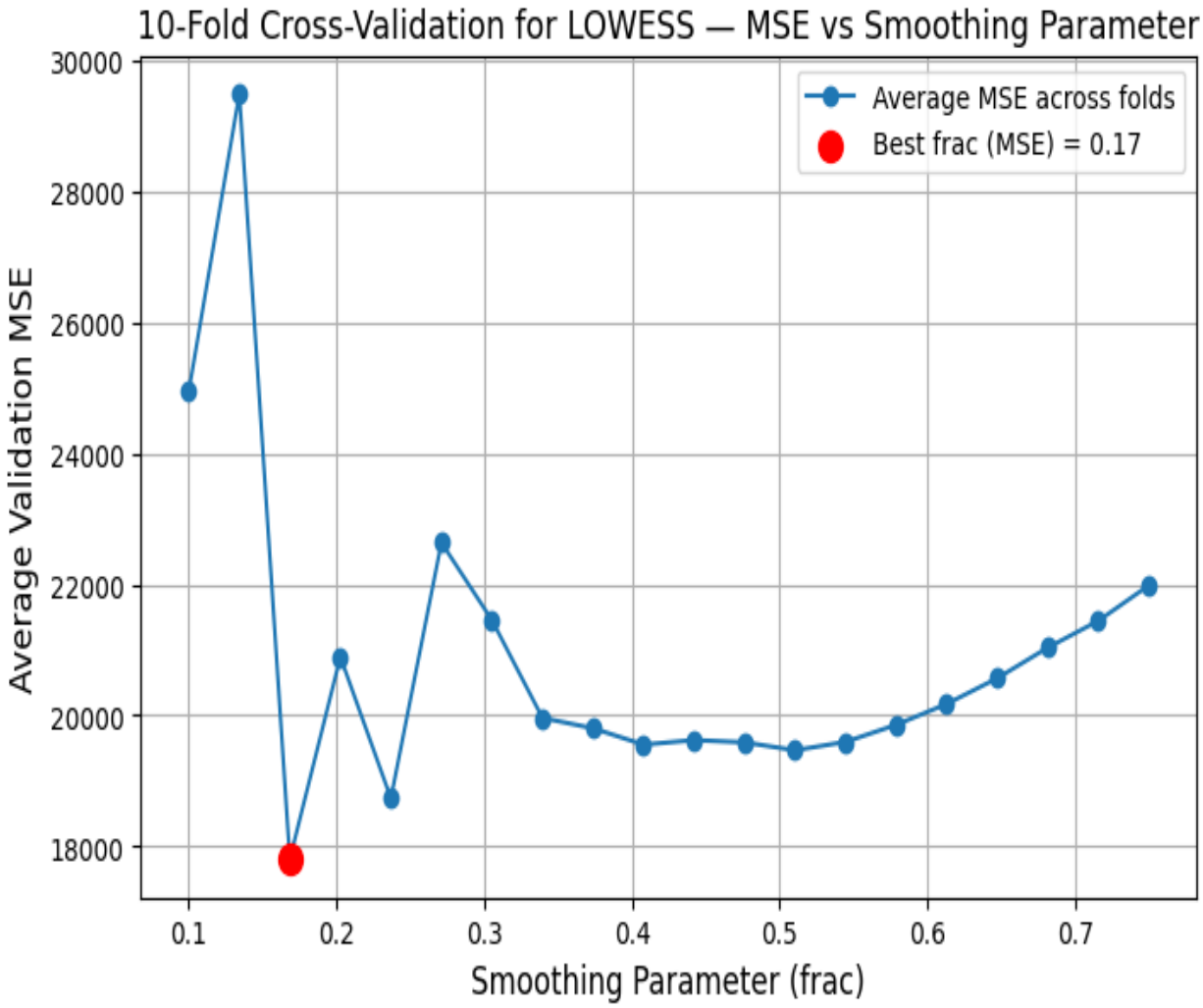
frac	Average MAE
0.100000	81.596268
0.134211	83.841850
0.168421	69.681222
0.202632	76.413172
0.236842	72.423883
0.271053	80.889251
0.305263	79.757097
0.339474	77.750763
0.373684	77.867793
0.407895	77.269857
0.442105	76.358075
0.476316	76.318784
0.510526	75.855224
0.544737	75.840032
0.578947	76.214321
0.613158	77.055452
0.647368	77.895780
0.681579	79.359678
0.715789	80.618083
0.750000	82.094265



Best frac= 0.16842105263157897

Hyperparameters and corresponding cross-validation MSE:

frac	Average MSE
0.100000	24941.526229
0.134211	29479.025621
0.168421	17808.045998
0.202632	20894.851947
0.236842	18735.959437
0.271053	22636.011238
0.305263	21454.050411
0.339474	19962.277984
0.373684	19804.881874
0.407895	19554.766919
0.442105	19624.002490
0.476316	19589.517746
0.510526	19469.166097
0.544737	19597.223745
0.578947	19856.704245
0.613158	20175.768675
0.647368	20567.376125
0.681579	21034.388673
0.715789	21442.359158
0.750000	21982.900967



Best frac= 0.16842105263157897

Final Test MSE = 21008.72276618028

What hyperparameter(s) did you tune for each smoother?

