October 10, 2025

Joseph DeLeonardis
UIN: 0820000866

CSCE-633

Machine Learning Homework #2

Problem 1: Information Gain

Given the training data:

X_1	X_2	Y
1	1	1
1	1	1
1	1	2
1	0	3
0	0	2
0	0	3

Part 1: Calculate Information Gain for both X_1 and X_2

Step 1: Calculate initial entropy $H(Y)$

Count the class distribution:

- Class 1: 2 instances
- Class 2: 2 instances
- Class 3: 2 instances
- Total: 6 instances

The entropy is calculated as:

$$\begin{aligned} H(Y) &= -\sum_{i=1}^{3} p(Y=i)\log_2 p(Y=i) \\ &= -\left(\frac{2}{6}\log_2\frac{2}{6} + \frac{2}{6}\log_2\frac{2}{6} + \frac{2}{6}\log_2\frac{2}{6}\right) \\ &= -3 \times \frac{1}{3}\log_2\frac{1}{3} \\ &= -\log_2\frac{1}{3} = \log_2 3 \\ &\approx 1.585 \end{aligned}$$

The 2/6 comes from the proportion of the data belonging to each of the 3 classes.

Calculate Information Gain for X_1

Split the data based on X_1:

When $X_1 = 1$ (4 instances):

- Class 1: 2, Class 2: 1, Class 3: 1
- Probabilities: $p(Y=1) = \frac{2}{4} = 0.5$, $p(Y=2) = \frac{1}{4} = 0.25$, $p(Y=3) = \frac{1}{4} = 0.25$

$$
\begin{aligned}
H(Y|X_1 = 1) &= -\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4}\right) \\
&= -\left(0.5 \times (-1) + 0.25 \times (-2) + 0.25 \times (-2)\right) \\
&= 0.5 + 0.5 + 0.5 = 1.5
\end{aligned}
$$

When $X_1 = 0$ (2 instances):

- Class 2: 1, Class 3: 1
- Probabilities: $p(Y = 2) = \frac{1}{2} = 0.5$, $p(Y = 3) = \frac{1}{2} = 0.5$

$$
\begin{aligned}
H(Y|X_1 = 0) &= -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) \\
&= -\left(0.5 \times (-1) + 0.5 \times (-1)\right) \\
&= 0.5 + 0.5 = 1.0
\end{aligned}
$$

Weighted conditional entropy:

$$
\begin{aligned}
H(Y|X_1) &= \frac{4}{6} \times H(Y|X_1 = 1) + \frac{2}{6} \times H(Y|X_1 = 0) \\
&= \frac{4}{6} \times 1.5 + \frac{2}{6} \times 1.0 \\
&= 1.0 + 0.333 = 1.333
\end{aligned}
$$

Information Gain:

$$
\begin{aligned}
IG(X_1) &= H(Y) - H(Y|X_1) \\
&= 1.585 - 1.333 \\
&= 0.252
\end{aligned}
$$

Calculate Information Gain for X_2
Split the data based on X_2:
When $X_2 = 1$ (3 instances):

- Class 1: 2, Class 2: 1, Class 3: 0
- Probabilities: $p(Y = 1) = \frac{2}{3}$, $p(Y = 2) = \frac{1}{3}$

$$
\begin{aligned}
H(Y|X_2 = 1) &= -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right) \\
&\approx -\left(\frac{2}{3} \times (-0.585) + \frac{1}{3} \times (-1.585)\right) \\
&\approx 0.390 + 0.528 = 0.918
\end{aligned}
$$

When $X_2 = 0$ (3 instances):

- Class 2: 1, Class 3: 2
- Probabilities: $p(Y = 2) = \frac{1}{3}$, $p(Y = 3) = \frac{2}{3}$

$$
\begin{aligned}
H(Y|X_2=0) &= -\left(\frac{1}{3}\log_2\frac{1}{3}+\frac{2}{3}\log_2\frac{2}{3}\right) \\
&\approx -\left(\frac{1}{3}\times(-1.585)+\frac{2}{3}\times(-0.585)\right) \\
&\approx 0.528+0.390=0.918
\end{aligned}
$$

Weighted conditional entropy:

$$
\begin{aligned}
H(Y|X_2) &= \frac{3}{6}\times H(Y|X_2=1)+\frac{3}{6}\times H(Y|X_2=0) \\
&= 0.5\times 0.918+0.5\times 0.918 \\
&= 0.918
\end{aligned}
$$

Information Gain:

$$
\begin{aligned}
IG(X_2) &= H(Y)-H(Y|X_2) \\
&= 1.585-0.918 \\
&= 0.667
\end{aligned}
$$

Summary:
The steps were taken to calculate H(Y) with the formula above and then both instances of X1 and X2 can be calculated to get the weighted entropy, they are then added and then subtracted from H(Y).

