Assignment -1

Exploring Nonlinear Relationships Using Smoothing Methods

Debjit Khaskel

M.Sc. Data Science (Sem: 3) Roll Number: 438

	vendor name	Model Name	MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX	PRP	ERP
0	adviser	32/60	125	256	6000	256	16	128	198	199
1	amdahl	470v/7	29	8000	32000	32	8	32	269	253
2	amdahl	470v/7a	29	8000	32000	32	8	32	220	253
3	amdahl	470v/7b	29	8000	32000	32	8	32	172	253
4	amdahl	470v/7c	29	8000	16000	32	8	16	132	132
...	...	...	...	...	...	...	...	...	...	...
204	sperry	80/8	124	1000	8000	0	1	8	42	37
205	sperry	90/80-model-3	98	1000	8000	32	2	8	46	50
206	sratus	32	125	2000	8000	0	2	14	52	41
207	wang	vs-100	480	512	8000	32	0	0	67	47
208	wang	vs-90	480	1000	4000	0	0	0	45	25

209 rows × 10 columns

Link: https://www.kaggle.com/datasets/abdelazizsami/computer-hardware

Dataset considered here is the "Computer Hardware

Performance Metrics and Specifications of Historical Computer Hardware" from kaggle.

Dataset Overview

Characteristics:

* Number of Instances: 209
* Number of Features: 10
* Missing Values: No

Description:
This dataset includes information about computer hardware with a focus on performance metrics.

Variables:

VendorName

  Role: Feature
  Type: Categorical
  Description: Names of the vendors (e.g., adviser, amdahl, apollo, etc.)
  Missing Values: No

ModelName

  Role: Feature
  Type: Categorical
  Description: Unique symbols for model names
  Missing Values: No

MYCT

  Role: Feature
  Type: Integer
  Description: Machine cycle time (in nanoseconds)
  Missing Values: No

MMIN

Role: Feature

Type: Integer

Description: Minimum main memory (in kilobytes)

Missing Values: No

MMAX

Role: Feature

Type: Integer

Description: Maximum main memory (in kilobytes)

Missing Values: No

CACH

Role: Feature

Type: Integer

Description: Cache memory (in kilobytes)

Missing Values: No

CHMIN

Role: Feature

Type: Integer

Description: Minimum channels (in units)

Missing Values: No

CHMAX

Role: Feature

Type: Integer

Description: Maximum channels (in units)

Missing Values: No

PRP

Role: Feature

Type: Integer

Description: Published relative performance

Missing Values: No

ERP

Role: Feature

Type: Integer

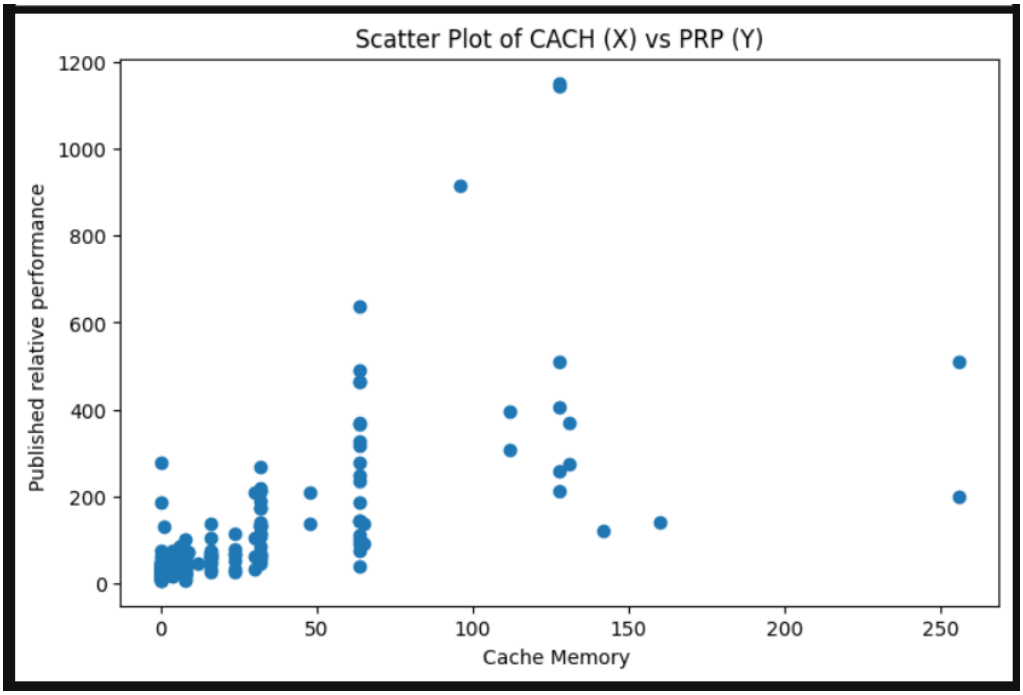
Description: Estimated relative performance from the original article

