DECISION TREES

Data mining and Business Intelligence CSE 385

NAME: KARIM WALID ELHAMMADY
ID: 16P3090

Table of Contents

List of Figures 2
List of Tables 2
List of Equations 2
Abstract 3
Introduction 4
Body 5
What is a Decision Tree and its terms? 5
Mathematical formulation of Decision Trees 6
ID3 and C4.5 algorithms 10
ID3 10
C4.5 12
ID3 Vs. C4.5 12
Building a decision tree 14
Classification in Decision Trees 16
Applications in real life 18
Python 19
Weka 22
Conclusion 24
Appendix 26

List of Figures

FIGURE 1 DECISION TREES FOR DIRECT MAILING RESPONSE FOR CUSTOMERS 5
FIGURE 2 GRAPH FOR ENTROPY AND GINI INDEX FUNCTIONS 9
FIGURE 3 ACCURACY COMPARISON BETWEEN ID3 AND C4.5 13
FIGURE 4 EXECUTION TIME COMPARISON BETWEEN ID3 AND C4.5 14
FIGURE 5 UNPRUNED TREE FOR HITTER DATA 16
FIGURE 6 TREE ANALYSIS FOR HITTER DATA 16
FIGURE 7 HEART DATA. TOP: UNPRUNED TREE, BOTTOM LEFT: ERROR FOR PRUNED TREE , BOTTOM RIGHT: PRUNED TREE 17
FIGURE 8 DATASET SUMMARY 19
FIGURE 9 ALL ATTRIBUTES GRAPHICAL REPRESENTATION 19
FIGURE 10 HISTOGRAM FOR SELECTED ATTRIBUTES 20
FIGURE 11 IMPULSE FOR SELECTED ATTRIBUTES 20
FIGURE 12 BOXPLOT BEFORE REMOVING OUTLIERS 21
FIGURE 13 BOXPLOT AFTER REMOVING OUTLIERS 21
FIGURE 14 STATISTICAL ANALYSIS FOR ATTRIBUTES 22
FIGURE 15 OUTPUT FOR DBSCAN CLUSTERING 22
FIGURE 16 OUTPUT FOR K-MEANS CLUSTERING 23

List of Tables

TABLE 1 ACCURACY COMPARISON BETWEEN ID3 AND C4.5 13
TABLE 2 EXECUTION TIME COMPARISON BETWEEN ID3 AND C4.5 13

List of Equations

EQUATION 1 INFORMATION NEEDED TO CLASSIFY A TUPLE IN *D* 7
EQUATION 2 HOW MUCH INFORMATION IS NEEDED FOR EXACT CLASSIFICATION 7
EQUATION 3 INFORMATION GAIN 7
EQUATION 4 SPLIT INFORMATION 8
EQUATION 5 GAIN RATIO 8
EQUATION 6 GINI INDEX 8
EQUATION 7 WEIGHTED SUM OF IMPURITY 8
EQUATION 8 IMPURITY REDUCTION 9
EQUATION 9 EQUATION FOR CONSTRUCTING DECISION TREES 15

Abstract

Data mining is the valuable method for exploring big data information. In data mining various ways & algorithms are found to be useful. Classification is the mostly common used by large database to find the mine rule. Decision tree approach is commonly used to classify objects or pictures as it is most easiest framework for comprehension and making decisions. Many ways for classification purposes are available in data mining based on the usage of Deep Learning and Neural Networks, Nearest Neighbor Rule & Baysen, on the other hand, mining in decision trees is very easy. ID3 and C4.5 algorithms were introducted which make good Decision Trees. In this research, Decision Trees concepts will be discussed thoroughly, as well as it uses in machine learning. The paper has two parts: theoretical and practical. In the theoretical section, we will discuss mathematical formulation of Decision Trees, various methods to construct Decision Trees and how are they related to probability, and a numerical example on how to build a Decision Tree and how it is used in classification In the practical part, a real dataset will be collected and used for classification, data analysis for the collected dataset by analyzing raw data, and use WEKA to perform further analysis.

