Supervised Learning In Quest SLIQ

Jaehyun Lee

Spring2023
EEL4887

Introduction

What is SLIQ?

- SLIQ (Supervised Learning in Quest) is a high speed and flexible decision tree classifier that allows to sort and interpret data.

- SLIQ can reduce costs using efficient and pre-sorting decision trees to sort through large data sets while accounting for differences in data types.

- This maintains competitive accuracy with the ability to scale and interpret larger data sets with multiple classes and attributes.

Advantages of SLIQ

- The algorithm aims to reduce the diversity of the tree at each split, resulting in more efficient and cost-effective sorting of data
- SLIQ does not require data normalization, making it easier to use with a variety of data types and features
- It scales well with data size, making it suitable for large data sets
- SLIQ is an improvement upon existing decision tree algorithms, offering better sorting and pruning of data

Disadvantages of SLIQ

- The algorithm requires more time to train the model compared to other decision tree algorithms.
- SLIQ is not suitable for regression or predictive modeling, limiting its usefulness in certain applications.
- SLIQ may not perform well with imbalanced datasets, where one class has significantly more instances than the others.

Application of SLIQ

- The main application of SLIQ is on classification tasks in data mining. Which can be applied in many domains, such as finance, healthcare, and marketing.

- One real life application is Bidding Decision System of Electricity Market. The SLIQ algorithm is applied to the bidding decision system of the electricity market, where the knowledge of the bidding unit's ability is gained by considering the market's demand, bidding price, and capacity [3].

Procedure

Procedure Overview

- Presorting the data
- Processing evaluation on splits using Gini index
- Updating the class list
- Tree pruning

Gini Index

- SLIQ uses a training set and a Gini index to prepare the data for the decision tree algorithm. These equations are what make SLIQ a, supervised learning, algorithms as the data is pre-sorted and pruned.

- For training set with n distinct classes the equation, where L is left of the the root node and R is the right of the root node. A and B are the class results.

Attribute Value < P	A	B
L	a1	a2
R	b1	b2

Histogram example

$$Gini\ Index = \frac{a1+a2}{n}\left[1-\left(\frac{a1}{a1+a2}\right)^2-\left(\frac{a2}{a1+a2}\right)^2\right]+ \frac{b1+b2}{n}\left[1-\left(\frac{b1}{b1+b2}\right)^2-\left(\frac{b2}{b1+b2}\right)^2\right] \quad (5)$$

with data classes a and b :

Tree Pruning

- The Minimum Description Length (MDL) principle is used in the tree pruning strategy of SLIQ. According to MDL, the best model for encoding data is the one that minimizes the sum of the cost of describing the data in terms of the model and the cost of describing the model. In the context of decision tree classifiers, the models are the set of trees obtained by pruning the initial decision tree, and the data is the training set [4].

- If M is a model that encodes the data D, the total cost of the encoding, cost(M, D), is defined as:

$$cost(M, D) = cost(D \mid M) + cost(M)$$

Example

Example 1 [4]

