Agreement 1) This assignment represents my own work. I did not work on this assignment with others. All coding was done by myself.

# Q1

**• Acquisition/Recording: (ii)**

Justification: In this scenario, the Swomee-Swans record the data of Truffula trees everyday by taking photos. This is data acquisition process.

**• Extraction/Cleaning/Annotation: (i)**

Justification: The Bar-Ba-Loots annotate the color and quality of the Truffula tree tuft and clean out ambiguous or dubious information.

**• Integration/Aggregation/Representation: (v)**

Justification: In this scenario, the Humming-Fish integrate all the data into the same format that is directly conducive to data mining.

**• Analysis/Modeling: (iv)**

Justification: The Swomee-Swans use the data to build models to predict when the Truffula trees will disappear. This is Analysis/Modeling process.

**• Interpretation: (iii)**

Justification: The Lorax look at the result of data analysis and find a sustainable plan for Thneed business. This is Interpretation process.

# Q2

**(a)**

**(1) Prove $I(X;Y) \geq 0$:**

Given that,

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) log p(x)$$

$$H(X|Y) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) log \frac{p(x,y)}{p(y)}$$

$$I(X;Y) = H(X) - H(X|Y)$$

Therefore,

$$I(X;Y) = -\sum_{x \in \mathcal{X}} p(x) log p(x) + \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) log \frac{p(x,y)}{p(y)}$$

$$I(X;Y) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) log p(x) + \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) log \frac{p(x,y)}{p(y)}$$

$$I(X;Y) = -(\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) log p(x) - \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) log \frac{p(x,y)}{p(y)})$$

$$I(X;Y) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) log \frac{p(x)p(y)}{p(x,y)}$$

Apply Jensen's Inequality $f(E[X]) \leq E[f(X)]$, where the convex function $f(X) = -\log(X)$:

$$-\log \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) \frac{p(x)p(y)}{p(x,y)} \leq -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) log \frac{p(x)p(y)}{p(x,y)}$$

$$-\log \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x)p(y) \leq I(X;Y)$$

Because $\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x)p(y) = \sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} p(y) = 1$

$$I(X;Y) \geq -\log 1$$

$$\boldsymbol{I(X;Y) \geq 0}$$

**(2) Prove $\boldsymbol{I(X;Y|Z) \geq 0}$:**

Given that,

$$H(X|Z) = -\sum_{x\in\mathcal{X},z\in\mathcal{Z}} p(x,z)log\frac{p(x,z)}{p(z)}$$

$$H(X|Y,Z) = -\sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p(x,y,z)log\frac{p(x,y,z)}{p(y,z)}$$

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z)$$

Therefore,

$$I(X;Y|Z) = -(\sum_{x\in\mathcal{X},z\in\mathcal{Z}} p(x,z)log\frac{p(x,z)}{p(z)} - \sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p(x,y,z)log\frac{p(x,y,z)}{p(y,z)})$$

$$I(X;Y|Z) = -(\sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p(x,y,z)log\frac{p(x,z)}{p(z)} - \sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p(x,y,z)log\frac{p(x,y,z)}{p(y,z)})$$

$$I(X;Y|Z) = -\sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p(x,y,z)log\frac{p(x,z)p(y,z)}{p(z)p(x,y,z)}$$

Apply Jensen's Inequality $f(E[X]) \leq E[f(X)]$, where the convex function $f(X) = -\log(X)$:

$$-log\sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p(x,y,z)\frac{p(x,z)p(y,z)}{p(z)p(x,y,z)} \leq -\sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p(x,y,z)log\frac{p(x,z)p(y,z)}{p(z)p(x,y,z)}$$

$$-log\sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} \frac{p(x,z)p(y,z)}{p(z)} \leq I(X;Y|Z)$$

$$I(X;Y|Z) \geq -log\sum_{z\in\mathcal{Z}} \frac{\sum_{x\in\mathcal{X}} p(x,z)\sum_{y\in\mathcal{Y}} p(y,z)}{p(z)}$$

Because $\sum_{x\in\mathcal{X}} p(x,z) = p(z), \sum_{y\in\mathcal{Y}} p(y,z) = p(z)$,