Introduction

Data Science is the refining of data and analysis discipline for the aim to extract valuable knowledge. The ability to make assumptions about certain events is one of the most important aims in data mining field. Prediction is clearly no easy thing. Even, half of the time we are using estimation effectively. For example, YouTube, one of the most popular websites, record everyone watching patterns, and analyze it so it can predict which other videos we might watch. The data mining sector offers the techniques and tools that enable the automated processing of enormous of data. Decision Trees is one of the valuable techniques that is commonly used. While a decision tree in data mining is based on statistics, which can be used to describe both classifiers and regression models, the decision trees in operations analysis apply to a hierarchy of decision network and their effects. The decision maker uses decision trees to find the best approach that will most likely achieve the required goal. It is known as a Classification Tree when classification activities are applied using decision trees, where data is represented in nodes rather than decisions. If used for functions of regression, that is called Regression Tree. A group of connected decision, where each decision has a different outcome from the others is referred to as a decision tree. This helps individuals and organizations, taking into consideration the risks, possibilities, and advantages, to decide future acts based on the predictions in hand. Usually, a single node is the building unit of trees, which it begins with and then branches out into different scenarios. Each of these findings leads to additional outcomes, branching to the next possibility. These units are combined together and give the shape of a tree. Decision trees are also used in areas of use such as accounting, communications, electronics, and medicine. You can also use a decision tree to help building models used for predicting that can be used in various fields like machine learning, data mining, and statistics. Widely known as learning mechanism of the decision tree, this method takes into consideration all observations for a particular item to predict the value of this specific item. As an exploratory tool the classification tree is useful. However, it does not aim to replace current conventional mathematical approaches, although there are several other strategies that can be used to identify or forecast instance association with a previously known set of groups, such as neural networks or vector machines help.

Body

What is a Decision Tree and its terms?

As mentioned before, the most common use for decision trees is in machine learning field to build predictive models, but what is really a decision tree and how can we present it? A classifier can be built by using decision tree that expresses itself as a recursive instance space partition. As described above, nodes is the building unit of a decision tree which help to form a rooted tree; all trees always have a node which is called “root of the tree”, which has no incoming edges connected to it. All nodes in the tree, except the root, has only a single entrance point. There is also another types of nodes, other than the root node, which has outgoing edges and they are called “internal” nodes. The rest of the nodes may be also commonly known as terminal, decision nodes, or leaves. The space is divided in most of the decision trees by the internal nodes of the instance into two more sub spaces according to the unique attributes that are found in the input vales. In the simplest and widely used way each measure takes into account only one attribute, so the partitioning process to produce the sub spaces occur depending on the value of these attributes. The state applies to a number, for all numeric properties.

Each leaf in the tree is assigned to a specific class which corresponds to the best objective value . Alternatively, a vector probability can be found in leaves, also called as affinity

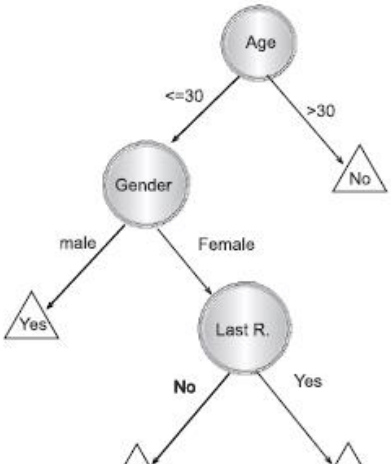


Figure 1 Decision Trees for Direct Mailing response for customers

Vector) which shows the possibility that a certain attribute has a specified value. Another example of the decision trees is shown in Figure 1 predicting if a customer will reply to the direct mail or not. Circles in the figure indicates internal nodes, while triangles are denoted as leaves. Each internal node can have two or more branches. Furthermore, Each node is mapped to a certain method, and the branches indicates the number of values. These value must have a specific range so it could be comprehensive and mutually exclusive.

The decision-makers obviously choose a decision-making tree that is not complicated because it is likely to be more intuitive. In addition, the size of the tree got a significant impact on the accuracy. The size of the tree usually is calculated by many characteristics that is found in the tree like : number of all nodes, number of all nodes, tree depth and total number of attributes.

Mathematical formulation of Decision Trees

The usefulness of a classification tree is a basic aspect of machine learning. As stated, the decision tree inducer receives an input set and creates a classification tree that can categorize an invisible instance. Two evaluation criteria can be used to assess the classification tree and the inductor. The evaluation is important to understand the classification tree quality and the parameters for the KDD iterative process.