TRAINING DATA

Age	Salary	Class
30	65	G
23	15	B
40	75	G
55	40	B
55	100	G
45	60	G

AFTER PRE-SORTING

Age	Class List Index
23	2
30	1
40	3
45	6
55	5
55	4

Age List

Salary	Class List Index
15	2
40	4
60	6
65	1
75	3
100	5

Salary List

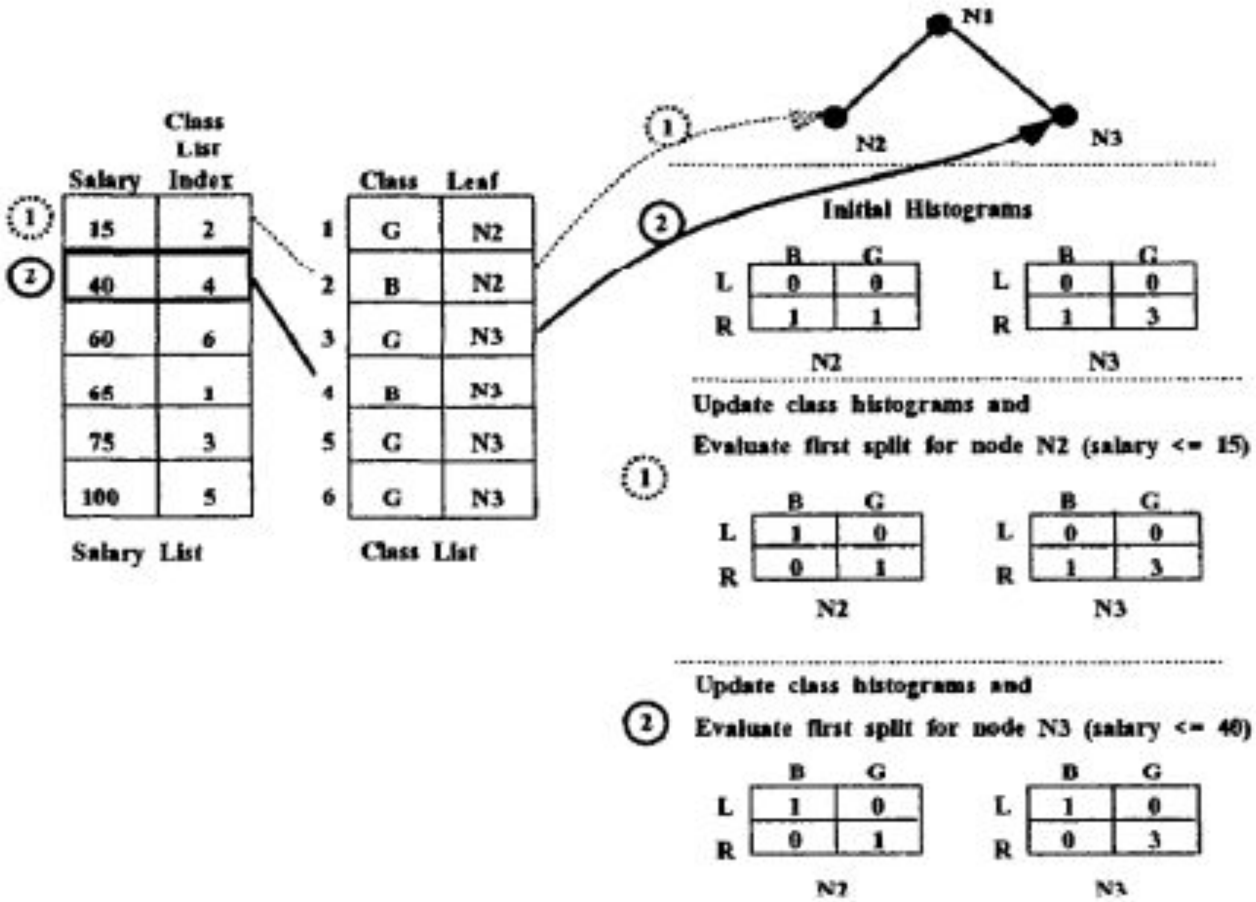
	Class	Leaf
1	G	N1
2	B	N1
3	G	N1
4	B	N1
5	G	N1
6	G	N1

Class List

Example 1 [4]

