3. AI AND MACHINE LEARNING VTU LAB | READ NOW

VTULOOP - October 20, 2021 - Machine Learning Lab - 0 Comments

MACHINE LEARNING VTU LAB

Program -3] WRITE A PROGRAM TO DEMONSTRATE THE WORKING OF THE DECISION TREE BASED ID3 ALGORITHM. USE AN APPROPRIATE DATA SET FOR BUILDING THE DECISION TREE AND APPLY THIS KNOWLEDGE TO CLASSIFY A NEW SAMPLE.

Program Code- lab3.py

Raw Copy Extern

```
import numpy as np
import math
import csv

def read_data(filename):
    with open(filename, 'r') as csvfile:
        datareader = csv.reader(csvfile, delimiter=',')
        headers = next(datareader)
        metadata = []
        traindata = []
        for name in headers:
            metadata.append(name)
        for row in datareader:
            traindata.append(row)

    return (metadata, traindata)

class Node:
    def __init__(self, attribute):
        self.attribute = attribute
        self.children = []
        self.answer = ""
```

```
    count = np.zeros((items.shape[0], 1), dtype=np.int32)

    for x in range(items.shape[0]):
        for y in range(data.shape[0]):
            if data[y, col] == items[x]:
                count[x] += 1

    for x in range(items.shape[0]):
        dict[items[x]] = np.empty((int(count[x]), data.shape[1]), dtype="|S32")
        pos = 0
        for y in range(data.shape[0]):
            if data[y, col] == items[x]:
                dict[items[x]][pos] = data[y]
                pos += 1
        if delete:
            dict[items[x]] = np.delete(dict[items[x]], col, 1)

    return items, dict

def entropy(S):
    items = np.unique(S)

    if items.size == 1:
        return 0

    counts = np.zeros((items.shape[0], 1))
    sums = 0

    for x in range(items.shape[0]):
        counts[x] = sum(S == items[x]) / (S.size * 1.0)

    for count in counts:
        sums += -1 * count * math.log(count, 2)
    return sums

def gain_ratio(data, col):
    items, dict = subtables(data, col, delete=False)

    total_size = data.shape[0]
    entropies = np.zeros((items.shape[0], 1))
    intrinsic = np.zeros((items.shape[0], 1))

    for x in range(items.shape[0]):
        ratio = dict[items[x]].shape[0]/(total_size * 1.0)
        entropies[x] = ratio * entropy(dict[items[x]][:, -1])
        intrinsic[x] = ratio * math.log(ratio, 2)

    total_entropy = entropy(data[:, -1])
    iv = -1 * sum(intrinsic)

    for x in range(entropies.shape[0]):
        total_entropy -= entropies[x]

    return total_entropy / iv

def create_node(data, metadata):
    if (np.unique(data[:, -1])).shape[0] == 1:
        node = Node("")
        node.answer = np.unique(data[:, -1])[0]
        return node

    gains = np.zeros((data.shape[1] - 1, 1))

    for col in range(data.shape[1] - 1):
        gains[col] = gain_ratio(data, col)

    split = np.argmax(gains)

    node = Node(metadata[split])
    metadata = np.delete(metadata, split, 0)

    items, dict = subtables(data, split, delete=True)

    for x in range(items.shape[0]):
        child = create_node(dict[items[x]], metadata)
        node.children.append((items[x], child))

    return node

def empty(size):
    s = ""
    for x in range(size):
        s += "   "
    return s
```

```
        for value, n in node.children:
            print(empty(level + 1), value)
            print_tree(n, level + 2)

    metadata, traindata = read_data("tennisdata.csv")
    data = np.array(traindata)
    node = create_node(data, metadata)
    print_tree(node, 0)
```

MACHINE LEARNING Program Execution – lab3.ipynb

Jupyter Notebook program execution.

```
import numpy as np
import math
import csv
```

```
def read_data(filename):
    with open(filename, 'r') as csvfile:
        datareader = csv.reader(csvfile, delimiter=',')
        headers = next(datareader)
        metadata = []
        traindata = []
        for name in headers:
            metadata.append(name)
        for row in datareader:
            traindata.append(row)

    return (metadata, traindata)
```

```
class Node:
    def __init__(self, attribute):
        self.attribute = attribute
        self.children = []
        self.answer = ""

    def __str__(self):
        return self.attribute
```

```
def subtables(data, col, delete):
    dict = {}
    items = np.unique(data[:, col])
    count = np.zeros((items.shape[0], 1), dtype=np.int32)

    for x in range(items.shape[0]):
        for y in range(data.shape[0]):
            if data[y, col] == items[x]:
                count[x] += 1

    for x in range(items.shape[0]):
        dict[items[x]] = np.empty((int(count[x]), data.shape[1]), dtype="|S32")
        pos = 0
        for y in range(data.shape[0]):
            if data[y, col] == items[x]:
                dict[items[x]][pos] = data[y]
                pos += 1
        if delete:
            dict[items[x]] = np.delete(dict[items[x]], col, 1)

    return items, dict
```

```
def entropy(S):
    items = np.unique(S)

    if items.size == 1:
        return 0

    counts = np.zeros((items.shape[0], 1))
    sums = 0

    for x in range(items.shape[0]):
        counts[x] = sum(S == items[x]) / (S.size * 1.0)

    for count in counts:
        sums += -1 * count * math.log(count, 2)
    return sums
```

```
def gain_ratio(data, col):
    items, dict = subtables(data, col, delete=False)

    total_size = data.shape[0]
    entropies = np.zeros((items.shape[0], 1))
    intrinsic = np.zeros((items.shape[0], 1))
```

```
    total_entropy = entropy(data[:, -1])
    iv = -1 * sum(intrinsic)

    for x in range(entropies.shape[0]):
        total_entropy -= entropies[x]

    return total_entropy / iv
```

```
def create_node(data, metadata):
    if (np.unique(data[:, -1])).shape[0] == 1:
        node = Node("")
        node.answer = np.unique(data[:, -1])[0]
        return node

    gains = np.zeros((data.shape[1] - 1, 1))

    for col in range(data.shape[1] - 1):
        gains[col] = gain_ratio(data, col)

    split = np.argmax(gains)

    node = Node(metadata[split])
    metadata = np.delete(metadata, split, 0)

    items, dict = subtables(data, split, delete=True)

    for x in range(items.shape[0]):
        child = create_node(dict[items[x]], metadata)
        node.children.append((items[x], child))

    return node
```

```
def empty(size):
    s = ""
    for x in range(size):
        s += "   "
    return s

def print_tree(node, level):
    if node.answer != "":
        print(empty(level), node.answer)
        return
    print(empty(level), node.attribute)
    for value, n in node.children:
        print(empty(level + 1), value)
        print_tree(n, level + 2)
```

```
metadata, traindata = read_data("tennisdata.csv")
data = np.array(traindata)
node = create_node(data, metadata)
print_tree(node, 0)
```