In this study, different nonparametric smoothers were implemented — namely K-Nearest Neighbors (KNN) smoothing, Kernel Regression, and LOWESS (Locally Weighted Scatterplot Smoothing).

Each method involves a key hyperparameter that controls the degree of smoothness and hence affects the bias–variance trade-off.

These hyperparameters were tuned using 10-fold cross-validation to minimize the validation error (MSE or MAE).

1. KNN Smoothing

- Tuned hyperparameter: Number of neighbors (K)

- Description:

The parameter `K` determines how many nearby observations are averaged to estimate the smoothed value at each point.

Smaller `K` values allow the smoother to follow the data more closely, leading to low bias but high variance.

Larger `K` values result in smoother fits with higher bias but lower variance.

The optimal `K` was selected based on the minimum cross-validated error.

2. Kernel Regression

- Tuned hyperparameter: Bandwidth (h)

- Description:

The bandwidth `h` controls the width of the kernel window, determining the range of data points that influence each prediction.

A small `h` leads to highly flexible fits that capture local fluctuations (low bias, high variance),

while a large `h` yields smoother, more stable curves (high bias, low variance).

The Gaussian kernel was used in this study, and the optimal bandwidth was identified via cross-validation.

3. LOWESS

- Tuned hyperparameter: Fraction of data used in each local regression (frac)

- Description:

The parameter `frac` specifies the proportion of data points used to fit each local regression.

Smaller values of `frac` produce highly local fits (low bias, high variance),

whereas larger values result in smoother trends (high bias, low variance).

The best fraction was chosen using cross-validation to achieve the lowest validation error.

Summary:

Smoother	Hyperparameter Tuned	Effect on Smoothness	Bias–Variance Behavior
KNN Smoothing	K: number of neighbors	Controls local averaging	↑K → ↑Bias, ↓Variance
Kernel Regression	h: bandwidth	Controls kernel window width	↑h → ↑Bias, ↓Variance
LOWESS	frac: fraction of data	Controls local window size	↑frac → ↑Bias, ↓Variance

How did the validation error change across the hyperparameter range?

Across all three smoothers, the validation errors (MSE and MAE) showed a U-shaped relationship with respect to the hyperparameter.

Too small a hyperparameter led to overfitting (high variance), while too large a value caused underfitting (high bias).

The minimum point of the curve represented the optimal trade-off between bias and variance.

What number of folds did you use in cross-validation, and why?

We used a 10-fold validation approach for our experiments. With only 124 samples, each fold allows ~90% of the data to be used for training and the remaining ~10% as a validation set, helping the model learn effectively. Increasing the number of folds reduces bias because each model sees more training data, while the slight increase in variance from smaller validation sets (~12–13 samples per fold) is minimal. Overall, 10-fold validation provides a good balance between reliable error estimation and efficient model training.

Which error metric(s) did you use, and what are the pros and cons of your choice?

We evaluated model performance using mean absolute error (MAE) and mean squared error (MSE) on the validation sets.

- MAE (Mean Absolute Error)

- Measures the average absolute difference between predicted and true values.

- **Pros:** Less sensitive to outliers; easy to interpret in the same units as the target.

- **Cons:** Does not penalize large errors as strongly as MSE.

- MSE (Mean Squared Error)

- Measures the average squared difference between predicted and true values.

- **Pros:** Penalizes larger errors more heavily, useful if large deviations are undesirable.

- **Cons:** Sensitive to outliers; values are in squared units, making interpretation less direct.

Using both metrics together provides a more complete picture: MAE reflects typical prediction error, while MSE highlights the impact of large deviations.

Which smoother performed best on the test data, and why might that be?

Smoother Type	Test MSE
LOWESS	21008.72
Kernel Smoother	13510.36
KNN Smoother	15396.136

From the results, Gaussian Kernel Regression performed best on the test data, as it achieved the lowest test MSE (13,510.362) compared to KNN and LOWESS.

This suggests that the Gaussian kernel smoother achieved a better bias–variance trade-off. While KNN and LOWESS either overfit local noise or oversmoothed the trend, the kernel regression with an appropriately tuned bandwidth provided enough flexibility to capture the underlying relationship without fitting random fluctuations. Its smooth weighting mechanism allows it to generalize better, resulting in improved predictive performance on unseen data.