$$I(X;Y|Z) \geq -log \sum_{z \in \mathcal{Z}} \frac{p(z)^2}{p(z)}$$

$$I(X;Y|Z) \geq -log \sum_{z \in \mathcal{Z}} p(z)$$

$$I(X;Y|Z) \geq -log1$$

$$\boldsymbol{I(X;Y|Z) \geq 0}$$

**(b)**

**(1) Prove $\boldsymbol{I(X;Y,Z) = I(X;Y|Z) + I(X;Z)}$:**

$$left = I(X;Y,Z) = H(X) - H(X|Y,Z)$$

$$right = I(X;Y|Z) + I(X;Z) = H(X|Z) - H(X|Y,Z) + H(X) - H(X|Z)$$

$$= H(X) - H(X|Y,Z) = left$$

Therefore,

$$\boldsymbol{I(X;Y,Z) = I(X;Y|Z) + I(X;Z)}$$

**(2) Prove $\boldsymbol{I(X;Y,Z) = I(X;Z|Y) + I(X;Y)}$:**

$$left = I(X;Y,Z) = H(X) - H(X|Y,Z)$$

$$right = I(X;Z|Y) + I(X;Y) = H(X|Y) - H(X|Y,Z) + H(X) - H(X|Y)$$

$$= H(X) - H(X|Y,Z) = left$$

Therefore,

$$\boldsymbol{I(X;Y,Z) = I(X;Z|Y) + I(X;Y)}$$

**(c)**

Given,

$$p(z|y,x) = p(z|y)$$

Therefore,

$$\frac{p(x,y,z)}{p(x,y)} = \frac{p(y,z)}{p(y)}$$

$$\frac{p(y)}{p(x,y)} = \frac{p(y,z)}{p(x,y,z)} \tag{1}$$

Given,

$$I(X;Y) = H(X) - H(X|Y)$$

$$I(X;Z) = H(X) - H(X|Z)$$

$$H(X|Y) = -\sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)log\frac{p(x,y)}{p(y)}$$

$$H(X|Z) = -\sum_{x\in\mathcal{X},z\in\mathcal{Z}} p(x,z)log\frac{p(x,z)}{p(z)}$$

Therefore,

$$I(X;Y) - I(X;Z) = H(X) - H(X|Y) - H(X) + H(X|Z)$$

$$I(X;Y) - I(X;Z) = H(X|Z) - H(X|Y)$$

$$I(X;Y) - I(X;Z) = -(\sum_{x\in\mathcal{X},z\in\mathcal{Z}} p(x,z)log\frac{p(x,z)}{p(z)} - \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)log\frac{p(x,y)}{p(y)})$$

$$I(X;Y) - I(X;Z) = -(\sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p(x,y,z)log\frac{p(x,z)}{p(z)} - \sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p(x,y,z)log\frac{p(x,y)}{p(y)})$$

$$I(X;Y) - I(X;Z) = -\sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p(x,y,z)log\frac{p(x,z)}{p(z)}\frac{p(y)}{p(x,y)} \tag{2}$$

Combine equation (1) and (2):

$$I(X;Y) - I(X;Z) = -\sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p(x,y,z)log\frac{p(x,z)}{p(z)}\frac{p(y,z)}{p(x,y,z)}$$

Apply Jensen's Inequality $f(E[X]) \leq E[f(X)]$, where the convex function $f(X) = -\log(X)$:

$$-log \sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p(x,y,z)\frac{p(x,z)}{p(z)}\frac{p(y,z)}{p(x,y,z)} \leq -\sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} p(x,y,z)log\frac{p(x,z)}{p(z)}\frac{p(y,z)}{p(x,y,z)}$$

$$-log \sum_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}} \frac{p(x,z)p(y,z)}{p(z)} \leq I(X;Y) - I(X;Z)$$

$$I(X;Y) - I(X;Z) \geq -log \sum_{z\in\mathcal{Z}} \frac{\sum_{x\in\mathcal{X}} p(x,z) \sum_{y\in\mathcal{Y}} p(y,z)}{p(z)}$$

Because $\sum_{x\in\mathcal{X}} p(x,z) = p(z), \sum_{y\in\mathcal{Y}} p(y,z) = p(z)$,

$$I(X;Y) - I(X;Z) \geq -log \sum_{z\in\mathcal{Z}} p(z)$$

Because $\sum_{z\in\mathcal{Z}} p(z) = 1$,

$$I(X;Y) - I(X;Z) \geq -log1$$

$$I(X;Y) - I(X;Z) \geq 0$$

$$\boldsymbol{I(X;Y) \geq I(X;Z)}$$

# Q3

**(a)**

From the dataset, the fraction of positives before split $p = \frac{5}{10} = \frac{1}{2}$

Therefore, $Gini\ index(p, 1-p) = 2 \times \frac{1}{2} \times \left(1 - \frac{1}{2}\right) = \frac{1}{2}$

**(1) If split Rain first:**

When $Rain = 0$, there is 5 positives and 1 negatives.

$\Rightarrow\ N_0 = 6,\ p_0 = \frac{5}{6}$

When $Rain = 1$, there is no positives and 4 negatives.

$\Rightarrow\ N_1 = 4,\ p_1 = 0$

Therefore,

$$Gini\ Reduction = Gini\ index(p, 1-p) - \sum_{children\ c} \frac{N_c}{\mathrm{N}} Gini\ index(p_c, 1-p_c)$$

$$= \frac{1}{2} - \left(\frac{6}{10} \times 2 \times \frac{5}{6} \times \left(1 - \frac{5}{6}\right) + \frac{4}{10} \times 2 \times 0 \times (1-0)\right) = \mathbf{\frac{1}{3}}$$

**(2) If split Good Strategy first:**

When $Good\ Strategy = 0$, there is 3 positives and 5 negatives.

$\Rightarrow\ N_0 = 8,\ p_0 = \frac{3}{8}$

When $Good\ Strategy = 1$, there is 2 positives and no negatives.

