Assignment 3

CS215

Abhineet Majety
23B0923

Mohana Evuri
23B1017

Saksham Jain
23B1074

Fall 2024

PROBLEM 1

Finding Optimal Bandwidth

1.1 Cross-Validation Estimator

The cross-validation estimator is defined as

$$\hat{J}(h) = \int \hat{f}(x)^2 dx - \frac{2}{n} \sum_{i=1}^{n} \hat{f}_{(-i)}(X_i),$$

where $\hat{f}_{(-i)}$ is the histogram estimator after removing the ith observation.

(a)

We can write the histogram estimator $\hat{f}(x)$ as

$$\begin{aligned}
\hat{f}(x) &= \sum_{k=1}^{m} \frac{\hat{p}_k}{h} \mathbb{I}[x \in B_k] \\
&= \sum_{k=1}^{m} \frac{v_k}{nh} \mathbb{I}[x \in B_k].
\end{aligned}$$

Hence,

$$\begin{aligned}
\int \hat{f}(x)^2 dx &= \int \left(\sum_{k=1}^{m} \frac{v_k}{nh} \mathbb{I}[x \in B_k] \right)^2 dx \\
&= \sum_{j=1}^{m} \int_{B_j} \left(\sum_{k=1}^{m} \frac{v_k}{nh} \mathbb{I}[x \in B_k] \right)^2 dx + 0 \\
&= \sum_{j=1}^{m} \int_{B_j} \left(\frac{v_j}{nh} \right)^2 dx \qquad (1.1) \\
&= \sum_{j=1}^{m} \frac{(v_j)^2}{n^2 h^2} h \\
&= \frac{1}{n^2 h} \sum_{j=1}^{m} (v_j)^2.
\end{aligned}$$

This is the required result.

(b)

Suppose $X_i \in B_j$. Then on removing X_i, the number of points in B_j is $v_j - 1$ and the total number of points is $n-1$. Hence, $\hat{f}_{(-i)}(X_i) = \frac{(v_j-1)}{(n-1)h}$. Using this result,

$$\begin{aligned}
\sum_{i=1}^{n} \hat{f}_{(-i)}(X_i) &= \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{v_j - 1}{(n-1)h}\mathbb{I}[X_i \in B_j] \\
&= \sum_{j=1}^{m}\sum_{i=1}^{n} \frac{v_j - 1}{(n-1)h}\mathbb{I}[X_i \in B_j] \\
&= \sum_{j=1}^{m} \frac{v_j - 1}{(n-1)h} \cdot v_j \\
&= \frac{1}{(n-1)h}\sum_{j=1}^{m}(v_j^2 - v_j).
\end{aligned} \tag{1.2}$$

This is the second part. We can combine the 1.1 and 1.2 parts to write the cross-validation estimator as

$$\hat{J}(h) = \frac{2}{(n-1)h} - \frac{n+1}{(n-1)h}\sum_{j=1}^{m}\hat{p}_j^2$$

1.2 Using the Cross-Validation Estimator

(a)

The probabilities were calculated using `numpy.histogram` function. The estimated probabilities $\hat{p}_j$ for all the bins are:

Bin	$\hat{p}_j$
1	0.20588235
2	0.48823529
3	0.04705882
4	0.04117647
5	0.13529412
6	0.05882353
7	0.00588235
8	0.0
9	0.01176471
10	0.00588235

Using the obtained values, the histogram was plotted using the `matplotlib.pyplot.hist` function. The histogram plot can be found in `images/10binhistogram.png`.

