H2-CART

Team Members: Vu Ngoc Nguyen, Xinyue Xu, Avinash Ahuja

Introduction

H2 is an open-source pure Java SQL database. The small footprint of H2 (~ 1 MB) and its capability to run on embedded systems have made it very popular. H2 has a web based GUI along with shell access that allow for interaction with the database.

Statistical and machine learning, have become a key part of many workflows. It is used in fields ranging from Economics, Insurance to Engineering and Natural Sciences. **Our goal is to integrate machine learning functions such as classification and regression into the database system.** Since database systems are widely adopted for storing data, by natively introducing ML functions into the database makes their adoption a lot less cumbersome and the workflow more streamlined.

We have chosen to implement Classification using Decision Trees.

Implemented Functions

To integrate Classification ability into the database, we added the following functions and files to H2's source code.

/src/main/org/h2/util/ID3.java :

This file contains all the code required to build decision trees (DTs). It includes various helper functions (such as entropy calculation, tree splitting etc) that are required to build DTs.

/src/main/org/h2/util/ Function.java :

The following functions were added to this file to enable SQL style queries

DECISIONTREE : This function enables SQL style command to build a DT.

Syntax: `CALL DECISIONTREE(TABLENAME, D1, D2, D3 ... Dn, L);`

where: D1 to Dn are the features and L is the label.

CLASSIFY: This function classifies the data inserted in the database and stores it the designated column.

Syntax :
```
ALTER TABLE 'TABLENAME' ADD COLUMN PREDICTION VARCHAR(255) AS
CLASSIFY(TABLENAME, D1, D2, D3 … Dn, L) AFTER L
```

Workflow

In the following section, we have described the workflow for our component along with an example.

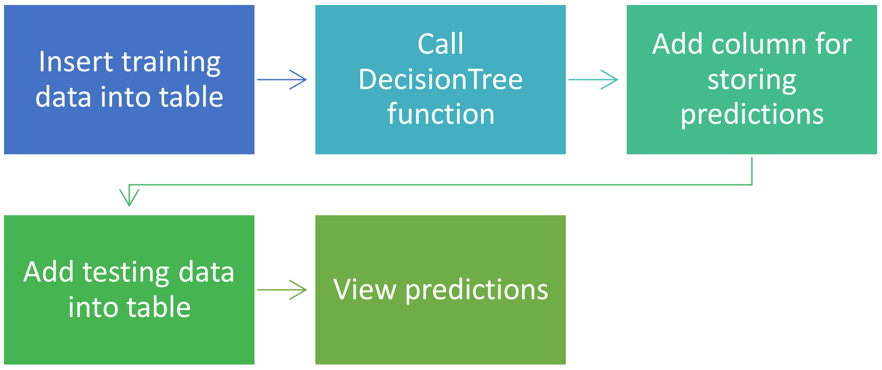
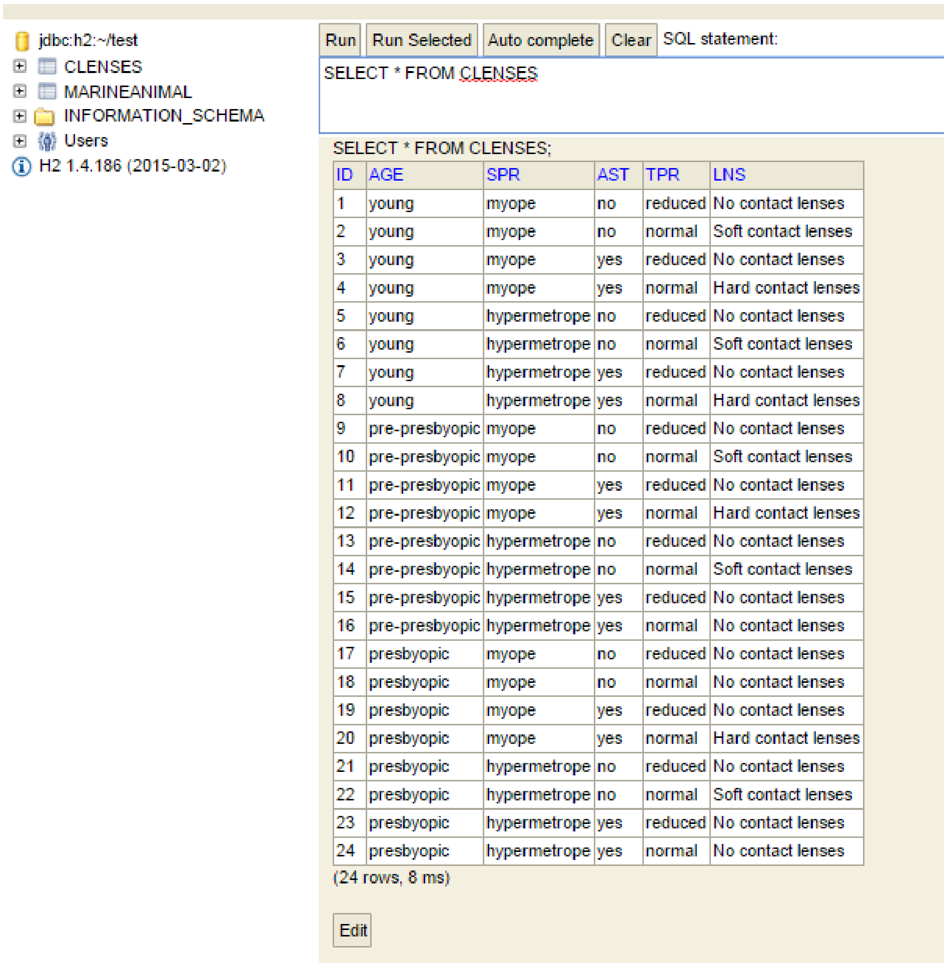