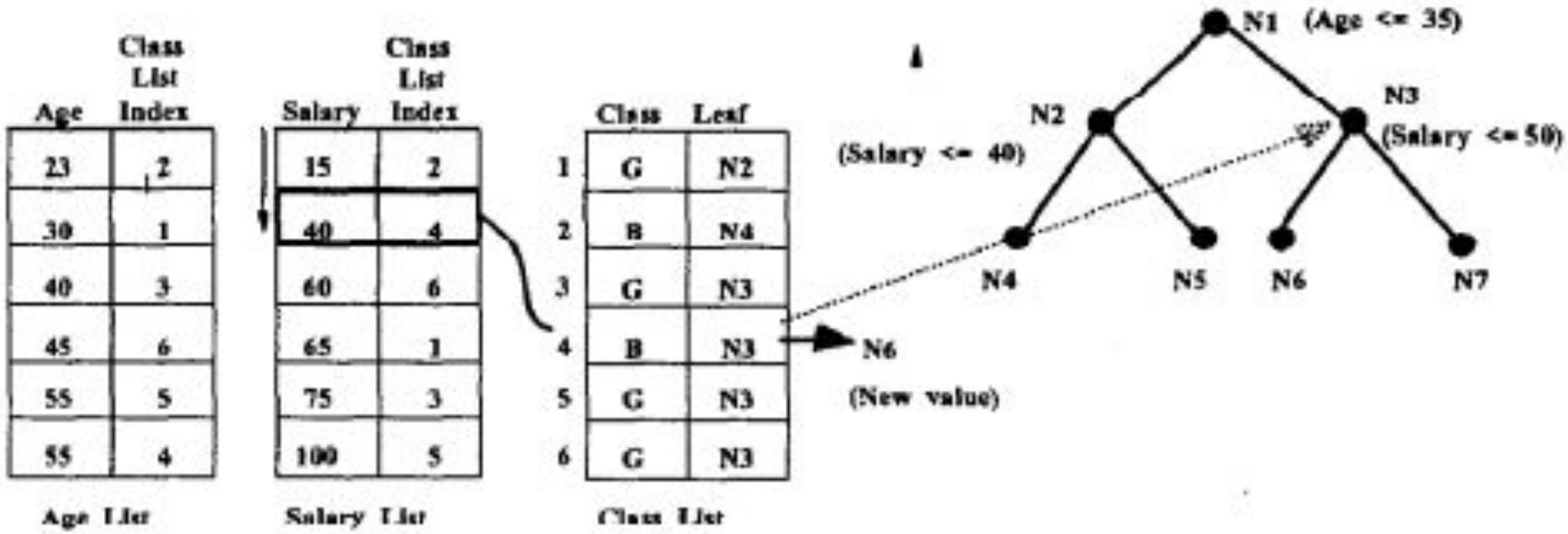
```
EvaluateSplits()
    for each attribute A do
        traverse attribute list of A
        for each value v in the attribute list do
            find the corresponding entry in the class list, and
                hence the corresponding class and the leaf node (say l)
            update the class histogram in the leaf l
            if A is a numeric attribute then
                compute splitting index for test (A ≤ v) for leaf l
        if A is a categorical attribute then
            for each leaf of the tree do
                find subset of A with best split
```

Example 1 [4]



N1
N2
N3
Salary
Class List Index
Class
Leaf
Salary List
Class List
Initial Histograms
Update class histograms and
Evaluate first split for node N2 (salary <= 15)
Update class histograms and
Evaluate first split for node N3 (salary <= 40)

Example 1 [4]



