Ain Shams University

Faculty Of Engineering

Computer Engineering and Software Systems Program

CSE 385 DataMining

Using Decision Trees for Classification in Data Mining

Submitted by:

Monica Mourad Elia 16p8223

Submitted to:

Dr. Mahmoud Mounir

TA Mohammed A.Farahat

Spring 2020

Abstract

The occurrence of the digital evolution (1950s to the late 1970s) made many companies shift from paperwork to digital computers record keeping, as a result after 50 years of storing huge amount of data daily on computers we started hearing terms like Big Data.

The explosive growth of data lead to the evolution of data mining which uses artificial intelligence, machine learning and statistics to analyze Big Data and extract from it important valuable knowledge that was previously unknown and that can be very useful for making successful market decisions by making accurate data-driven predictions based on customer behaviors.

In this paper, we will discuss the classification function of data mining which is used to classify data into classes. We will mainly focus on the implementation techniques and the use of decision trees and we will prove how they are used to predict the right class for every case in our huge data accurately.

1 Introduction

Big Data arose the problem of dealing with rapid growing data sets that are difficult and nearly impossible to get analyzed by traditional data-processing due to its size and complexity. Big Data has been expanding that it has reached various engineering and science domains (biological, physical, bio-medical sciences).

As a result, database and information technology have been evolving since, attempting to meet the rising demand; it evolved from file processing systems in 1960 into indexed relational database systems accompanied with data access and modeling tools through DBMS query languages in the 1970's that lead eventually to the development of the data mining concept [1].

Data Mining which is also known as "knowledge mining from data" means simply searching for knowledge in data; this knowledge discovery is done through the following steps [1]:

1. **Data cleaning:** removing irrelevant data (noise)
2. **Data integration:** combing different data sources together

Data selection: picking out data that will be useful to analyze from the data warehouse

3. **Data transformation:** change data into a more applicable form to be easily mined
4. **Data mining:** extracting knowledge (useful patterns) using various technologies
5. **Pattern evaluation:** identify the useful patterns that satisfy the needed knowledge measures
6. **Knowledge presentation:** present the achieved knowledge to users

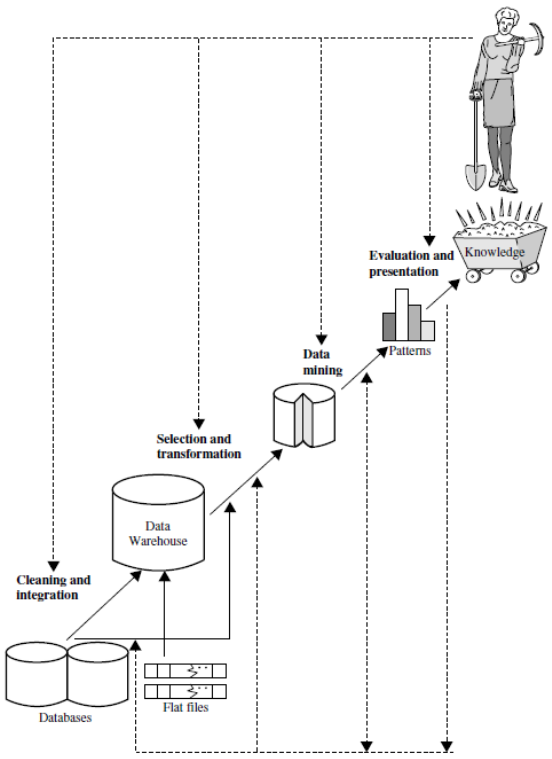


Figure 1 Data mining process

Data mining can mine specific patterns which are called data mining functionalities [1]:

- **Characterization and discrimination**

  Data characterization means describing each class of data in precise terms that summarize its data characteristics; the output is presented in statistical forms such as bar charts, curves and pie charts.

  Data discrimination compares the features of a class data with the features of another class data, it is represented in similar ways to data discrimination but with taking more comparative measures to highlight the difference between the two classes.

- **mining of frequent patterns, associations, and correlations**

  Frequent patterns mean the patterns (item sets, subsequences, substructures) that are repeated frequently in my data like an item set that appear together in a dataset or a subsequence of specific items that are repeated in the same order which form a sequential pattern. Substructures take specific forms like graphs and trees which are built of item sets or subsequences and may be repeated in the data set. Mining these frequent patterns finds valuable associations and correlations within the original data.

- **classification and regression**

  classification analyses data set items (training data) that belong to known classes in order to drive model that can distinguish and describe each class in the data set so that it can be used to predict the class of each unidentified item in the data set.

  This model may take the form of decision trees, neural networks, classification rules or a mathematical formulae.

- **clustering analysis and outlier analysis**

  clustering is the process of grouping data as it tends to put each set item in its group that have a specific class label but unlike classification there is no training data (data set items with known classes in the beginning).

  Outliers are data set items that fall outside the classes that define most of our data set; analysis of this outliers are needed in some applications like fraud detection.

Out of the most advanced technologies that data mining frequently employs is machine learning; there are two main learning techniques that are of our concern [1]:

Supervised learning

This is the **classification** functionality that we mentioned earlier; it means teaching the system through providing it with class labeled data set items the characteristics of each class so that it can create a model (e.g. decision trees) that can classify any new unlabeled data into its correct class.

Unsupervised learning

This is the **clustering** functionality that we mentioned earlier; we don't provide the system with any labeled training data, we let the system discover classes that are within our data.

In this paper we will write about Decision trees and we will prove that it is a very effective supervised machine learning technique that is frequently used in various fields such as engineering, medicine, finance and even marketing due to its wide popularity for being simple, transparent and self-explanatory [3].

In section 2, we will explain in details the definition, mathematical formulation and the terms of the decision trees. Then, in section 3, we will write about constructing the decision trees and how they are related to probability; in section 4, we will see a numerical example that shows the way of building a decision tree. After that, in section 5, we will understand how to use a decision tree. Later on, in section 6, we will show different real life applications that uses decisions trees; we will also classify and analyze a real data set with the help of Weka and python. Finally, in section 7, we will emphasize the research's main points representing the conclusions that we would have reached by the end of our research.

2 Decision Trees

A decision tree is a machine learning tool that is used to predict a qualitative response and also mainly in classifying data items in big data sets into classes. It poses a series of questions about the characteristics of the data items, these questions are presented in a tree-like structure where each node represents one question and the answer to any question that is contained in an internal node must lead to a single child node (another question) or to a leaf node [5].

The questions proposed in the decision tree can be complicated as long as the tree remain self-explanatory and the questions' answers can be traced effectively. Answers should be either values (from a set) like {a, b, c} or yes/no if the questions asks about the variable condition is like for example (Is N < 3 ? , gender is male) [5].

The items' classes are chosen when following the tree from up the root down to the last node in its path (leaf node that doesn't have any children) where the item is assigned to class associated with last node it reached [5].

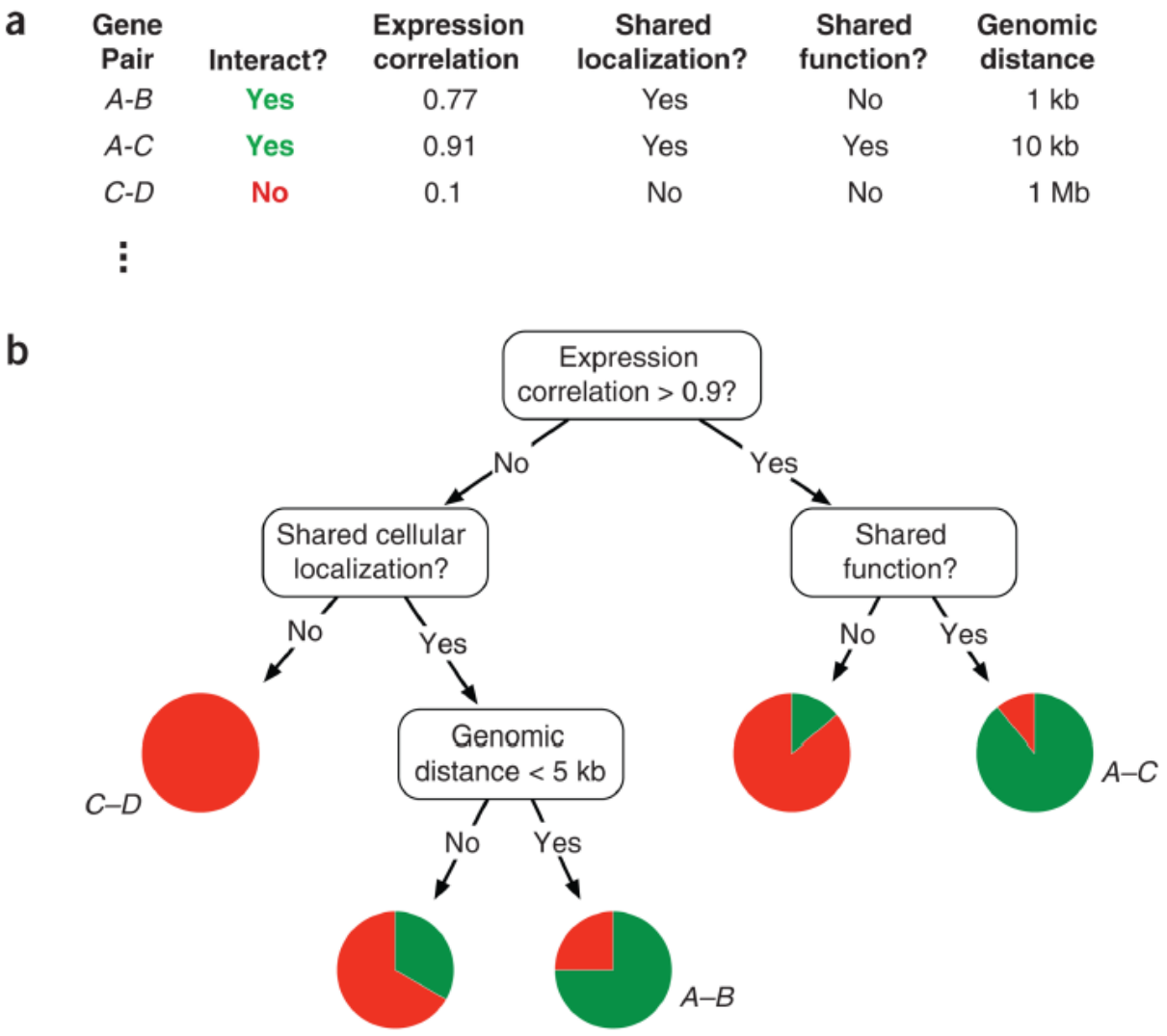


Figure 2 An example of how decision tree can predict protein-protein interactions

Decision trees are easier to understand than other approaches such as neural networks, and extracting the decision rules is far more easy and although sometimes a minor change in input may lead to a huge change in the trees, they are still effectively used in many applications. The reason is mainly due to their flexibility that make them able to handle items that may miss some features also due to the fact that as soon as they are constructed they can be used to classify new items quickly [5].

Mathematical formulation of Decision Trees

In decision tress we split the data into two or more sets based on the most differentiator of our data; there are many different criteria for deciding where to split as we need to make strategic splits that will make our final tree accurate and ensure its nodes' purity. The chosen criteria mathematical formulation must be based upon the target variables, as the nodes' purity increases with respect to the target variables [4].

The most used algorithms in decision trees are [2]:

classification error rate (E) [2]

Classification error rate (E) is the fraction of training data that was observed in this region that don't belong to most common class; it is calculated using the following formula:

$$E = 1 - \max_k(\hat{p}_{mk}).$$

$\hat{p}_{mk}$ refer to the fraction of the training data observations in the mth region that are from the kth class.

Gini index [2]

Gini index (G) measures the total variance across the *K* classes; it is calculated using the following formula:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

The Gini index will have a low value if most $\hat{p}_{mk}$ are close to zero or one. That's why the Gini index is an accurate measure of node *purity*; the smaller the value the purer the tree nodes are.

Entropy [2]

The entropy (D) can be viewed as an alternative to the gini index as they are both very similar numerically; it is calculated using the following formula:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

It is implied that $0 \leq \hat{p}_{mk} \leq 1$, therefore we conclude that $0 \leq -\hat{p}_{mk} \log \hat{p}_{mk}$.

It can be also seen that the entropy will have a low value if most $\hat{p}_{mk}$ are close to zero or one. Therefore, entropy will have a low value if the *m*th node is pure.

Information gain

Information gain is a measure of the entropy or in other words the degree of disorganization of the system; purer nodes require less information to describe them while less pure nodes require more information to be fully described [2].

