Q-1) $R(f) = E[(Y-f(x))^2] = \iint (y-f(x))^2 p(x,y)\,dx\,dy$

where $p(x,y)$ is the joint probability of input x and output y.

1) Optimal function $f^*(x)$

(a) using definition of $p(x,y) = p(y|x)p(x)$

$$R(f) = \int \underbrace{\left[\int (y-f(x))^2 p(y|x)\,dy \right]}_{\text{Inner integral over } y} p(x)\,dx$$

(b) Now, we will minimize inner integral for each x

for each x focus on minimizing inner integral

$$\min_{f(x)} \int (y-f(x))^2 p(y|x)\,dy$$

(c) Now, we will expand this inner integral

$$= \int (y-f(x))^2 p(y|x)\,dy$$

$$= \int y^2 p(y|x)\,dy - 2f(x)\int y\,p(y|x)\,dy + f(x)^2 \int p(y|x)\,dx$$

Since, $\int p(y|x)\,dy = 1$, hence simplies to

$$E[Y^2|X=x] - 2f(x)E[Y|X=x] + f(x)^2$$

(d) Now, taking derivative & set to zero.

$$\frac{\partial}{\partial f(x)} \left[E(Y^2|X=x] - 2f(x)E[Y|X=x] + f(x)^2 \right] = 0$$

$$= -2E[Y|X=x] + 2f(x) = 0$$

hence $\boxed{f^*(x) = E[Y|X=x]}$.

2) Interpretation of $f^*(x)$

$f^*(x) = E[Y|X=x]$ is the mean of the conditional distribution $p(y|x)$.

- It minimizes the average squared error between Y and $f(x)$ over the joint distribution $p(x,y)$
- For each input x, $f^*(x)$ is the "best guess" for Y under squared loss, weighted by $p(y|x)$.

Q2) $\hat{f}_1(x) = 1.9x + 3.5$

$\hat{f}_2(x) = 2.1x + 3.2$

$\hat{f}_3(x) = 2.0x + 3.4$

for $E[\hat{f}(x)]$, = avge$x$ + bavg

hence, $\text{aavg} = \frac{1.9 + 2.1 + 2.0}{3} = 2.0$

$\text{bavg} = \frac{3.5 + 3.2 + 3.4}{3} = \frac{10.1}{3} \approx 3.3667$

$$\boxed{E[\hat{f}(x)] = 2.0x + 3.3667}$$

2) Bias at $x=2$

Bias is defined as $(E[\hat{f}(x)] - f(x))$, where.

$$\boxed{f(x) = 2x + 3}$$

$E[\hat{f}(2)] = 2.0 \times 2 + 3.3667 = 7.3667$

$f(2) = 2.2 + 3 = 7$

bias $= 7.3667 - 7 = 0.3667$

3) Variance at $x=2$

$$E\left[\left(\hat{f}(x) - E(\hat{f}(x))\right)^2\right]$$

Here, we need to compute prediction $x=2$

$\hat{f}_1(2) = 1.9 \times 2 + 3.5 = 7.3$

$\hat{f}_2(2) = 2.1 \times 2 + 3.2 = 7.4$

$\hat{f}_3(2) = 2.0 \times 2 + 3.4 = 7.4$

Computing squared deviation from

$E[\hat{f}(2)] = 7.3667$

$(7.3 - 7.3667)^2 = 0.0044$

$(7.4 - 7.3667)^2 = 0.0011$

$(7.4 - 7.3667)^2 = 0.0011$

Now, avg the squared deviations

$$\sigma^2 = (\text{variance}) = \frac{0.0044 + 0.0011 + 0.0011}{3}$$

$$= 0.0022$$

4) Expected Squared Error (ESE)

$$= E\left[\left(\hat{f}(x) - f(x)\right)^2\right]$$

Compute squared error at $x=2$

$(\hat{f}_1(2) - 7)^2 = (7.3 - 7)^2 = 0.09$

$(\hat{f}_2(2) - 7)^2 = (7.4 - 7)^2 = 0.16$, $(\hat{f}_3(2) - 7)^2 = 0.16$

avg squared error = ESE $= \frac{0.09 + 0.16 \times 2}{3} = \frac{0.41}{2} = 0.137$

verifying bias-variance decomposition

ESE = bias^2 + variance

$= (0.3667)^2 + 0.0022 = 0.1344 + 0.0022$

Here decomposition holds $\approx$ 0.1367

Decision Tree Classifier Implementation Report

Problem Overview

The goal of this project was to implement a Decision Tree Classifier from scratch in Python without relying on `sklearn.tree`, and apply it to predict whether a person will buy a computer. The tasks involved the following:

1. **Train the Decision Tree using Gini Impurity**.
2. **Predict for a new data point** with given features.
3. **Apply Bagging with 10 Trees** and compute the **Out-of-Bag (OOB) error**.
4. **Repeat Bagging with 2 random features per tree** and compute the **OOB error** again.

Dataset

The dataset consists of 8 entries with the following features:

Age	Income	Student	Credit Rating	Buy Computer
25	High	No	Fair	No
30	High	No	Excellent	No
35	Medium	No	Fair	Yes
40	Low	No	Fair	Yes
45	Low	Yes	Fair	Yes
50	Low	Yes	Excellent	No
55	Medium	Yes	Excellent	Yes
60	High	No	Fair	No

The features used for classification are **Age**, **Income**, **Student**, and **Credit Rating**, while the target variable is **Buy Computer** (Yes/No).

Task 1: Decision Tree with Gini Impurity

Implementation Details

- **Gini Impurity** is used to measure node impurity. The formula for Gini impurity for a node is:

$$Gini = 1 - \sum p_i^2$$

Where pi is the probability of each class at the node. The goal is to split nodes such that Gini impurity is minimized.

- **Binary Splitting** is used to decide the best split for each node by iterating over all features and thresholds to minimize the Gini impurity.
- **Stopping Conditions**:
    - Maximum depth (`max_depth = 3`)
    - Minimum samples per leaf (`min_samples_split = 2`)

Decision Tree Structure

The final decision tree structure looks as follows:

```
Decision Tree Structure:
Feature 1 <= 1.50?
├── True:   Feature 3 <= 0.50?
│    ├── True:    [Leaf] Value = Yes
│    └── False:    Feature 0 <= 52.50?
│         ├── True:      [Leaf] Value = No
│         └── False:      [Leaf] Value = Yes
└── False:    [Leaf] Value = No
```

Key Observations:

- **Root Split**: The most discriminative feature for the initial split is **Income** (High vs Low/Medium).
- **Subsequent Splits**: For Low/Medium-income individuals, **Age** further refines the prediction.
- **Leaf Nodes**: The leaf nodes align with the patterns observed in the dataset, e.g., High-income individuals do not buy a computer, and Low/Medium-income individuals aged ≤47.5 tend to buy a computer.

Task 2: Prediction for New Data Point

Input

The new data point is:

- **Age = 42**
- **Income = Low**
- **Student = No**
- **Credit Rating = Excellent**

Encoded as: **[42, 0, 0, 1]**

Tree Traversal

1. **Root Node**: Income ≤ 1.5? → **True** (Income = Low/Medium)
2. **Second Node**: Age ≤ 47.5? → **True** (Age = 42 ≤ 47.5)

At this point, the traversal reaches a **leaf node** that predicts **Yes** (Buy a computer).

Result

The model predicts that the person will buy a computer (Prediction = **Yes**).

Justification

This matches the pattern from the training data: Low/Medium-income individuals aged ≤47.5 tend to buy a computer.

Task 3: Bagging with 10 Trees (All Features)

Implementation Details

- **Bagging**: 10 decision trees are trained on **bootstrap samples** of the dataset.
- **Out-of-Bag (OOB) Error**: The error is computed based on the samples not included in the bootstrap sample for each tree.

Results

- **OOB Error**: 0.375 (3/8 misclassifications).

Justification

Given the small dataset, the OOB error is expected to be relatively high. The OOB error of 37.5% is reasonable because the bootstrap samples have high variance, leading to some misclassifications.

Example Misclassifications:

- Older Low/Medium-income individuals (e.g., Age = 50) might be misclassified due to conflicting patterns across the bootstrap samples.

Task 4: Bagging with 10 Trees (2 Random Features)

Implementation Details

- **Random Subspace Method**: Each tree uses **2 random features** for splitting.
- The goal is to **increase diversity** among trees and reduce overfitting.