$\Rightarrow\ N_1 = 2,\ p_1 = 1$

Therefore,

$$Gini\ Reduction = Gini\ index(p, 1-p) - \sum_{children\ c} \frac{N_c}{\mathrm{N}} Gini\ index(p_c, 1-p_c)$$

$$= \frac{1}{2} - \left(\frac{8}{10} \times 2 \times \frac{3}{8} \times \left(1 - \frac{3}{8}\right) + \frac{2}{10} \times 2 \times 1 \times (1 - 1)\right) = \mathbf{\frac{1}{8}}$$

**(3) If split Qualifying first:**

When $Qualifying = 0$, there is 1 positives and 3 negatives.

$\Rightarrow N_0 = 4,\ p_0 = \frac{1}{4}$

When $Qualifying = 1$, there is 4 positives and 2 negatives.

$\Rightarrow N_1 = 6,\ p_1 = \frac{4}{6}$

Therefore,

$$Gini\ Reduction = Gini\ index(p, 1-p) - \sum_{children\ c} \frac{N_c}{\mathrm{N}} Gini\ index(p_c, 1-p_c)$$

$$= \frac{1}{2} - \left(\frac{4}{10} \times 2 \times \frac{1}{4} \times \left(1 - \frac{1}{4}\right) + \frac{6}{10} \times 2 \times \frac{4}{6} \times \left(1 - \frac{4}{6}\right)\right) = \mathbf{\frac{1}{12}}$$

**Conclusion:**

Comparing (1), (2), (3), it is found that when splitting $Rain$ firstly, its Gini Reduction is the greatest. **Therefore, the feature $\boldsymbol{Rain}$ should be split first.**

**(b)**

$$Gain(S, Rain) = H\left(\left[\frac{1}{2}, \frac{1}{2}\right]\right) - \left(\frac{6}{10} H\left(\left[\frac{5}{6}, \frac{1}{6}\right]\right) + \frac{4}{10} H([0,1])\right)$$

$$Gain(S, Good\ Strategy) = H\left(\left[\frac{1}{2}, \frac{1}{2}\right]\right) - \left(\frac{8}{10} H\left(\left[\frac{3}{8}, \frac{5}{8}\right]\right) + \frac{2}{10} H([1,0])\right)$$

$$Gain(S, Qualifying) = H\left(\left[\frac{1}{2}, \frac{1}{2}\right]\right) - \left(\frac{4}{10} H\left(\left[\frac{1}{4}, \frac{3}{4}\right]\right) + \frac{6}{10} H\left(\left[\frac{4}{6}, \frac{2}{6}\right]\right)\right)$$

Knowing that,

$$H([p, 1-p]) = -plog_2(p) - (1-p)log_2(1-p)$$

Therefore,

$$Gain(S, Rain) = 1 - \left(\frac{6}{10} \times 0.65 + \frac{4}{10} \times 0\right) = 0.61$$

$$Gain(S, Good\ Strategy) = 1 - \left(\frac{8}{10} \times 0.95 + \frac{2}{10} \times 0\right) = 0.24$$

$$Gain(S, Qualifying) = 1 - \left(\frac{4}{10} \times 0.81 + \frac{6}{10} \times 0.92\right) = 0.12$$

Because $Gain(S, Rain) > Gain(S, Good\ Strategy) > Gain(S, Qualifying)$ ,

**$\boldsymbol{Rain}$ is chosen to be split first, which is same as (a).**

# Q4

**(a)**

The total number of binary trees can be expressed using recursive function as below:

$$Num(d,h) = Num^2(d-1,h-1) \times d + 1 \tag{1}$$

Where $d$ is the number of binary features, $h$ is the maximum depth, and $Num(0,1) = 1$.

Therefore,

$$Num(1,2) = Num^2(0,1) \times 1 + 1 = 2$$

$$Num(2,3) = Num^2(1,2) \times 2 + 1 = 9$$

$$\boldsymbol{Num(3,4) = Num^2(2,3) \times 3 + 1 = 244}$$

$$\boldsymbol{Num(4,5) = Num^2(3,4) \times 4 + 1 = 238145}$$

**(b)**

Because there is one feature that is always 1, splitting on this feature make no sense to minimize 0-1 misclassification loss, $\ell(f,X,y)$. In addition, due to $\lambda > 0$, splitting based on this feature increases the number of leaves, and hence the objective function, $R(f,X,y)$. Therefore, all trees split by this feature will not be the optimal tree. In this case, when $d = 4$ and $h = 5$, the effective features number $d$ should be 3 instead of 4. Therefore, the number of trees after reduction is calculated by using equation (1):

$$N(0,2) = 1$$

$$N(1,3) = 2$$

$$N(2,4) = 9$$

$$\boldsymbol{N_{reduced} = Num(4-1,5) = Num(3,5) = 244}$$

**(c)**

when splitting $(p_k, \hat{y}_k)$ into $(p_{k1}, \hat{y}_{k1})$ and $(p_{k2}, \hat{y}_{k2})$, and/or further splitting $(p_{k1}, \hat{y}_{k1})$ or $(p_{k2}, \hat{y}_{k2})$, the minimum increase in $\lambda s(f)$ is $\lambda$. In addition, because the predicted label $\hat{y}_k$ is the majority label in leaf $k$, the maximum decrease in $\ell(f, X, y)$ occurred when the number of positive labels is equivalent to negative labels in leaf $k$, which is $\frac{m_k}{2n}$.