The information gain is calculated using the following formula which depends on the Entropy:

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

For example: G (play golf, outlook) = E (play golf)- E (play golf, outlook) where T and X are variables. Higher Information Gain means more Entropy was removed

Gain ratio [3]

This algorithm is used to normalize the information gain; it is calculated using the following formula:

$$Gain\ Ratio|(a_i, S) = \frac{\text{Information Gain}(a_i, S)}{Entropy(a_i, S)}$$

Denominator can't be zero or the Gain ratio will be undefined; the ratio may favor the attributes that makes the denominator very small. To calculate the gain ratio, we first calculate the information gain for all the attributes and then their gain ratio; then we select the attribute that gives the best gain ratio.
This algorithm gives more accurate results that the information gain algorithm.

Chi-square [4]

Chi-square is used to find the statistical significance between the differences that exists between the decision tree's parent and child nodes emerging from it. It is defined as the sum of squares of the differences between the expected and observed frequencies of target variable. It is calculated for each node using the following formula:

$$\text{Chi-square} = ((\text{Actual} - \text{Expected})^2 / \text{Expected})^{1/2}$$

The higher the Chi-square the higher the statistical significance between the differences that exists between the decision tree's parent and child nodes emerging from it.

As it can be observed, gini index and entropy is more preferred over classification error rate because it is not sensitive enough to be used in tree-growing. In General, choosing the criteria is based upon our need; if we need to evaluate the quality of a particular split then the gini index or the entropy is used as these approaches are more sensitive to node purity than classification error rate. classification error rate approach is preferable if our goal is the prediction accuracy of the tree [2].

Terminology of the Decision Trees [4]

Root: represents the entire data which will be divided into two or maybe more homogenous sets.

Splitting: The division of a parent node into at least two sub-nodes or maybe more.

Decision node: It is a node that splits into sub-nodes.

Leaf node: It is a node that won't split anymore, sometimes called terminal node.

Pruning: It is the opposite of splitting, it is the removing of the Decision node 's subnodes.

Branch: It is a sub-section (part) of the decision tree, may be called Sub-tree.

Parent node: It is any node that divides into sub-nodes; this node will be considered the parent of the sub-nodes.

Child node: It is a sub-node that emerged from its parent node.

3 Decision trees induction algorithms

The two most popular algorithms that use the splitting and tree pruning criteria mentioned to construct the decision tree are:

ID3

This algorithm is very simple; it is based upon the information gain and uses this criterion to decide where to split. The Information Gain criterion makes the tree stop growing (stop splitting) when the best information gain equals zero or when all data items have a single value of a target feature [3].

As we have seen earlier the information gain = entropy source – entropy of branches and of course calculating probability is a must to calculate the entropy using its equation:

$$\text{Entropy} = -(\text{probability}(a) * \log_2(\text{probability}(a))) - (\text{probability}(b) * \log_2(\text{probability}(b)))$$

The ID3 algorithm neither can handle missing values or numeric attributes nor uses any pruning criterion which is a huge disadvantage. Unfortunately, these are not the only disadvantages as ID3 is notorious of other features such as [3]:

- It can be easily stuck in local optimums as a result it doesn't always lead to an optimal solution; this problem is due to the use of greedy strategy to reach the solution; searching with the help backtracking can avoid this problem.
- overfitting the training data; the algorithm produces small trees but they are not always the smallest possible trees which lead to the overfit problem.
- Only nominal attributes are used; continuous data must be converted to nominal bins to be used.

Although ID3 may have many disadvantages, it was frequently used due to its simplicity. Over time a new advanced algorithm having the same advantage with less drawbacks evolved from the ID3 algorithm; the new algorithm C4.5 became more preferable by many practitioners [3].

C4.5

This algorithm was presented by the same author of the ID3 algorithm; it is based upon the gain ratio and uses this criterion to decide where to split. The gain ratio criterion makes the tree stop growing (stop splitting) when the number of instances to be split is below a certain threshold [3].

Gain ratio's equation involves calculating entropy and of course calculating probability is a must to calculate the entropy using its equation:

$$\text{Entropy} = -(\text{probability}(a) * \log2(\text{probability}(a))) - (\text{probability}(b) * \log2(\text{probability}(b)))$$

One of the main differences and advantages of the C4.5 algorithm over the ID3 algorithm is that the C4.5 can handle numerical attributed in addition to attributes with missing values using the corrected gain ratio criteria; moreover, it applies pruning on the tree after growing it. C4.5 provides many improvements to the ID3 algorithm such as [3]:

- It prunes the tree after growing it by removing the tree's branches that may lead to a decrease in accuracy and replaces them with nodes
- It allows working and dealing with attributes that have missing values
- It can work with continuous valued attributes as it splits the values into two subsets, then it tries and find the best threshold that will result in the maximum gain ratio; the first subset gets all the values that are smaller than the threshold and the second subset takes the remaining values.

A new more updated version of the C4.5 was recently presented; version C5.0 is much more efficient in memory and computation time, provides a much higher speedup than C4.5 and also supports a boosting mechanism that have better predictions performance than C4.5 [3].

Nevertheless, recent studies found that C4.5 have a more accurate consistent performance when comparing it with C4.5 with J48 especially when using small datasets [3].

4 Building a decision tree [6]

Check this small training data set; every item is assigned to either class N or P.

No.	Attributes				Class
	Outlook	Temperature	Humidity	Windy	
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

Figure 3 training data set

We have 14 items; each have four attributes that describes it:

Outlook {sunny, overcast, rain}
Temperature {cool, mild, hot}
Humidity {high, normal}
Winds {true, false}

Our mission is to build a decision tree that can classify any new unlabeled item to its class (P or N) based on its attribute and how they are similar to the common attributes of each class. We will build it depending on the training data in figure 3 where each item has four fixed attribute values and a corresponding known outcome (class).

We will start comparing between the information gain of each attribute.

Step 1

Entropy(class) = -5/14(log 5/14)-9/14(log9/14)=0.5305+0.4098=0.9403

The training data outlook attribute has 5 rain (2N, 3P), 4 overcast (0N, 4P) and 5 sunny (3N, 2P)
Entropy(outlook) = 5/14 H(2,3)+4/14 H(0,4)+5/14 H(3,2)
= 5/14 (-2/5 log (2/5) – 3/5 log (3/5)) + 4/14 (-4/4 log (4/4)) +5/14 (-3/5 log (3/5) – 2/5 log (2/5)) = 0.3468 + 0 +0.3468 = 0.6936
Gain (class, outlook) = Entropy(class) - Entropy(outlook) = 0.9403 – 0.6936 = 0.2467

The training data Temperature attribute has 4 cool (1N, 3P), 6 mild (2N, 4P) and 4 hot (2N, 2P)

Entropy(Temperature) = 4/14 H (1,3) + 6/14 H (2,4) + 4/14 H (2,2)

=4/14 (-1/4 log (1/4) – 3/4 log (3/4)) + 6/14 (-2/6 log (2/6)-4/6 log (4/6) +4/14 (-2/4 log (2/4) – 2/4 log (2/4)) = 0.2318 + 0.3936 + 0.2857 = 0.9111

Gain (class, Temperature) = Entropy(class) - Entropy(Temperature) = 0.9403 – 0.9111 = 0.0292

The training data Humidity attribute has 7 high (4N, 3P) and 7 normal (1N, 6P)

Entropy(Humidity) = 7/14 H (4,3) + 7/14 H (1,6)

=7/14 (-4/7 log (4/7) – 3/7 log (3/7)) + 7/14 (-1/6 log (1/6) - 6/7 log (6/7))

=0.4926 + 0.2958 = 0.7884

Gain (class, Humidity) = Entropy(class) - Entropy(Humidity) = 0.9403 – 0.7884 = 0.1519

The training data winds attribute has 6 true (3N, 3P) and 8 false (2N, 6P)

Entropy(winds) = 6/14 H (3,3) + 8/14 H (2,6)

=6/14 (-3/6 log (3/6) – 3/6 log (3/6)) + 8/14 (-2/8 log (2/8)-6/8 log (6/8))

=0.4286 + 0.4636 = 0.8922

Gain (class, winds) = Entropy(class) - Entropy(winds) = 0.9403 – 0.8922 =0.0481

Attribute	**Information Gain**
Outlook	0.2467
Temperature	0.0292
Humidity	0.1519
winds	0.0481

The above results show that the outlook will be in the root node. Outlook has 3 outputs sunny, rain and overcast; Entropy (overcast) = 4/4 (-4/4 log (4/4)) = 0 as it will always lead to a P.

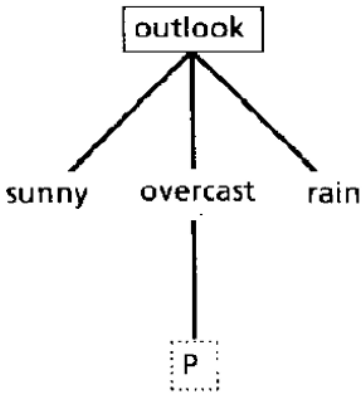


Figure 4 decision tree building –step 1

Step 2

Entropy(sunny) = (-3/5 log (3/5) -2/5 log (2/5)) = 0.971

The training data Temperature attribute has 1 cool (0N, 1P), 2 mild (1N, 1P) and 2 hot (2N, 0P)

Entropy(Temperature) = 1/5 H (0,1) + 2/5 H (1,1) + 2/5 H (2,0)

=1/5 (-1 log (1)) +2/5 (-1/2 log (1/2) -1/2 log (1/2)) + 2/5 (-2/2 log (2/2))

= 0 + 0.4 + 0 = 0.4

Gain (sunny, Temperature) = Entropy(sunny) - Entropy(Temperature) = 0.971 – 0.4 = 0.571

The training data humidity attribute has 3 high (3N, 0P) and 2 normal (0N, 2P)

Entropy(humidity) = 3/5 H (3,0) + 2/5 H (0,2)

=3/5 (-1 log (1)) +2/5 (-1 log (1))

= 0 + 0 = 0

Gain (sunny, humidity) = Entropy(sunny) - Entropy(humidity) = 0.971 – 0 = 0.971

The training data windy attribute has 3 false (2N, 1P) and 2 true (1N, 1P) Entropy(winds) = 3/5 H (2,1) + 2/5 H (1,1)

=3/5 (-2/3 log (2/3)-1/3 log (1/3)) +2/5 (-1/2 log (1/2)-1/2 log (1/2))

= 0.551 + 0.26 = 0.811

Gain (sunny, winds) = Entropy(sunny) - Entropy(winds) = 0.971 – 0.811 = 0.16

Attribute	**Information Gain**
Temperature	0.571
Humidity	0.971
winds	0.16

The above results show that the humidity will be used in the splitting of the items with sunny outlook. Entropy (high) = 3/5 (-3/3 log (3/3)) = 0 as it always leads to N and Entropy (normal) = 2/5 (-2/2 log (2/2)) = 0 as it always leads to P.

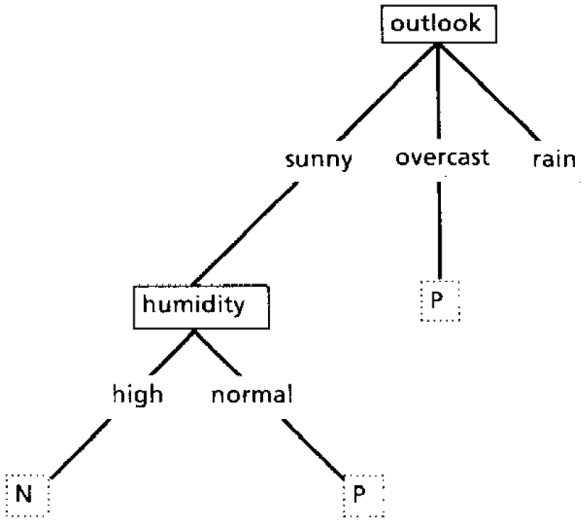


Figure 5 decision tree building - step 2

Step 3

Entropy(rain) = (-3/5 log (3/5) -2/5 log (2/5)) = 0.971

The training data Temperature attribute has 2 cool (1N, 1P), 3 mild (1N, 2P) and 0 hot
Entropy(Temperature) = 2/5 H (1,1) + 3/5 H (1,2)
= 2/5 (-1/2 log (1/2) -1/2 log (1/2)) + 3/5 (-1/3 log (1/3) -2/3 log (2/3))
= 0.4 + 0.551 = 0.951
Gain (rain, Temperature) = Entropy(rain) - Entropy(Temperature) = 0.971 – 0.951 = 0.02

The training data winds attribute has 3 false (0N ,3P) and 2 true (2N,0P)
Entropy(winds) = 3/5 H (0,3) + 2/5 H (2,0)
= 3/5 (-3/3 log (3/3)) + 2/5 (-2/2 log (2/2))
= 0
Gain (rain, winds) = Entropy(rain) - Entropy(winds) = 0.971 – 0= 0.971

Attribute	Information Gain
Temperature	0.02
winds	0.971

The above results show that the winds will be used in the splitting of the items with rain outlook. Entropy (false) = 3/5 (-3/3 log (3/3)) = 0 as it always leads to P and Entropy (true) = 2/5 (-2/2 log (2/2)) = 0 as it always leads to N.