Although there is many criteria for the evaluation of classification trees' predictive performance, other criteria, such as computational complexity or classification comprehensibility, may also be important.

Information gain is used for measuring the selection of attributes. Let N node serve or carry D tuples on the partition. The largest knowledge gain attribute is chosen as the splitting attribute for node N. This feature minimizes the details required to distinguish between the tuples in the resulting partitions and represents the least randomness or impurity in these partitions. This procedure minimizes the estimated number of tests needed to identify an

individual tuple and ensures that a simple (but not necessarily the easiest) tree is identified**.** The expected formulation is:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

Equation 1 information needed to classify a tuple in *D*

where pi is the nonzero probability that an arbitrary tuple in D belongs to class Ci. *Info*(*D*) is also known as the **entropy** of *D*.

Ideally, we would like this partitioning to create a precise tuple classification. That is, we 'd like to be pure on any partition. However, the partitions are reasonably likely to be impure. The precise amount of Information needed after the partitioning to achieve exact classification is measured by

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j).$$

Equation 2 how much information is needed for exact classification

Information gain is defined as the difference between the information originally given condition (i.e. based exclusively on class proportion) and current condition (i.e., achieved after partitioning on A) is given by:

$$Gain(A) = Info(D) - Info_A(D).$$

Equation 3 Information Gain

A sort of standardization is applied to knowledge gain using a value described analogously as "split information", which can be represented in this equation:

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right).$$

Equation 4 Split Information

It varies from knowledge value, which tests the knowledge that is gained on the basis of the same partitioning as regards classification. The proportion of gains is defined as:

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}.$$

Equation 5 Gain Ratio

For CART, the index Gini is used. The Gini index tests the impurity of D, a data partition or a set of training tuples using the notation previously mentioned, as

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

Equation 6 Gini Index

where *pi* is the probability that a tuple in *D* belongs to class *Ci* .The sum is computed over *m* classes.

We calculate a weighted sum of the impurity for each resulting partition when considering a binary split. For example, if a binary divides A partitions D into D1 and D2, given that partitioning, the Gini index D is

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2).$$

Equation 7 Weighted Sum of Impurity

Each attribute is considered to be possible binary splits. The subset that gives the minimum Gini index for that attribute is selected as its dividing subset for a discrete-valued attribute.

The impurity reduction which a binary split on a discrete or continuously valued attribute A would incur is

$$\Delta Gini(A) = Gini(D) - Gini_A(D).$$

Equation 8 Impurity Reduction

The attribute that maximizes impurity elimination (or has the same attribute Minimum Gini index) is chosen as the dividing attribute. This attribute and either its splitting sub-set (for a discrete-valued splitting attribute) or split-point (for a continuously valued splitting attribute) form the splitting criteria together.

It was found that both the Entropy and Gini index use the same concept, a graphical representation for both equations was plotted to recognize the resemblance.

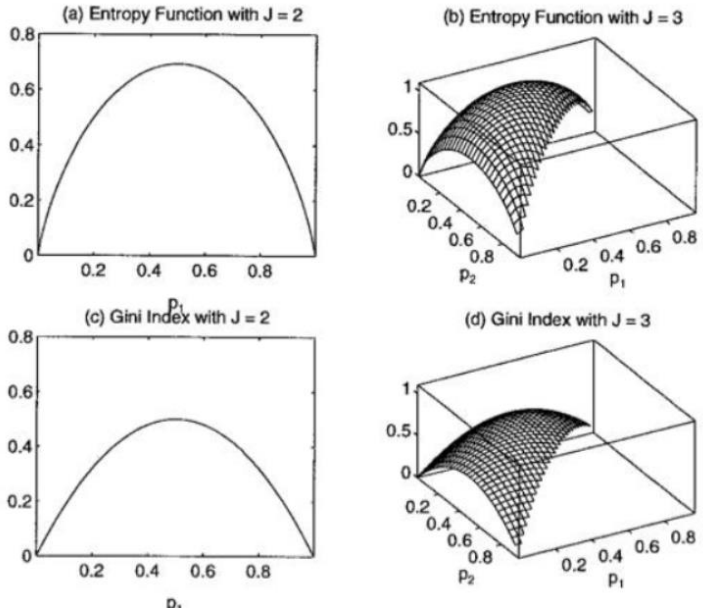


Figure 2 Graph for Entropy and Gini Index functions

ID3 and C4.5 algorithms

Throughout this section, we will shortly review some of the common induction algorithms throughout decision trees, including: ID3 and C4.5. Both these algorithms use criterion splitting and methods of pruning. The purpose of this section, therefore, is merely to indicate which setting each algorithm uses, what are the pros and cons of each algorithm and how they could be linked to probability.