Missing Values: No

Therefore, the reason behind choosing Cache Memory as the predictor is that it often scales nonlinearly (power-law) with performance. We may expect weight to increase disproportionately as Cache Memory increases, hence nonlinear curve.

The goal here is to use nonparametric smoothing to explore and model this relationship.

Exploratory Data Analysis and Data Cleaning



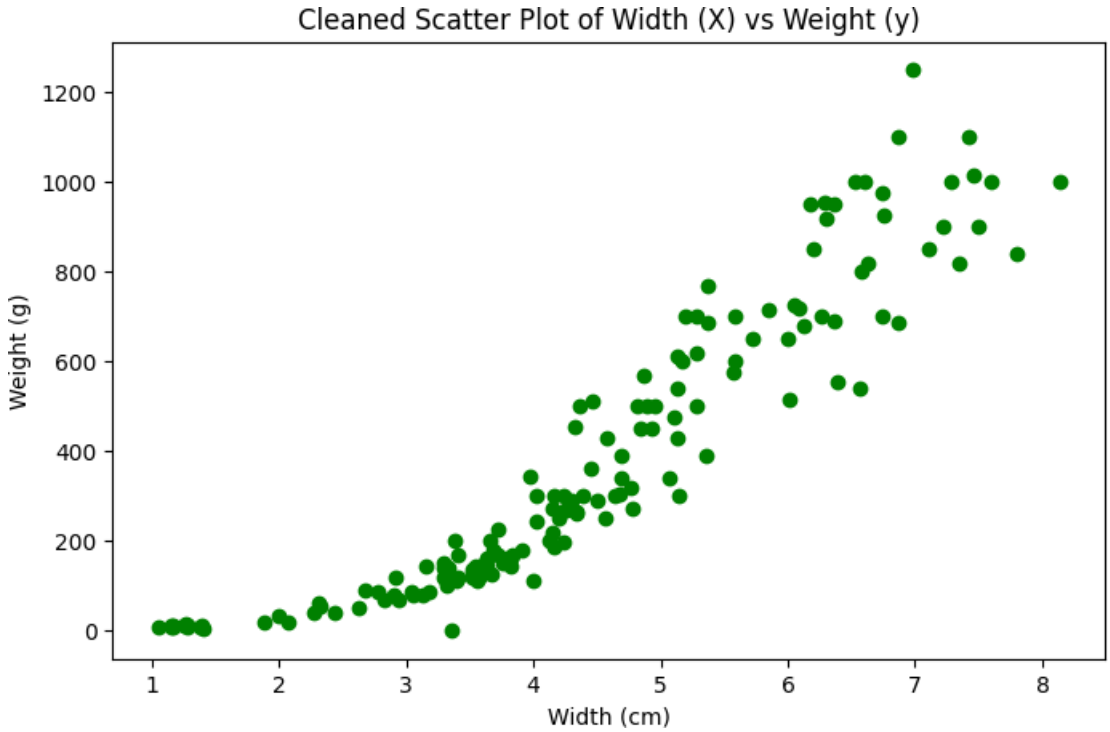
The scatter plot shows that there seems to be an exponential relationship between X and Y.

```
Missing values in dataset:

PRP       0

CACH      0

dtype: int64
```

There are no missing entries in the dataset, so none of the rows get dropped from the subset (X and Y).