5 Using a Decision tree

After building the complete decision tree (figure 6) it is time to conclude from it the decision rules which describes the path that an unlabeled data item should take to be classified to either P or N class at the end.

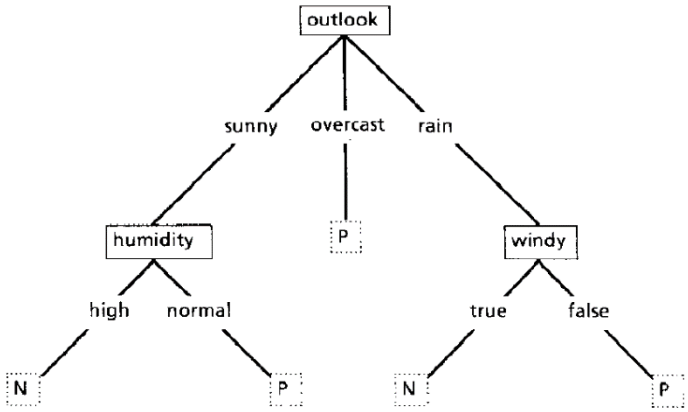


Figure 6 constructed decision tree

Rule 1: **If** the (outlook = sunny) and (humidity = normal) **Then** item is classified as P.
Rule 2: **If** the (outlook = sunny) and (humidity = high) **Then** item is classified as N.
Rule 3: **If** the (outlook = overcast) **Then** the item is always classified as P.
Rule 4: **If** the (outlook = rain) and (windy = true) **Then** the item is classified as N.
Rule 5: **If** the (outlook = rain) and (windy = false) **Then** the item is classified as P.

Now after completing our model we can classify the following unlabeled data as follows:

Number	outlook	temperature	Humidity	windy
1	Sunny	hot	high	true
2	Overcast	hot	high	true
3	Rain	hot	high	true
4	Sunny	hot	normal	true
5	Overcast	hot	normal	true
6	Rain	hot	normal	true
7	Sunny	hot	high	false
8	Overcast	hot	high	false
9	Rain	hot	high	false
10	Sunny	hot	normal	false
11	Overcast	hot	normal	false
12	Rain	hot	normal	false
13	Sunny	Cold	high	true
14	Overcast	Cold	high	true
15	Rain	Cold	high	true
16	Sunny	Cold	normal	true
17	Overcast	Cold	normal	true
18	Rain	Cold	normal	true
19	Sunny	Cold	high	false
20	Overcast	Cold	high	false
21	Rain	Cold	high	false
22	Sunny	Cold	normal	false
23	Overcast	Cold	normal	false
24	Rain	Cold	normal	false

The previous 24 items will be classified as following:

Number	Decision rule	class
1	Rule 2	N
2	Rule 3	P
3	Rule 4	N
4	Rule 1	P
5	Rule 3	P
6	Rule 4	N
7	Rule 2	N
8	Rule 3	P
9	Rule 5	P
10	Rule 1	P
11	Rule 3	P
12	Rule 5	P
13	Rule 2	N
14	Rule 3	P
15	Rule 4	N
16	Rule 1	P
17	Rule 3	P
18	Rule 4	N
19	Rule 2	N
20	Rule 3	P
21	Rule 5	P
22	Rule 1	P
23	Rule 3	P
24	Rule 5	P

6 Applications of Decision Trees in real life

This dataset [a] is used for binary classification (room occupancy) based upon Temperature, Humidity, Light and CO2. Ground-truth occupancy was obtained from time stamped pictures that were taken every minute.

o Exploratory data analysis for the collected dataset [e]

Measures of central tendency

We used python to analyze our data set; the source code is below [e]

Mean

```
Temperature         20.906212
Humidity            27.655925
Light              130.756622
CO2                690.553276
HumidityRatio        0.004249
dtype: float64
```

Median

```
Temperature         20.700000
Humidity            27.290000
Light                0.000000
CO2                565.416667
HumidityRatio        0.004337
dtype: float64
```

Mode

```
   Temperature  Humidity  Light    CO2  HumidityRatio
0        20.39     30.39    0.0  439.0       0.004793
```

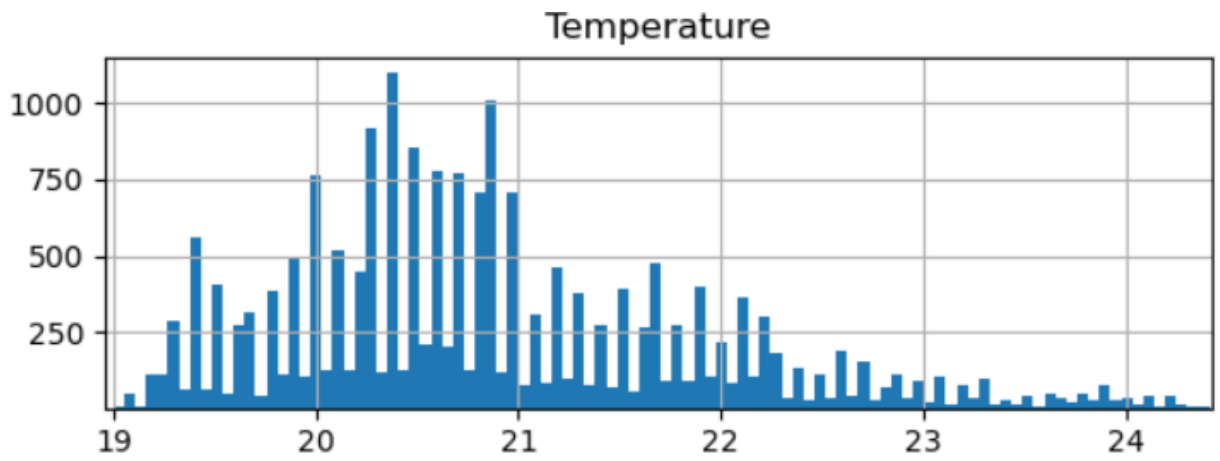
Skew

```
skewness:
Temperature       0.801036
Humidity         -0.013548
Light             1.309864
CO2               1.654305
HumidityRatio    -0.154726
dtype: float64
```

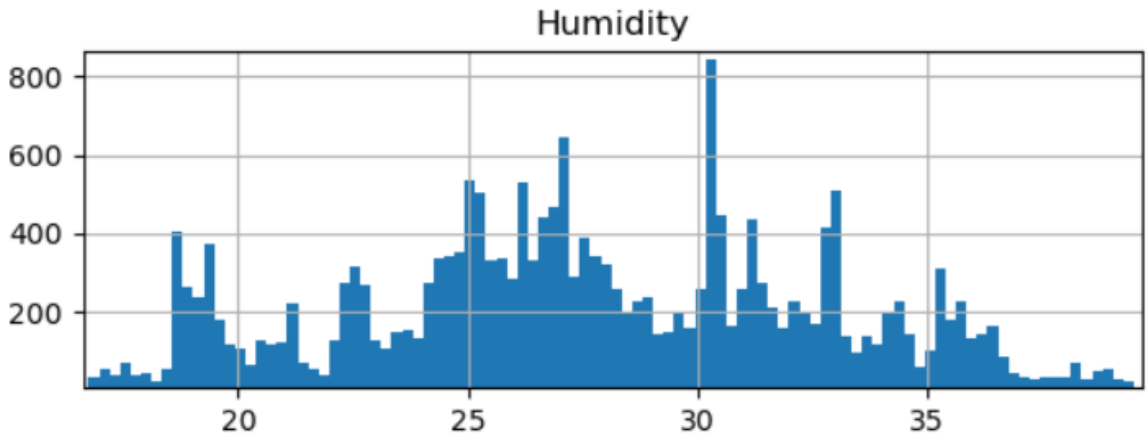
Comment

As it can be observed:

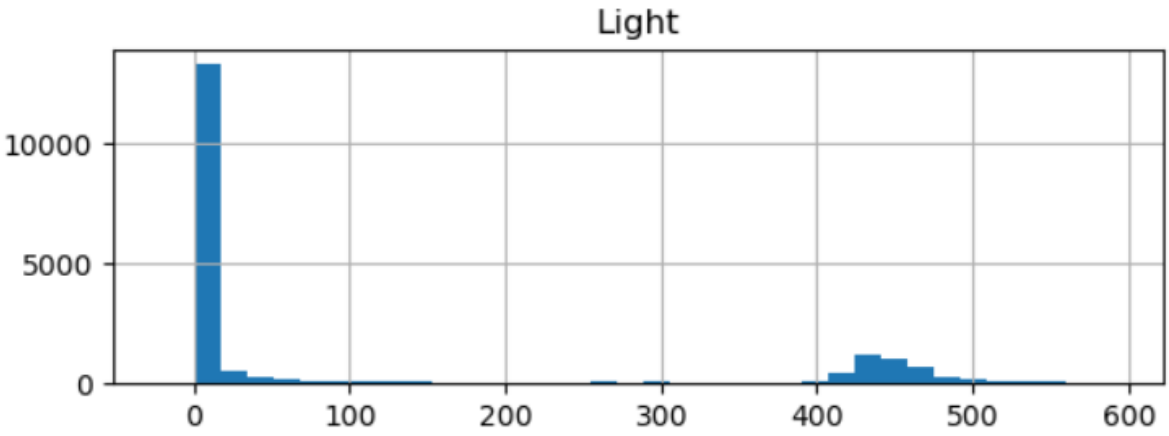
- The temperature attribute values are slightly skewed to the right as their skewness is a positive value (0.801036); also its mean (20.9) > median (20.7) > mode (20.39)



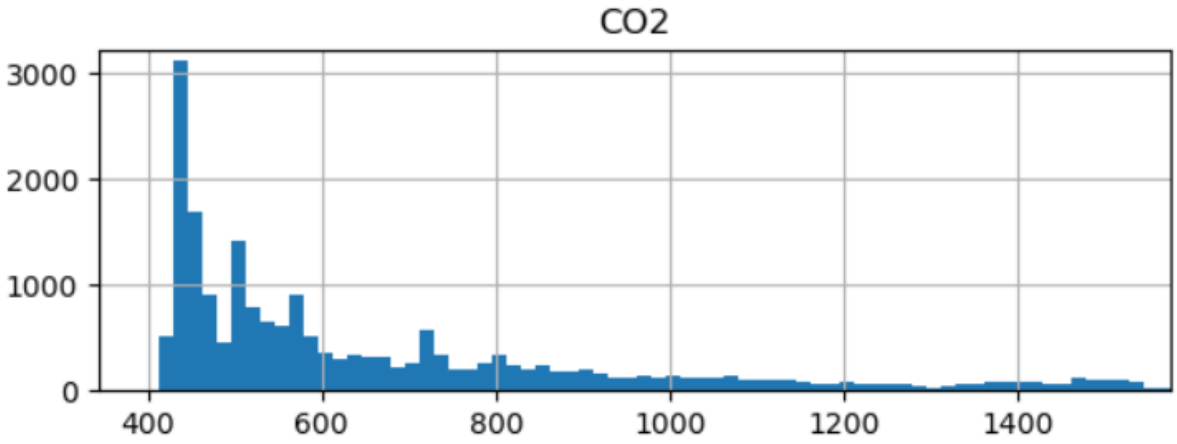
- The Humidity attribute values are slightly skewed to the left as their skewness is a negative value (-0.013548); also its mean (27.66) & median (27.29) < mode (30.39)



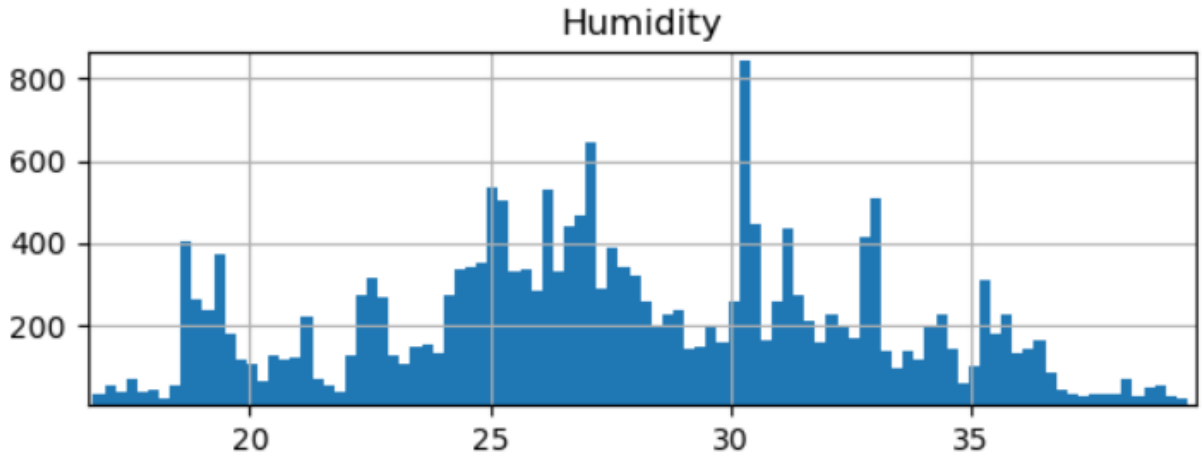
- The light attribute values are skewed to the right as their skewness is a positive value (1.309864); also its mean (130.76) > median (0) => mode (0)



- The CO2 attribute values are skewed to the right as their skewness is a positive value (1.654305); also its mean (690.6) > median (565.4) > mode (439)



- The HumidityRatio attribute values are slightly skewed to the left just like Humidity as their skewness is a negative value (-0.154726); also its mean (0.004249) < median (0.004337) < mode (0.004749).