Given that,

$$m_k < 2n\lambda$$

Therefore,

$$\frac{m_k}{2n} < \lambda$$

Therefore,

$$maximum\ decrease\ in\ \ell(f, X, y) < minimum\ increase\ in\ \lambda s(f)$$

**Therefore, the objective functions of $\boldsymbol{f'}$ and further split trees are always greater than the objective function of $\boldsymbol{f}$, which means that $\boldsymbol{f'}$ and further split trees cannot be optimal.**

**(d)**

Given $\lambda = 0.3$ and $n = 8$,

$$2n\lambda = 2 \times 0.3 \times 8 = 4.8$$

Based on part (c), in this case, for a leaf k, if $m_k < 4.8$, this leaf does not need to be further split.

Based on part (b), in this case, the feature $x_4$ is not an effective feature. Therefore, all trees split by $x_4$ will be skipped.

**Start evaluating:**

**(1) 0 split and only 1 leaf**

The objective function is calculated:

$$R = \ell + \lambda s = \frac{3}{8} + 0.3 \times 1 = 0.675$$

$m_k = 8 > 2n\lambda$, so, this leaf should be further split.

**(2) 1 split and 2 leaf (split by $x_1$)**

The left leaf (when $x_1 = 0$) $(p_{k0}, \hat{y}_{k0})$ has 1 positive label and 3 negative labels; The right leaf (when $x_1 = 1$) $(p_{k1}, \hat{y}_{k1})$ has 4 positive labels and 0 negative label. Therefore, the objective function is calculated:

$$R = \ell + \lambda s = \frac{1}{8} + 0.3 \times 2 = 0.725$$

Because $m_{k0} = m_{k1} = 4 < 2n\lambda$, the trees produced by further splitting $(p_{k0}, \hat{y}_{k0})$ or $(p_{k1}, \hat{y}_{k1})$ can be skipped from evaluating.

**(3) 1 split and 2 leaf (split by $x_2$)**

The left leaf (when $x_2 = 0$) $(p_{k0}, \hat{y}_{k0})$ has 2 positive labels and 2 negative labels; The right leaf (when $x_2 = 1$) $(p_{k1}, \hat{y}_{k1})$ has 3 positive labels and 1 negative label. Therefore, the objective function is calculated:

$$R = \ell + \lambda s = \frac{3}{8} + 0.3 \times 2 = 0.975$$

Because $m_{k0} = m_{k1} = 4 < 2n\lambda$, the trees produced by further splitting $(p_{k0}, \hat{y}_{k0})$ or $(p_{k1}, \hat{y}_{k1})$ can be skipped from evaluating.

**(4) 1 split and 2 leaf (split by $x_3$)**

The left leaf (when $x_3 = 0$) $(p_{k0}, \hat{y}_{k0})$ has 2 positive labels and 2 negative labels; The right leaf (when $x_3 = 1$) $(p_{k1}, \hat{y}_{k1})$ has 3 positive labels and 1 negative label. Therefore, the objective function is calculated:

$$R = \ell + \lambda s = \frac{3}{8} + 0.3 \times 2 = 0.975$$

Because $m_{k0} = m_{k1} = 4 < 2n\lambda$, the trees produced by further splitting $(p_{k0}, \hat{y}_{k0})$ or $(p_{k1}, \hat{y}_{k1})$ can be skipped from evaluating.

**Conclusion:**

**From the evaluation above, the optimal tree is the tree that has 0 split and only 1 leaf, with a minimum objective value of $\mathbf{0.675}$.**

**In addition, due to the bound from part (b) and (c), the number of trees I need to evaluate is 4, and the proportion is calculated below:**

$$\boldsymbol{proportion} = \frac{\boldsymbol{N_{evaluated}}}{\boldsymbol{N_{total}}} = \frac{\mathbf{4}}{\mathbf{238145}} = \mathbf{0.00168\%}$$

# Q5

**(a)**

The relationship between max depth and average model performance is plotted in the code below. Based on the plot, the accuracy, F1 score, and AUC are all the highest when the max depth is 5. Therefore, in this case, $max\ depth\ =\ 5$ will be chosen for the final model.

**(b)**

The best parameters for the decision tree are shown as below:

$$\{'criterion': 'gini', 'max\_depth': 6, 'splitter': 'best'\}$$

The tree is plotted in the code below.

**(c)**

GOSDT is used. The F1 score is $0.714$ and AUC Score is $0.746$.

For the constructed tree, the parameter 'depth_budget' is tuned with 5-fold cross-validation. In addition, the relationships between depth_budget and performance are plotted in the code below.