ID3

The ID3 algorithm is a very basic decision tree algorithm. The ID3 algorithm uses information as a differentiation criterion to avoid evolving when all instances are part of a single value of a target item, or whether there is no more than zero maximal information gain. No pruning process applies to ID3 nor does it handle missing numerical attributes or values.

ID3 's principal quality is its versatility. Because of this, ID3 algorithm is also used for teaching purposes. ID3 however has several drawbacks:

1. ID3 does not promise an optimal solution, since it uses a greedy approach, it will get trapped in the local optimums. Backtracking can be used during the hunt to prevent optimal local conditions.
2. ID3 may be overfitting to the training details. Smaller decision trees should be favored to bigger ones to prevent overfitting. Usually this algorithm produces small trees, but it doesn't always produce the smallest tree possible.
3. ID3 is for the minimal characteristics. Therefore, continuous data can be used only after translating them to nominal bins.

Because of the aforementioned disadvantages, most practitioners favor the C4.5 algorithm over ID3 primarily since C4.5 is an extension of the ID3 algorithm that aims to fix its drawbacks

ID3 is a guided learning algorithm that generates an instance decision tree. For the classification of potential samples is used the corresponding tree. ID3 algorithm creates tree based on knowledge gain from training instances and uses it in order to identify the results of the study. ID3 algorithm usually uses minimum classification attributes that have not been missed.

Inputs: *R: a set of non- target attributes, C: the target* ***attribute****, S: training data.*

Output*: returns a decision tree*

Start

Initialize to empty tree;

If S is empty then

Return *a single node failure value*

End If

If *S is made only for the values of the same target* ***then***

Return *a single node of this value*

End if

If *R is empty* ***then***

Return *a single node with value as the most common value of the target attribute values found in S*

End if

D ← the attribute that has the largest Gain (D, S) among all the attributes of R

{dj j = 1, 2, ..., m} ← Attribute values of D

{Sj with j = 1, 2, ..., m} ←The subsets of S respectively constituted of dj records attribute value D

Return *a tree whose root is D and the arcs are labeled by d1, d2, ..., dm and going to sub-trees ID3 (R-{D}, C, S1), ID3 (R-{D} C, S2), .., ID3 (R-{D}, C, Sm)*

End

C4.5

C4.5, an ID3 evolution provided by the same author, uses the gain ratio as criterion for splitting. If the number of occasions to be divided is beyond a given level, the separation stops. After the through process error-based pruning is carried out. C4.5 can handle attributes in numeric form. It can also induce by using modified benefit ratio criteria from a training set which integrates missing values.

C4.5 algorithm provides many improvements to ID3. The most notable changes are:

1. C4.5 uses a pruning technique to detach and replace branches that do not lead to consistency with leaf nodes
2. Attributes values are permitted to be missing in C4.5
3. In order to split the spectrum of the attribute value into two sub-sets, C4.5 preserves continuous attributes. Specifically, the right threshold is sought to optimize the profit ratio criterion. The first sub-set contains all values above the threshold and the second subcommon values.

Unlike ID3, the information is sorted for division at any tree node. It uses the impurity ratio method to measure division of the product. Decision trees are built into C4.5 using training data collection or data sets like ID3. C4.5 selects at each tree node one data attribute which divides its samples most effectively into subsets enriched in each class. Its parameters are a weighted gain of information (entropy difference) as a result of the choosing of a data separation attribute. The most uniform advantage of knowledge is selected to make the choice.

ID3 Vs. C4.5

ID3 algorithm selects the best attribute for tree creation based on the principle of entropy and information gain.