Measures of central tendency

Range

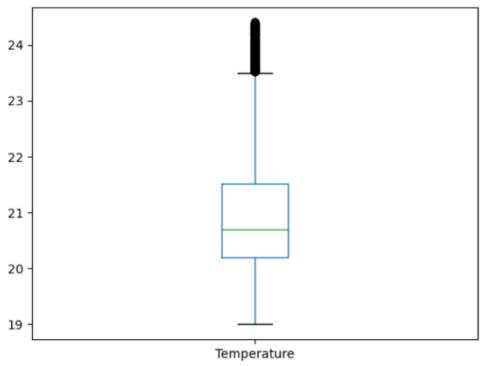
```
range:
Temperature          5.408333
Humidity            22.755000
Light             1697.250000
CO2               1663.750000
HumidityRatio        0.003802
dtype: float64
```

Inter Quartile range

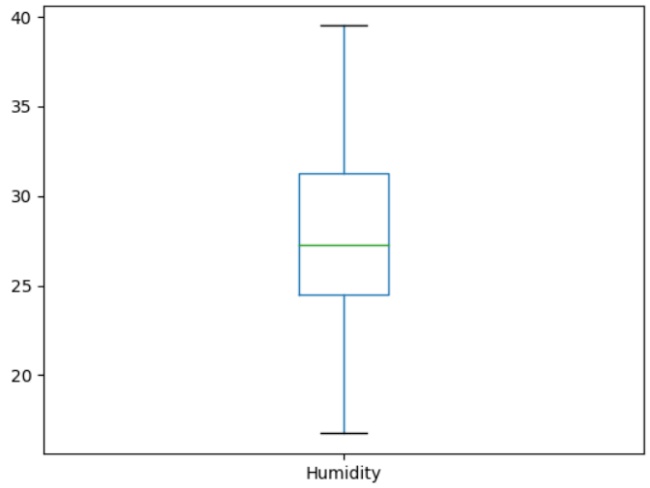
```
inter quartile range:
Temperature          1.325000
Humidity             6.790000
Light              301.000000
CO2                344.666667
HumidityRatio        0.001113
dtype: float64
```

Comment

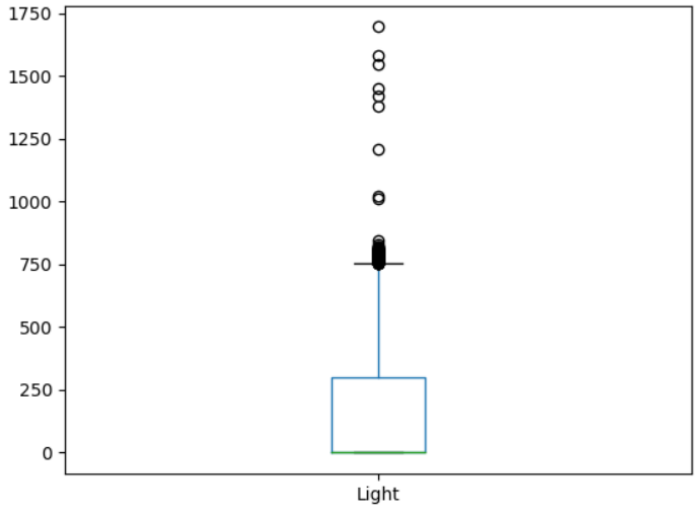
- As we can observe; the temperature attribute has very uneven distribution as there is a huge difference between the range (5.41) and the inter quartile range (1.325) which indicates the presence of many outliers as the box plot shows.



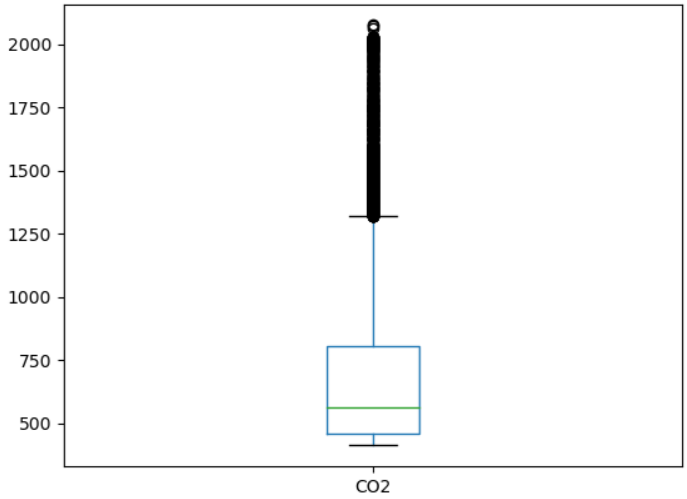
- As we can observe; the humidity attribute has very uneven distribution as there is a huge difference between the range (22.75) and the inter quartile range (6.79) which indicates the presence of many outliers as the box plot shows.



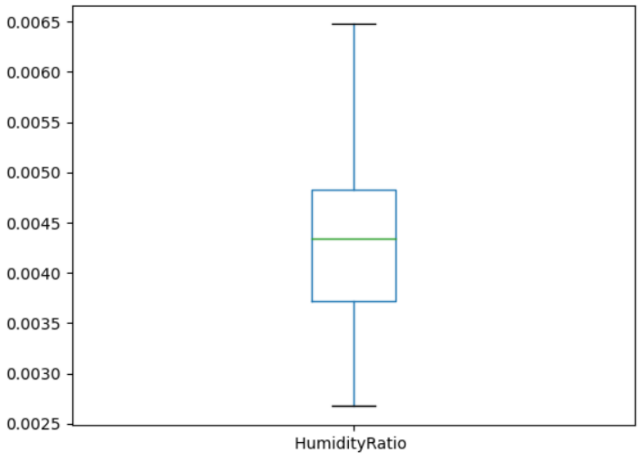
- As we can observe; the light attribute has very uneven distribution as there is a huge difference between the range (1697.25) and the inter quartile range (301) which indicates the presence of many outliers as the box plot shows.



- As we can observe; the CO2 attribute has very uneven distribution as there is a huge difference between the range (1663.75) and the inter quartile range (344.67) which indicates the presence of many outliers as the box plot shows.



- As we can observe; the HumidityRatio attribute has an even distribution as the range (0.003802) and the inter quartile range (0.001113) is nearly the same which indicates uniform distribution of data as the plot box shows.



Standard deviation

```
Temperature          1.055315
Humidity             4.982154
Light              210.430875
CO2                311.201281
HumidityRatio        0.000772
dtype: float64
```

Variance

```
Temperature       1.113689e+00
Humidity          2.482185e+01
Light             4.428115e+04
CO2               9.684624e+04
HumidityRatio     5.958454e-07
dtype: float64
```

Comment:

- The temperature and the humidity values have the lowest variance as their values don't spread out around the mean
- The light and the HumidityRatio have larger variance which means that the values are more spread out around the mean
- The CO2 have the highest variance value which means that the values are the most spread out around the mean

Removing outliers:

We used z score to remove values that are far from mean with more than 3 standard deviation; the new cleaned data set can be found here [e]

```
#   removing outliers
print('\nremoving outliers using z-score:')
z_scores = stats.zscore(DataSet)
abs_z_scores = np.abs(z_scores)
filtered_entries = (abs_z_scores < 3).all(axis=1)
new_DataSet = DataSet[filtered_entries]
new_DataSet.to_csv('C:\\Users\\Monica\\Desktop\\College\\semester 8\\Data Mining\\Final Research\\occupancy data\\DataMiningFinal\\NewDataSet.csv')
```

o Using Weka:

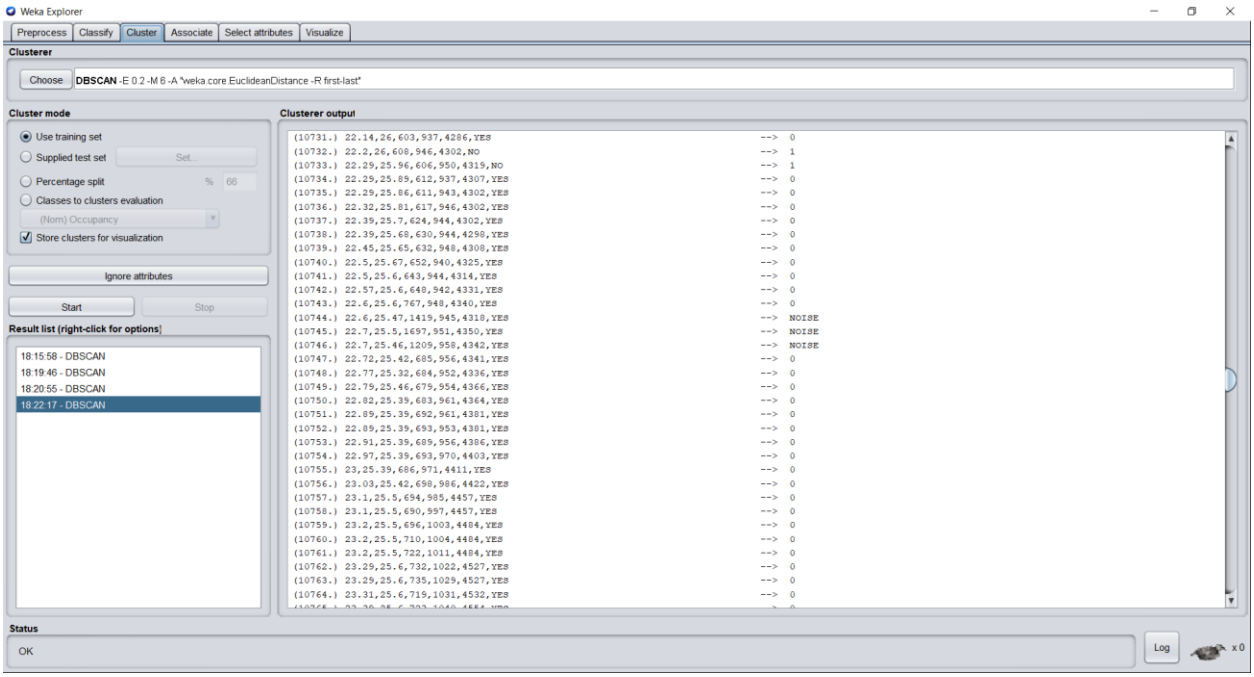
▪ **Perform outlier analysis using DBSCAN.**

Steps followed:

1. *Opened csv file on preprocessor tab; removed date (irrelevant), temperature, humidity, light, humidity ratio and CO2 attributes as we are going to use approximate values (Readme.txt [a])*
2. *From cluster tab, click chose button, then DBSCAN then start*
3. *Tried different* DBSCAN ϵ and minpoints values and kept trying and chose the best combinations that will give best results
4. *The optimal parameters found and used were ϵ = 0.2 and min points = 6*


```
Time taken to build model (full training data) : 21.68 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       4732 ( 23%)
1      15808 ( 77%)
2          9 (  0%)

Unclustered instances : 11
```

