CSC2515 Assignment 2

Hon Wa Ng

18 October 2024

Question 1a

To show that the sample average estimator $h_{\text{avg}} = \frac{1}{n}\sum_{i=1}^{n} Y_i$ is the solution of the following optimization problem:

$$h_{\text{avg}}(\mathcal{D}) = \arg\min_{m \in R} \frac{1}{n}\sum_{i=1}^{n}(Y_i - m)^2$$

This is an optimization problem where the sum of squared differences between the observed values Y_i and an arbitrary mean m is minimized.

1. Objective function: To minimize the following function:

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - m)^2$$

2. Take the derivative: To find the minimum, we need to differentiate this function with respect to m, and set the derivative equal to zero:

$$\frac{\partial}{\partial m}\left(\frac{1}{n}\sum_{i=1}^{n}(Y_i - m)^2\right)$$

Expanding the sum term:

$$\frac{\partial}{\partial m}\left(\frac{1}{n}\sum_{i=1}^{n}\left(Y_i^2 - 2Y_i m + m^2\right)\right)$$

As we are differentiating with respect to m, Y_i^2 is a constant, it can be ignored and the remaining terms is:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial m}\left(-2Y_i m + m^2\right)$$

Taking the derivative of each term:

$$\frac{1}{n}\sum_{i=1}^{n}\left(-2Y_i + 2m\right)$$

$$\frac{1}{n}\left(-2\sum_{i=1}^{n} Y_i + 2nm\right)$$

3. Set the derivative to zero: To find the minimizing value of m, we set the derivative equal to zero:

$$-2\sum_{i=1}^{n} Y_i + 2nm = 0$$

Simplifying:

$$\sum_{i=1}^{n} Y_i = nm$$

Solving for m gives:

$$m = \frac{1}{n}\sum_{i=1}^{n} Y_i$$

This shows that the value of m that minimizes the sum of squared differences is the **sample average**:

$$h_{\text{avg}} = \frac{1}{n}\sum_{i=1}^{n} Y_i$$

Thus, the sample average is the solution to the given optimization problem.

Question 1b

The bias-variance decomposition for a general estimator $h(\mathcal{D})$ is given by:

$$E_{\mathcal{D}}\left[(h(\mathcal{D}) - \mu)^2\right] = (E_{\mathcal{D}}[h(\mathcal{D})] - \mu)^2 + E_{\mathcal{D}}\left[(h(\mathcal{D}) - E_{\mathcal{D}}[h(\mathcal{D})])^2\right]$$

where the first term represents the bias squared and the second term represents the variance.

For the sample average estimator h_{avg}, we have:

$$h_{\text{avg}} = \frac{1}{n}\sum_{i=1}^{n} Y_i$$

We can now compute the bias and variance for this estimator.

Bias Calculation:

In this and following answer, the bias is defined as:

$$\text{Bias}(h_{\text{avg}}) = E_{\mathcal{D}}[h_{\text{avg}}] - \mu$$

Since h_{avg} is just the sample mean, we have:

$$E[h_{\text{avg}}] = E\left[\frac{1}{n}\sum_{i=1}^{n} Y_i\right] = \frac{1}{n}\sum_{i=1}^{n} E[Y_i]$$

Given that $E[Y_i] = \mu$, we get:

$$E[h_{\text{avg}}] = \frac{1}{n} \sum_{i=1}^{n} \mu = \mu$$

Thus, the bias of h_{avg} is:

$$\text{Bias}(h_{\text{avg}}) = \mu - \mu = 0$$

Variance Calculation:
The variance of h_{avg} is given by:

$$\text{Var}(h_{\text{avg}}) = E\left[(h_{\text{avg}} - E[h_{\text{avg}}])^2\right] = \text{Var}\left(\frac{1}{n} \sum_{i=1}^{n} Y_i\right)$$

Since the Y_i's are independent and identically distributed (i.i.d.), the variance of the sum is:

$$\text{Var}\left(\frac{1}{n} \sum_{i=1}^{n} Y_i\right) = \frac{1}{n^2} \sum_{i=1}^{n} \text{Var}(Y_i) = \frac{1}{n^2} \cdot n \cdot \sigma^2 = \frac{\sigma^2}{n}$$

Thus, the variance of h_{avg} is:

$$\text{Var}(h_{\text{avg}}) = \frac{\sigma^2}{n}$$

Conclusion: The bias of the sample average estimator h_{avg} is 0. And the variance of h_{avg} is $\frac{\sigma^2}{n}$.

