CS215 - Assignment 3

Ujwal L Shankar (23b1050)
Madhav R Babu (23b1060)
Marumamula Venkata Pranay (23b1073)

Contents

1 Finding Optimal Bandwidth

1. (a) To prove:

$$\int \hat{f}(x)^2\,dx = \frac{1}{n^2 h}\sum_{j=1}^{m} v_j^2$$

For $x \in B_j$, the histogram estimate ($\hat{f}(x)$) is the fraction (p_j) of data points in bin B_j divided by the bandwidth h.

$$\hat{f}(x) = \frac{p_j}{h} = \frac{v_j}{nh}$$

where v_j is the number of data points in B_j. Now,

$$\int \hat{f}(x)^2\,dx = \int_{B_1} \hat{f}(x)^2\,dx + \int_{B_2} \hat{f}(x)^2\,dx + \cdots \int_{B_m} \hat{f}(x)^2\,dx$$

$$\int \hat{f}(x)^2\,dx = \int_{B_1} \left(\frac{v_1}{nh}\right)^2 dx + \int_{B_2} \left(\frac{v_2}{nh}\right)^2 dx + \cdots \int_{B_m} \left(\frac{v_m}{nh}\right)^2 dx$$

The inner term of every integral is independent of x, and hence can be pulled out. We also have $\int_{B_j} dx = h$

$$\int \hat{f}(x)^2\,dx = \frac{1}{n^2h^2} \cdot \left(\sum_{i=1}^{m} v_i^2\right) \cdot h = \frac{1}{n^2 h} \cdot \sum_{i=1}^{m} v_i^2$$

Hence proved.

(b) After removing X_i where $X_i \in B_j$, there are $v_j - 1$ data points in bin B_j and $n-1$ points in total.

$$\hat{f}_{-i}(X_i) = \frac{v_j - 1}{(n-1)\,h}$$

For each bin B_k, there are v_k number of points for which some $X_j \in B_k$. Transform the summation in terms of the bin number j.

$$\sum_{i=1}^{n} \hat{f}_{-i}(X_i) = \sum_{j=1}^{m} v_j \cdot \frac{v_j - 1}{(n-1)\,h} = \frac{1}{(n-1)\,h} \cdot \sum_{j=1}^{m} \left(v_j^2 - v_j\right)$$

2. (a) The estimated probabilities are mentioned in the table below

Interval	**Frequency**
0.0-0.4	0.0059
0.4-0.8	0.3824
0.8-1.2	0.3235
1.2-1.6	0.0529
1.6-2.0	0.1118
2.0-2.4	0.1000
2.4-2.8	0.0059
2.8-3.2	0.0000
3.2-3.6	0.0118
3.6-4.0	0.0059

(b) This probability estimation is an under fit for the true probability. This is because of too few number of bins, the smoothing effect is so much that the essence of the distribution is partially lost.

(c) The plot of Cross validation score vs bandwidth has been saved in "crossvalidation.png"

(d) Optimal value of bin width is 0.0833. This has been verified using the code written in 1.py

(e) The histogram obtained by using the optimal bandwidth is saved in "optimalhistogram.png". This obtained plot having more bins than the original plot, allows for lesser smoothing. This helps the obtained plot to capture the essence of the distribution in a better manner.

(f) The code for this entire question has been saved in "1.py".

2 Detecting Anomalous Transactions using KDE

2.1 Designing a custom KDE class

A custom Epanechnikov KDE class has been implemented in file 2.py.

2.2 Estimating Distribution of Transcations

1. Epanechnikov class has been initialized with bandwidth 1.0, this can be modified if required and None is used as a placeholder for data.
2. Data present in 'transcation_data.npz' has been fitted.
3. A reasonably smooth curve was obtained using the Epanechnikov KDE class, the resulting distribution contains **2 modes**.