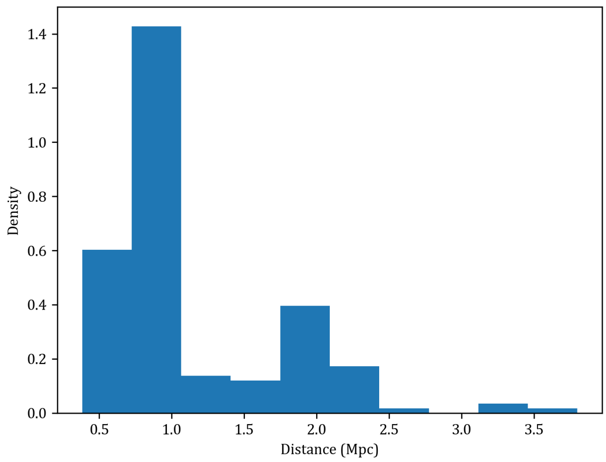

Figure 1.1: 10 Bin Histogram

(b)

The probability distribution is **oversmoothed**. Lower values of h yield a lower cross-validation score.

(c)

The cross-validation score for the number of bins from 1 to 1000 was calculated using: `numpy.histogram`, `numpy.square` and `numpy.sum` functions. The plot of cross-validation score versus h can be found in `images/crossvalidation.png`

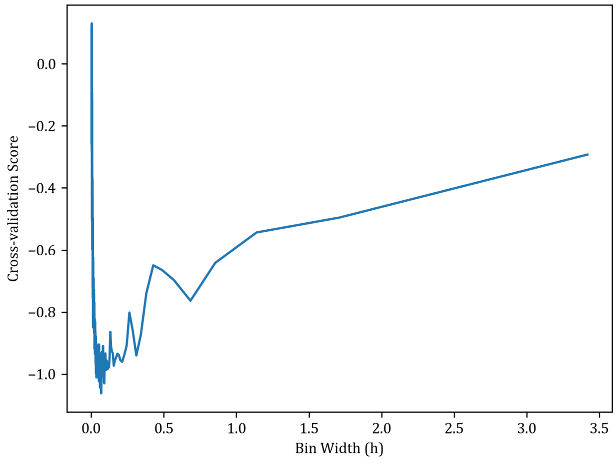

Figure 1.2: Cross Validation

(d)

The optimal bin width is the value of h for which the cross-validation score is minimum. From the plot, this corresponded to 50 bins, for which the value of h is **0.06835999** Mpc.

(e)

The histogram with optimal value $h^* = 0.06835999$ is present in `images/optimalhistogram.png`.

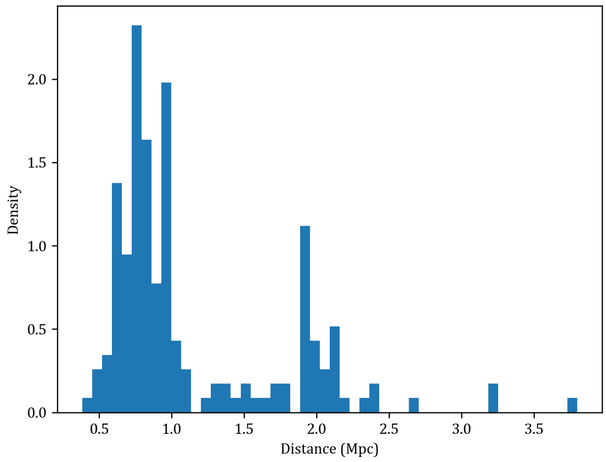

Figure 1.3: Cross Validation

(f)

The code for all the parts is present in `code/1.py`.

PROBLEM 2

Detecting Anomalous Transactions using KDE

2.1 Designing a custom KDE Class

The implemented code for the class is:

```
class EpanechnikovKDE:
    def __init__(self, bandwidth=1.0):
        """Initialize with given bandwidth."""
        self.bandwidth = bandwidth
        self.data = None

    def fit(self, data):
        """Fit the KDE model with the provided data."""
        self.data = np.array(data)

    def epanechnikov_kernel(self, x, xi):
        """Epanechnikov kernel function for 2D using vectorized operations."""
        norm_squared = np.sum(((xi - x) / self.bandwidth) ** 2, axis=-1)
        return ((2 / np.pi) * (1 - norm_squared)) * (norm_squared <= 1)

    def evaluate(self, x):
        """Evaluate the KDE at multiple points x in 2D."""
        return self.epanechnikov_kernel(x, self.data).mean() / (self.bandwidth ** 2)
```