**(d)**

The traditional decision tree in (b) with best parameters has a test accuracy of $0.6875$ and a test F1 score of $0.667$; while the GOSDT with $depth\_budget = 6$ has a test accuracy of $0.75$ and a test F1 score of $0.714$. Therefore, the GOSDT implementation performed the best.

**The intact code and answers for Q5 are shown below.**

# HW2_Q5

September 25, 2023

# 1 Q5 Decision Trees Implementations

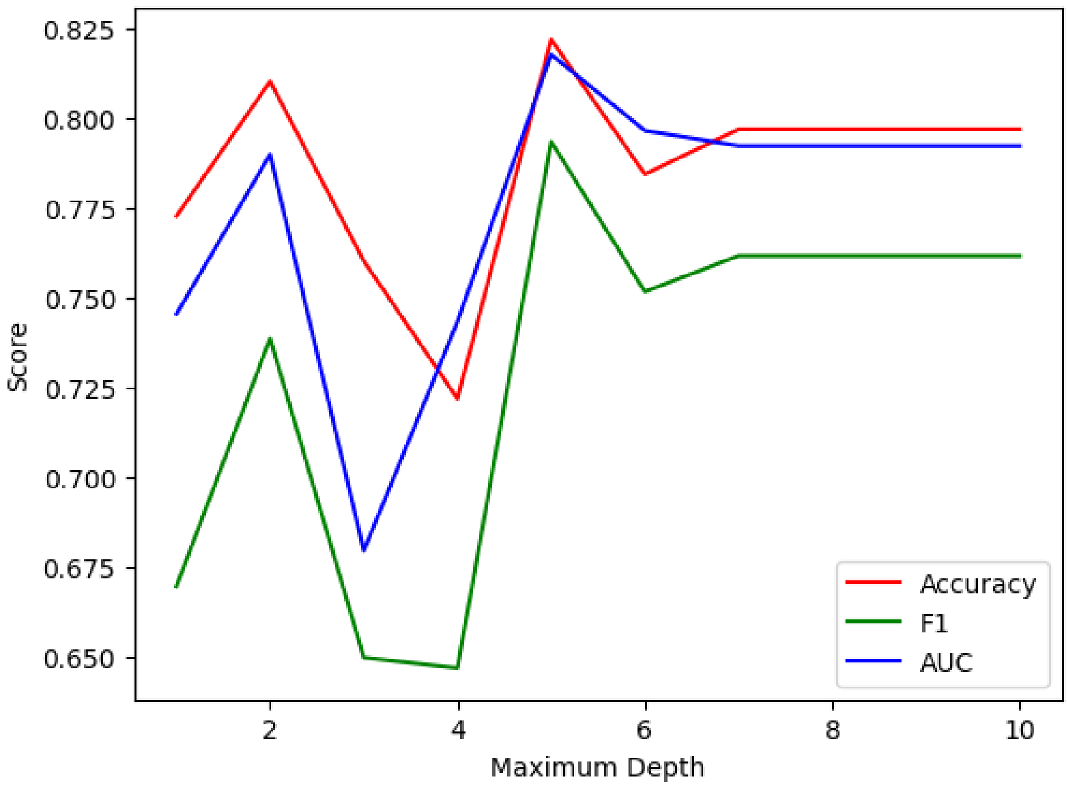
## 1.1 Initialization

[1]:
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import cross_validate, train_test_split,␣
↪GridSearchCV
from sklearn.metrics import f1_score, roc_auc_score, accuracy_score
from sklearn.utils import shuffle
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from gosdt.model.gosdt import GOSDT
from gosdt.model.threshold_guess import compute_thresholds, cut

# set random seed as 42
np.random.seed(42)

# read data from csv file
data = pd.read_csv("kindey stone urine analysis.csv")
```

## 1.2 (a)

[2]:
```
# shuffle
shuffle_data = shuffle(data, random_state=42)
# print(data)
# print(shuffle_data)
x = shuffle_data[['gravity', 'ph', 'osmo', 'cond', 'urea', 'calc']]
y = shuffle_data.target
# print(x)
# print(y)

accuracy_scores, f1_scores, auc_scores, max_depths = [], [], [], []
best_accuracy, best_f1, best_auc = 0, 0, 0
best_accuracy_depth, best_f1_depth, best_auc_depth = 0, 0, 0

# tuning max depth from 1 to 10
for i in range(10):
```

```
    model = DecisionTreeClassifier(criterion='gini', splitter='best',␣
↪max_depth=i+1, random_state=42)
    scores = cross_validate(model, x, y, scoring=['accuracy', 'f1', 'roc_auc'],␣
↪cv=5)

    accuracy_scores.append(np.mean(scores['test_accuracy']))
    f1_scores.append(np.mean(scores['test_f1']))
    auc_scores.append(np.mean(scores['test_roc_auc']))
    max_depths.append(i+1)