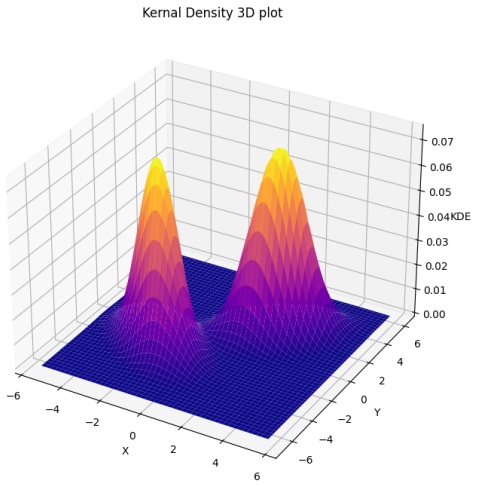


Figure 1: Transcation Distribution

4. Plot obtained is present in transaction_distribution.png.

3 Higher-Order Regression

3.1 Part 1

In a simple linear regression model Y=Bx+ A, where

$$B = \frac{\Sigma x_i y_i - n\bar{x}\bar{y}}{\Sigma x_i^2 - n\bar{x}^2}$$

$$A = \bar{y} - B\bar{x}$$

At $x = \bar{x}$, $Y = B\bar{x} + \bar{y} - B\bar{x} = \bar{y} \implies (\bar{x}, \bar{y})$ lies on the least-square regression line.

3.2 Part 2

$Y = \beta_0^* + \beta_1^* \cdot (x - \bar{x}) + \epsilon \implies \beta_1^* \cdot x + \beta_0^* - \beta_1^* \cdot \bar{x} + \epsilon$
comparing coefficients with the equation obtained by performing MLE on $Y = Bx + A + \epsilon$ we obtain $\beta_1^* = B = \frac{\Sigma x_i y_i - n\bar{x}\bar{y}}{\Sigma x_i^2 - n\bar{x}^2}$ and $\beta_0^* = \bar{y}$
Similarity: both B and β_1^* are equal
Difference: A and β_0^* differ

3.3 Part 3

a) On performing Higher Order regression on the given data by and using cross-validation to evaluate the model we obtained the ideal value of m to be 5, and its corresponding coefficients $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, and \beta_5$ are present in 3_wights.pkl file. Predictions for Test data has been saved to 3_predictions.csv file.

b) **m=5** gives the best fit. Data is under fitted when m < 5 and overfitted when in m > 5. Additional plots for better visualization of overfitting and underfitting have also been added.

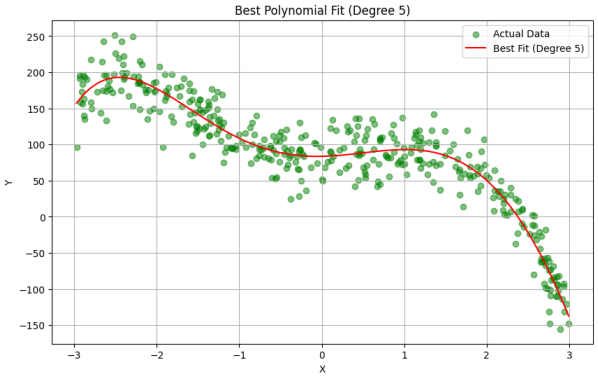
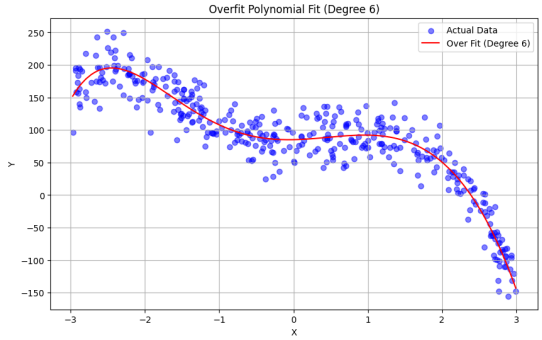
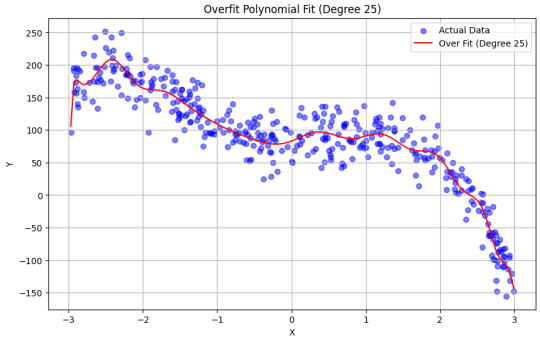