Code 2.1: 2D Epanechnikov KDE class

2.2 Estimating Distribution of Transactions

For the distribution that is obtained from the given data, we have 2 modes. The 3D graph for the given data is as follows:

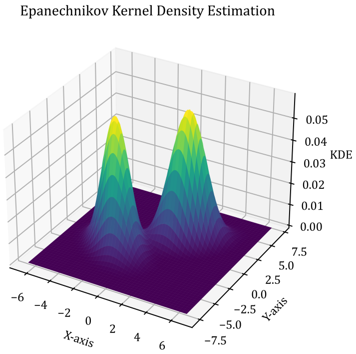

Figure 2.1: Transaction Distribution

PROBLEM 3

Higher-Order Regression

3.1 Showing that the Point $(\bar{x}, \bar{y})$ lies on the Least-Squares Regression Line

In simple linear regression, the model is given by:

$$Y = \beta_0 + \beta_1 x + \epsilon$$

Where:

- β_0 is the intercept,
- β_1 is the slope, and
- ϵ is the error term.

The least squares regression line, which minimizes the sum of squared residuals, is given by the following equation:

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

To find the least-squares estimates $\hat{\beta}_0$ and $\hat{\beta}_1$, we have to minimize the sum of squared residuals (SSR):

$$\text{SSR} = \sum_{i=1}^{n} \left(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)\right)^2$$

Partial derivative with respect to $\hat{\beta}_0$:

$$\frac{\partial \text{SSR}}{\partial \hat{\beta}_0} = -2 \sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i\right) = 0$$
$$\sum_{i=1}^{n} y_i = n\hat{\beta}_0 + \hat{\beta}_1 \sum_{i=1}^{n} x_i$$
$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Partial derivative with respect to $\hat{\beta}_1$:

$$\frac{\partial \text{SSR}}{\partial \hat{\beta}_1} = -2 \sum_{i=1}^{n} x_i \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i\right) = 0$$
$$\sum_{i=1}^{n} x_i y_i = \hat{\beta}_0 \sum_{i=1}^{n} x_i + \hat{\beta}_1 \sum_{i=1}^{n} x_i^2$$
$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Now, substituting $\bar{x}$ into the regression line equation gives

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 \bar{x}$$

Substitute $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$:

$$\hat{Y} = (\bar{y} - \hat{\beta}_1 \bar{x}) + \hat{\beta}_1 \bar{x}$$

Simplifying this:

$$\hat{Y} = \bar{y}$$

Thus, the point $(\bar{x}, \bar{y})$ lies exactly on the least-squares regression line.

3.2 New model using $z = x - \bar{x}$

The original least-squares estimate for β_1:

$$\hat{\beta}_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

Since $z_i = x_i - \bar{x}$, this can be rewritten in terms of z:

$$\hat{\beta}_1^* = \frac{\sum z_i(y_i - \bar{y})}{\sum z_i^2}$$

The original least squares estimate for β_0:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

For the intercept in the new model, we know that z is centered around 0, which means $\bar{z} = 0$. Therefore, the least squares estimate of β_0^* is simply the average of y, i.e., $\bar{y}$:

$$\hat{\beta}_0^* = \bar{y}$$

Since $z = x - \bar{x}$, this is just a change in the predictor variable. Therefore, the slope of the regression line does not change, and we have the following:

$$\hat{\beta}_1^* = \hat{\beta}_1$$

In the new model, the least squares estimates of β_0 are:

$$\hat{\beta}_0^* = \bar{y}$$

On the other hand, in the original model, the least squares estimate of β_0 is:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Thus, the relationship between β_0 and β_0^* is:

$$\hat{\beta}_0^* = \hat{\beta}_0 + \hat{\beta}_1 \bar{x}$$

Model Differences: The original model estimates both the intercept and slope based on uncentered values of x, while the transformed model estimates the intercept at the mean of Y and uses a centered version of x (now z). The slope remains the same in both models.
Interpretation Differences: The intercept in the original model reflects the value of Y when x is zero (or of a different origin), while in the new model, it represents the mean of Y at the mean of x, providing a different baseline from which predictions are made.