Question 1c

The ℓ_2-regularized mean estimator $h_\lambda(\mathcal{D})$ is defined as the solution to the following minimization problem:

$$h_\lambda(\mathcal{D}) = \arg\min_{m \in R} \left[\frac{1}{n} \sum_{i=1}^{n} (Y_i - m)^2 + \lambda m^2\right]$$

Expand the Objective Function
First, expand $(Y_i - m)^2$ to get:

$$(Y_i - m)^2 = Y_i^2 - 2Y_i m + m^2$$

Thus, the objective function becomes:

$$f(m) = \frac{1}{n} \sum_{i=1}^{n} \left(Y_i^2 - 2Y_i m + m^2\right) + \lambda m^2$$

Simplifying:

$$f(m) = \frac{1}{n} \sum_{i=1}^{n} Y_i^2 - 2m \cdot \frac{1}{n} \sum_{i=1}^{n} Y_i + m^2 + \lambda m^2$$

$$f(m) = \frac{1}{n}\sum_{i=1}^{n} Y_i^2 - 2m \cdot \frac{1}{n}\sum_{i=1}^{n} Y_i + (1+\lambda)m^2$$

Taking the derivative with respect to m:

$$\frac{d}{dm} f(m) = -2 \cdot \frac{1}{n}\sum_{i=1}^{n} Y_i + 2(1+\lambda)m$$

Setting the derivative equal to zero to find the minimum:

$$-2 \cdot \frac{1}{n}\sum_{i=1}^{n} Y_i + 2(1+\lambda)m = 0$$

Simplifying:

$$m(1+\lambda) = \frac{1}{n}\sum_{i=1}^{n} Y_i$$

Solving for m:

$$m = \frac{1}{n}\sum_{i=1}^{n} Y_i \cdot \frac{1}{1+\lambda}$$

Thus, the explicit formula for the ℓ_2-regularized mean estimator is:

$$h_\lambda(\mathcal{D}) = \frac{h_{\text{avg}}(\mathcal{D})}{1+\lambda}$$

Question 1d

We are now asked to compute the bias and variance of the ℓ_2-regularized estimator $h_\lambda(\mathcal{D})$.

From part (c), we found the expression for the regularized estimator:

$$h_\lambda(\mathcal{D}) = \frac{h_{\text{avg}}(\mathcal{D})}{1+\lambda}$$

Where $h_{\text{avg}}(\mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n} Y_i$, which is the sample mean.

Bias of $h_\lambda(\mathcal{D})$

The bias is given by:

$$\text{Bias}(h_\lambda) = E[h_\lambda(\mathcal{D})] - \mu$$

Now, we calculate the expectation of $h_\lambda(\mathcal{D})$:

$$E[h_\lambda(\mathcal{D})] = E\left[\frac{h_{\text{avg}}(\mathcal{D})}{1+\lambda}\right] = \frac{1}{1+\lambda} E[h_{\text{avg}}(\mathcal{D})]$$

Since $E[h_{\text{avg}}(\mathcal{D})] = \mu$, we know:

$$E[h_\lambda(\mathcal{D})] = \frac{\mu}{1+\lambda}$$

Now, the bias becomes:

$$\text{Bias}(h_\lambda) = \frac{\mu}{1+\lambda} - \mu$$

$$\text{Bias}(h_\lambda) = \mu\left(\frac{1}{1+\lambda} - 1\right)$$

$$\text{Bias}(h_\lambda) = -\mu \cdot \frac{\lambda}{1+\lambda}$$

Variance of $h_\lambda(\mathcal{D})$

The variance is given by:

$$\text{Var}(h_\lambda) = E\left[(h_\lambda(\mathcal{D}) - E[h_\lambda(\mathcal{D})])^2\right]$$