Summarize the results for all smoothing methods in a comparison table that includes:

o Method name

o Best hyperparameter value

o Validation Error

o Test Error

Comparison Table

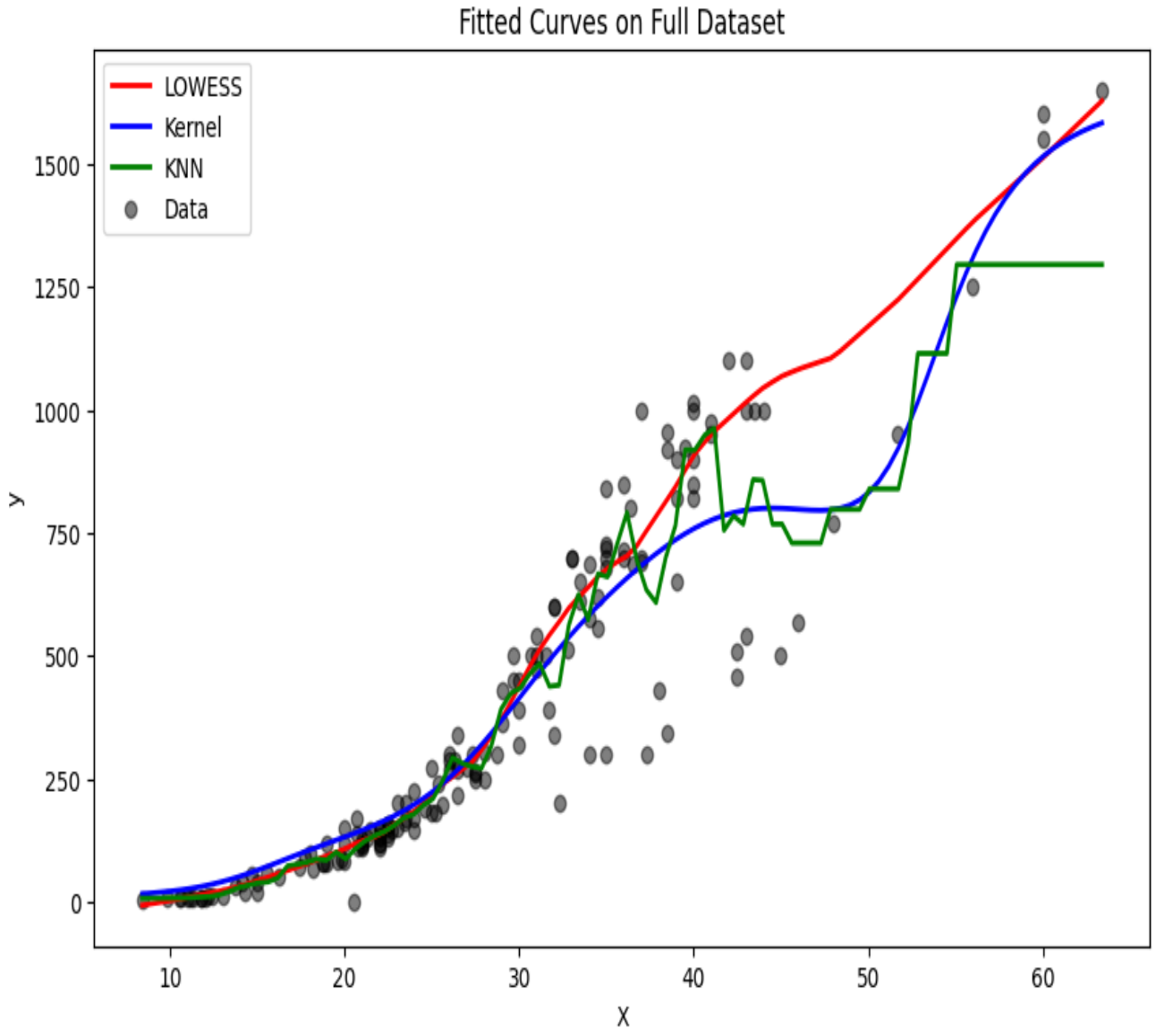
The table below summarizes the best hyperparameter, validation, and test errors for each smoothing method.

Method	Best Hyperparameter (MAE)	Validation MAE	Best Hyperparameter (MSE)	Validation MSE	Test MSE
K-Nearest Neighbors (KNN)	k = 6	73.516	k = 6	16080.547	15396.136
Gaussian Kernel Regression	bandwidth = 1.724	80.794	bandwidth = 3.816	18357.236	13510.362
LOWESS	frac = 0.168	69.681	frac = 0.168	17808.046	21008.723

Visualize the fitted curves from all best models over the same scatterplot for comparison.

