Q1. Detecting multicollinearity in the feature set

(a)

In [9]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
```

In [10]:
```
df = pd.read_csv("Xtrain.csv",header=None)
df = df.iloc[:,0:13]
```

In [11]:
```
def calculate_vif(df):
    features = df.columns
    Vif = []
    for feature in features:
        y = df[feature]
        x = df[[i for i in features if i != feature]]
        R2 = LinearRegression().fit(x,y).score(x,y)
        vif = 1/(1-R2)
        Vif.append(vif)
    df_vif = pd.DataFrame([Vif],columns = features)
    df_vif.index = ["VIF"]
    return df_vif
```

In [12]:
```
calculate_vif(df)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
VIF	1.008912	1.008647	1.011746	1.014471	1.006488	1.014334	1.013079	1.015148	1.011172	1.010408	1.010806	1.011713	1.009419

(b)

I can see each column has almost the same VIF value and near 1, which means these features are almost irrelevant. However, if we are forced to remove one feature, I will choose the 8th column which has the highest VIF 1.015148.

Q2 :

(a)

x↓, y→	0	1	x
0	1/3	1/3	2/3
1	0	1/3	1/3
y	1/3	2/3	

$$P(x) = \begin{bmatrix} 2/3 \\ 1/3 \end{bmatrix} \qquad P(y) = \begin{bmatrix} 1/3 \\ 2/3 \end{bmatrix}$$

$$P(x,y) = \begin{bmatrix} 1/3 & 1/3 \\ 0 & 1/3 \end{bmatrix}$$

$$P(x|y) = \frac{P(x,y)}{P(y)} = \begin{bmatrix} 1 & 1/2 \\ 0 & 1/2 \end{bmatrix}$$

$$P(y|x) = \frac{P(x,y)}{P(x)} = \begin{bmatrix} 1/2 & 1/2 \\ 0 & 1 \end{bmatrix}$$

$$H(x) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

$$= -[2/3 \cdot \log_2(2/3) + 1/3 \cdot \log_2(1/3)]$$

$$= 0.918$$

$$H(y) = -[1/3 \cdot \log_2(1/3) + 2/3 \cdot \log_2(2/3)]$$

$$= 0.918$$

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

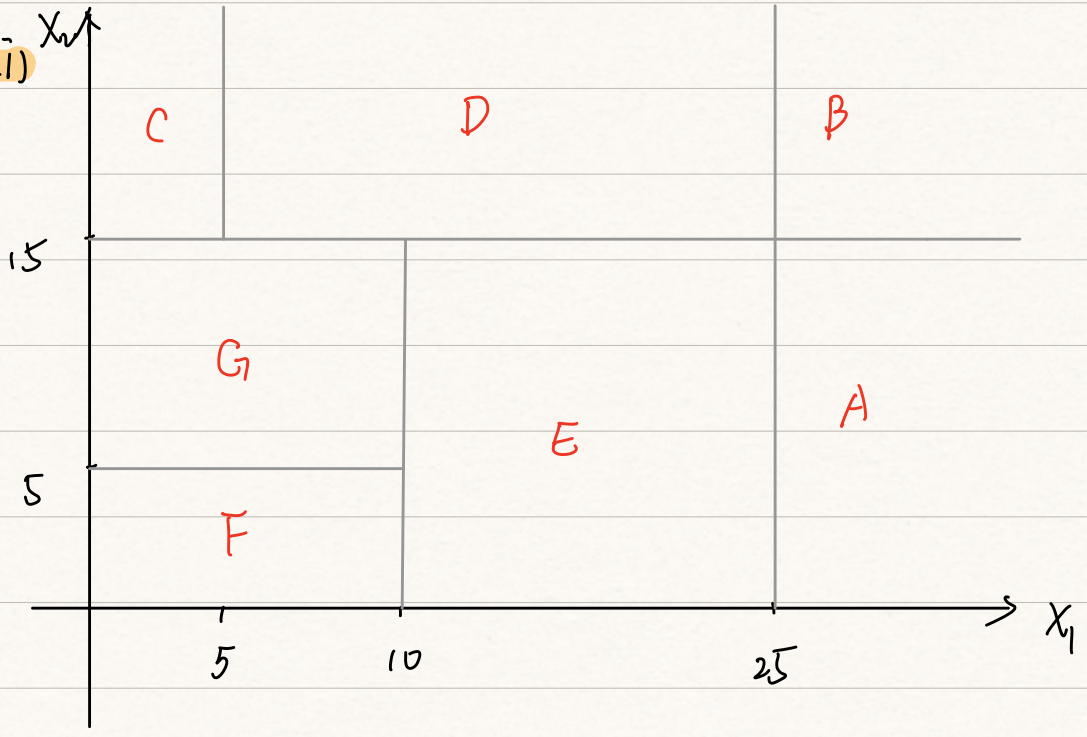
$$= -(1/3 \cdot \log_2 1 + 0 \cdot \log_2 0 + 1/3 \log_2(1/2) + 1/3 \log_2(1/2))$$