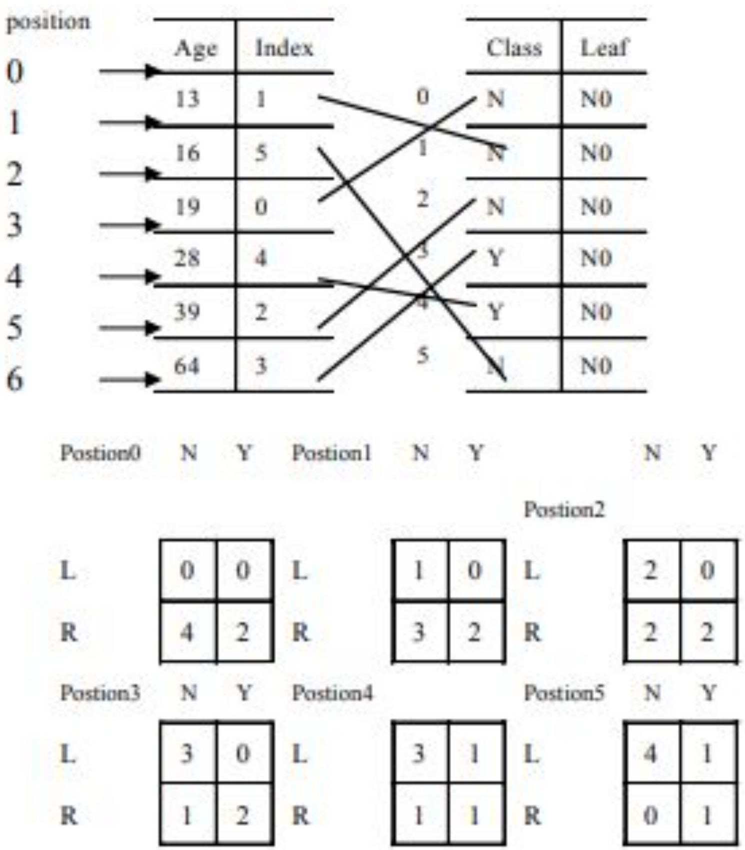
Age
Class List Index
23 2
30 1
40 3
45 6
55 5
55 4
Age List
Salary
Class List Index
15 2
40 4
60 6
65 1
75 3
100 5
Salary List
Class
Leaf
1 G N2
2 B N4
3 G N3
4 B N3
5 G N3
6 G N3
Class List
N6
(New value)
N1 (Age <= 35)
N2
(Salary <= 40)
N3
(Salary <= 50)
N4
N5
N6
N7

Example 2 [3]

Rid	Aag	Credit-rating	class
0	19	Excellent	N
1	13	Fair	N
2	39	Fair	N
3	64	Excellent	Y
4	28	Medium	Y
5	16	Excellent	N

credit-rating list		Age list			class list	
credit-rating	Index	Age	Index	Rid	Class	Leaf
Excellent	0	13	1	0	N	N0
Excellent	3	16	5	1	N	N0
Excellent	5	19	0	2	N	N0
Fair	1	28	4	3	Y	N0
Fair	2	39	2	4	Y	N0
Medium	4	64	3	5	N	N0

Example 2 [3]



S1: $Gini(s_1 < 14.5) = 1 - (\frac{1}{1})^2 = 0$

S2: $Gini(s_2 > 14.5) = 1 - ((\frac{3}{5})^2 + (\frac{2}{5})^2) = 0.48$

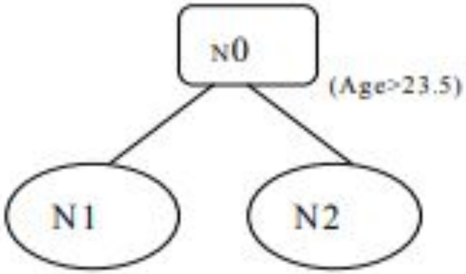
$$Gini\_split(14.5) = \frac{1}{6} \times 0 + \frac{5}{6} \times 0.48 = 0.4$$

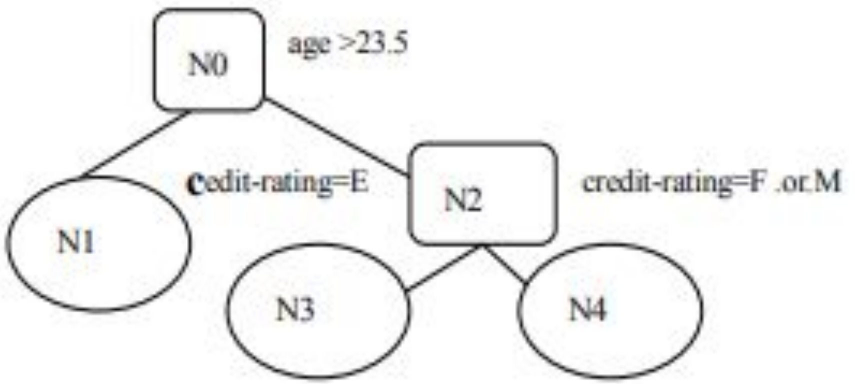
In the same way, $Gini\_split(17.5) = 0.33$