    # choose best max depth
    if best_accuracy < np.mean(scores['test_accuracy']):
        best_accuracy = np.mean(scores['test_accuracy'])
        best_accuracy_depth = i + 1
    if best_f1 < np.mean(scores['test_f1']):
        best_f1 = np.mean(scores['test_f1'])
        best_f1_depth = i + 1
    if best_auc < np.mean(scores['test_roc_auc']):
        best_auc = np.mean(scores['test_roc_auc'])
        best_auc_depth = i + 1

# plot relationship between max depth and average model performance
plt.plot(max_depths, accuracy_scores, color='r')
plt.plot(max_depths, f1_scores, color='g')
plt.plot(max_depths, auc_scores, color='b')
plt.xlabel('Maximum Depth')
plt.ylabel('Score')
plt.legend(['Accuracy', 'F1', 'AUC'])
plt.show()

# best max depth for different model performance
print('When accuracy is the highest, the max_depth is', best_accuracy_depth)
print('When F1 score is the highest, the max_depth is', best_f1_depth)
print('When AUC is the highest, the max_depth is', best_auc_depth)
```



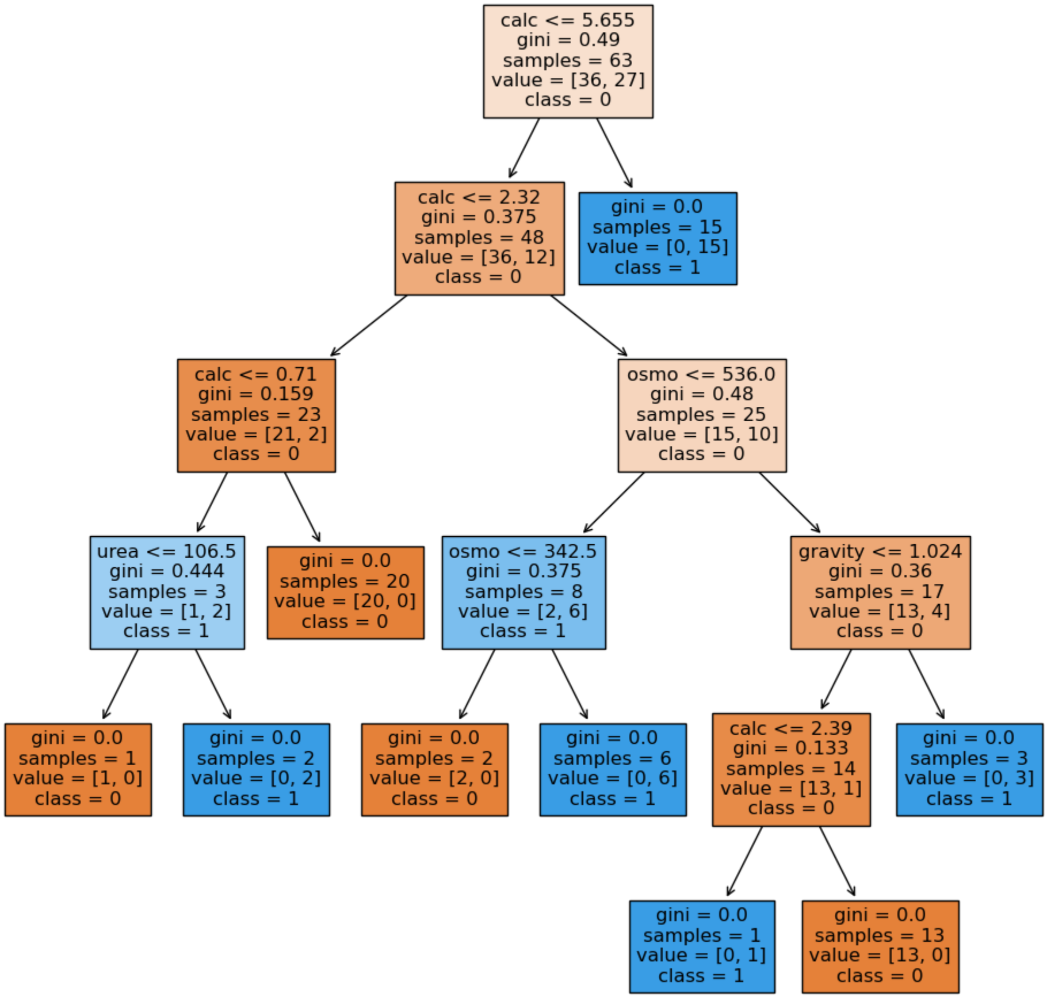

```
When accuracy is the highest, the max_depth is 5
When F1 score is the highest, the max_depth is 5
When AUC is the highest, the max_depth is 5
```

# 2 (b)

[3]:
```
# split train and test dataset
x = data[['gravity', 'ph', 'osmo', 'cond', 'urea', 'calc']]
y = data.target
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
 ↪train_size=0.8, random_state=42, shuffle=True, stratify=y)

# GridSearch 5-fold Cross Validation on criterion, splitter, and max_depth
model = DecisionTreeClassifier(random_state=42)
param = {'criterion':['gini', 'entropy', 'log_loss'], 'splitter':['best',␣
 ↪'random'], 'max_depth':[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}