$$= 0.667$$

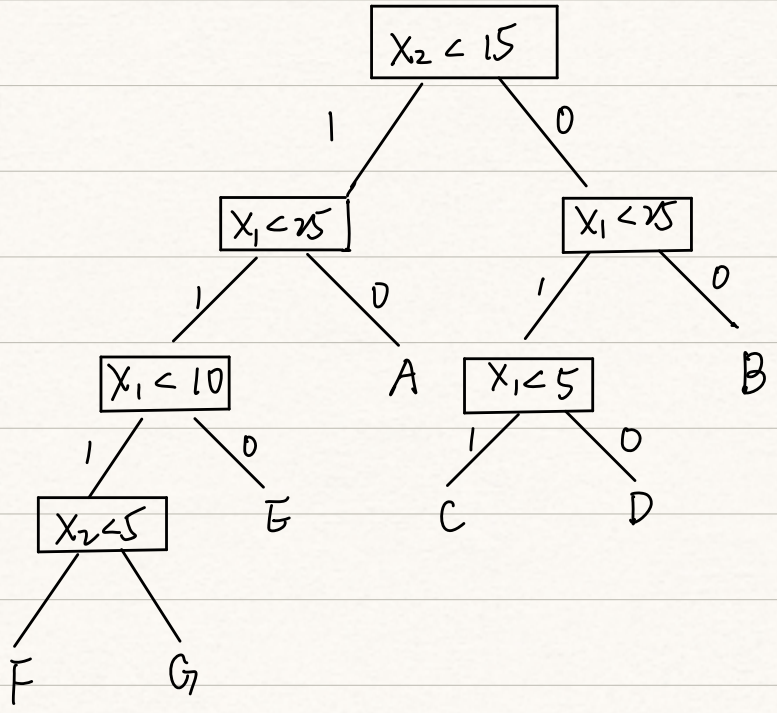
$$H(y|x) = -(1/3 \cdot \log_2(1/2) + 1/3 \cdot \log_2(1/2) + 0 \cdot \log_2 0 + 1/3 \cdot \log_2 1)$$
$$= 0.667$$
$$IG(x|y) = H(x) - H(x|y) = 0.918 - 0.667 = 0.251$$

$$Gini(x) = 1 - (1/3^2 + 2/3^2) = \frac{4}{9}$$
$$Gini(y) = 1 - (2/3^2 + 1/3^2) = \frac{4}{9}$$

(b) (i)

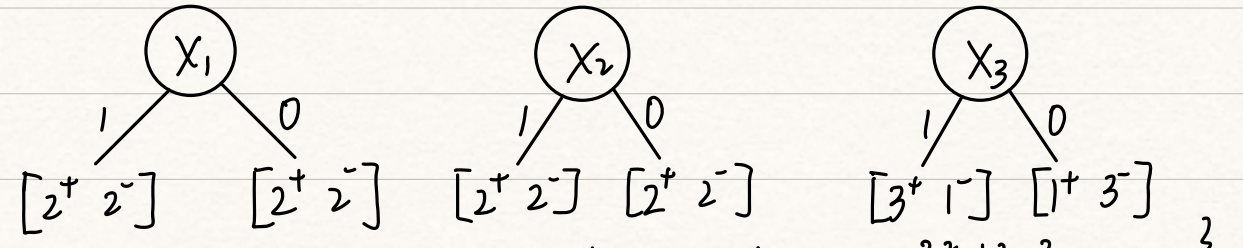


(b) (ii)