Figure 2: Correct Fit



(a) Overfit for m=6

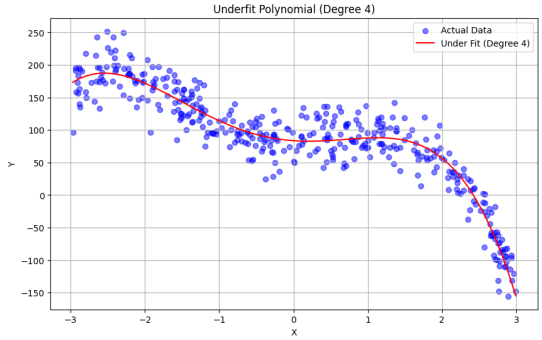


(b) Overfit for better visualization

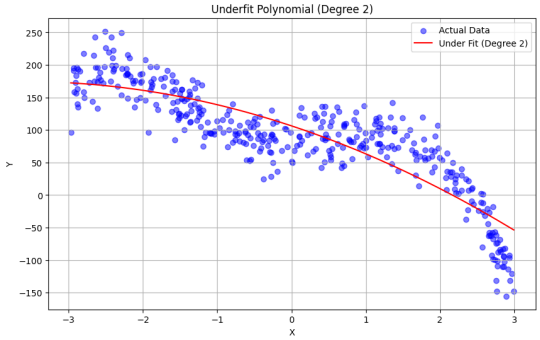
c) SS_R and R^2 for m=4, 5, and 6 (averaged over various folds):

i m=4, $SS_R = 62586.046367, R^2 = 0.897516$
ii m=5, $SS_R = 59810.615845, R^2 = 0.902064$
iii m=6, $SS_R = 59970.102075, R^2 = 0.901837$

SS_R and R^2 for m = 1 to 25 have been printed in the Python notebook.



(a) Underfit for m=4



(b) Underfit for better visualization

4 Non-parametric Regression

4.1 Code

The code required for the correct implementation of Nadaraya-Watson Kernal regression has been uploaded in file 4.ipynb.
For cross-validation, the **k-fold** method has been implemented.