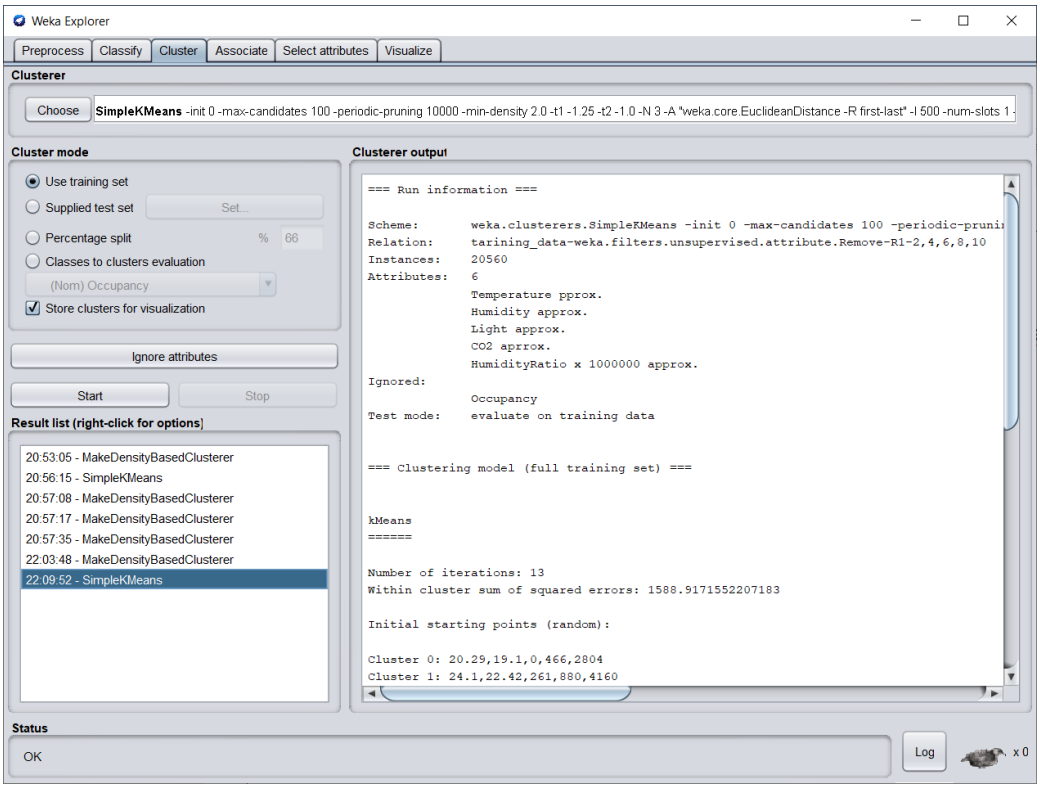
The results show that our data have very little outliers as most of them are very coherent; these results' code can be seen when using weka to run files at [b].

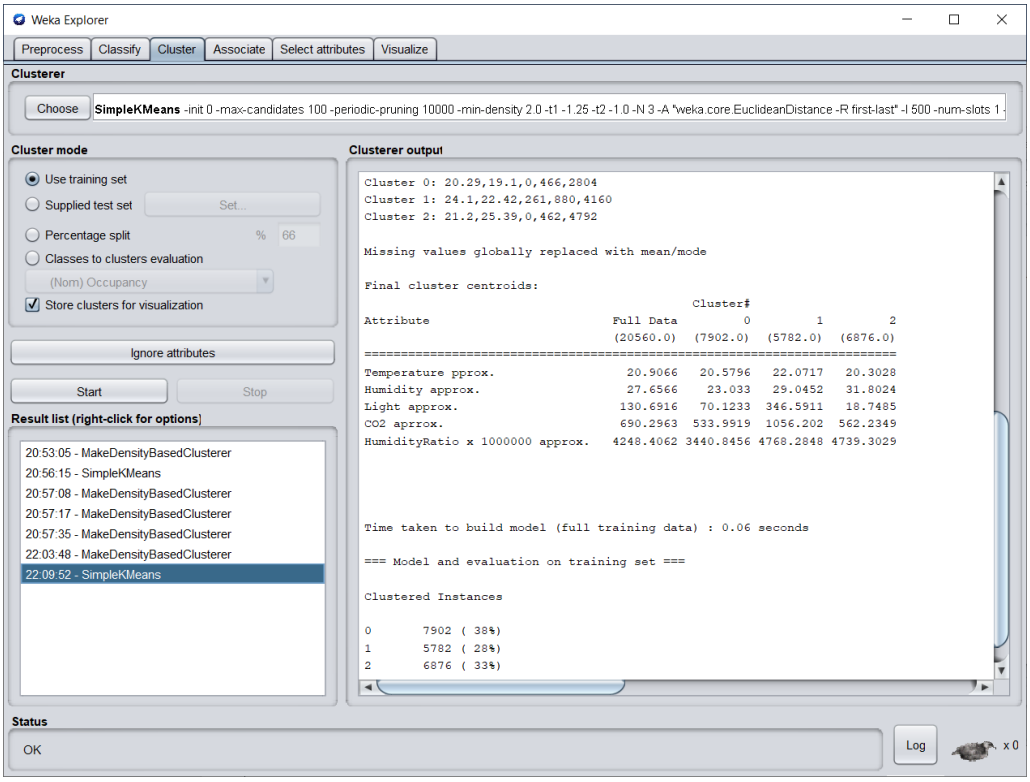


▪ **Perform cluster analysis to your data using a suitable technique of clustering.**

Steps followed:

1. *Opened csv file on preprocessor tab; removed date (irrelevant), temperature, humidity, light, humidity ratio and CO2 attributes as we are going to use approximate values (Readme.txt [a])*
2. *From cluster tab, click chose button, then SimpleKMeans and set number of clusters to be 3 then start*

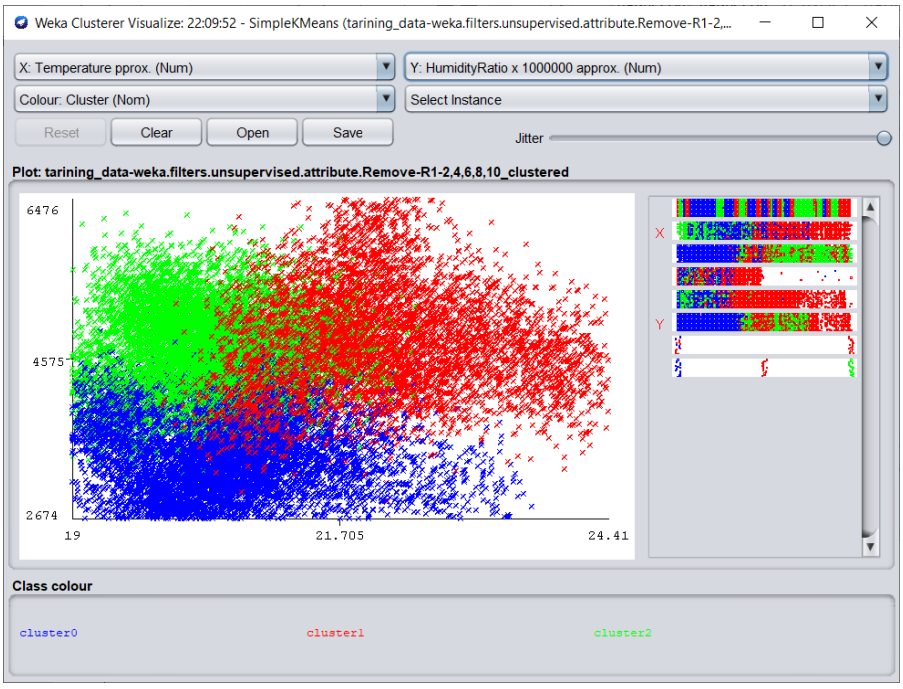

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) Occupancy
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
20:53:05 - MakeDensityBasedClusterer
20:56:15 - SimpleKMeans
20:57:08 - MakeDensityBasedClusterer
20:57:17 - MakeDensityBasedClusterer
20:57:35 - MakeDensityBasedClusterer
22:03:48 - MakeDensityBasedClusterer
22:09:52 - SimpleKMeans
Clusterer output
=== Run information ===
Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruni
Relation: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10
Instances: 20560
Attributes: 6
Temperature pprox.
Humidity approx.
Light approx.
CO2 aprrox.
HumidityRatio x 1000000 approx.
Ignored:
Occupancy
Test mode: evaluate on training data
=== Clustering model (full training set) ===
kMeans
======
Number of iterations: 13
Within cluster sum of squared errors: 1588.9171552207183
Initial starting points (random):
Cluster 0: 20.29,19.1,0,466,2804
Cluster 1: 24.1,22.42,261,880,4160
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) Occupancy
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
20:53:05 - MakeDensityBasedClusterer
20:56:15 - SimpleKMeans
20:57:08 - MakeDensityBasedClusterer
20:57:17 - MakeDensityBasedClusterer
20:57:35 - MakeDensityBasedClusterer
22:03:48 - MakeDensityBasedClusterer
22:09:52 - SimpleKMeans
Clusterer output
Cluster 0: 20.29,19.1,0,466,2804
Cluster 1: 24.1,22.42,261,880,4160
Cluster 2: 21.2,25.39,0,462,4792
Missing values globally replaced with mean/mode
Final cluster centroids:
Cluster#
Attribute Full Data 0 1 2
(20560.0) (7902.0) (5782.0) (6876.0)
Temperature pprox. 20.9066 20.5796 22.0717 20.3028
Humidity approx. 27.6566 23.033 29.0452 31.8024
Light approx. 130.6916 70.1233 346.5911 18.7485
CO2 aprrox. 690.2963 533.9919 1056.202 562.2349
HumidityRatio x 1000000 approx. 4248.4062 3440.8456 4768.2848 4739.3029
Time taken to build model (full training data) : 0.06 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 7902 (38%)
1 5782 (28%)
2 6876 (33%)
Status
OK
Log
x 0


Weka Clusterer Visualize: 22:09:52 - SimpleKMeans (tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,...
X: Temperature pprox. (Num)
Y: Humidity approx. (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10_clustered
39.5
28.125
16.75
19
21.705
24.41
X
Y
Class colour
cluster0
cluster1
cluster2


Weka Clusterer Visualize: 22:09:52 - SimpleKMeans (tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,...
X: Temperature pprox. (Num)
Y: Light approx. (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10_clustered
1697
848.5
0
19
21.705
24.41
X
Y
Class colour
cluster0
cluster1
cluster2

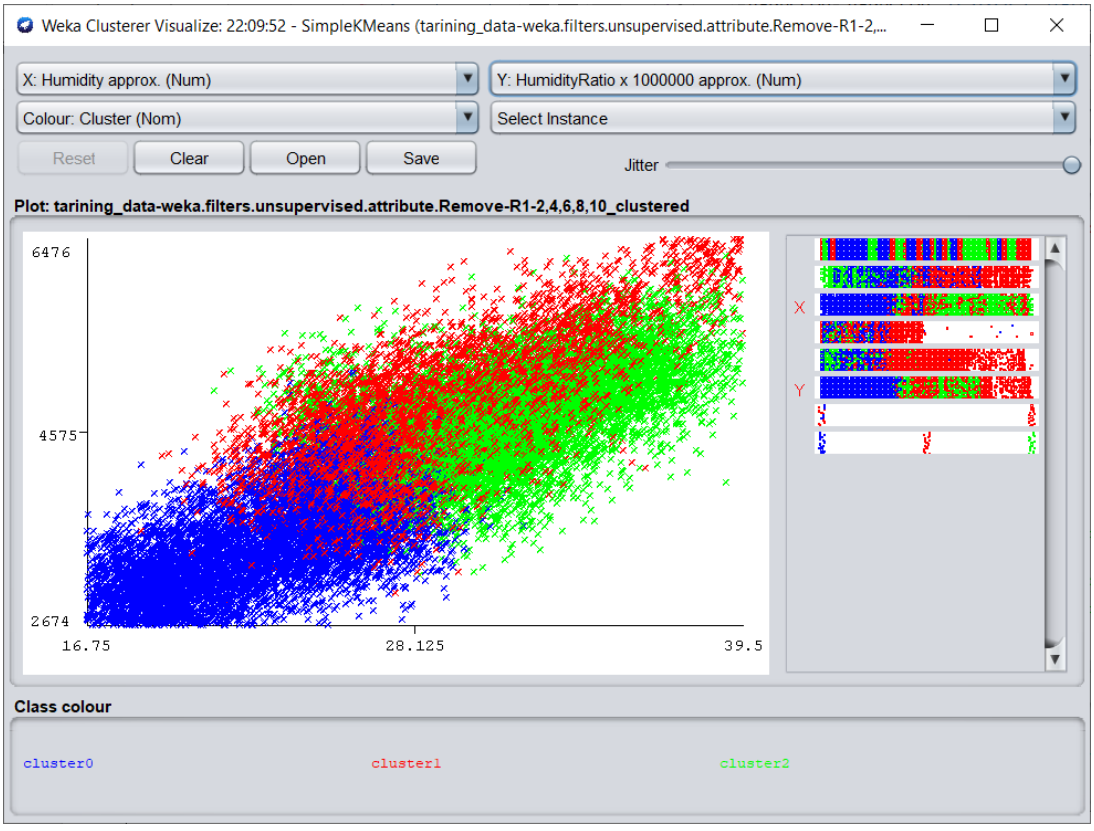

Weka Clusterer Visualize: 22:09:52 - SimpleKMeans (tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,...
X: Temperature pprox. (Num)
Y: CO2 aprrox. (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10_clustered
2076
1244
412
19
21.705
24.41
X
Y
Class colour
cluster0
cluster1
cluster2