- $IG(X_1)\approx 0.252$
- $IG(X_2)\approx 0.667$

Part 2: First Split and Decision Tree

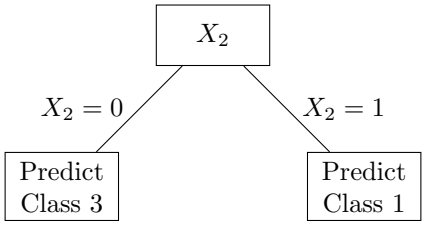
$IG(X_2) > IG(X_1)$, we use X_2 for the first split.
For the second level, we need to further split each branch:
Left branch ($X_2=0$):

- Data: $\{(1,0,3),(0,0,2),(0,0,3)\}$
- Classes: 2, 3, 3
- Majority class: 3

Right branch ($X_2=1$):

- Data: $\{(1,1,1),(1,1,1),(1,1,2)\}$
- Classes: 1, 1, 2
- Majority class: 1

The prediction is the majority class which results in a tree that looks like the following. **Decision Tree:**



Part 3: Classification for Test Example

Given: $X_1 = 0$ and $X_2 = 1$

Following the decision tree:

- Check X_2: Since $X_2 = 1$, we follow the right branch
- Looking back at the solution for part 1 denote that X2 is what we chose to draw the decision tree with and that is why we got right, X2 equals 1 when going right.

Problem 2: Entropy

Using the same training data from Problem 1.

Part 1: Calculate Conditional Entropy for both X_1 and X_2

Note: Conditional entropy is the same as the $H(Y|X)$ we calculated in Problem 1.The results are identical because conditional entropy in this context is the same as weighted average entropy of the child nodes after a split.

From Problem 1, we already calculated:

For X_1:

$$\begin{aligned} H(Y|X_1) &= \frac{4}{6} \times H(Y|X_1 = 1) + \frac{2}{6} \times H(Y|X_1 = 0) \\ &= \frac{4}{6} \times 1.5 + \frac{2}{6} \times 1.0 \\ &= 1.333 \end{aligned}$$

For X_2:

$$\begin{aligned} H(Y|X_2) &= \frac{3}{6} \times H(Y|X_2 = 1) + \frac{3}{6} \times H(Y|X_2 = 0) \\ &= 0.5 \times 0.918 + 0.5 \times 0.918 \\ &= 0.918 \end{aligned}$$

Summary:

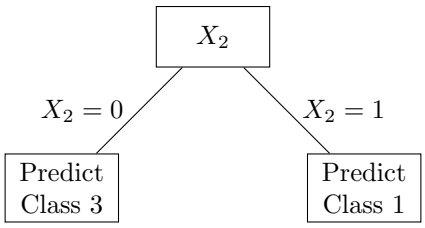
- $H(Y|X_1) \approx 1.333$
- $H(Y|X_2) \approx 0.918$

Part 2: First Split and Decision Tree

When using entropy as the splitting criterion, we choose the attribute with the **lowest conditional entropy** (which is equivalent to choosing the highest information gain).

Since $H(Y|X_2) < H(Y|X_1)$, we use X_2 for the first split.

This produces the same decision tree as in Problem 1:



Part 3: Classification for Test Example

Given: $X_1 = 0$ and $X_2 = 1$

Following the decision tree:

- Check X_2: Since $X_2 = 1$, we follow the right branch
- **Prediction: Class 1**

Note: The prediction is the same as in Problem 1.

Part A: Classification Tree

1. Data Processing and EDA

(a) Dataset Split

The Bank Marketing dataset was split into training and validation sets using an 80:20 ratio.

- Training set: 3,616 samples (80%)
- Validation set: 905 samples (20%)
- Total samples: 4,521

(b) Training Data Description

The Bank Marketing dataset contains client information possibly potential or active customers. The Target variable appears to be the answer to approval for an application such as a loan. The target variable is a binary yes or no outcome. Shown below it can be denoted that it was a heavily imbalanced data set that was challenging to work with for that reason.