The fitted curves from all three best models were plotted together on a single scatterplot for comparison.

- The **LOWESS** curve (red) is **the smoothest visually**, producing a continuous and flexible line that follows the general nonlinear trend well.
- The **KNN** smoother (green) produces blocky, step-like segments because it averages over local neighborhoods, resulting in less continuity.
- The **Gaussian Kernel Regression** (blue) also provides a smooth fit, though slightly stiffer than LOWESS, and captures the overall shape effectively without excessive local oscillation.



Discuss which method fits your dataset best, and whether the result aligns with your expectations.

Among all smoothing methods, the **Gaussian Kernel Regression** achieved the lowest test error (MSE = 13,510.362), indicating that it generalizes best to unseen data. This method maintains an effective balance between bias and variance, producing a smooth fit that avoids both underfitting and overfitting.

The **LOWESS** smoother produced the most visually smooth curve, adapting well to local variations in the data. However, its test error (MSE = 21,008.723) was higher, suggesting slight overfitting in regions where observations are dense or noisy.

The **K-Nearest Neighbors (KNN)** smoother, with $k = 6$, performed moderately well (Test MSE = 15,396.136) but exhibited localized overfitting. Smaller neighborhood sizes made the fitted line more sensitive to random fluctuations, resulting in a jagged and less continuous appearance compared to the other smoothers.

Overall, the **Gaussian Kernel Regression** provided the best predictive accuracy, while **LOWESS** offered the smoothest and most visually natural representation of the data. This outcome aligns with theoretical expectations: **KNN** tends to have low bias but higher variance for small k values, **LOWESS** is flexible and smooth but can overfit locally, and the **Gaussian Kernel** method achieves a desirable compromise by producing accurate and stable results.

Which method achieved the lowest test error?

The Kernel Regression with Gaussian Kernel method achieved the lowest test error, with a test MSE of 13,510.362. This indicates that it generalized best to unseen data compared to KNN and LOWESS.

Was the best-performing model also the smoothest visually?

No, the best-performing model (Gaussian Kernel Regression) was not the smoothest visually. While it achieved the lowest test MSE, its curve (in blue) still shows moderate fluctuations, especially in regions with sparse data. The LOWESS curve (in red) appears much smoother and follows a more gradual trend across the range of x-values.

This indicates that the model with the lowest numerical test error does not necessarily provide the smoothest visual fit. Gaussian Kernel Regression captured more local variations that helped it reduce test error, whereas LOWESS produced a smoother but slightly less accurate fit overall.

In what situations could another smoother outperform this one?

In some situations, another smoother such as KNN or LOWESS could outperform the Gaussian Kernel Regression. If the true relationship in the data has local or abrupt changes, KNN may adapt better since it

relies on local neighborhoods rather than global smoothing. Similarly, if the data are non-uniformly distributed or contain clusters, KNN can provide more flexible local fits. LOWESS could also outperform when the data exhibit heteroscedasticity or when adaptive smoothing is needed, as it adjusts the amount of smoothing across regions of the dataset. In these cases, methods like KNN or LOWESS can capture patterns that a globally smoothed Gaussian kernel might miss.

What does this project teach you about the importance of hyperparameter tuning in nonparametric regression?

This project highlights that hyperparameter tuning is essential in nonparametric regression because the performance and smoothness of models like KNN, LOWESS, and Gaussian Kernel Regression depend heavily on the chosen parameters. Hyperparameters such as the number of neighbors in KNN, the smoothing fraction in LOWESS, and the bandwidth in kernel regression control the model's flexibility. Poorly chosen values can lead to underfitting or overfitting—either oversmoothing important patterns or capturing random noise. Through systematic tuning, such as cross-validation, we can identify the balance that minimizes error while preserving the true data structure. This demonstrates that proper hyperparameter selection is key to achieving accurate and generalizable nonparametric models.

Conclusion:

Among all three smoothers, Gaussian Kernel Regression achieved the lowest test MSE (13,510.36), indicating the best predictive performance on unseen data.

This suggests that the chosen bandwidth effectively balanced bias and variance, providing an accurate yet flexible fit.

While the LOWESS smoother offered a smoother and visually appealing fit, it slightly underperformed in predictive accuracy, possibly due to over-smoothing.

The KNN smoother, although simpler, performed reasonably well but was slightly less smooth compared to the kernel regression curve.

Overall, this project highlights that different smoothers may perform better under different data structures — and hyperparameter tuning (choosing k, frac, or bandwidth) is crucial to achieving the right trade-off between smoothness and model accuracy.