# custom refit strategy
def custom_refit_strategy(cv_results):
    cv_results_ = pd.DataFrame(cv_results)
```

```python
    return np.argmax(cv_results_["mean_test_accuracy"] +␣
↪cv_results_["mean_test_f1"] + cv_results_["mean_test_roc_auc"])

model = GridSearchCV(model, param_grid=param, scoring=['accuracy', 'f1',␣
↪'roc_auc'], cv=5, refit=custom_refit_strategy)
model.fit(x_train, y_train)

# find the best parameters
print('The best parameters:', model.best_params_)
# print(model.best_index_)

# train the whole training set
model = model.best_estimator_
model.fit(x_train, y_train)

# plot the tree
plt.figure(figsize=(12,12))
_ = plot_tree(
    model,
    feature_names=['gravity', 'ph', 'osmo', 'cond', 'urea', 'calc'],
    class_names=['0', '1'], # target
    filled=True
)
```

```
The best parameters: {'criterion': 'gini', 'max_depth': 6, 'splitter': 'best'}
```




# 3 (c)

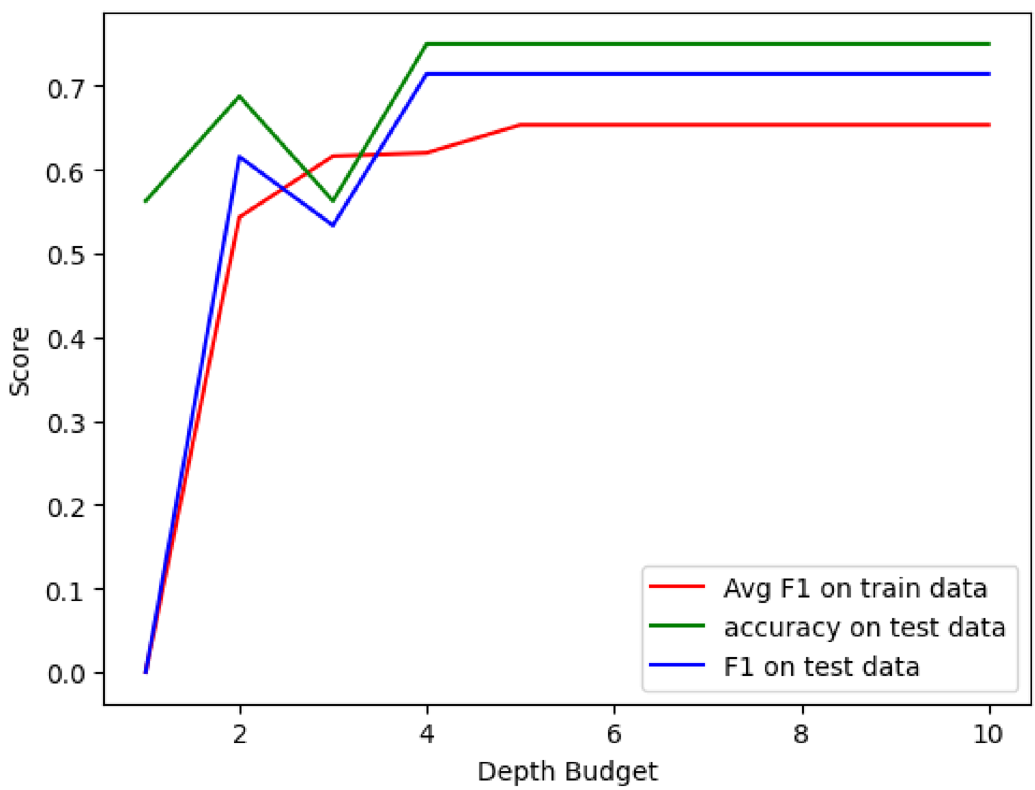
[4]:
```
import warnings
import os, sys

# ignore warnings
warnings.filterwarnings("ignore")
# ignore gosdt's print information
class ignorePrint:
    def __enter__(self):
        self._stdout = sys.stdout
        sys.stdout = open(os.devnull, 'w')
```

```
    def __exit__(self, exc_type, exc_val, exc_tb):
        sys.stdout.close()
        sys.stdout = self._stdout

# generate binary features
x_train_bin, thresholds, header, _ = compute_thresholds(x_train.copy(),
 ↪y_train, 60, 1)
# print(x_train_bin.shape)
x_test_bin = cut(x_test.copy(), thresholds)
x_test_bin = x_test_bin[header]
# print(x_test_bin.shape, x_train_bin.shape)

# train
model = GOSDT({
    'regularization': 0.02,
    'depth_budget': 6,
    'similar_support': False
})
with ignorePrint():
    model.fit(x_train_bin, y_train)

# test GOSDT
y_pred = model.predict(x_test_bin)
# F1 scores and AUC Scores on test dataset
print('\nF1 score on test dataset (GOSDT):', f1_score(y_test, y_pred))
print('AUC score on test dataset (GOSDT):', roc_auc_score(y_test, y_pred), '\n')