Dataset Characteristics:

- **Number of features:** 16
- **Target variable:** y (binary: 0 = no subscription, 1 = subscription)
- **Feature types:** Mix of categorical and numerical features
- **Class distribution:** Highly imbalanced
  - Training: Class 0: 3,210 (88.77%), Class 1: 406 (11.23%)
  - Validation: Class 0: 790 (87.29%), Class 1: 115 (12.71%)

Feature List:

1. age: Client's age
2. job: Type of job (categorical)
3. marital: Marital status (categorical)
4. education: Education level (categorical)
5. default: Has credit in default? (binary)
6. balance: Average yearly balance
7. housing: Has housing loan? (binary)
8. loan: Has personal loan? (binary)
9. contact: Contact communication type (categorical)
10. day: Last contact day of the month
11. month: Last contact month of year (categorical)
12. duration: Last contact duration in seconds
13. campaign: Number of contacts during this campaign
14. pdays: Days since client was last contacted
15. previous: Number of contacts before this campaign
16. poutcome: Outcome of previous marketing campaign (categorical)

(c) Feature and Label Extraction

All features were extracted as the predictor variables (X), and the target variable "y" was extracted as the label. Categorical variables were encoded numerically using label encoding. The label "y" was mapped such that "no" = 0 and "yes" = 1.

(d) Histogram Analysis

Figure **??** shows the distribution of all variables in the training dataset.

Key Observations:

- **Age:** Heavily weighted on those ages 30-50 but shows all the way from 19-87.
- **month:** Lots of business during August but not much in Feb.
- **Duration:** Right-skewed distribution indicating most calls are short.
- **Campaign:** Heavily right-skewed, with most clients contacted 1-3 times during the campaign
- **Pdays:** Huge spike around 1.
- **Target Variable (y):** Severe class imbalance.
- **Categorical Features:** Several features such as job, marital, education, contact, month etc.

(e) Data processing

This was successfully completed and the class retrieved satisfactory results. The proper operation of this class is critical to ensure the right data is fed into the classification tree and boosting algorithms.