Outlook

Overcast

b'Yes'

Rainy

Windy

b'False'

b'Yes'

b'True'

b'No'

Sunny

Humidity

b'High'

b'No'

b'Normal'

b'Yes'

Alternative – LAB 3 Alt.ipynb

```
# Import neccessary libaries
import pandas as pd
```

```
data = pd.read_csv('tennisdata.csv')
print("The first 5 values of data is \n",data.head())
```

The first 5 values of data is

Outlook Temperature Humidity Windy PlayTennis

0 Sunny Hot High False No

1 Sunny Hot High True No

2 Overcast Hot High False Yes

3 Rainy Mild High False Yes

4 Rainy Cool Normal False Yes

```
# Obtain Train data and Train output
X = data.iloc[:,:-1]
print("\nThe first 5 values of Train data is \n",X.head())
```

The first 5 values of Train data is

Outlook Temperature Humidity Windy

0 Sunny Hot High False

1 Sunny Hot High True

2 Overcast Hot High False

3 Rainy Mild High False

4 Rainy Cool Normal False

```
y = data.iloc[:,-1]
print("\nThe first 5 values of Train output is \n",y.head())
```

The first 5 values of Train output is

0 No

1 No

2 Yes

3 Yes

4 Yes

Name: PlayTennis, dtype: object

```
# Convert them in numbers
le_outlook = LabelEncoder()
X.Outlook =  le_outlook.fit_transform(X.Outlook)

le_Temperature = LabelEncoder()
X.Temperature =  le_Temperature.fit_transform(X.Temperature)

le_Humidity = LabelEncoder()
X.Humidity =  le_Humidity.fit_transform(X.Humidity)

le_Windy = LabelEncoder()
X.Windy =  le_Windy.fit_transform(X.Windy)
print("\nNow the Train data is",X.head())
```

Now the Train data is Outlook Temperature Humidity Windy

0 2 1 0 0

1 2 1 0 1

2 0 1 0 0

3 1 2 0 0

4 1 0 1 0

```
le_PlayTennis = LabelEncoder()
y =  le_PlayTennis.fit_transform(y)
print("\nNow the Train data is\n",y)
```

Now the Train data is

[0 0 1 1 1 0 1 0 1 1 1 1 1 0]

```
## Function to encode input
def labelEncoderForInput(list1):
    list1[0] =  le_outlook.transform([list1[0]])[0]
    list1[1] =  le_Temperature.transform([list1[1]])[0]
    list1[2] =  le_Humidity.transform([list1[2]])[0]
    list1[3] =  le_Windy.transform([list1[3]])[0]
    return [list1]

## predict for an input
inp = ["Rainy","Mild","High","False"]
inp1=["Rainy","Cool","High","False"]
pred1 = labelEncoderForInput(inp1)
y_pred = classifier.predict(pred1)
y_pred

print("\nfor input {0}, we obtain {1}".format(inp1, le_PlayTennis.inverse_transform(y_
```

for input [1, 0, 0, 0], we obtain Yes

Download the dataset

Dataset

TAGS: ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING, VTU CSE LAB

← Previous Post
2. AI AND MACHINE LEARNING VTU LAB | READ NOW

Next Post →
4. AI AND MACHINE LEARNING VTU LAB | READ NOW

YOU MIGHT ALSO LIKE

Machine Learning AO* Algorithm | Read Now

2. AI AND MACHINE LEARNING VTU LAB | READ

Leave a Reply

Your comment here...

Name (required)

Email (required)

Website

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Contact Info

VTULOOP shares the free digital notes, Text books, Tutorials, Tools, etc for engineering Students.

Address:
Bangalore India

Phone:
not available

Email:
support@vtuloop.com

Website:
VTULOOP.COM

Recent Posts

Zoho Computer Programming Questions and Answers
MARCH 14, 2022 / 0 COMMENTS

Apriori Algorithm | Read Now
FEBRUARY 23, 2022 / 0 COMMENTS

Association Rule Learning | Read Now
FEBRUARY 23, 2022 / 0 COMMENTS

Store

- Women
- Men
- Brands A - Z
- Latest fashion arrivals

Useful Links

- About Us
- Contact Us
- Privacy Policy
- Disclaimer

Tags

1ST AND 2ND SEM TEXTBOOK
ENGINEERING TEXTBOOKS
MATHEMATICS TEXTBOOK
PHYSICS TEXTBOOK

Follow Us