We know that:

$$h_\lambda(\mathcal{D}) = \frac{h_{\text{avg}}(\mathcal{D})}{1+\lambda}$$

and

$$E[h_\lambda(\mathcal{D})] = \frac{\mu}{1+\lambda}$$

Thus:

$$\text{Var}(h_\lambda) = E\left[\left(\frac{h_{\text{avg}}(\mathcal{D})}{1+\lambda} - \frac{\mu}{1+\lambda}\right)^2\right]$$

Factoring out $\frac{1}{1+\lambda}$:

$$\text{Var}(h_\lambda) = \frac{1}{(1+\lambda)^2} E\left[(h_{\text{avg}}(\mathcal{D}) - \mu)^2\right]$$

Since $E\left[(h_{\text{avg}}(\mathcal{D}) - \mu)^2\right] = \frac{\sigma^2}{n}$ (the variance of the sample mean), we have:

$$\text{Var}(h_\lambda) = \frac{1}{(1+\lambda)^2} \cdot \frac{\sigma^2}{n}$$

Final Expressions

- **Bias:**

$$\text{Bias}(h_\lambda) = -\mu \cdot \frac{\lambda}{1+\lambda}$$

- **Variance:**

$$\text{Var}(h_\lambda) = \frac{\sigma^2}{n(1+\lambda)^2}$$

Question 1e

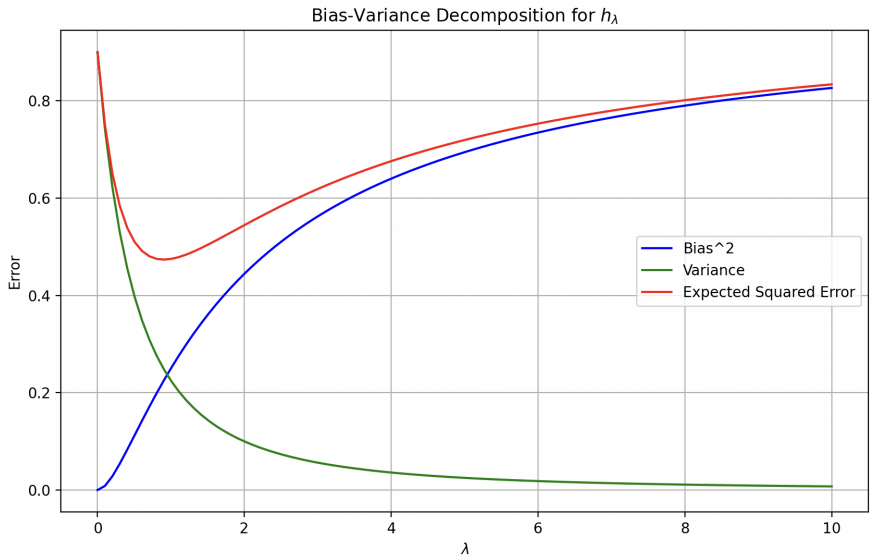


Figure 1: Visualization

Question 1f

The visualization illustrates the relationship between the regularization parameter λ and the bias-variance decomposition of the estimator h_λ, with the squared bias, variance, and expected squared error plotted against λ. As λ increases, the impact of regularization becomes stronger, which directly affects the bias and variance components and, consequently, the expected squared error.

When λ is close to zero, there's minimal regularization, so the estimator h_λ can closely match the data. This results in a low squared bias because the estimator remains close to the true mean of the data. However, this flexibility comes with high variance, as the model becomes more sensitive to variations in the data, increasing the risk of overfitting. In this low λ region, the high variance significantly raises the overall expected squared error.

As λ increases, the regularization term $\lambda|m|^2$ becomes more dominant, making the model simpler and less flexible. This reduction in flexibility lowers the variance as the model becomes less sensitive to fluctuations in the data, resulting in a more stable estimate. However, this comes at the cost of an increase in squared bias, since the regularization causes the estimator to deviate from the true mean, leading to underfitting. The expected squared error is a combination of this rising bias and decreasing variance.

The expected squared error curve reaches its minimum at an intermediate value of λ, where the reduction in variance is sufficient to offset the increase in

bias. At this point, the bias-variance trade-off is optimized, providing the best balance between model complexity and stability. According to the graph, this optimal λ value appears to be around 1 to 2. Beyond this range, as λ continues to increase, the squared bias becomes the dominant contributor to the expected squared error, causing it to rise again.