# tuning depth_budget with 5-fold cross-validation
# create 5 folds evenly
folds_index = []
x_train_split = x_train.copy()
n_int = x_train_split.shape[0] // 5
n_remain = x_train_split.shape[0] % 5
# print(n_int, n_remain)
for i in range(5):
    n = n_int
    if n_remain != 0:
        n += 1
        n_remain -= 1
    index = x_train_split.head(n)._stat_axis.values.tolist()
    folds_index.append(index)
    x_train_split.drop(index, axis=0, inplace=True)

# tuning depth_budget (k + 1)
F1_avg_train_scores, accuracy_test_scores, F1_test_scores, depth_budgets = [],
 ↪[], [], []
```

```
for k in range(10):
    model = GOSDT({
        'regularization': 0.02,
        'depth_budget': k + 1,
        'similar_support': False
    })
    F1_sum_cv = 0
    for i in range(5):
        eval_index_cv = folds_index[i]
        train_index_cv = []
        for j in range(5):
            if i != j:
                train_index_cv += folds_index[j]

        #get eval and train set
        eval_x_cv = x_train.loc[eval_index_cv]
        eval_y_cv = y_train.loc[eval_index_cv]
        train_x_cv = x_train.loc[train_index_cv]
        train_y_cv = y_train.loc[train_index_cv]

        train_x_cv_bin, thresholds, header, _ = compute_thresholds(train_x_cv.
 ↪copy(), train_y_cv, 60, 1)
        eval_x_cv_bin = cut(eval_x_cv.copy(), thresholds)
        eval_x_cv_bin = eval_x_cv_bin[header]

        with ignorePrint():
            model.fit(train_x_cv_bin, train_y_cv)
        eval_y_pred = model.predict(eval_x_cv_bin)
        F1_cv = f1_score(eval_y_cv, eval_y_pred)
        F1_sum_cv += F1_cv

    # get average F1 score
    F1_avg_train_score = F1_sum_cv / 5

    # test
    model = GOSDT({
        'regularization': 0.02,
        'depth_budget': k + 1,
        'similar_support': False
    })
    with ignorePrint():
        model.fit(x_train_bin, y_train)
    test_y_pred = model.predict(x_test_bin)
    # get accuracy and F1 score for test dataset
    accuracy_test_score = accuracy_score(y_test, test_y_pred)
    F1_test_score = f1_score(y_test, test_y_pred)
```

```python
    F1_avg_train_scores.append(F1_avg_train_score)
    accuracy_test_scores.append(accuracy_test_score)
    F1_test_scores.append(F1_test_score)
    depth_budgets.append(k+1)

# plot
plt.plot(depth_budgets, F1_avg_train_scores, color='r')
plt.plot(depth_budgets, accuracy_test_scores, color='g')
plt.plot(depth_budgets, F1_test_scores, color='b')
plt.xlabel('Depth Budget')
plt.ylabel('Score')
plt.legend(['Avg F1 on train data', 'accuracy on test data', 'F1 on test data'])
plt.show()
```

```
F1 score on test dataset (GOSDT): 0.7142857142857143
AUC score on test dataset (GOSDT): 0.746031746031746
```



## 3.1 (d)

[5]:
```
# for traditional decision tree in (b)
# using the best parameters obtained in (b)
# print(x_train.shape, y_train.shape, x_test.shape, y_test.shape)
model = DecisionTreeClassifier(criterion='gini', max_depth=6, splitter='best',␣
↪random_state=42)
model.fit(x_train, y_train)
test_y_pred_trad = model.predict(x_test)
# get accuracy and F1 score for test dataset
accuracy_test_score_trad = accuracy_score(y_test, test_y_pred_trad)
F1_test_score_trad = f1_score(y_test, test_y_pred_trad)
print('traditional decision tree: accuracy_test =', accuracy_test_score_trad,␣
↪', F1_test =', F1_test_score_trad)

# for GOSDT
# From the diagram above, the best performance for gosdt can be obtained when␣
↪Depth Budget >= 5
# using Depth Budget = 6
model = GOSDT({
        'regularization': 0.02,
        'depth_budget': 6,
        'similar_support': False
})
x_train_bin, thresholds, header, _ = compute_thresholds(x_train.copy(),␣
↪y_train, 60, 1)
x_test_bin = cut(x_test.copy(), thresholds)
x_test_bin = x_test_bin[header]
with ignorePrint():
    model.fit(x_train_bin, y_train)
test_y_pred_gosdt = model.predict(x_test_bin)
# get accuracy and F1 score for test dataset
accuracy_test_score_gosdt = accuracy_score(y_test, test_y_pred_gosdt)
F1_test_score_gosdt = f1_score(y_test, test_y_pred_gosdt)
print('gosdt: accuracy_test =', accuracy_test_score_gosdt, ', F1_test =',␣
↪F1_test_score_gosdt)
print('Therefore, gosdt has better performance than traditional decision tree.')
```

```
traditional decision tree: accuracy_test = 0.6875 , F1_test = 0.6666666666666666
gosdt: accuracy_test = 0.75 , F1_test = 0.7142857142857143
Therefore, gosdt has better performance than traditional decision tree.
```