4.2 Plots

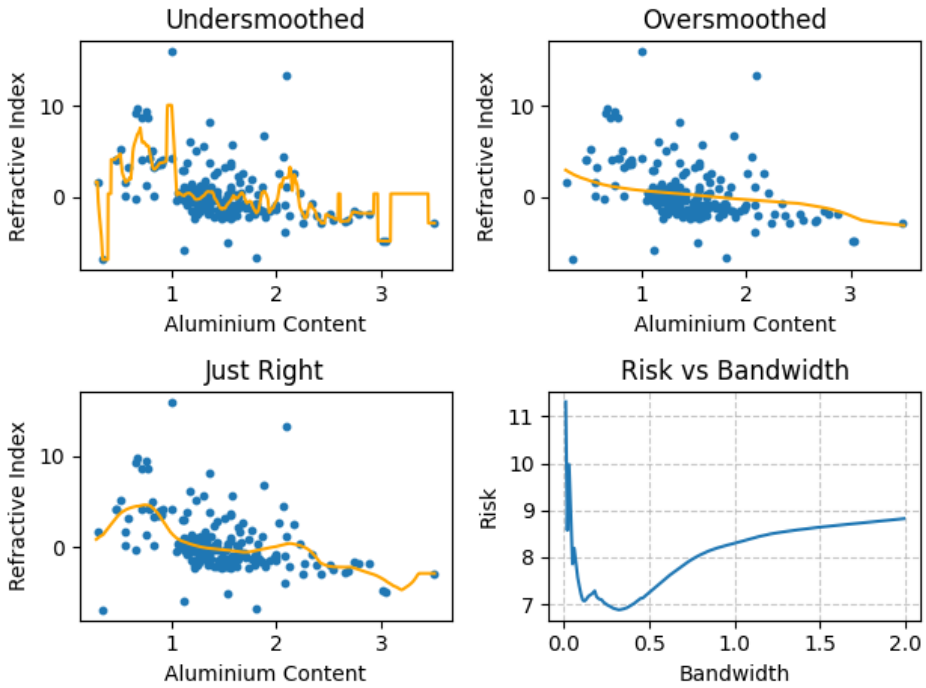


Figure 5: Epanechnikov Kernal Regression

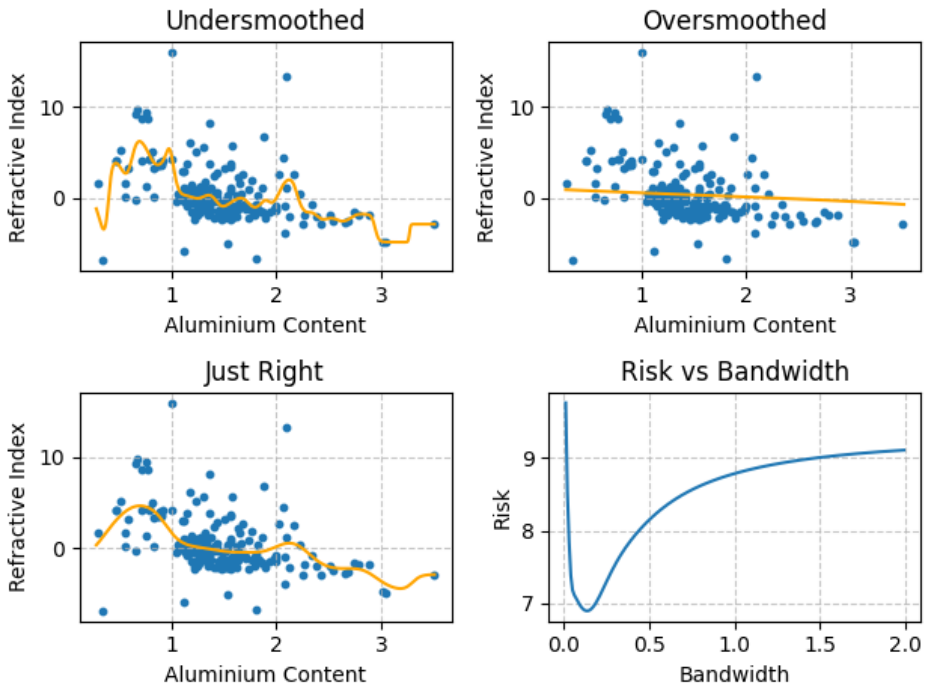


Figure 6: Gaussian Kernal Regression

4.3 Bandwidth

Bandwidth corresponding to minimum estimated risk:

- Epanechnikov Kernal Regression: 0.32
- Gaussian Kernal Regression: 0.13

4.4 Similarities and Differences

First and foremost, the gaussian kernel follows a normal curve whereas the Epanechnikov kernel follows a downward parabolic path in $[-1, 1]$ and flattens to 0 everywhere else.

The Gaussian kernel usually generates smoother prediction curves because it is less local, while considering points farther away with decreasing weights.

On the other hand, the Epanechnikov kernel produces less-smooth predictions, as it gives zero consideration for points which are beyond a certain distance. This makes it more influenced by local patterns.

For dense data points, both of them perform rather similarly, but for sparse data spreads, the difference is significant due to the reasons mentioned above.
Even the Risk vs Bandwidth plots for the Gaussian kernel is smoother compared to that of the Epanechnikov kernel.

5 Multivariate Insights Unlocked!

The code required for the correct implementation of Multi-Variate regression has been uploaded in file 5.ipynb.

It was found that the parametric estimation gave the most accurate results and least error, while taking lesser time to compute and execute.

Insights, analysis of data, along with supporting plots are present in the jupyter notebook 5.ipynb.
Best Score on Kaggle so far: 265.546.