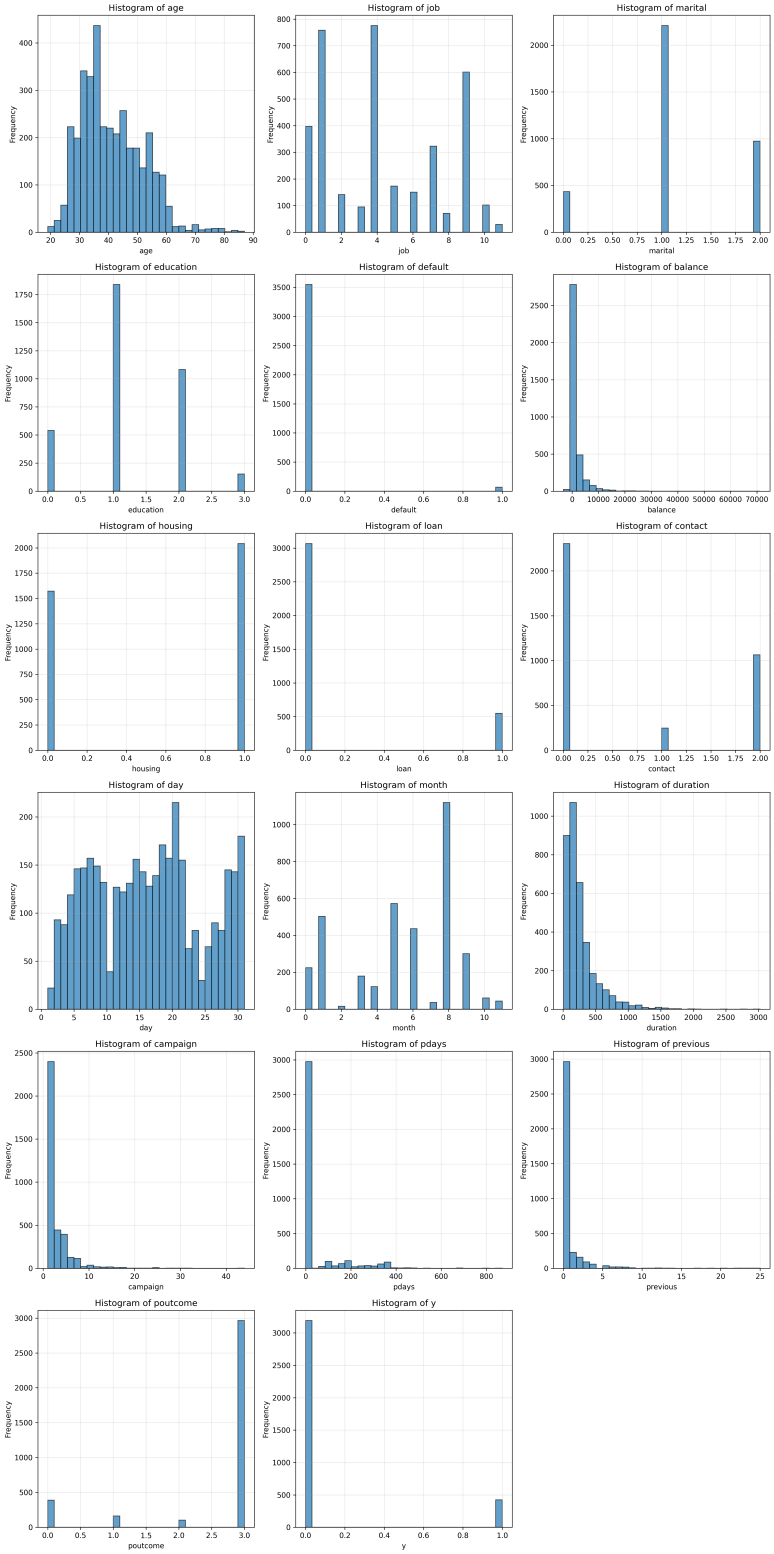


Figure 1: Histograms of all variables in the Bank Marketing dataset

2. Classification Tree Implementation

(a) Implementation Details

A decision tree classifier was implemented from scratch without using any machine learning libraries. The implementation includes using the Gini impurity to calculate a split criteria. A tree classification tree was then built with a max depth of 8 several values were experimented with but 8 led to the best results when being ran locally. There is other ways to find a split but this is the easiest way computationally.

The Gini impurity is calculated as:

$$\text{Gini}(y) = 1 - \sum_{i=1}^{C} p_i^2$$

where p_i is the proportion of class i in the node and C is the number of classes.

(b) Training and Validation Results

The classification tree was trained and the following results were confirmed:

Metric	**Training**	**Validation**
Accuracy	95.05%	88.62%
F1 Score (Binary)	0.7454	0.4718

(c) Performance Analysis

Achievement: The validation F1 score of 0.4718 exceeds the required threshold of 0.4, successfully meeting the assignment requirement.

Observations:

- The training accuracy (95.05%) is significantly higher than validation accuracy (88.62%).
- The gap between training F1 (0.7454) and validation F1 (0.4718) indicates the model has memorized some training data patterns and it did not generalize well. Several attempts to fix this were made but the strong imbalance in the data set made it a challenge.
- The maximum depth constraint (8) provided the best results.

Part B: Boosting with XGBoost

1. Model Training with L1 Regularization

An XGBoost model was trained using bootstrapping with 100 iterations to evaluate different L1 regularization parameter (α) values. The following alpha values were tested: [0.001, 0.01, 0.1, 1, 10, 100, 1000].

Bootstrapping Process:

- 100 bootstrap iterations led to 1 being the best alpha, however I also noticed that sometimes it was 1 and this depended on how other hyper-parameters were set.

Initializing the best model:

- The model was initialized and the best alpha was later added.

F1 score Results:

- Locally the best score was about 0.57, however when submitting to Gradescope it was roughly 0.51. This is explained in greater detail in the later sections.