Insert training data into table
Call DecisionTree function
Add column for storing predictions
Add testing data into table
View predictions

Figure 1: Classification workflow

Step 1: Insert the data training data into the database. The current implementation of the classification function works only with categorical data. Figure 2 shows the dataset that we will be working with. This dataset was obtained from https://archive.ics.uci.edu/ml/datasets/Lenses.

Figure 2: Lens dataset



```
SELECT * FROM CLENSES
```

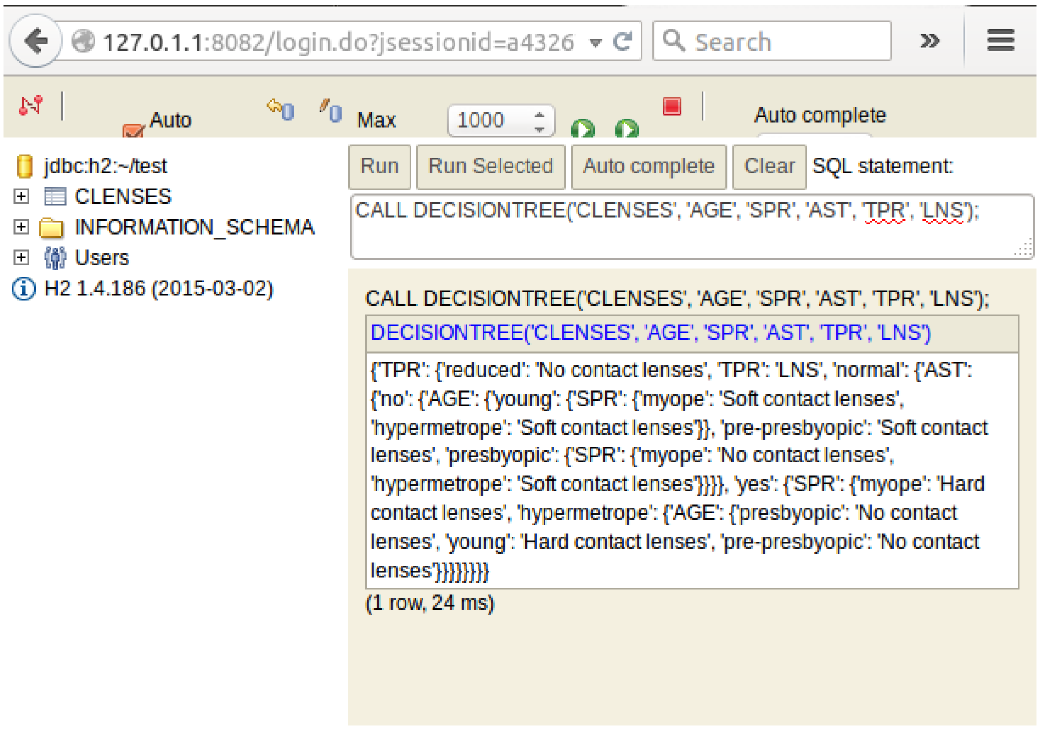
SELECT * FROM CLENSES;