C4.5 algorithm behaves like ID3 but strengthens some ID3 characteristics:

- Continuous data can be used

- Missing values are used
- Attributes of different weights are allowed to be used
- Trees pruning after being created

An accuracy comparison between both algorithms is represented in Table 1 , also its graphical representation is represented in Figure 3

Size of Data Set	Algorithm	
	ID3 (%)	C4.5 (%)
14	94.15	96.2
24	78.47	83.52
35	82.2	84.12

Table 1 accuracy comparison between ID3 and C4.5

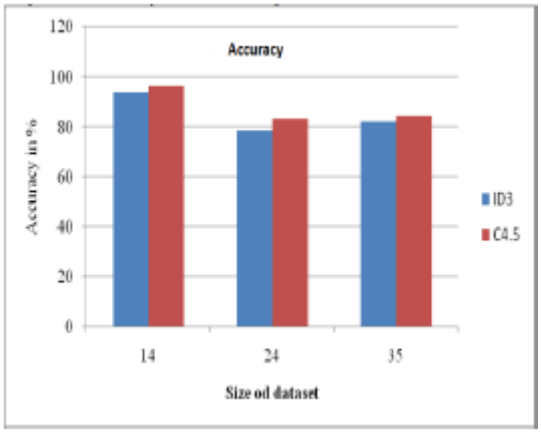


Figure 3 accuracy comparison between ID3 and C4.5

Another parameter was used for the comparison, which is execution time. The numerical data evaluated on various datasets is represented in Table 2, along with the graphical representation in Figure 4.

Size of Data Set	Algorithm	
	ID3 (%)	C4.5 (%)
14	0.215	0.0015
24	0.32	0.17
35	0.39	0.23

Table 2 Execution time comparison between ID3 and C4.5

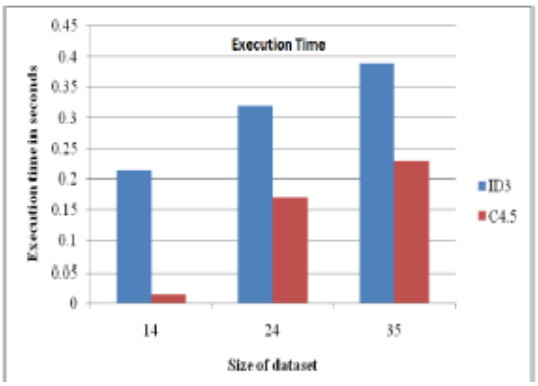


Figure 4 Execution time comparison between ID3 and C4.5

Building a decision tree

In order to build a decision tree there are certain steps that should be followed:

1. Using recursive binary splitting to expand a broad tree on the training results, stopping only when each terminal node has fewer observations than a minimum number.
2. To obtain a sequence of the best sub-trees, apply cost-complexity pruning to the large tree as a function of α.
3. Using the cross-validation K-fold to pick an α. That is, divide findings of the training into folds of K. For every k = 1,. . – K:
    a) Repeat steps 1 and 2 on all of the training data except the kth fold.
    b) Assess, as a function of α, the mean squared prediction error on the data in the left-out kth fold.
    c) Average the results for each value of *α*, and pick *α* to minimize the average error.

4. Return the sub-tree from step 2 corresponding to the value chosen for α

For each value of *α* there corresponds a subtree $T \subset T0$ such that:

$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{Rm})^2 + \alpha |T|$$

Equation 9 equation for constructing decision trees

Hitter's knowledge set was used to measure the salaries for a baseball player based on Years (the number of years that he played in the major leagues). They then remove measures of missed pay values and log-transform salaries to make the allocation more conventional.

Figures 5, 6 and 9 of the functions display the results of the application and pruning of the regression tree on the Hitters data. Next, we split the collection of data by randomness into two parts: 132 in the course set and 131 in the test set. We created a broad regression tree and variated α in (8.4) on training results, to build sub-trees with specific node numbers for the terminal. Eventually, we performed semi-validated cross validation to estimate the cross-validated MSE of the trees according to α.

The unpruned regression line is seen in Figure 5. The CV error will be shown in Figure 6 in the green curve as a function of the leaves,2 while the orange curve will represent the test error. Generic error bars are also visible which cover approximate mistakes. For comparison, the exercise error curve is in black. The CV error is an equitable estimated of the test error: a three-node tree is the most commonly reported in a CV error, whereas the test error is always the lowest in a 10-node tree.