Outlier detection: One common technique for detecting outliers is the Z score. It is a statistical measurement that describes how far a data point is from the mean, expressed in terms of standard deviations. It helps us to identify if a data point is relatively higher or lower than the mean and how far it deviates from the average value. Commonly, data points with a Z-score greater than 3 or less than -3 are considered outliers as they lie more than 3 standard deviations away from the mean.

```
Before outlier detection:  (209, 10)

After outlier detection:  (198, 10)
```



So eleven points (potential outliers) got eliminated.

From the cleaned scatter plot, there seems to be a curved and increasing trend, as the Cache Memory (Kb) increases Performance also seems to increase in an exponential manner.

Here instead of a parametric regression that would impose a specific functional form, we adopt non parametric approaches such as KNN smoother, Kernel smoother, LOESS, that would be completely based on the local structure of the data. But for comparison purpose, we will fit a parametric regression and draw a comparison with the rest of the non-parametric techniques.

Cross-validation for hyperparameter tuning

If we choose a very small number of folds,

the training sets are smaller. Models trained on smaller datasets tend to be less representative of the true model that would be learned from the entire dataset, leading to higher bias. Smaller training set sizes across the folds can result in models that are less stable and more sensitive to the specific data points in each fold, leading to more variability in the performance estimates across different folds. It results in a lower computational cost because the model is trained fewer times.

And if we choose very large number of folds, each model is trained on a larger portion of the data. This leads to a more accurate and less biased estimate of the model's performance because the training data for each fold is more representative of the full dataset. When the number of folds increases, the validation set for each fold becomes smaller. This can cause a higher variance in the model's performance across the different folds. It results in a significant increase in the computational cost and time.

We have 158 observations in the training set, choosing 4-fold cross validation will be suitable here as approximately 39 observations would be there in each fold, which provides a good trade-off between bias and variance and not results in a very high computational cost and time.

So, we choose 4 folds for cross-validation.

For KNN smoother, hyperparameter is the number of neighbors (K), if we increase the number of neighbors, so more data points will be included in the neighborhood in order to obtain the estimated value of the function at a particular point. The estimator will be more stabilized, variance will reduce but bias will increase since points which are farther away are considered in the neighborhood of the estimation point which might not reflect the local behavior at that point, and hence introduces bias.

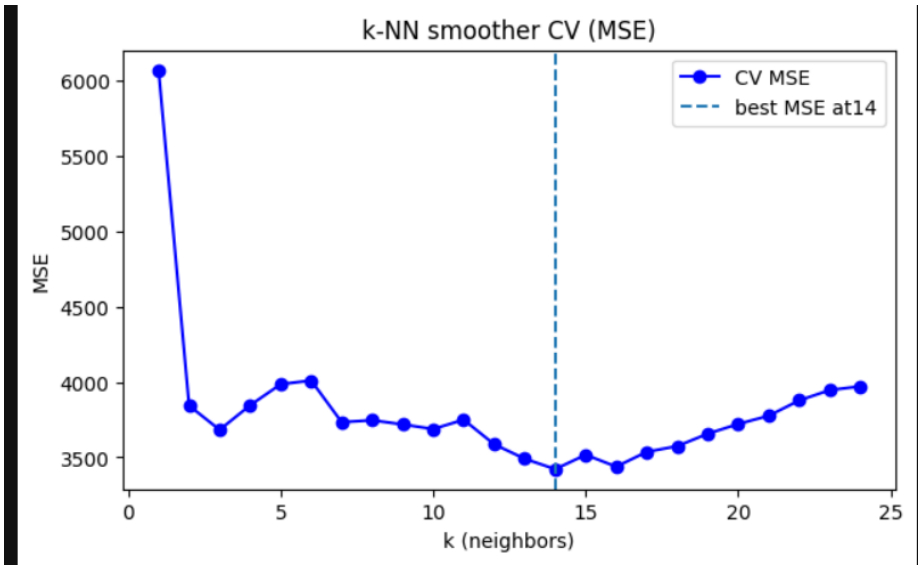
For Kernel smoother, the hyperparameter is the bandwidth (h), it defines the width of the neighborhood. If we increase h, then the width of the neighborhood will increase, variance will decrease and bias will increase.