Q3:

(a)



$GI = 1 - \frac{1}{2}^2 - \frac{1}{2}^2 = \frac{1}{2}$ $GI = \frac{1}{2}$ $GI = \frac{1}{2}$ $GI = \frac{1}{2}$ $GI = 1 - \frac{3}{4}^2 - \frac{1}{4}^2 = \frac{3}{8}$ $GI = \frac{3}{8}$

$GI(X_1) = \frac{4}{8} \times \frac{1}{2} + \frac{4}{8} \times \frac{1}{2} = \frac{1}{2}$ $GI(X_2) = GI(X_1) = \frac{1}{2}$ $GI(X_3) = \frac{4}{8} \times \frac{3}{8} + \frac{4}{8} \times \frac{3}{8} = \frac{3}{8}$

$\because GI(X_3) < GI(X_1) = GI(X_2)$

$\therefore$ Choose X_3 to be the root of the decision tree.

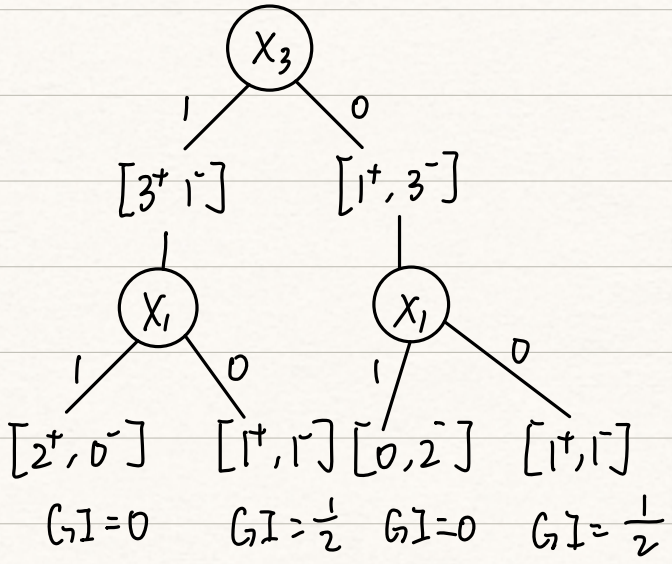
(b) Stopping criteria:

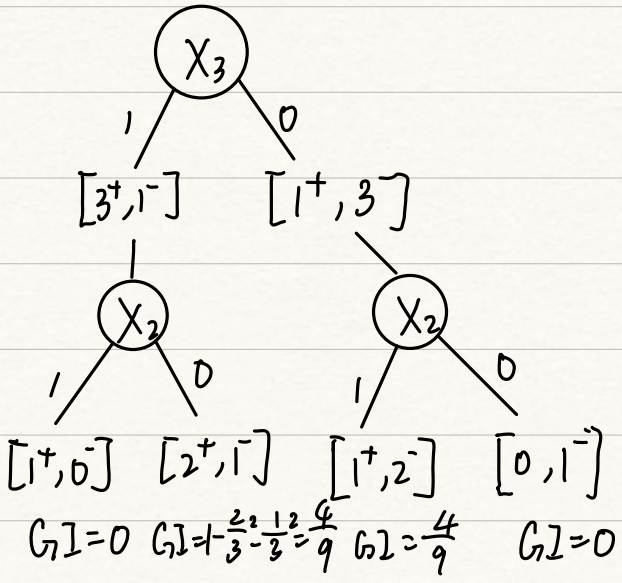
① The subsets of training examples have the same output.

② The subsets of training examples have the same values for all input attributes.

③ There are no training examples for a particular leaf node.