$Gini\_split(23.5) = 0.22$; $Gini\_split(33.5) = 0.42$

Example 2 [3]

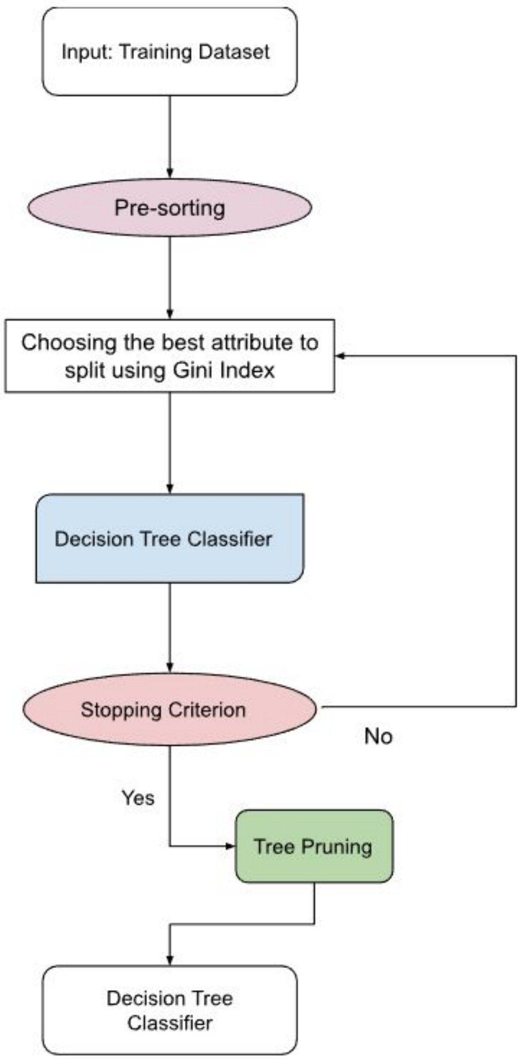
	Class	Leaf
0	N	N1
1	N	N1
2	N	N2
3	Y	N2
4	Y	N2
5	N	N1