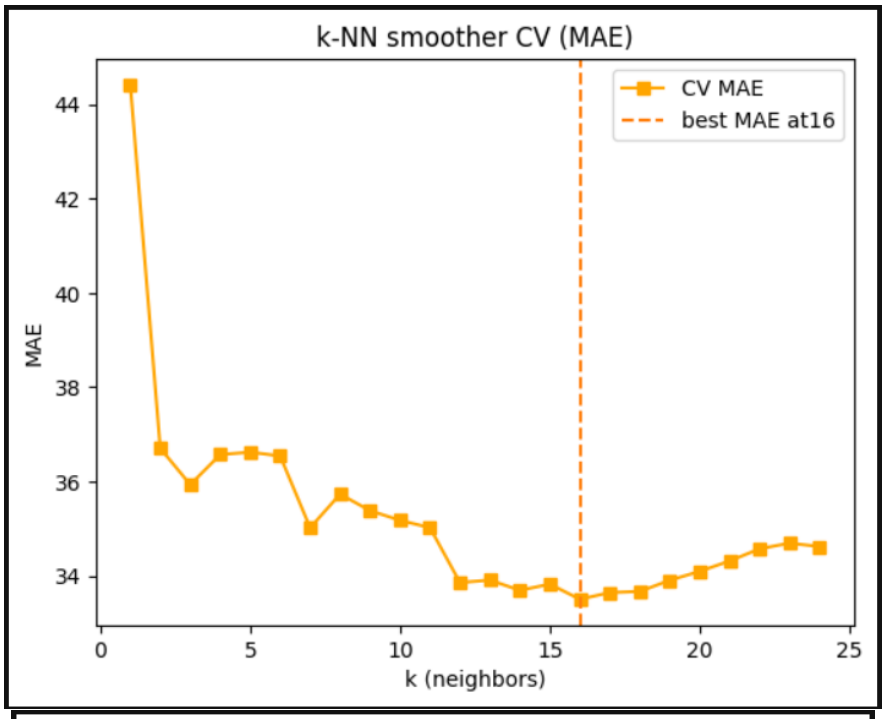
For LOWESS, the hyperparameter is the frac. It specifies the fraction of data points used to fit the local regression for each estimate. If we increase frac, then more data points will be considered in the local fit, resulting in lesser variance and increased bias. Here, we have considered a grid of values for each of the smoothing technique to perform cross validation:

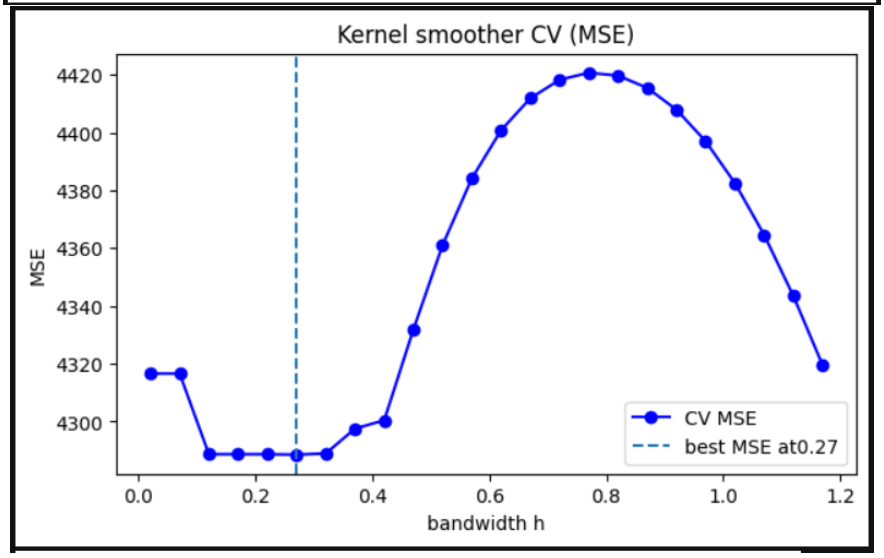
- KNN : 1,2,3,....25
- Kernel : 0.02, 0.07, 0.12,..,1.2
- LOESS : 0.5 to 0.9 of size 24