(C)



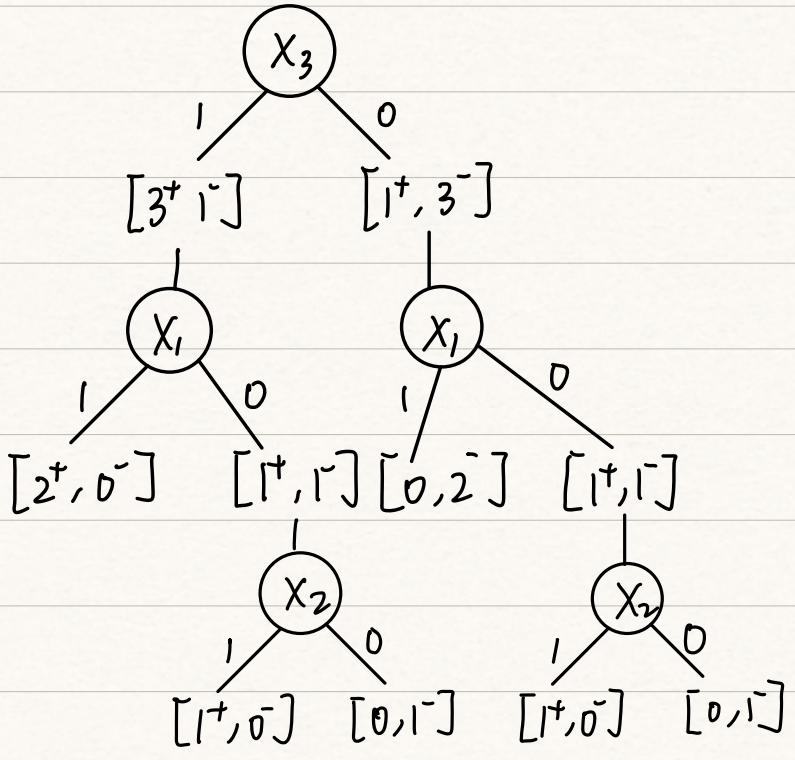


$GI(X_1) = \frac{2}{8} \times \frac{1}{2} + \frac{2}{8} \times \frac{1}{2} = \frac{1}{4}$

$GI(X_2) = \frac{3}{8} \times \frac{4}{9} + \frac{3}{8} \times \frac{4}{9} = \frac{1}{3}$

$\because GI(X_1) > GI(X_2)$

$\therefore$ Choose X_1 to be the second attribute.



$\because$ Every data can be trained perfectly

$\therefore$ error (train) = 0

(d)

Instance	X_1	X_2	X_3	Y
9	1	1	1	1
10	1	0	0	0
11	0	1	1	1

(e) I think we can use test data to evaluate if the tree is overfitting. When number of nodes becomes larger, the accuracy of train data will increase, but the accuracy of test data will go up first and then go down. If accuracy is high in training data and low in test data, the tree is overfitting.

Avoid overfitting:

① Defines the minimum number of samples which are required in a node to be considered for splitting.

② Defines the minimum samples required in a terminal node or leaf.

③ Defines the maximum depth of tree.

(f) I think the structure of tree will change. Because the original tree is trained perfectly will 0 error, which means the tree is sensitive. If we change train data, the tree will change as well. Also, the leaf nodes will change. Instances 5 and 6, 7 and 8 have the same attribute input, but they don't have the same output. Therefore, we can't create a tree with GI = 0, which means the new tree can't have the same leaf nodes as the original tree.