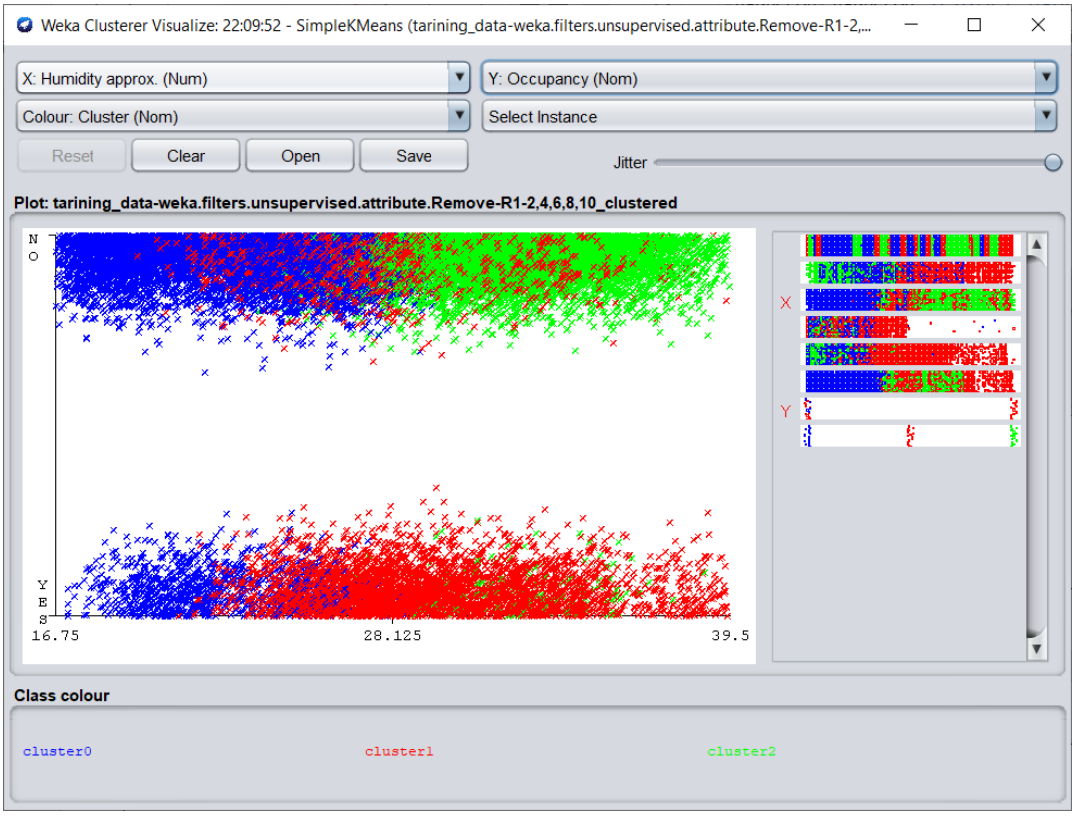

Weka Clusterer Visualize: 22:09:52 - SimpleKMeans (tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,...
X: Temperature pprox. (Num)
Y: HumidityRatio x 1000000 approx. (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10_clustered
6476
4575
2674
19
21.705
24.41
X
Y
Class colour
cluster0
cluster1
cluster2


Weka Clusterer Visualize: 22:09:52 - SimpleKMeans (tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,...
X: Temperature pprox. (Num)
Y: Occupancy (Nom)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10_clustered
NO
YES
19
21.705
24.41
X
Y
Class colour
cluster0
cluster1
cluster2


Weka Clusterer Visualize: 22:09:52 - SimpleKMeans (tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,...
X: Humidity approx. (Num)
Y: Light approx. (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10_clustered
1697
848.5
0
16.75
28.125
39.5
X
Y
Class colour
cluster0
cluster1
cluster2


Weka Clusterer Visualize: 22:09:52 - SimpleKMeans (tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,...
X: Humidity approx. (Num)
Y: CO2 aprrox. (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10_clustered
2076
1244
412
16.75
28.125
39.5
X
Y
Class colour
cluster0
cluster1
cluster2


Weka Clusterer Visualize: 22:09:52 - SimpleKMeans (tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,...
X: Humidity approx. (Num)
Y: HumidityRatio x 1000000 approx. (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10_clustered
6476
4575
2674
16.75
28.125
39.5
X
Y
Class colour
cluster0
cluster1
cluster2


Weka Clusterer Visualize: 22:09:52 - SimpleKMeans (tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,...
X: Humidity approx. (Num)
Y: Occupancy (Nom)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10_clustered
N
O
Y
E
S
16.75
28.125
39.5
X
Y
Class colour
cluster0
cluster1
cluster2

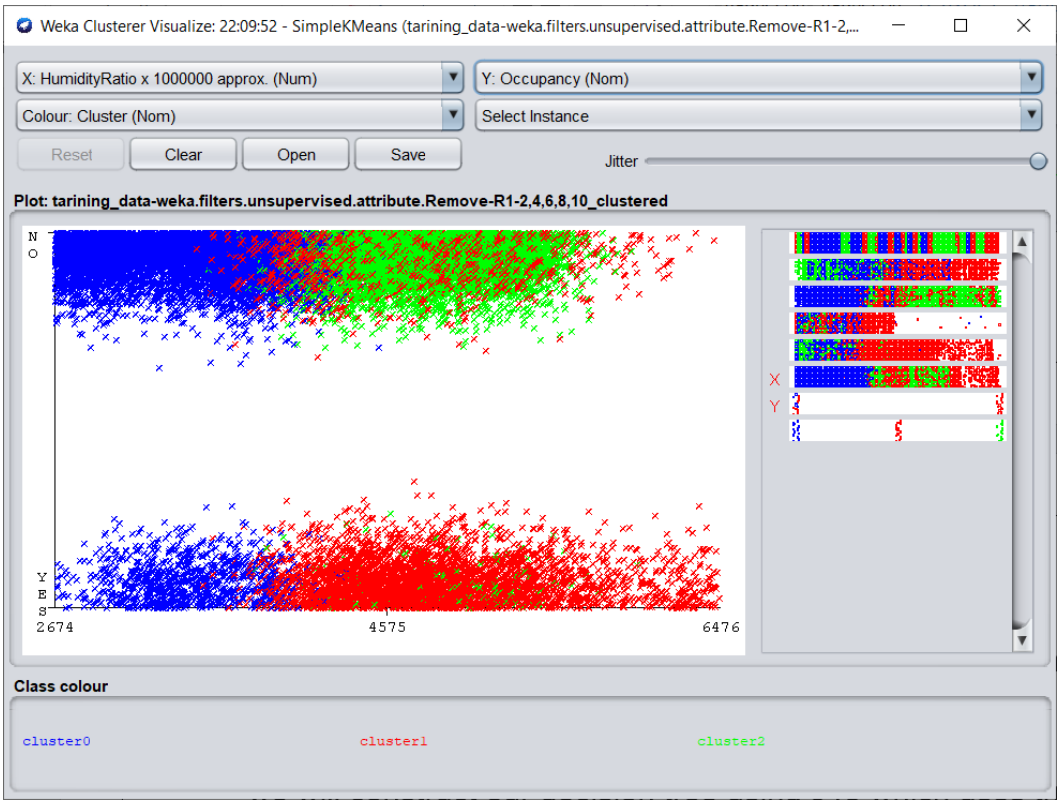

Weka Clusterer Visualize: 22:09:52 - SimpleKMeans (tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,...
X: Light approx. (Num)
Y: CO2 aprrox. (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10_clustered
2076
1244
412
0
848.5
1697
X
Y
Class colour
cluster0
cluster1
cluster2


Weka Clusterer Visualize: 22:09:52 - SimpleKMeans (tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,...
X: Light approx. (Num)
Y: HumidityRatio x 1000000 approx. (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10_clustered
6476
4575
2674
0
848.5
1697
X
Y
Class colour
cluster0
cluster1
cluster2


Weka Clusterer Visualize: 22:09:52 - SimpleKMeans (tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,...
X: Light approx. (Num)
Y: Occupancy (Nom)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10_clustered
N
O
Y
E
S
0
848.5
1697
X
Y
Class colour
cluster0
cluster1
cluster2


Weka Clusterer Visualize: 22:09:52 - SimpleKMeans (tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,...
X: CO2 aprrox. (Num)
Y: HumidityRatio x 1000000 approx. (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10_clustered
6476
4575
2674
412
1244
2076
X
Y
Class colour
cluster0
cluster1
cluster2


Weka Clusterer Visualize: 22:09:52 - SimpleKMeans (tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,...
X: CO2 aprrox. (Num)
Y: Occupancy (Nom)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: tarining_data-weka.filters.unsupervised.attribute.Remove-R1-2,4,6,8,10_clustered
N
O
Y
E
S
412
1244
2076
X
Y
Class colour
cluster0
cluster1
cluster2



All cluster analysis file can be found here [d]

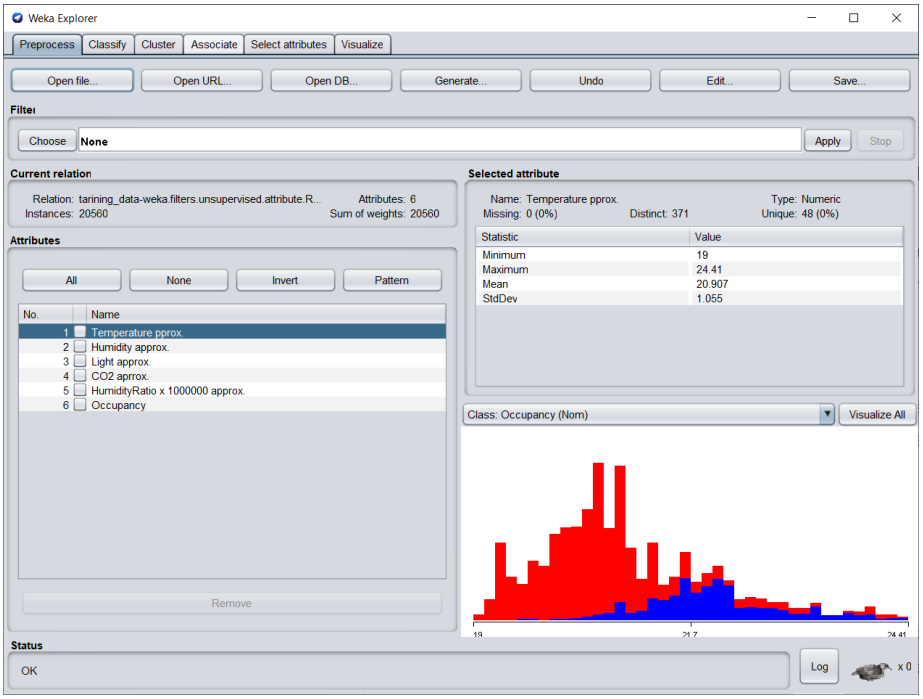
▪ Use Decision Trees to classify the data based on some aspects in the context of your collected real dataset

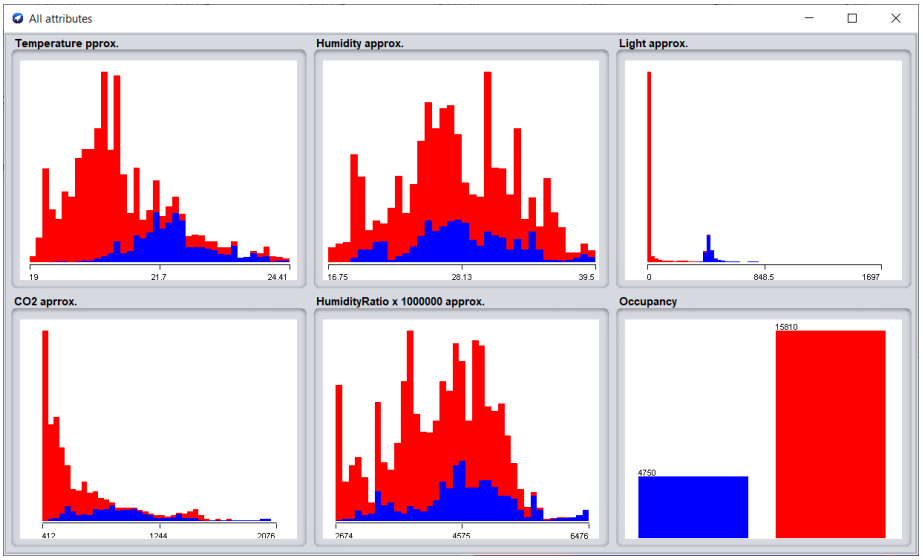
we will construct our decision tree using J48 which uses the same algorithm as ID3 but it is just a more updated version.