ID	AGE	SPR	AST	TPR	LNS
1	young	myope	no	reduced	No contact lenses
2	young	myope	no	normal	Soft contact lenses
3	young	myope	yes	reduced	No contact lenses
4	young	myope	yes	normal	Hard contact lenses
5	young	hypermetrope	no	reduced	No contact lenses
6	young	hypermetrope	no	normal	Soft contact lenses
7	young	hypermetrope	yes	reduced	No contact lenses
8	young	hypermetrope	yes	normal	Hard contact lenses
9	pre-presbyopic	myope	no	reduced	No contact lenses
10	pre-presbyopic	myope	no	normal	Soft contact lenses
11	pre-presbyopic	myope	yes	reduced	No contact lenses
12	pre-presbyopic	myope	yes	normal	Hard contact lenses
13	pre-presbyopic	hypermetrope	no	reduced	No contact lenses
14	pre-presbyopic	hypermetrope	no	normal	Soft contact lenses
15	pre-presbyopic	hypermetrope	yes	reduced	No contact lenses
16	pre-presbyopic	hypermetrope	yes	normal	No contact lenses
17	presbyopic	myope	no	reduced	No contact lenses
18	presbyopic	myope	no	normal	No contact lenses
19	presbyopic	myope	yes	reduced	No contact lenses
20	presbyopic	myope	yes	normal	Hard contact lenses
21	presbyopic	hypermetrope	no	reduced	No contact lenses
22	presbyopic	hypermetrope	no	normal	Soft contact lenses
23	presbyopic	hypermetrope	yes	reduced	No contact lenses
24	presbyopic	hypermetrope	yes	normal	No contact lenses

(24 rows, 8 ms)

This dataset has 4 feature columns namely Age (AGE), Spectacle Prescription (SPR), Astigmatic (AST), Tear Production Rate (TPR) and a label column Lens (LNS). **Note:** This step can be skipped if data already exists in separate training and testing tables.

Step 2: To generate a decision tree based on the data, we use the function DECISIONTREE as shown in Figure 3. The decision tree is created and stored in JSON format. The syntax has been discussed in the previous section.

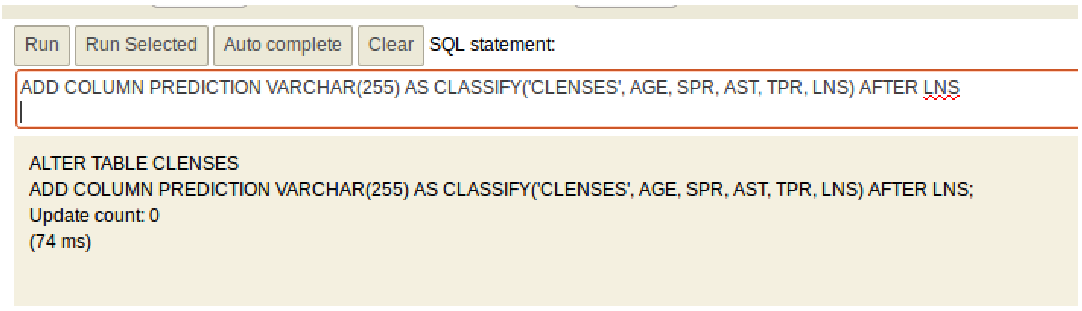


```
CALL DECISIONTREE('CLENSES', 'AGE', 'SPR', 'AST', 'TPR', 'LNS');
```

```
CALL DECISIONTREE('CLENSES', 'AGE', 'SPR', 'AST', 'TPR', 'LNS');
DECISIONTREE('CLENSES', 'AGE', 'SPR', 'AST', 'TPR', 'LNS')
{'TPR': {'reduced': 'No contact lenses', 'TPR': 'LNS', 'normal': {'AST': {'no': {'AGE': {'young': {'SPR': {'myope': 'Soft contact lenses', 'hypermetrope': 'Soft contact lenses'}}, 'pre-presbyopic': 'Soft contact lenses', 'presbyopic': {'SPR': {'myope': 'No contact lenses', 'hypermetrope': 'Soft contact lenses'}}}}, 'yes': {'SPR': {'myope': 'Hard contact lenses', 'hypermetrope': {'AGE': {'presbyopic': 'No contact lenses', 'young': 'Hard contact lenses', 'pre-presbyopic': 'No contact lenses'}}}}}}}}
(1 row, 24 ms)
```