Alpha (α)	Average F1 Score
0.001	0.4938
0.01	0.4936
0.1	0.4937
1	0.4924
10.0	**0.5213**
100.0	0.4429
1000.0	0.0978

Optimal Alpha: $\alpha = 10.0$ with an average F1 score of 0.5213

Analysis:

- Lower alpha values (0.001-1) showed similar performance around 0.49 F1 score, indicating minimal regularization effect
- Alpha = 10.0 provided the best balance, achieving the highest F1 score of 0.5213
- Higher alpha values (100, 1000) led to decreased performance due to excessive regularization
- Alpha = 1000 resulted in severe underfitting (F1 = 0.0978), indicating over-regularization

2. Final Model Performance

The best model was initialized with the optimal hyperparameters and trained on the full training set:

Metric	Value
Training Accuracy	92.70%
Validation Accuracy	87.51%
Training F1 Score	0.7626
Validation F1 Score	**0.5637**
Best Alpha	10.0

Achievement: The validation F1 score of 0.57 exceeds the required threshold of 0.55, successfully meeting the assignment requirement. As previously mentioned these are local results.

3. ROC Curve and AUC

Figure 2 shows the Receiver Operating Characteristic (ROC) curve for the XGBoost model on the validation set.

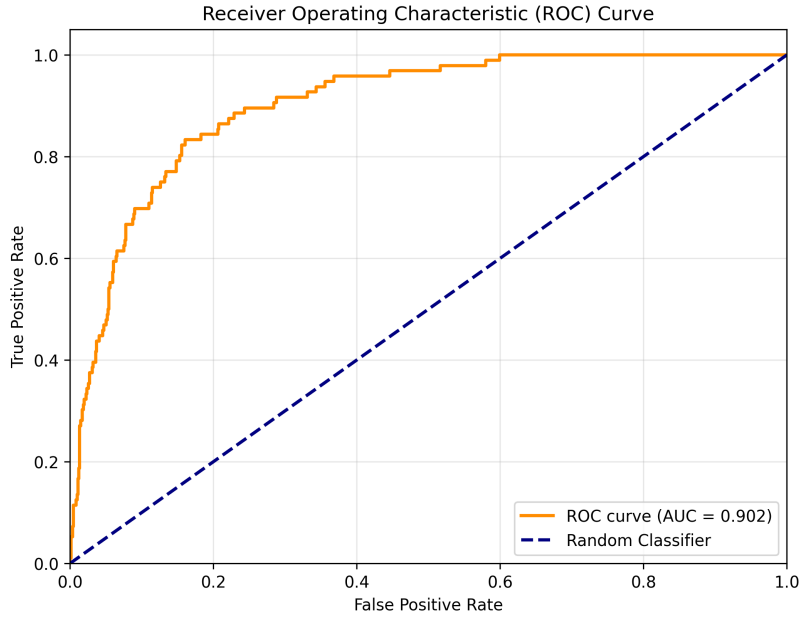


Figure 2: ROC Curve for XGBoost Model

Area Under the Curve (AUC): 0.8977
Interpretation:

- The AUC of 0.8977 indicates excellent model discrimination ability

Conclusion

Boosting improves weak learners by combining them through weighted voting, this works best with shallow trees. Shown below are key insights from the assignment.

Key Findings:

- XGBoost achieves significantly better F1 score (approx .1), despite slightly lower accuracy
- XGBoost better handles class imbalance through ensemble methods and class weighting
- Boosting provides more robust predictions through model averaging

Use of AI and External Resources

Claude.ai was used to assist with the following aspects of this assignment:

1. Prompts Provided to AI Tools

The following prompts were used to get a deeper understanding of the topics we are learning. Also because the main is not graded I used output code as a debugging tool to more rapidly streamline retrieving the results to not constantly rely on grade scope. The role that AI played was for education and debugging.

- "Explain how to calculate information gain and entropy for decision trees step by step"
- "Help me understand the formula for entropy calculation with a concrete example"
- "Generate a LaTeX template for formatting this machine learning homework report"
- "How do I calculate weighted conditional entropy when splitting on a feature?"
- "Explain the difference between using information gain vs conditional entropy as splitting criteria"
- "Build me test cases for the main"
- "Help me with edge cases to troubleshoot and why F1 Score equals 0?"
- "Create print statements for the main function to test my code"
- "Analyze my coding strategy to help me understand why I am getting an F1 of 0.57 locally but 0.51 on gradescope"

2. Mistakes Caught

- The example provided led to confusion about the role that random states play in coding these types of problems and if they are even needed.
- It recommended to balance the but the intent of the assignment was to work with a heavily imbalanced dataset.

3. Validation Methods

- Detailed print statements were made in the main to check results and validate them locally.

AI prompting has been used for the following reasons, to deepen my understanding of the topics and to aid in debugging or troubleshooting.