We will use 10-fold cross-validation test mode to test our constructed model (decision tree). The tree is used as a model that will help identify if room is occupied (YES) or not (NO) based on its conditions.

Steps followed:

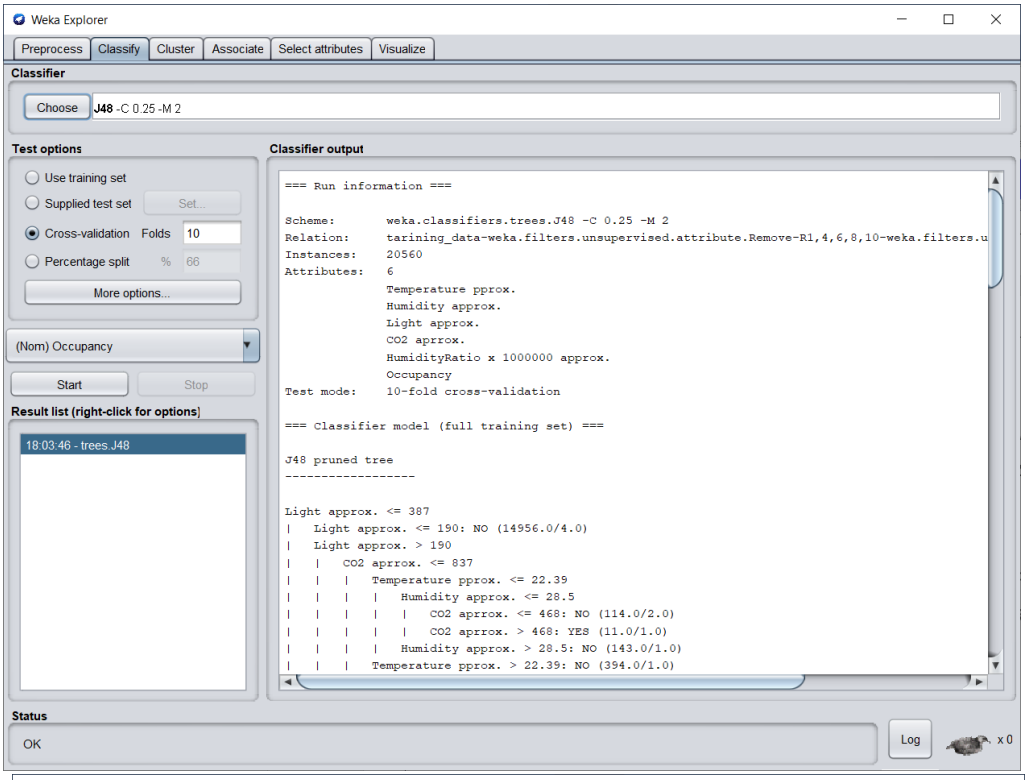
1. *Opened csv file on preprocessor tab; removed date (irrelevant), temperature, humidity, light, humidity ratio and CO2 attributes as we are going to use approximate values (Readme.txt [a])*

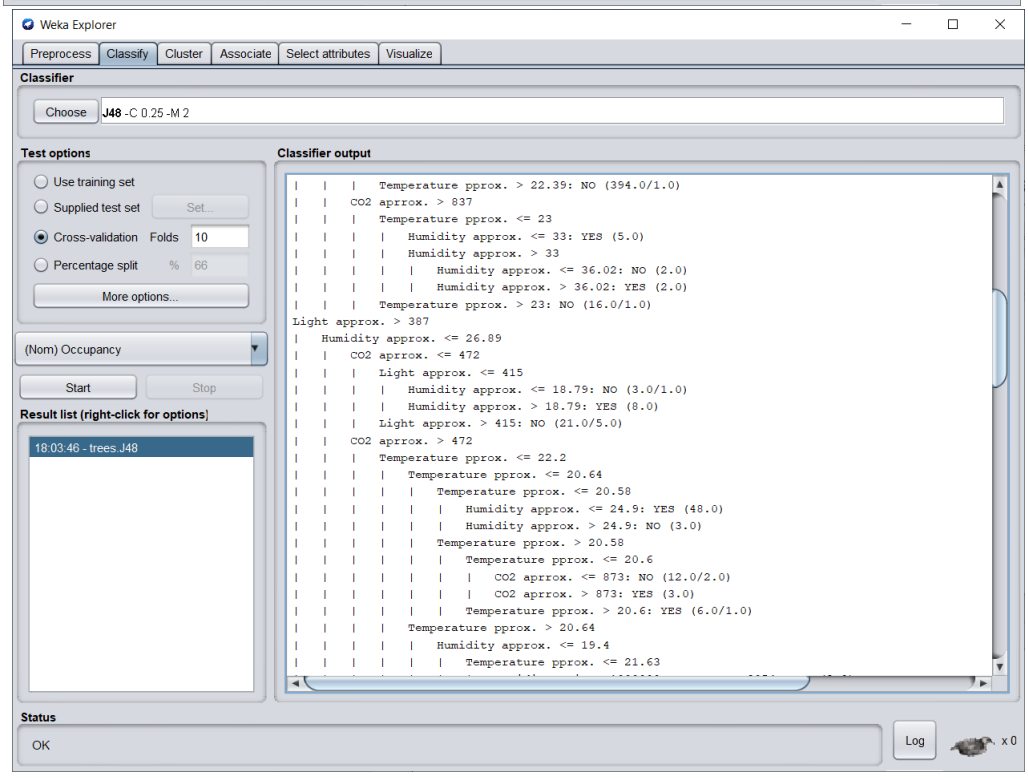


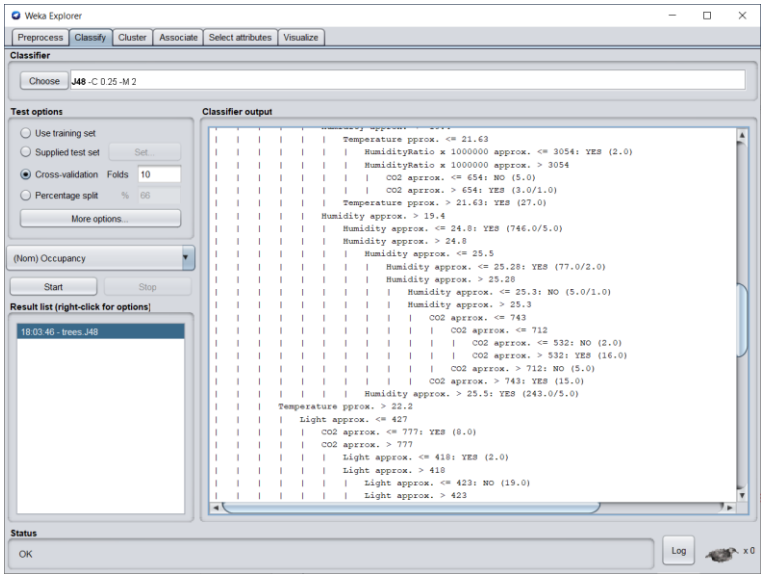


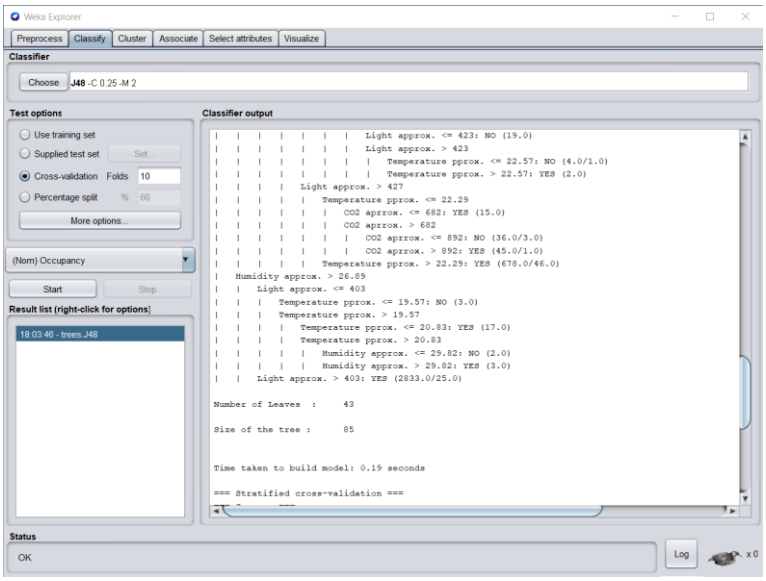
Although data is numeric and continuous we don't need to apply any preprocessing filter on it as J48 effectively discretizes numeric attributes as it goes along.

2. *From classify tab, click choose button, then trees>>J48*
3. *Choose cross-validation with 10 folds then click start*


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) Occupancy
Start
Stop
Result list (right-click for options)
18:03:46 - trees.J48
Classifier output
=== Run information ===
Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: tarining_data-weka.filters.unsupervised.attribute.Remove-R1,4,6,8,10-weka.filters.u
Instances: 20560
Attributes: 6
Temperature pprox.
Humidity approx.
Light approx.
CO2 aprrox.
HumidityRatio x 1000000 approx.
Occupancy
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
Light approx. <= 387
| Light approx. <= 190: NO (14956.0/4.0)
| Light approx. > 190
| | CO2 aprrox. <= 837
| | | Temperature pprox. <= 22.39
| | | | Humidity approx. <= 28.5
| | | | | CO2 aprrox. <= 468: NO (114.0/2.0)
| | | | | CO2 aprrox. > 468: YES (11.0/1.0)
| | | | Humidity approx. > 28.5: NO (143.0/1.0)
| | | Temperature pprox. > 22.39: NO (394.0/1.0)
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) Occupancy
Start
Stop
Result list (right-click for options)
18:03:46 - trees.J48
Classifier output
| | | Temperature pprox. > 22.39: NO (394.0/1.0)
| | CO2 aprrox. > 837
| | | Temperature pprox. <= 23
| | | | Humidity approx. <= 33: YES (5.0)
| | | | Humidity approx. > 33
| | | | | Humidity approx. <= 36.02: NO (2.0)
| | | | | Humidity approx. > 36.02: YES (2.0)
| | | Temperature pprox. > 23: NO (16.0/1.0)
Light approx. > 387
| Humidity approx. <= 26.89
| | CO2 aprrox. <= 472
| | | Light approx. <= 415
| | | | Humidity approx. <= 18.79: NO (3.0/1.0)
| | | | Humidity approx. > 18.79: YES (8.0)
| | | Light approx. > 415: NO (21.0/5.0)
| | CO2 aprrox. > 472
| | | Temperature pprox. <= 22.2
| | | | Temperature pprox. <= 20.64
| | | | | Temperature pprox. <= 20.58
| | | | | | Humidity approx. <= 24.9: YES (48.0)
| | | | | | Humidity approx. > 24.9: NO (3.0)
| | | | | Temperature pprox. > 20.58
| | | | | | Temperature pprox. <= 20.6
| | | | | | | CO2 aprrox. <= 873: NO (12.0/2.0)
| | | | | | | CO2 aprrox. > 873: YES (3.0)
| | | | | | Temperature pprox. > 20.6: YES (6.0/1.0)
| | | | Temperature pprox. > 20.64
| | | | | Humidity approx. <= 19.4
| | | | | | Temperature pprox. <= 21.63
Status
OK
Log
x 0