In summary, the expected squared error is minimized when there is an ideal balance between the bias and variance. At lower values of λ, variance dominates, leading to a higher error, while at higher values of λ, bias dominates, causing the error to rise. The optimal λ effectively balances these components, resulting in a model that avoids both overfitting and underfitting, achieving the lowest possible prediction error.

Question 2a:

We are given a dataset $\{Z_1, Z_2, \ldots, Z_N\}$, where each Z_i is independently drawn from a Poisson distribution with an unknown parameter λ. The probability mass function of the Poisson distribution is given by:

$$p(Z = k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k \in \{0, 1, 2, \ldots\}$$

Likelihood Function

The likelihood function for λ given the data $\{Z_1, Z_2, \ldots, Z_N\}$ is:

$$L(\lambda; Z_1, Z_2, \ldots, Z_N) = \prod_{i=1}^{N} p(Z_i; \lambda) = \prod_{i=1}^{N} \frac{\lambda^{Z_i} e^{-\lambda}}{Z_i!}$$

Log-Likelihood Function

The log-likelihood function is the natural logarithm of the likelihood function:

$$\log L(\lambda; Z_1, Z_2, \ldots, Z_N) = \sum_{i=1}^{N} \log \left(\frac{\lambda^{Z_i} e^{-\lambda}}{Z_i!} \right)$$

Simplifying:

$$\log L(\lambda; Z_1, Z_2, \ldots, Z_N) = \sum_{i=1}^{N} \left(Z_i \log(\lambda) - \lambda - \log(Z_i!) \right)$$

Since $\log(Z_i!)$ does not depend on λ, it can be ignored for maximization purposes. So we have:

$$\log L(\lambda; Z_1, Z_2, \ldots, Z_N) = \sum_{i=1}^{N} Z_i \log(\lambda) - N\lambda$$

Differentiate the Log-Likelihood

To find the maximum likelihood estimate of λ, we take the derivative of the log-likelihood with respect to λ and set it equal to zero:

$$\frac{d}{d\lambda} \log L(\lambda) = \frac{\sum_{i=1}^{N} Z_i}{\lambda} - N = 0$$

Solve for λ
Solving for λ:

$$\frac{\sum_{i=1}^{N} Z_i}{\lambda} = N \quad \Rightarrow \quad \lambda = \frac{1}{N} \sum_{i=1}^{N} Z_i$$

Thus, the Maximum Likelihood Estimator (MLE) for λ is the sample mean:

$$\hat{\lambda} = \frac{1}{N} \sum_{i=1}^{N} Z_i$$

Question 2(b)

Define the Likelihood Function

For a dataset $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$, where $y^{(i)}$ represents the observed count and $x^{(i)}$ represents the input vector, the probability mass function (PMF) of the Poisson distribution for each observation is given by:

$$p(y^{(i)}|x^{(i)}; \mathbf{w}) = \frac{\exp(y^{(i)} \mathbf{w}^\top x^{(i)}) \cdot \exp(-\exp(\mathbf{w}^\top x^{(i)}))}{y^{(i)}!}$$

Write the Log-Likelihood Function

The log-likelihood for the entire dataset is the sum of the log of the PMF over all observations:

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{N} \left[y^{(i)} \mathbf{w}^\top x^{(i)} - \exp(\mathbf{w}^\top x^{(i)}) - \log(y^{(i)}!) \right]$$

The term $\log(y^{(i)}!)$ does not depend on $\mathbf{w}$ and thus can be ignored when optimizing the log-likelihood.