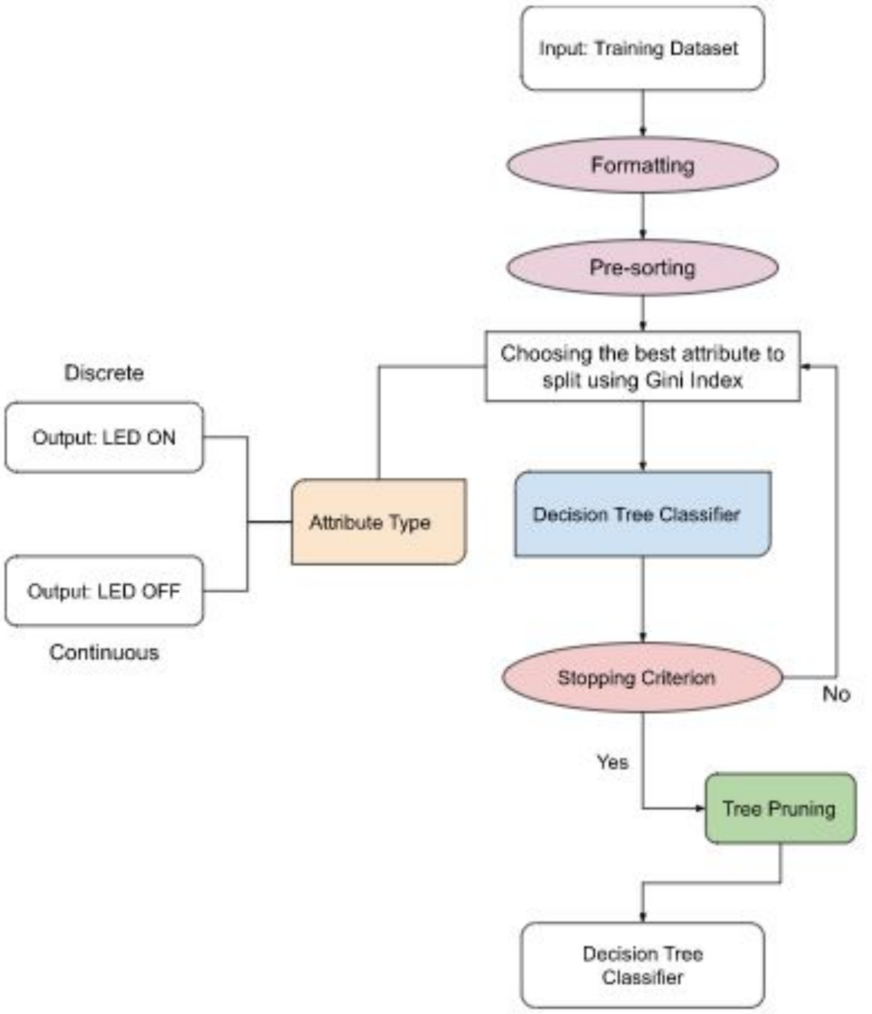


Methodology

Data Flow Graph



Data Flow Graph



Data Flow Graph

Training Data

```
Status,Married,Dependents,Education,Self_Employed,Applicant_Income,Coapplicant_Income,Loan_Amount,Credit_History
Y,0,0,1,0,584900,0,15000000,1
N,1,1,1,0,458300,150800,12800000,1
Y,1,0,1,1,300000,0,6600000,1
Y,1,0,0,0,258300,235800,12000000,1
Y,0,0,1,0,600000,0,14100000,1
Y,1,1,1,1,541700,419600,26700000,1
Y,1,0,0,0,233300,151600,9500000,1
N,1,1,1,0,303600,250400,15800000,0
Y,1,1,1,0,400600,152600,16800000,1
N,1,1,1,0,1284100,1096800,34900000,1
Y,1,1,1,0,320000,70000,7000000,1
Y,1,1,1,0,250000,184000,10900000,1
Y,1,1,1,0,307300,810600,20000000,1
N,0,0,1,0,185300,284000,11400000,1
Y,1,1,1,0,129900,108600,1700000,1
Y,0,0,1,0,495000,0,12500000,1
Y,0,1,0,0,359600,0,10000000,0
N,0,0,1,0,351000,0,7600000,0
N,1,0,0,0,488700,0,13300000,1
Y,1,0,1,0,260000,350000,11500000,1
N,1,0,0,0,766000,0,10400000,0
Y,1,1,1,0,595500,562500,31500000,1
N,1,0,0,0,260000,191100,11600000,0
N,1,1,0,0,336500,191700,11200000,0
N,1,1,1,0,371700,292500,15100000,0
Y,1,0,1,1,956000,0,19100000,1
Y,1,0,1,0,279900,225300,12200000,1
Y,1,1,0,0,422600,104000,11000000,1
N,0,0,0,0,144200,0,3500000,1
Y,0,1,1,0,375000,208300,12000000,1
N,1,1,1,0,416600,336900,20100000,0
N,0,0,1,0,316700,0,7400000,1
N,0,1,1,1,469200,0,10600000,1
Y,1,0,1,0,350000,166700,11400000,1
N,0,1,1,0,1250000,300000,32000000,1
Y,1,0,1,0,227500,206700,0,1
N,1,0,1,0,182800,133000,10000000,0
Y,1,0,1,0,366700,145900,14400000,1
Y,0,0,1,0,416600,721000,18400000,1
Y,0,0,0,0,374800,166800,11000000,1
N,0,0,1,0,360000,0,8000000,1
Y,0,0,1,0,180000,121300,4700000,1
Y,1,0,1,0,240000,0,7500000,0
Y,1,0,1,0,394100,233600,13400000,1
Y,1,0,0,1,469500,0,9600000,1
Y,0,0,1,0,341000,0,8800000,1
```