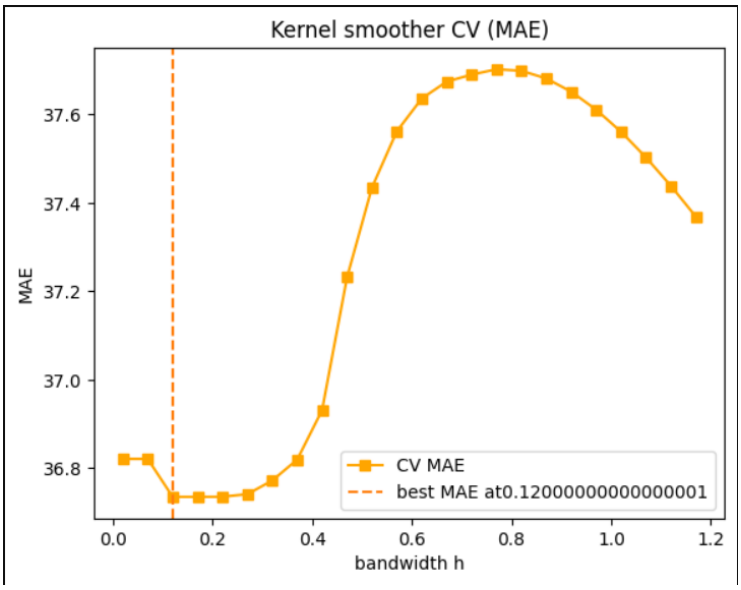
This grid of values has been decided by performing experimentations. Here, we have chosen MSE (mean squared error) and MAE (mean absolute error) as the error metrics. MSE penalizes larger errors heavily and is sensitive to outliers; MAE is robust and easier to interpret.

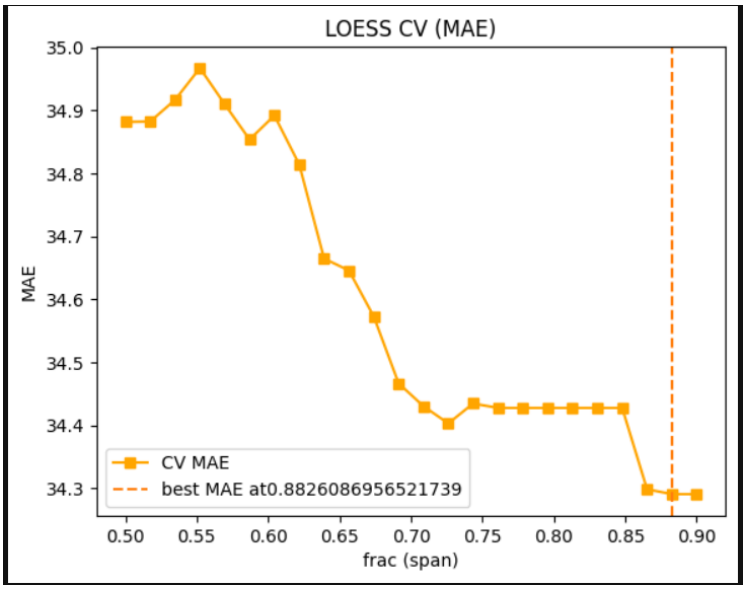
Here, we have chosen MSE to choose the optimal hyperparameters.