Simplify the Log-Likelihood for Optimization

Simplifying the log-likelihood by ignoring constants, we get:

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{N} \left[y^{(i)} \mathbf{w}^\top x^{(i)} - \exp(\mathbf{w}^\top x^{(i)}) \right]$$

Define the Negative Log-Likelihood (Loss Function)

Since maximum likelihood estimation involves maximizing the log-likelihood, we can alternatively minimize the negative log-likelihood:

$$\text{Loss}(\mathbf{w}) = -\sum_{i=1}^{N} \left[y^{(i)} \mathbf{w}^\top x^{(i)} - \exp(\mathbf{w}^\top x^{(i)}) \right]$$

By simplifying, we get the final answer as:

$$\text{Loss}(\mathbf{w}) = \sum_{i=1}^{N} \left[\exp(\mathbf{w}^\top x^{(i)}) - y^{(i)} \mathbf{w}^\top x^{(i)} \right]$$

Question 2c

Define the Weighted Least Squares Cost Function

The weighted least squares problem is defined as:

$$\mathbf{w}^* \leftarrow \arg\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^{N} a_i^{(i)} \left(y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)} \right)^2,$$

where $a_i^{(i)}$ represents the weights associated with each data point $(x^{(i)}, y^{(i)})$.

Rewrite in Matrix Form

Let $\mathbf{X}$ be the design matrix containing the input vectors $\mathbf{x}^{(i)}$ as rows, and $\mathbf{y}$ be the vector of target values $y^{(i)}$. Let $\mathbf{A}$ be a diagonal matrix with the weights $a_i^{(i)}$ along its diagonal:

$$\mathbf{A} = \text{diag}(a_1^{(1)}, a_2^{(2)}, \ldots, a_N^{(N)}).$$

The cost function can then be written in matrix form as:

$$\text{Cost}(\mathbf{w}) = \frac{1}{2} (\mathbf{y} - \mathbf{X}\mathbf{w})^\top \mathbf{A} (\mathbf{y} - \mathbf{X}\mathbf{w}).$$

Take the Gradient of the Cost Function

To find the minimum, take the derivative of the cost function with respect to $\mathbf{w}$ and set it equal to zero:

$$\nabla_{\mathbf{w}} \text{Cost}(\mathbf{w}) = -\mathbf{X}^\top \mathbf{A} (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0.$$

Solve for $\mathbf{w}^*$

To solve for $\mathbf{w}$, we reorganize the equation:

$$\mathbf{X}^\top \mathbf{A}\mathbf{y} = \mathbf{X}^\top \mathbf{A}\mathbf{X}\mathbf{w}.$$

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{A}\mathbf{X})^{-1} \mathbf{X}^\top \mathbf{A}\mathbf{y}.$$

Final Answer

The solution to the weighted least squares problem is:

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\top \mathbf{A}\mathbf{y},$$

where $\mathbf{X}$ is the design matrix, $\mathbf{A}$ is a diagonal matrix containing the weights, and $\mathbf{y}$ is the vector of target values.

Question 2d

Understand the Weight Function

The weight function for each data point in locally weighted least squares is given by:

$$a^{(i)}(x) = \frac{\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}^{(i)}\|^2}{2\tau^2}\right)}{\sum_{j=1}^{N} \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}^{(j)}\|^2}{2\tau^2}\right)}.$$

This function assigns a weight to each training example based on its distance from the query point $\mathbf{x}$, where τ controls how strong this weighting effect is.

Behavior as $\tau \to 0$

When τ approaches 0, the exponent in the weight function becomes very large for training points that are close to $\mathbf{x}$ and very small for those that are far away. Thus, as $\tau \to 0$:

- The weights $a^{(i)}(x)$ become concentrated on the training point $\mathbf{x}^{(i)}$ that is closest to $\mathbf{x}$.
- The locally weighted regression will heavily rely on the closest training point, making the prediction essentially the same as the response of that nearest neighbor.
- This is similar to a nearest neighbor regression approach where the prediction is influenced almost exclusively by the closest point.

Behavior as $\tau \to \infty$

When τ approaches ∞, the exponent in the weight function becomes small for all distances, which means:

- The weights $a^{(i)}(x)$ become more uniform, meaning that all training points contribute equally to the regression.
- As τ increases, the locally weighted regression starts to behave more like ordinary least squares (OLS) regression, where every data point has the same influence, regardless of its distance.
- This results in a global linear fit across all data points rather than a localized fit around the query point.

Disadvantage in Terms of Computational Complexity

Locally weighted least squares regression requires calculating a weight for each training point for every query point $\mathbf{x}$, which means:

- The algorithm has to compute the distance between the query point and each training point, resulting in a time complexity of $O(N)$ for each query.
- Since this process is repeated for every prediction, the overall complexity becomes $O(N^2)$ for N queries, making it computationally expensive for large datasets.
- In contrast, ordinary least squares (OLS) regression has a complexity of $O(N^2)$ for training due to the matrix inversion, but then it only takes $O(1)$ time to make predictions after the model is fitted.