3.3 Regression for a Dataset

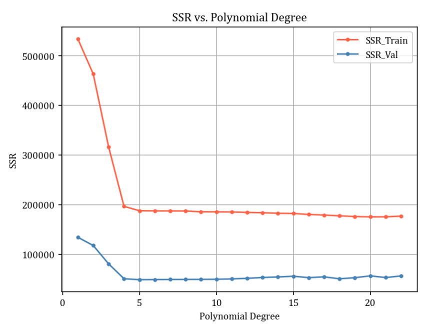

Figure 3.1: The SSR graph for various degrees

By the SSR graph above, degree = 5 seems to be the optimal degree for the polynomial.

3.3.1 Correct Fit

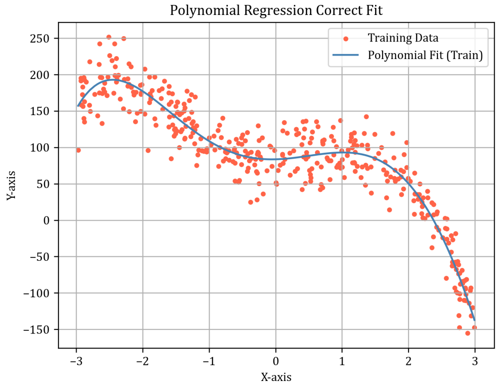

Figure 3.2: Correct Fit at degree = 5

3.3.2 UnderFit

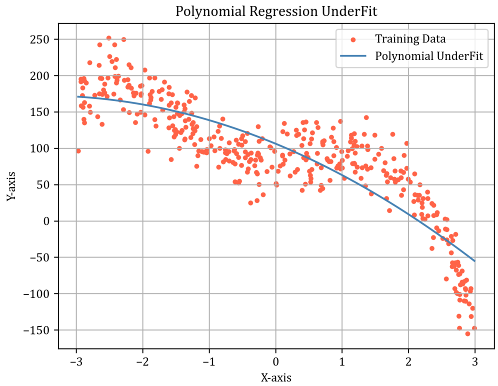

Figure 3.3: UnderFit at degree = 2

3.3.3 OverFit

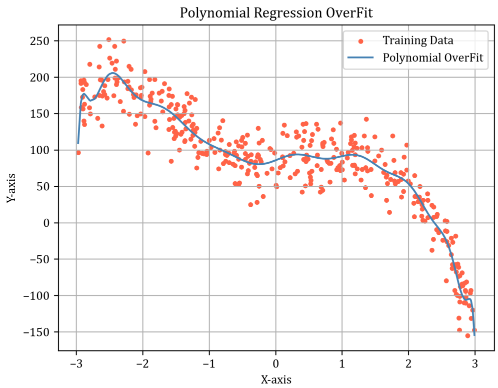

Figure 3.4: OverFit at degree = 20

PROBLEM 4

Non-parametric Regression

4.1

The two kernel functions chosen are the **Gaussian** kernel and the **Epanechnikov** kernel. We have used **k-fold cross-validation** to find the bandwidth corresponding to minimum estimated risk for each kernel. For estimating risk, the data was shuffled using `pandas.DataFrame.sample` method. The data was then split into $k = 10$ folds to perform cross-validation. The code can be found in `code/4.ipynb`.

4.2

The optimal bandwidths found are:

Kernel	Optimal bandwidth
Gaussian	0.13183673469387755
Epanechnikov	0.49734693877551023

The plots obtained for the Gaussian kernel are:

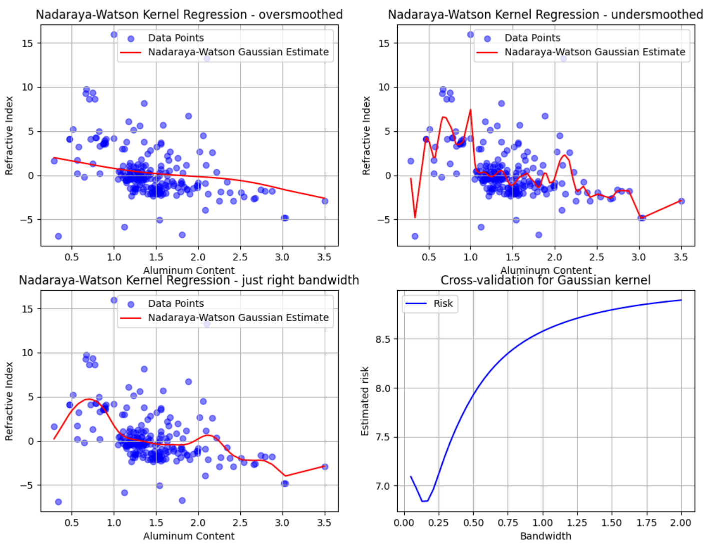

Figure 4.1: Gaussian kernel

The plot obtained for the Epanechnikov kernel are:

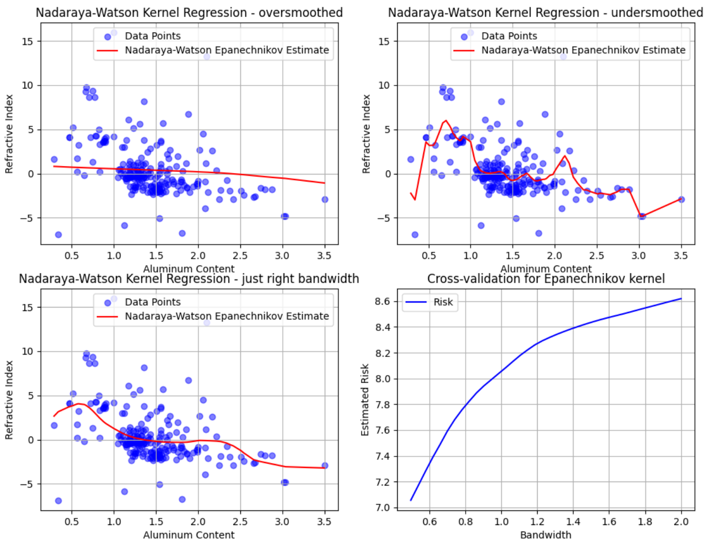

Figure 4.2: Epanechnikov kernel

4.3

The Gaussian kernel gives a lower minimum risk than the Epanechnikov kernel. The values on on one run are:

Kernel	Minimum risk
Gaussian	6.85
Epanechnikov	7.21

4.3.1 Differences

This shows that Gaussian kernel is better for this dataset. The risk-bandwidth graph for the Gaussian kernel reaches a minimum and then increases. On the other hand, the risk-bandwidth graph for the Epanechnikov kernel starts suddenly and then increases. For smaller bandwidth than the start point (the minimum-risk bandwidth), the risk tends to ∞ as the value $\left|\frac{x-x_i}{h}\right|$ is always greater than 1 for the given data.

4.3.2 Similarities

Some similarities between regression using both kernels are: the risk increases for large bandwidth. On overlapping the curves, both curves are found to be similar, as seen below

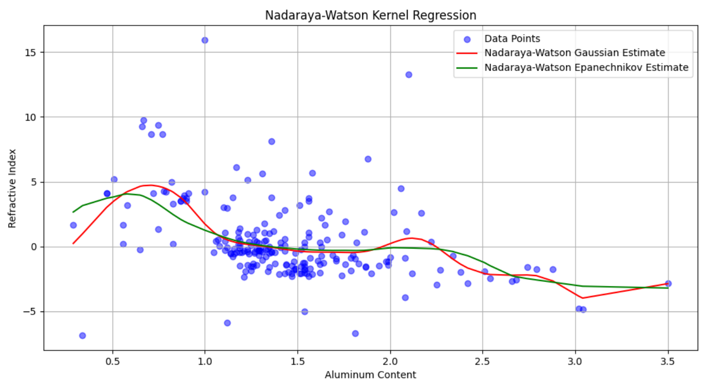

Figure 4.3: Overlapped plots