Presorting the Data

```python
# sort the data and save as .csv file
def sort_data(data, attr):
    return sorted(data, key=lambda x: x[attr])

for attr in attributes:
    sorted_attr_data = sort_data(data, attr)
    sorted_file_path = f"sorted_{attr}.csv"
    write_csv_data(sorted_file_path, header, sorted_attr_data)

def write_csv_data(file_path, header, data):
    with open(file_path, "w") as f:
        f.write(",".join(header) + "\n")
        for row in data:
            f.write(",".join([str(row[h]) for h in header]) + "\n")
```

```
index,Loan_Amount,Credit_Status
36,0,Y
17,10000000,Y
37,10000000,N
21,10400000,N
33,10600000,N
12,10900000,Y
28,11000000,Y
40,11000000,Y
24,11200000,N
14,11400000,N
34,11400000,Y
20,11500000,Y
23,11600000,N
4,12000000,Y
30,12000000,Y
49,12000000,N
27,12200000,Y
16,12500000,Y
2,12800000,N
19,13300000,N
44,13400000,Y
5,14100000,Y
38,14400000,Y
48,14400000,Y
1,15000000,Y
25,15100000,N
8,15800000,N
9,16800000,Y
15,1700000,Y
39,18400000,Y
26,19100000,Y
13,20000000,Y
31,20100000,N
6,26700000,Y
22,31500000,Y
35,32000000,N
```

Code

```python
# Imported file
file_path = "loan_train_em.csv"
attributes = ["Married", "Dependents", "Education", "Self_Employed", "Applicant_Income", "Coapplicant_Income", "Loan_Amount", "Credit_History"]

# Read and format the imported data file
def read_csv_data(file_path):
    with open(file_path, "r") as f:
        lines = f.readlines()
    header = lines[0].strip().split(",")
    data = [dict(zip(["index"] + header, [i] + line.strip().split(","))) for i, line in enumerate(lines[1:], 1)]
    return data

data = read_csv_data(file_path)
header = ["index"] + list(data[0].keys())[1:]
```

```python
# Class for each node, contains informations about the node and the split on the node
class Node:
    def __init__(self, data, attributes, target, depth=0):
        self.data = data
        self.attributes = attributes
        self.target = target
        self.depth = depth
        self.split_attr = None
        self.children = []

# Sort the data
def sort_data(data, attr):
    return sorted(data, key=lambda x: x[attr])
```

Code

```python
# Count the dependent variable for the gini
def count_credit_status(data):
    counts = {"Y": 0, "N": 0}
    for row in data:
        counts[row["Credit_Status"]] += 1
    return counts

# Calculate the gini index for the data and for each attribute
def gini_index(data):
    counts = count_credit_status(data)
    total = sum(counts.values())
    gini = 1 - sum([(count / total) ** 2 for count in counts.values()])
    return gini

def gini_index_for_attribute(data, attr):
    sorted_data = sort_data(data, attr)
    total_gini = 0
    for i in range(len(sorted_data) - 1):
        left_data = sorted_data[: i + 1]
        right_data = sorted_data[i + 1 :]
        left_gini = gini_index(left_data)
        right_gini = gini_index(right_data)
        total_gini += (len(left_data) * left_gini + len(right_data) * right_gini) / len(sorted_data)
    return total_gini / (len(sorted_data) - 1)
```

Code

```python
# Build decision tree classifier using the informations obtained
def build_tree(node, max_depth):
    if stopping_criterion(node, max_depth):
        return

    node.split_attr = best_split(node.data, node.attributes, node.target)
    for child_data in split_data(node.data, node.split_attr):
        child_attributes = list(node.attributes)
        child_attributes.remove(node.split_attr)
        child = Node(child_data, child_attributes, node.target, node.depth + 1)
        node.children.append(child)
        build_tree(child, max_depth)
```