Conclusion

1. **As** $\tau \to 0$: The algorithm behaves like a nearest neighbor model, focusing on the training point closest to $\mathbf{x}$.
2. **As** $\tau \to \infty$: The algorithm behaves like ordinary least squares regression, with all points contributing equally.
3. **Disadvantage**: Locally weighted least squares has higher computational complexity, $O(N^2)$ for making predictions, compared to ordinary least squares, due to the need to calculate weights for each query.

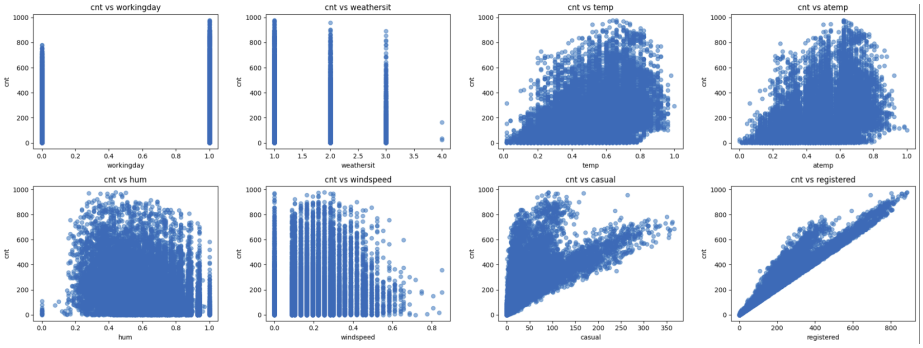
Question 3.1a

Please refer to code file.