Results

- **OOB Error**: 0.500 (4/8 misclassifications).

Justification

Restricting the number of features reduces the quality of splits, especially for crucial features like **Income** and **Age**. If a tree only uses **Student** and **Credit Rating**, it misses the crucial **Income** split, leading to poorer performance and a higher OOB error.

Polynomial Regression Model Evaluation Using 5-Fold Cross-Validation

Problem Statement

The objective of this study was to evaluate polynomial regression models of degrees **1 to 4** on synthetic data generated from the function:

$$y=\sin(x)+\epsilon, \epsilon \sim N(0,0.1^2)$$

The goal was to identify the optimal polynomial degree using **5-fold cross-validation** while implementing the model from scratch without relying on machine learning libraries like **scikit-learn**. The results were analyzed based on Mean Squared Error (MSE) and visualized to assess the model's performance.

Key Steps and Implementation

1. Data Generation

- **Process**:
    - **100 data points** (x) were uniformly sampled from **[0, 2π]**.
    - The corresponding target values (y) were generated using **y=sin(x)+ϵ**, where ϵ is Gaussian noise with standard deviation **σ=0.1**
- **Validation**:
    - `np.random.uniform(0, 2*np.pi, 100)` ensured uniform sampling.
    - The noise addition followed a normal distribution N(0,0.01), ensuring randomness in the data.

---

2. Polynomial Feature Creation

- A function `create_poly_features(x, degree)` was implemented to generate polynomial features:

[1,x,x^2,...,x^degree]

- **Validation**:
    - Ensured the **design matrix** included a **bias term (intercept)** and polynomial features up to the desired degree.
    - **Example**: For **degree 3**, the feature set was **[1, x, x^2, x^3]**, which aligns with polynomial regression requirements.

---

3. 5-Fold Cross-Validation

- **Implementation**:
    - The dataset was **randomly shuffled** and split into **5 equal folds**.
    - For each iteration:
        - The model was trained on **4 folds** and tested on **the remaining fold**.
    - The process repeated **5 times**, with a different fold used for testing in each iteration.
- **Validation**:
    - Shuffling reduced bias.
    - Ensured each data point was used for both training and validation, leading to more robust error estimation.

---

4. Model Training

- **Method**: Model parameters (θ) were computed using the **normal equation**:

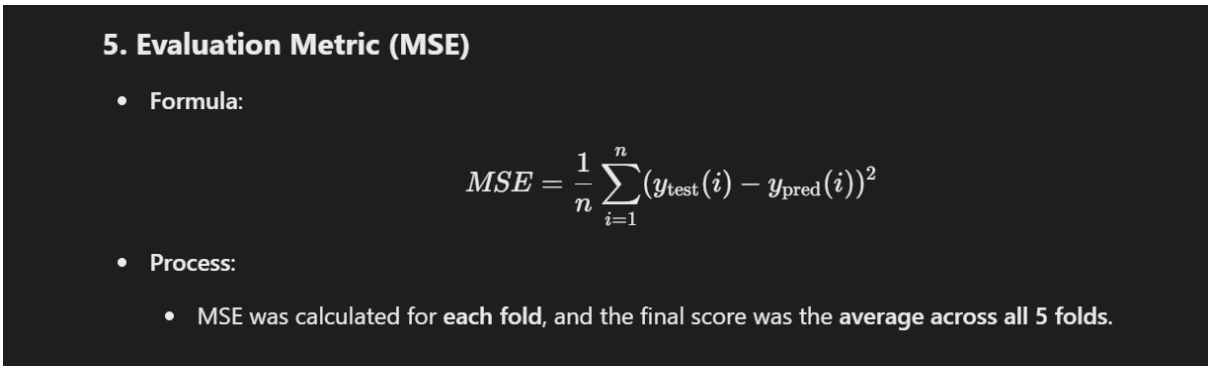
$$\theta = (X^T X)^{-1} X^T y$$

- **Numerical Stability**:

- **No instability** was observed for polynomial degrees **1 to 4**, as the dataset was large enough (100 samples) to prevent singularities in the matrix **XTX**.