Figure 3: Generating a Decision Tree

Step 3: Now we add a column in the existing table as CLASSIFY as shown in Figure 4. This column can be added in a new table as well if the user wants to separate the training and testing data.

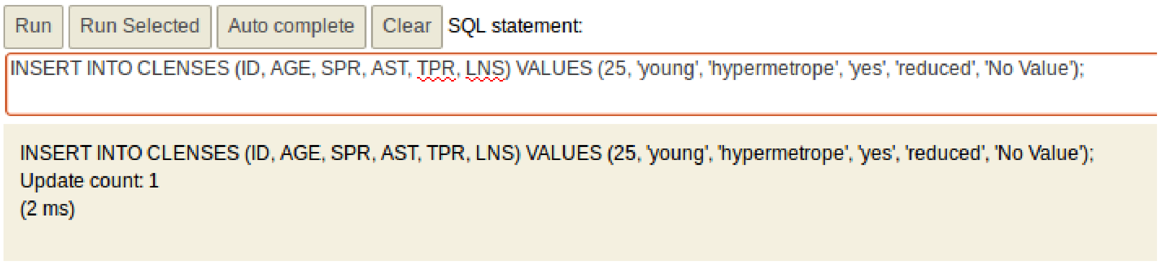


```
ADD COLUMN PREDICTION VARCHAR(255) AS CLASSIFY('CLENSES', AGE, SPR, AST, TPR, LNS) AFTER LNS
```

```
ALTER TABLE CLENSES
ADD COLUMN PREDICTION VARCHAR(255) AS CLASSIFY('CLENSES', AGE, SPR, AST, TPR, LNS) AFTER LNS;
Update count: 0
(74 ms)
```

Figure 4: Adding prediction column

Step 4: Now the user can insert the test data (the data that needs to be classified) into the table that has the prediction column. If the data already exists in a separate table, it can be classified by adding the prediction column in **Step 3** to this table. This is shown in Figure 5. **Note**: Since the type of lens is being predicted here the LNS column need not have any data. Here we replace it with 'No Value'

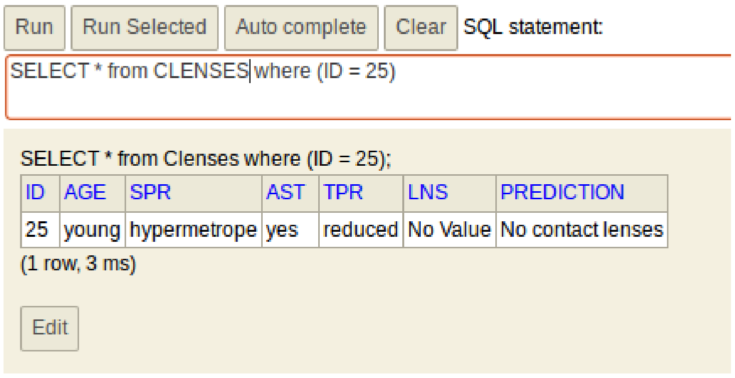


```
INSERT INTO CLENSES (ID, AGE, SPR, AST, TPR, LNS) VALUES (25, 'young', 'hypermetrope', 'yes', 'reduced', 'No Value');
```

```
INSERT INTO CLENSES (ID, AGE, SPR, AST, TPR, LNS) VALUES (25, 'young', 'hypermetrope', 'yes', 'reduced', 'No Value');
Update count: 1
(2 ms)
```

Figure 5: Inserting data into the testing table

Step 5: The results of the prediction can now be viewed in the table containing the test data.



```
SELECT * from CLENSES where (ID = 25)
```

SELECT * from Clenses where (ID = 25);

ID	AGE	SPR	AST	TPR	LNS	PREDICTION
25	young	hypermetrope	yes	reduced	No Value	No contact lenses

(1 row, 3 ms)