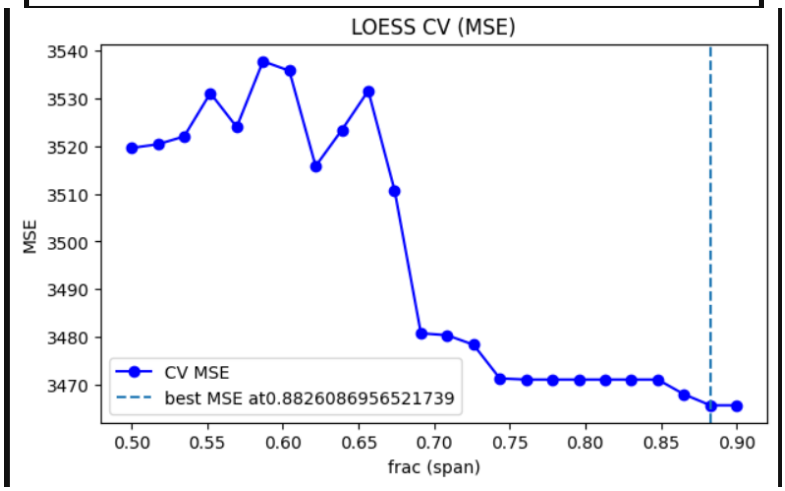



k-NN smoother CV (MAE)
CV MAE
best MAE at16
MAE
k (neighbors)
44
42
40
38
36
34
0
5
10
15
20
25


Kernel smoother CV (MSE)
CV MSE
best MSE at0.27
MSE
bandwidth h
4420
4400
4380
4360
4340
4320
4300
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Kernel smoother CV (MAE)
MAE
bandwidth h
CV MAE
best MAE at0.12000000000000001


LOESS CV (MAE)
MAE
frac (span)
CV MAE
best MAE at0.8826086956521739


LOESS CV (MSE)
MSE
frac (span)
CV MSE
best MSE at0.8826086956521739

In all cases except for LOESS, the CV error (MSE and MAE) decreases as the hyperparameter values increase, attains a minimum value and then again increases. The value of the hyperparameter for which the CV MSE attains the lowest value is the optimal choice of the hyperparameter.

	KNN - K	KNN - MSE	KNN - MAE	Kernel-h	Kernel - MSE	Kernel - MAE	LOESS - frac	LOESS - MSE	LOESS - MAE
0	1	6072.145513	44.413462	0.02	4316.612620	36.821051	0.500000	3519.597661	34.881636
1	2	3843.345353	36.705609	0.07	4316.612620	36.821051	0.517391	3520.364128	34.882252
2	3	3682.658102	35.935737	0.12	4288.713366	36.735218	0.534783	3522.000844	34.916605
3	4	3845.507492	36.572276	0.17	4288.713361	36.735218	0.552174	3531.076211	34.967151
4	5	3985.418231	36.623654	0.22	4288.707664	36.735410	0.569565	3523.977374	34.909694

Optimal choice of hyperparameter:

- KNN smoothing(K) = 14
- Kernel smoothing (h) = 0.27
- LOWESS (frac) = 0.8826086956521739

Comparison between the Models (techniques)