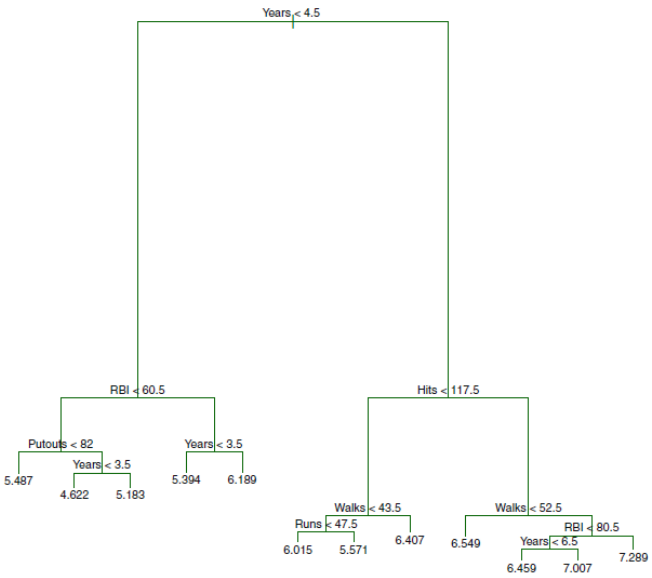


Figure 5 unpruned tree for Hitter data

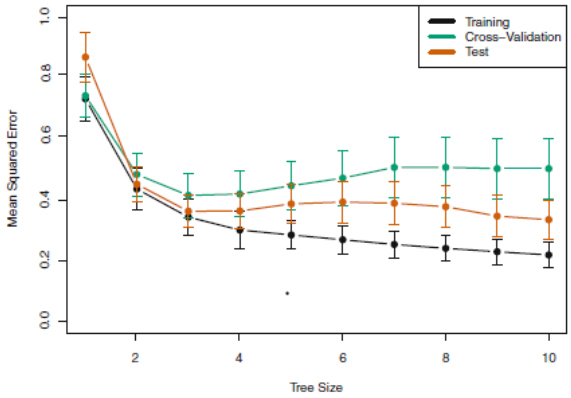


Figure 6 tree analysis for Hitter data

Classification in Decision Trees

A classification tree is used for qualitative and not quantitative answers. Any observation in a classification tree should belong to the most commonly occurring type of training observations. In assessing the results of a classification tree, we take into account not only

the class prognosis for a given node position but also the class proportions between observations for the training in this area.

We use recursive binary splitting to create a list in the construction of a classification tree. Since we plan to attribute the most common error rate to observations of training in a given area, the classifying error rate is simply a fraction of the observations of training that are not one of the most frequent in that area.

The Heart Info Collection is explained in Figure 5. These results give a conditional HD result for 303 patients suffering from chest pain. Yes, there is a result that suggests that heart disease occurs on the basis of an angiographic test; No suggests no heart defect. There are 13 predictors: age , sex, chol (a cholesterol indicator), and other heart and lung function measurements. The result is a tree with six terminal nodes.

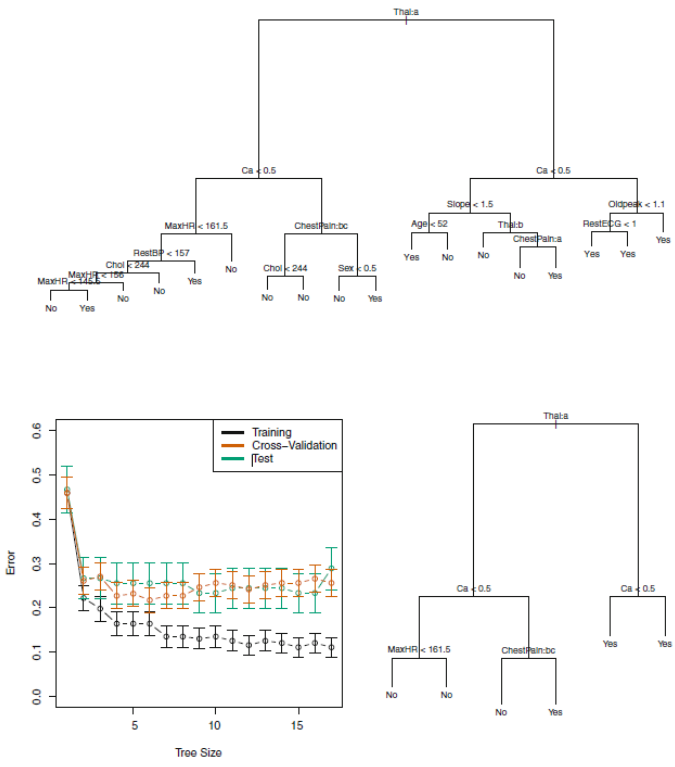


Figure 7 Heart Data. Top: unpruned tree, bottom left: error for pruned tree , bottom right: pruned tree