Classifier output

```
Number of Leaves  :     43

Size of the tree :      85


Time taken to build model: 0.19 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       20407               99.2558 %
Incorrectly Classified Instances       153                0.7442 %
Kappa statistic                          0.9791
Mean absolute error                      0.0109
Root mean squared error                  0.0834
Relative absolute error                  3.0585 %
Root relative squared error             19.7852 %
Total Number of Instances            20560

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Cl
                 0.988    0.006    0.980      0.988    0.984      0.979    0.993     0.974     YE
                 0.994    0.012    0.996      0.994    0.995      0.979    0.993     0.995     NO
Weighted Avg.    0.993    0.011    0.993      0.993    0.993      0.979    0.993     0.990

=== Confusion Matrix ===

     a     b   <-- classified as
  4693    57 |     a = YES
    96 15714 |     b = NO
```

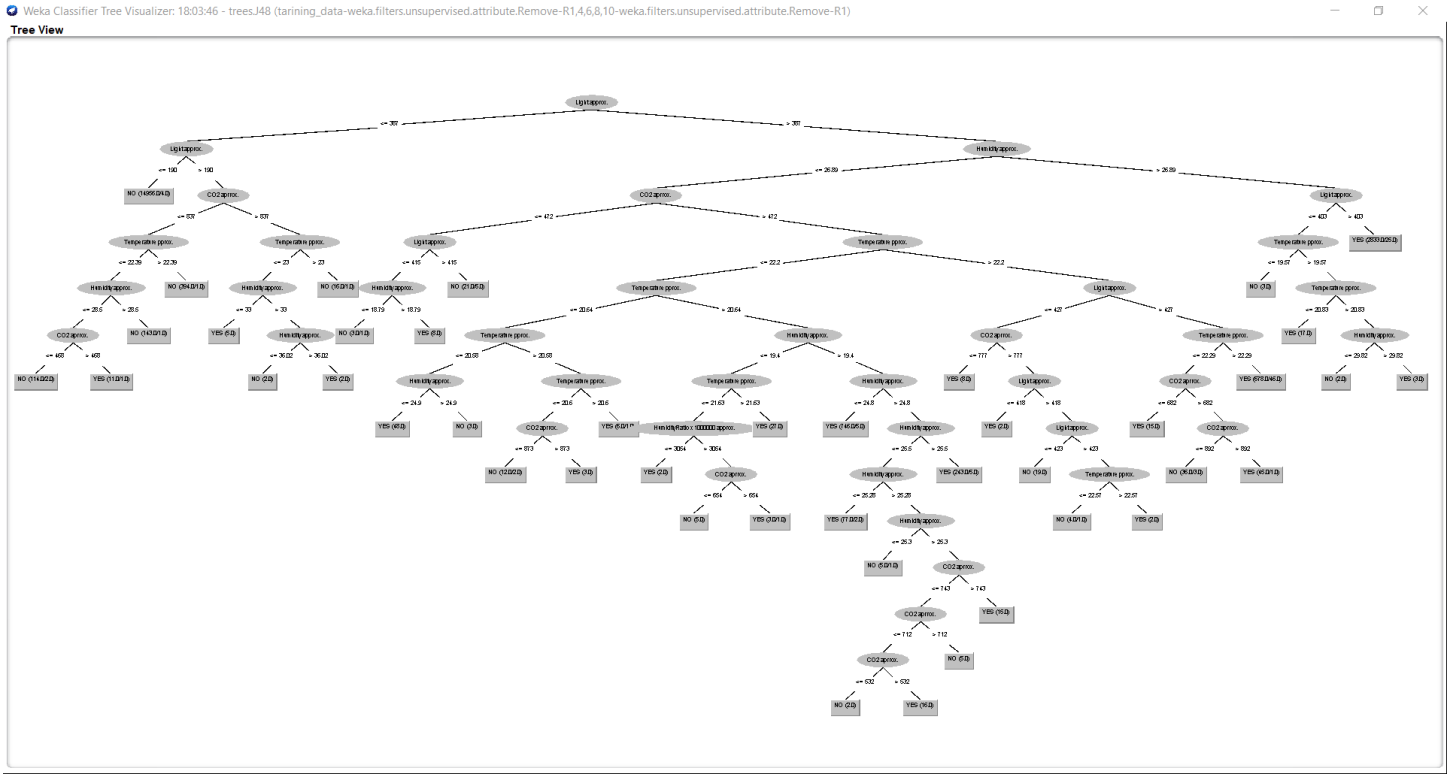


Figure 7 visualized J48 decision tree

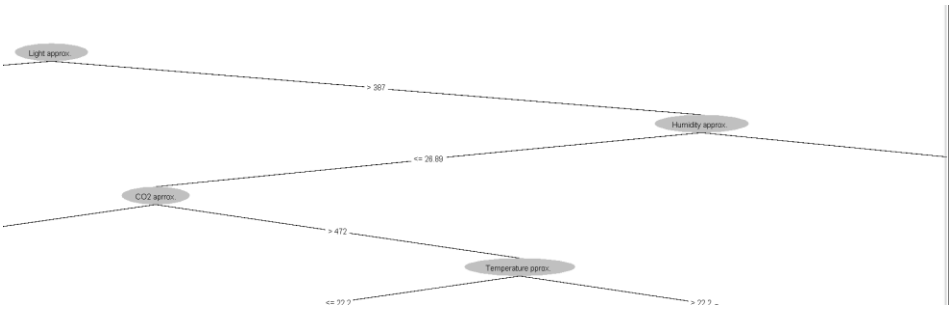
The result buffer and the tree model are attached below [c]

To have a better view let's consider 2 random instance from our data set

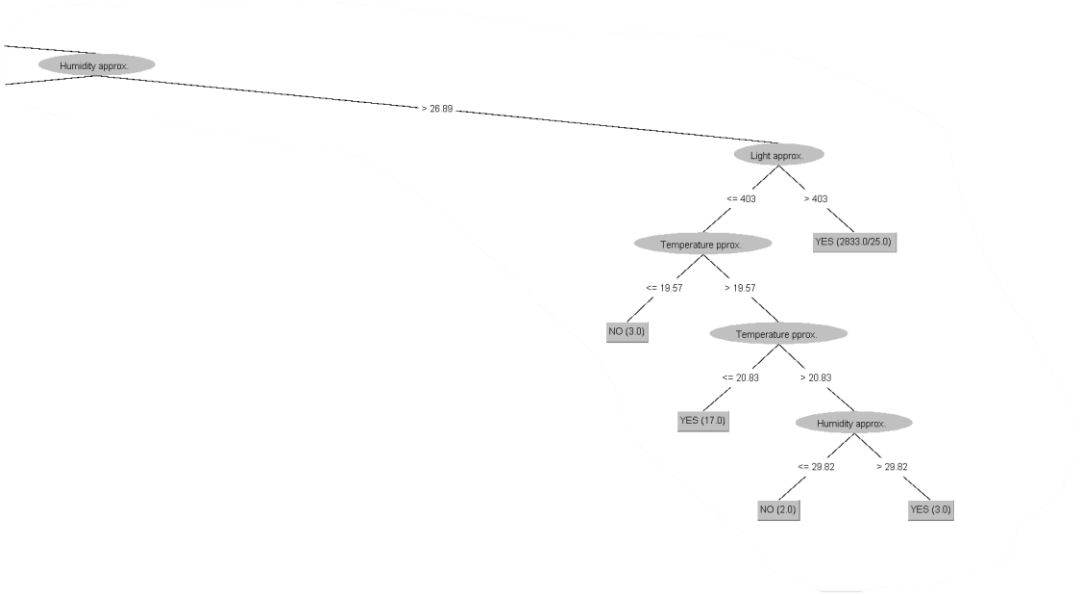
	date	Temperature	Temperature pprox.	Humidity	Humidity approx.	Light	Light approx.	CO2	CO2 aprrox.	HumidityF	HumidityRatio x 1000000 approx.	Occupancy
17	2004-02-15 18:06	23	23	27.125	27.13	418.5	418	680.5	680	0.00479	4792	YES
18	2004-02-15 18:07	23	23	27.2	27.2	0	0	681.5	681	0.00479	4792	NO

Item 17

Light approx. is > 387 and humidity approx. >26.89

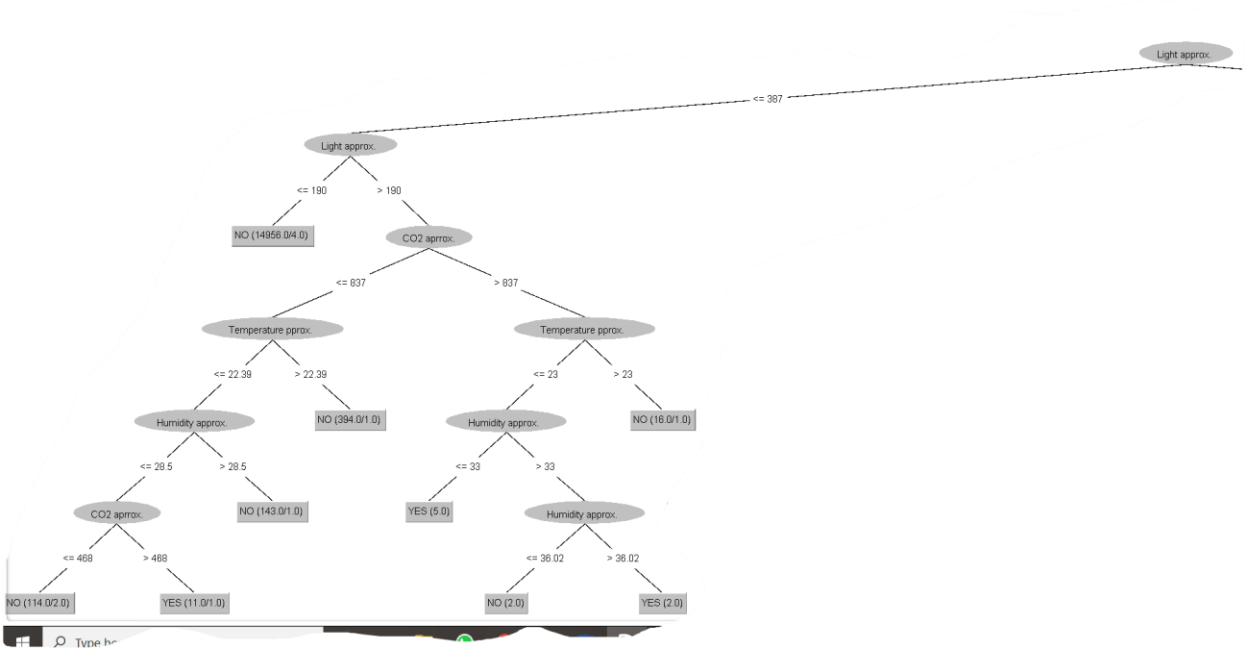


Light approx. is > 403 that's why the room is occupied (True)



Item 18

Light approx. is < 387 and < 190; that's why the room is unoccupied (true)



7 Conclusion

As we have proved throughout the research; classification is highly needed in big data to automatically classify data registered everywhere every day. Also, we discussed decision trees and how they work and how they are related to probability. Finally, we observed how decision trees are highly effective classification methods that result in accurate future prediction up to 99%.

9 Attachment list

[a] https://drive.google.com/drive/folders/1cPMAaIZZQ_kEZ0MFDnpUC1JIVqD8Dcxd?usp=sharing

[b] https://drive.google.com/drive/folders/1EWIUs3BzL5a5QiIF9ge6dr9sf6KDQbpw?usp=sharing

[c] https://drive.google.com/drive/folders/1uGJfK3qomuUrqRqTMzmiNIKthZ68KgR3?usp=sharing

[d] https://drive.google.com/drive/folders/1o4pt_mvFaI8FRSeWi8e0pFdjyhoWLgt2?usp=sharing

[e] https://drive.google.com/drive/folders/15N_glJT8NL5YC288pgko9IVzsgPdLEt6?usp=sharing

8 References

[1] Han, J., “Data mining: Concepts and techniques”, third edition, 2012, Waltham, Mass.: Morgan Kaufmann Publishers.

[2] Gareth James, et al., “An Introduction to Statistical Learning with Applications in R”, Springer.

[3] *Lior Rokach and Oded Maimon. 2014. Data Mining With Decision Trees: Theory and Applications (2nd. ed.). World Scientific Publishing Co., Inc., USA.*

[4] Dileep Kumar G. and Dileep Kumar G. 2018. Machine Learning Techniques for Improved Business Analytics (1st. ed.). IGI Global, USA.

[5] Kingsford, C., & Salzberg, S. L. (2008). What are decision trees?. *Nature biotechnology*, *26*(9), 1011–1013. https://doi.org/10.1038/nbt0908-1011

[6] Quinlan, J.R. Induction of decision trees. Mach Learn 1, 81–106 (1986). https://doi.org/10.1007/BF00116251