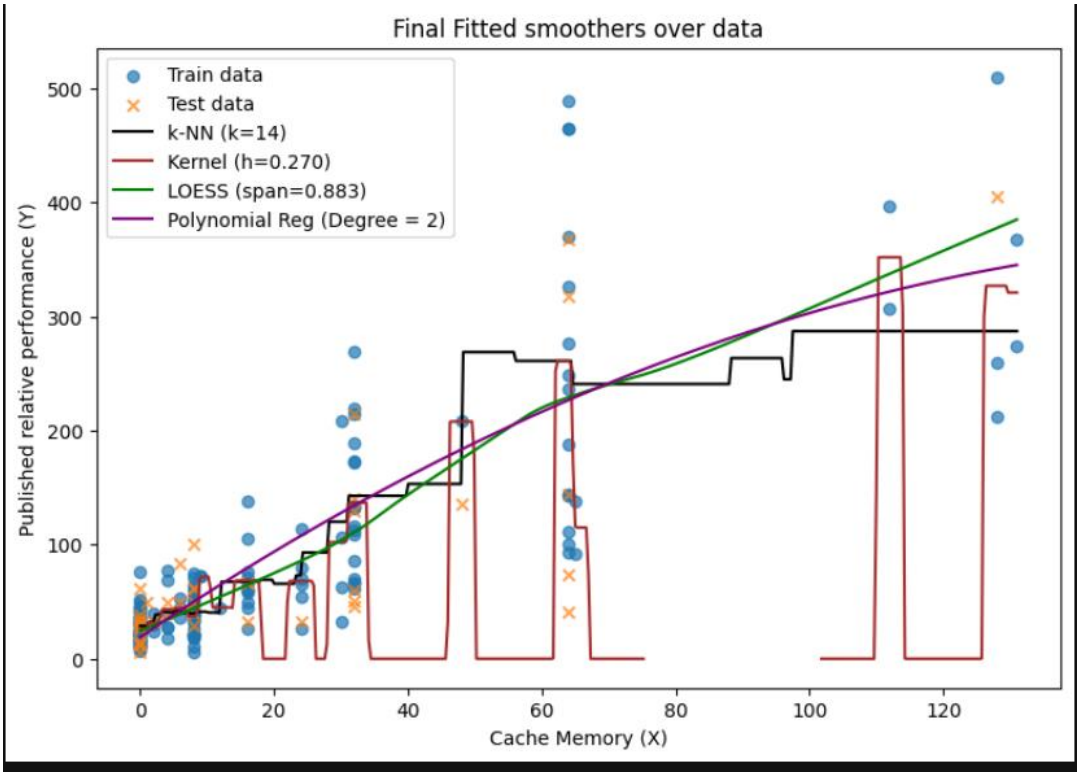
Using the optimal choice of the hyperparameters, we fit the final model (smoothing techniques) on the entire train set, and obtain the Test MSE and Test MAE based on the test set to conclude which model performs best (generalises well on the unseen observations).

	Method	Optimal Hyperparameter	CV MSE	Test_MSE	Test_MAE
0	KNN	14	3421.472503	4361.632270	41.594643
1	Kernel	0.27	4288.563968	4074.476696	39.873677
2	LOESS	0.882609	3465.717821	3287.576145	37.157700
3	Polynomial Regression (Deg = 2)	-	-	3578.296968	40.095214

Out of the three non-parametric techniques considered, LOESS performs the best. It was expected because LOESS typically produces a smoother, more flexible curve and offers built-in robustness to

outliers, since it fits polynomial regression model locally to the weighted data points rather than just averaging. As a result, it better captures the curvature of the data.

And in order to obtain a comparison, we also performed polynomial regression of degree 2 (Parametric), fitted globally. It outperforms LOESS in this scenario.



From the visual inspection, LOESS (best-performing model among non-parametric techniques) seems to appear smoothest. KNN produces the most jagged curve.

There can be situations where a different smoother can perform better, as none of the methods are universally best; rather, each has distinct advantages depending on the specific data and context. Some of the scenarios are mentioned here:

- If the true function contains abrupt local features, a more local smoother (smaller span / smaller bandwidth) could be better.
- If the data is very noisy but the true function is simple, a coarser smoother or parametric model might outperform flexible smoothers (here, polynomial regression of degree = 2 (fitted globally) outperforms the best non-parametric smoothing technique).
- If outliers are present, robust approaches (or MAE-focused tuning) may prefer a different smoother.

This project demonstrated that hyperparameters directly control the bias-variance tradeoff. Cross-validation is essential to choose hyperparameters that generalise well. Visual inspection of fitted curves alongside error metrics helps ensure the model is both accurate and interpretable.