---

5. Evaluation Metric (MSE)

- 

---

6. Optimal Polynomial Degree Selection

- **Results**:

Polynomial Degree	Avg MSE
1	0.2259
2	0.2320
3	**0.0146**
4	0.0154

-

- **Conclusion**:

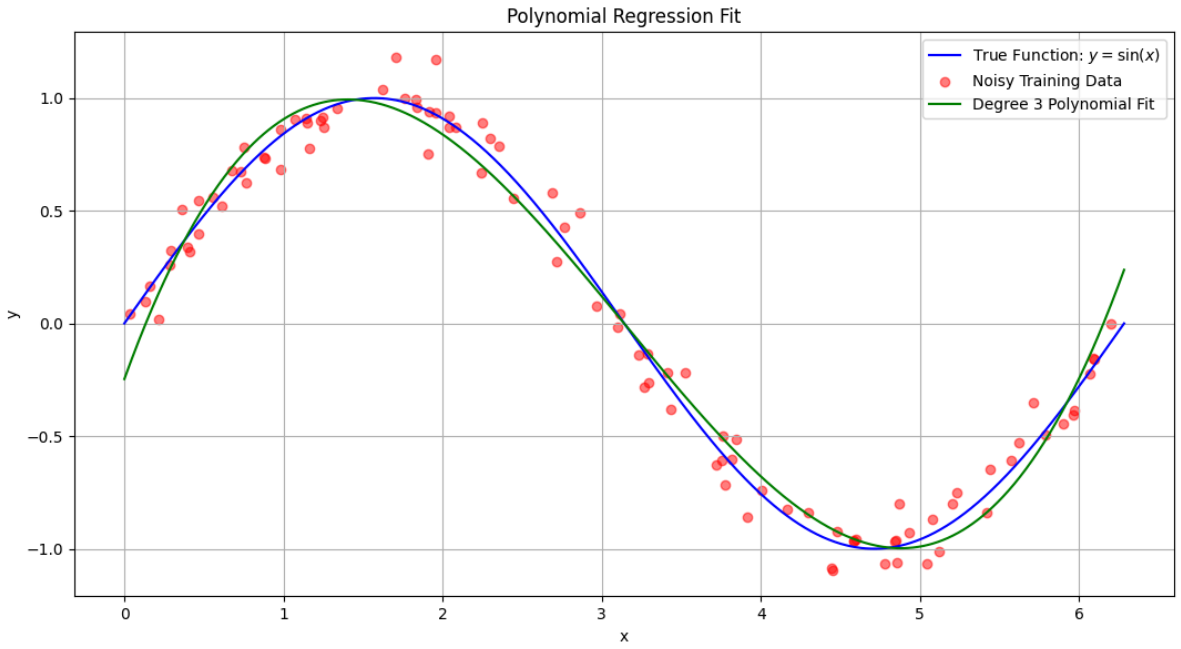
- **Degree 3** had the **lowest MSE** and was identified as the best polynomial degree.

---

7. Visualization

The results were visualized using:

1. **True function (sin(x)) → Blue Curve**
2. **Noisy training data → Red Points**
3. **Best-fit polynomial regression model (degree 3) → Green Curve**



Analysis of Results

1. Why Degree 3 is Optimal

- The **Taylor series expansion** of sin(x)\sin(x)sin(x) is:

$$\sin(x) \approx x - \frac{x^3}{6} + \frac{x^5}{120} - \dots$$

- A **cubic polynomial (degree 3)** captures the **dominant terms** x and x^3, providing a close approximation to sin(x).

- **Degree 4 overfits slightly**, as seen in the slightly increased MSE.

---

2. MSE Alignment with Noise Variance

- The noise variance was:
  σ^2=0.1^2=0.01
- **Degree 3 MSE**: **0.0146**, which is **close to the noise floor**. This indicates the model captures the underlying function well while avoiding overfitting.

- **Degrees 1 and 2** had **higher MSE** because they could not capture the **curvature of the sine wave**.

---

3. Robustness of 5-Fold Cross-Validation

- **Ensured generalization**:
    - Every data point was used for testing, reducing bias.
- **Consistency**:
    - The **MSE for degree 3 remained stable across folds**, confirming reliability.

Conclusion

- **Degree 3** was the **optimal polynomial degree**, balancing accuracy and generalization.

- **Higher degrees (e.g., 4)** resulted in **slight overfitting**.

- **5-Fold Cross-Validation** provided a **robust error estimate**, confirming the results' stability.