Q3. Decision Trees

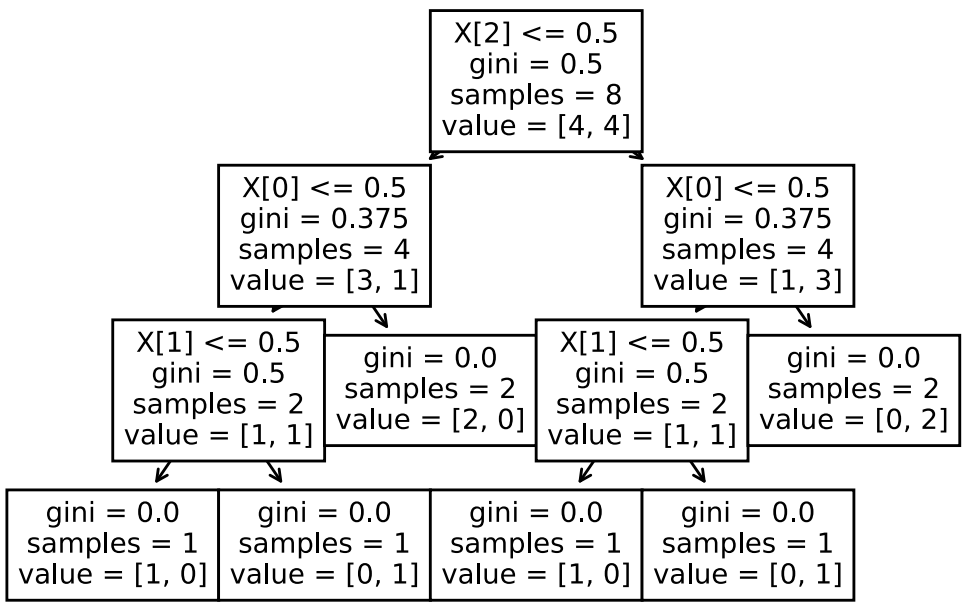
(g)

In [47]:
```
from sklearn import tree
from sklearn.datasets import load_iris
from sklearn import tree
X = [[0, 0, 0], [0, 0, 1],[0, 1, 0],[0,1,1],[1,0,1],[1,0,1],[1,1,0],[1,1,0]]
Y = [0,0,1,1,1,1,0,0]
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, Y)
pred_x = [[1, 1, 1], [1, 0, 0],[0, 1, 1]]
clf.predict(pred_x)
```

Out[47]:
```
array([1, 0, 1])
```

In [48]:
```
tree.plot_tree(clf)
```

Out[48]:
```
[Text(186.0, 190.26, 'X[2] <= 0.5\ngini = 0.5\nsamples = 8\nvalue = [4, 4]'),
 Text(111.60000000000001, 135.9, 'X[0] <= 0.5\ngini = 0.375\nsamples = 4\nvalue = [3, 1]'),
 Text(74.4, 81.53999999999999, 'X[1] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(37.2, 27.180000000000007, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(111.60000000000001, 27.180000000000007, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(148.8, 81.53999999999999, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(260.40000000000003, 135.9, 'X[0] <= 0.5\ngini = 0.375\nsamples = 4\nvalue = [1, 3]'),
 Text(223.20000000000002, 81.53999999999999, 'X[1] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(186.0, 27.180000000000007, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(260.40000000000003, 27.180000000000007, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(297.6, 81.53999999999999, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]')]
```



Q4. Classification and Regression Tree(CART) Implementation

(a) Gini impurity

In [49]:
```
GI = 1-((6/31)**2+(10/31)**2+(10/31)**2+(2/31)**2+(3/31)**2)
GI
```

Out[49]:
```
0.7408949011446411
```

I think the weather in Pittsburgh is not homogeneous.

(b) Code completion

```python
def _gini(self, y):
    """Compute Gini impurity of a non-empty node.
    Gini impurity is defined as sum p(1-p) over all classes, with p the frequency of a
    class within the node. Since sum p = 1, this is equivalent to 1 - sum p^2.
    """
    m = y.size

    # 1. Your code goes here (fill in variable for self._grow_tree)
    # you will need the variable self.n_classes_ which is the number of classes you need
    # to calculate the gini impurity
    num = [np.sum(y == c) for c in range(self.n_classes_)]
    gini_impurity = 1.0 - sum((n / m) ** 2 for n in num)

    # 1. Your code goes here above
    return gini_impurity
```

Screen Shot 2022-03-05 at 17.06.46

```python
# Split recursively until maximum depth is reached.
if depth < self.max_depth:
    idx, thr = self._best_split(X, y)
    if idx is not None:
        indices_left = X[:, idx] < thr
        X_left, y_left = X[indices_left], y[indices_left]
        X_right, y_right = X[~indices_left], y[~indices_left]
        node.feature_index = idx
        node.threshold = thr
        # 2. Your code goes here (fill in variable for self._grow_tree)
        node.left = self._grow_tree(X_left, y_left, depth+1 )
        node.right = self._grow_tree(X_right, y_right, depth+1 )
        # 2. Your code goes here above

return node
```