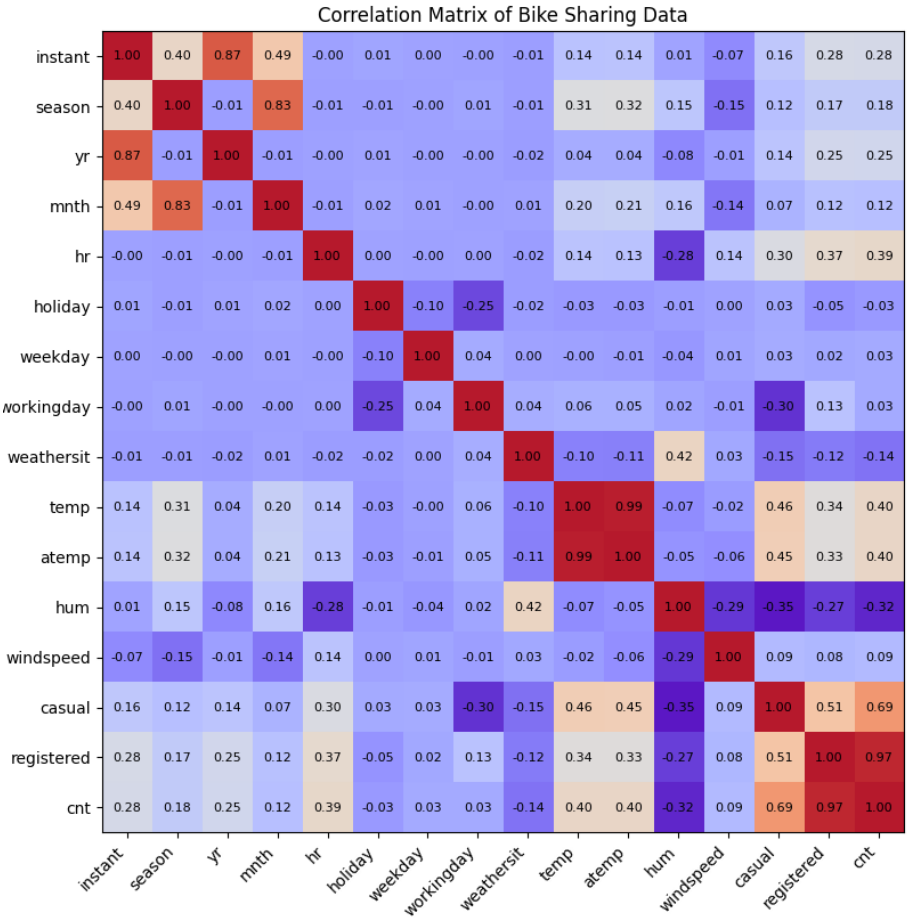
Question 3.1b

The dataset consists of 17,379 data points with 17 features, including both continuous and categorical variables. The continuous variables (*float64*) include *temp*, *atemp*, *hum*, *windspeed*, *casual*, *registered*, and the target variable *cnt*, which represents the total bike counts. Categorical or discrete variables (*int64*) include *instant*, *season*, *yr*, *mnth*, *hr*, *holiday*, *weekday*, *workingday*, and *weathersit*. Additionally, *dteday* (*object/string*) represents the date and time of each record.

Question 3.1c



Question 3.1d



Most positively correlated with 'cnt': cnt (correlation: 0.9721) ;Most negatively correlated with 'cnt': hum (correlation: -0.3229) ;Least correlated with 'cnt': weekday (correlation: 0.0269)

Question 3.1e

Please refer to code file.

Question 3.1f

Please refer to code file.

Question 3.2a

Please refer to code file.

Question 3.2b

R^2 score for the testing set is: 0.3768695152440047.

Question 3.2c

Please refer to code file.

Question 3.2d

R^2 score (updated) for the testing set after encoding is: 0.6817966649545326

Question 3.2e

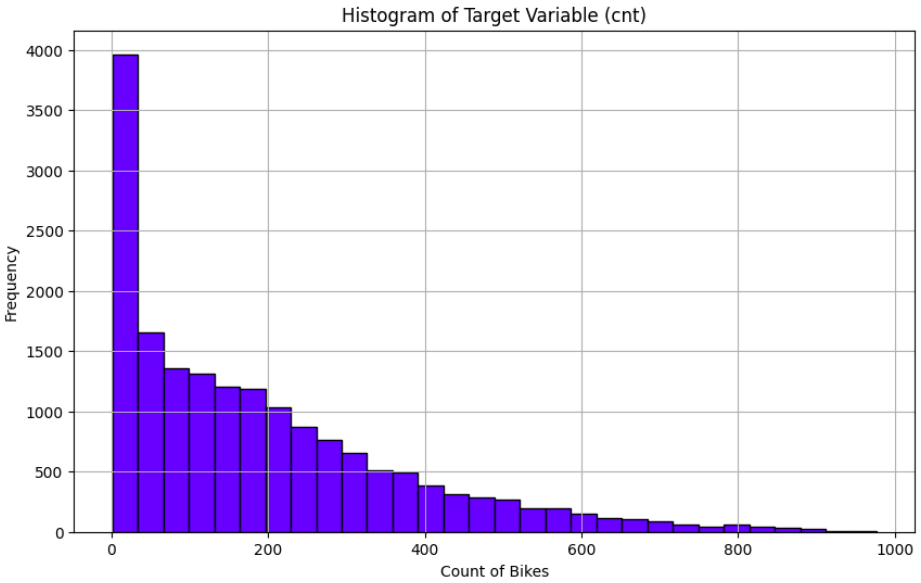
Please refer to code file.

Question 3.2f

R^2 score for Locally Weighted Regression with $\tau = 1$: 0.8401

For this evaluation, we used a reduced test set consisting of the first 200 samples from the full test set, as permitted by the problem statement. As τ approaches 0, the model behaves like k-nearest neighbors (k=1), with predictions being highly influenced by the nearest training points, resulting in very localized predictions and potential overfitting. Conversely, as τ increases, the model becomes more like ordinary least squares (OLS) regression, where all training points contribute almost equally, leading to a smoother, more generalized fit. With $\tau = 1$, the model strikes a balance between local sensitivity and global generalization, providing an R^2 score of 0.8401.

Question 3.2g



The target follows a Poisson distribution. The histogram of the target variable (‘cnt‘) exhibits a right-skewed shape, with a high frequency of lower counts and a long tail towards higher values. This distribution is characteristic of count data and aligns with the properties of a Poisson distribution, where counts are often skewed and concentrated around lower values.