Code

```python
# LED on if gini exist, LED off if gini does not exist
def is_continuous_attribute(attr):
    continuous_attributes = ["Applicant_Income", "Coapplicant_Income", "Loan_Amount"]
    return attr in continuous_attributes

led = machine.Pin(25, machine.Pin.OUT)

def delay():
    for i in range(0, 195785):
        pass

def response():
    response = sys.stdin.readline()
    if response == "\n":
        pass

for attr in attributes:
    response()
    if is_continuous_attribute(attr):
        gini = 0
        led.off()
        print(f"Gini index for {attr}: {gini}")
        delay()
    else:
        gini = gini_index_for_attribute(data, attr)
        led.on()
        print(f"Gini index for {attr}: {gini}")
        delay()
        led.off()
    response()
```

Alternative Code

```
# Get the sorted indices of the target attribute values
sorted_indices = sorted(range(len(dataset[target_attribute])), key=lambda i: dataset[target_attribute][i])

for attribute in dataset:
    if attribute != target_attribute:
        # Initialize the attribute list for the current attribute
        values = [dataset[attribute][i] for i in sorted_indices]
        classes = [dataset[target_attribute][i] for i in sorted_indices]
        attribute_lists[attribute] = AttributeList(attribute, values, classes)

return attribute_lists
```

Result

Result

Shell

```
>>> %Run -c $EDITOR_CONTENT

 Gini index for Married: 0.4624609

 Gini index for Dependents: 0.4569524

 Gini index for Education: 0.4588546

 Gini index for Self_Employed: 0.4560198

 Gini index for Applicant_Income: 0

 Gini index for Coapplicant_Income: 0

 Gini index for Loan_Amount: 0

 Gini index for Credit_History: 0.4034547

>>>
```

References

[1] Huacheng Zhang, Wu Xie, "Improvement of SLIQ Algorithm and its Application in Evaluation", Genetic and Evolutionary Computing 2009. WGEC '09. 3rd International Conference on, pp. 77-80, 2009.

[2] Xie Wu, Huacheng Zhang, Huimin Zhang, "Study of comprehensive evaluation metho of undergraduates based on data mining", Intelligent Computing and Integrated Systems (ICISS) 2010 International Conference on, pp. 541-543, 2010.

[3] Hongwen Yan, Rui Ma and Xiaojiao Tong, "SLIQ in data mining and application in the generation unit's bidding decision system of electricity market," 2005 International Pow Engineering

[4] Mehta, M., Agrawal, R., Rissanen, J. (1996). SLIQ: A fast scalable classifier for data mining. In: Apers, P., Bouzeghoub, M., Gardarin, G. (eds) Advances in Database Technology — EDBT '96. EDBT 1996. Lecture Notes in Computer Science, vol 1057. Springer, Berlin, Heidelberg.

[5] S. Sivagama Sundhari, "A knowledge discovery using decision tree by Gini coefficient 2011 International Conference on Business, Engineering and Industrial Applications, 2011, pp. 232-235.

[6] N. Prasad, P.K. Reddy and M. M. Naidu, "A Novel Decision Tree Approach for the Prediction of Precipitation Using Entropy in SLIQ," 2013 UKSim 15th International Conference on Computer Modelling and Simulation, 2013, pp. 209-217

[7] Hongwen Yan, Rui Ma and Xiaojiao Tong, "SLIQ in data mining and application in the generation unit's bidding decision system of electricity market," 2005 International Power Engineering Conference, 2005, pp. 1-137.

[8] B. Chandra and V.P. Paul, "A Robust Algorithm for Classification Using Decision Trees," 2006 IEEE Conference on Cybernetics and Intelligent Systems, 2006, pp. 1-5

Demo

Thank You