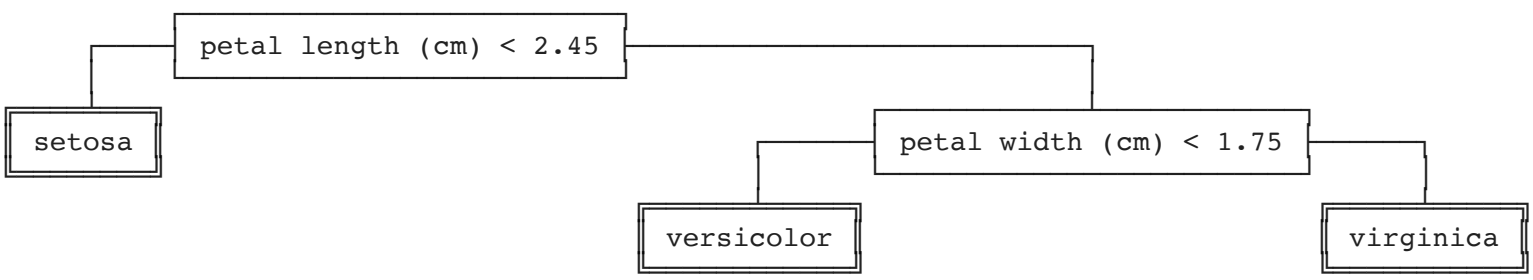
Screen Shot 2022-03-05 at 17.14.23

(c) Experiment with the completed model

(1) Dataset iris with Python Decision tree

In [50]:
```
!python cart.py --dataset="iris" --max_depth=2 --hide_details
```

```
Input: [0, 0, 3.0, 2.5]
Prediction: virginica
```



(2) Dataset iris with sklearn Decision tree

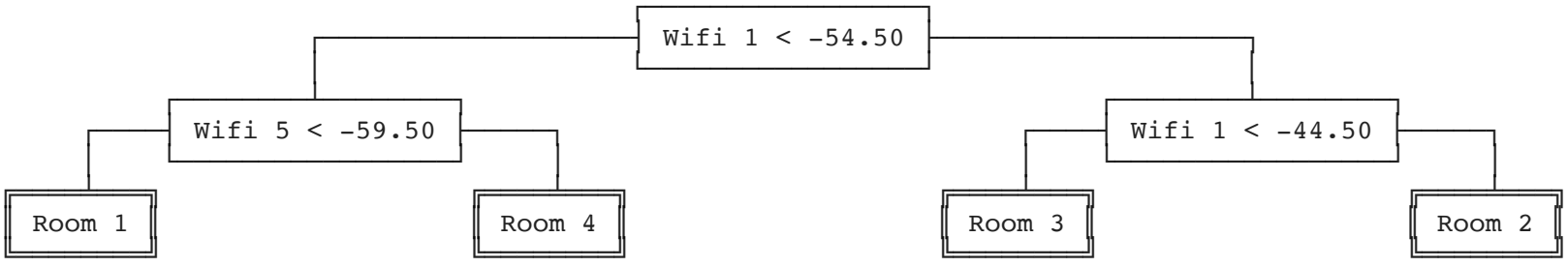
In [51]:
```
!python cart.py --dataset="iris" --max_depth=2 --hide_details --use_sklearn
```

```
Input: [0, 0, 3.0, 2.5]
Prediction: virginica
Done. To convert to PNG, run: dot -Tpng tree.dot -o tree.png
```

(3) Dataset wifi with Python Decision tree

In [52]:
```
!python cart.py --dataset="wifi" --max_depth=2 --hide_details
```

```
Input: [-70, 0, 0, 0, -40, 0, 33]
Prediction: Room 4
```



(4) Dataset wifi with sklearn Decision tree

In [53]:
```
!python cart.py --dataset="wifi" --max_depth=3 --use_sklearn
```

```
Input: [-70, 0, 0, 0, -40, 0, 33]
Prediction: Room 4
Done. To convert to PNG, run: dot -Tpng tree.dot -o tree.png
```