Question 3.2h

Please refer to code file.

Question 3.2i

R score for the testing set after encoding is: 0.8004934605372034.

Question 3.2j

Analysis of Feature Weights for Linear and Poisson Regression

```
Final weights for Linear Regression:
[  20.86263432   83.95015855  -16.40567981   16.23995431  258.88030375
  -88.76921633  -28.82720772  -30.68178659   12.36671697    2.54584195
   36.63186199    3.140151      4.01023771   10.06138103    0.61031588
   10.93950716   -7.166017    -21.82870696   -3.48447965   26.51893339
   11.71620429  -10.09789072   -3.5570018  -120.87674216 -139.06714585
 -148.45460157 -156.6766717  -159.69337307 -142.43987977  -84.28057816
   53.05931373  182.13605622   41.1092887   -14.48846878   12.09843646
   44.81275751   40.9857761    20.7873331    40.70195348   96.70550286
  254.69868054  222.91320726  109.29207238   30.47876753  -17.26287953
  -51.8251018   -93.85106915    5.15890024   -2.06052893   -2.92439302
    3.00408168   -0.50803004    2.32314481   15.86945958   44.66947584
   33.43354028  -19.0384036   -38.2019782 ]

Final weights for Poisson Regression:
[ 2.31764052e+00  3.04001572e-01 -9.02126202e-02  2.24089320e-02
  6.18675580e-01 -2.09772779e-01  9.50088418e-03  9.84864279e-02
  2.98967811e-01  2.04105985e-01  5.01440436e-01  2.14542735e-01
  2.07610377e-01  2.01216750e-01  2.04438632e-01  2.03336743e-01
  1.14940791e-01  9.79484174e-02  2.03130677e-01  1.91459043e-01
  7.44701018e-02  1.06536186e-01  1.08644362e-01  9.25783498e-02
 -1.57730245e+00 -1.61454366e+00 -1.89954241e+00 -2.00187184e+00
 -1.58467221e+00  5.14693588e-01  8.01549474e-01  9.03781625e-01
  7.91122143e-01  6.80192035e-01  6.96520879e-01  7.01563490e-01
  7.12302907e-01  7.12023798e-01  6.96126732e-01  7.96976972e-01
  8.89514470e-01  8.85799821e-01  7.82937298e-01  7.19507754e-01
  6.94903478e-01  5.95619257e-01  4.87472046e-01 -9.22250964e-02
 -1.61389785e-02 -2.12740280e-03 -8.95466561e-03  3.86902498e-03
 -5.10805138e-03 -1.18063218e-02  9.97181777e-02  1.04283319e-01
 -9.93348278e-02 -2.29697528e+00]

Most significant feature for Linear Regression: temp
Least significant feature for Linear Regression: weekday_3

Most significant feature for Poisson Regression: weathersit_3
Least significant feature for Poisson Regression: weekday_1
```

Linear Regression:

- **Most significant feature:** `temp`
- **Least significant feature:** `weekday_3` (1-hot index 3)

Poisson Regression:

- **Most significant feature:** `weathersit_3` (1-hot index 3)
- **Least significant feature:** `weekday_1` (1-hot index 1)

Rationale of the Significance:

- The significance of each feature is determined by the magnitude of its coefficient. Larger absolute values indicate a greater influence on the target variable (`cnt`). Features with larger coefficients will have a more pronounced effect on the prediction results, either positively or negatively.
- **Categorical Features:** For one-hot encoded variables, each category is treated as a binary feature. The most significant categorical feature is identified by considering the absolute value of its corresponding weight.

Visualizing weight contributions for Locally Weighted Linear Regression is not meaningful because the weights vary dynamically for each query point, unlike in global models like ordinary least squares or Poisson regression. In Locally Weighted Linear Regression (LWR), the model learns a different set of weights for each prediction point based on its proximity to the training points. Unlike global models like Linear or Poisson Regression, which have a fixed set of coefficients, LWR assigns higher weights to nearby points and lower weights to distant ones. This makes the weight contributions dynamic and dependent on each query point. As a result, it is not possible to produce a single set of interpretable weights that apply across the entire dataset, making visualizations of weight contributions impractical.