It was assumed that continuous values are taken in by the predictor variables. However, even in the presence of qualitative predictor variables, the decision trees can be constructed. For example, some of the predictors in the Heart results, such as Age, Thal (Thallium stress test), and Chest Pain, are qualitative. A division on one of these variables thus leads to assigning some of the qualitative values to one branch and assigning the others to the other branch.

Some of the nodes correspond to disconnected contextual factors. For example , the top inner node refers to the division of Thal. The text Thal: reveals that the left branch from this node consists of observations with the first (normal) value of the Thal portion and the remaining observations (fixed or reversible defects) consist of the right node. Chest Pain text bc breaks the tree on the left, implying that the 1st branch of that node consists of the results with the 2nd and 3rd values for Chest Pain where the possible angina, atypical angina and nonanginal pain are common.

Applications in real life

For the practical part in this research, a real dataset for Mobile prices classification has been used , where mobile prices have been defined as ranges (low, medium and high) and the price of the mobile is classified according to the given specifications of the mobile phone (attributes).

In order to analyse the dataset, we used pandas library in python ,source code can be found in appendix [1], to perform statistical and graphical analysis like: mean, mode, scatter graphs etc., and also used Weka software to perform clustering (DBSCAN, K-means) and classification using decision trees. The output obtained will be illustrated in this section.

Python

The size of the used dataset is (2000,21) , in which there is 21 attributes, represented in the number of columns, and 2000 examples in each attribute represented in the number of rows. A summary for the dataset is shown in Figure 8.

The figure illustrates the count of values found in each attribute, along with the attribute name and datatypes. The attributes are all of real mobile phone specifications like: Front Camera (fc), mobile weight (mobile_wt), etc.

```
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
 16  talk_time      2000 non-null   int64
 17  three_g        2000 non-null   int64
 18  touch_screen   2000 non-null   int64
 19  wifi           2000 non-null   int64
 20  price_range    2000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

Figure 8 Dataset Summary

Graphical representations were also used to illustrate the dataset and find the relation between attributes. In figure 9, all attributes were represented in a single graph using broken lines.

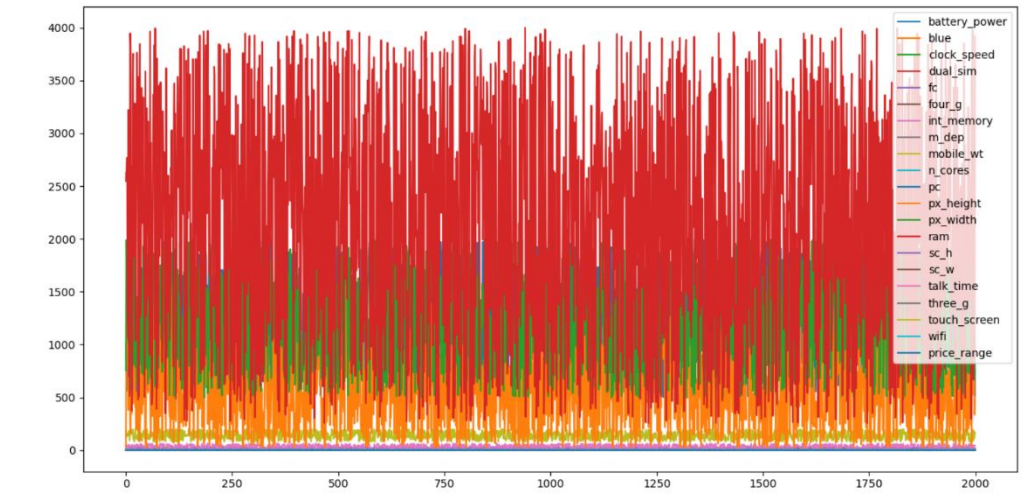


Figure 9 all attributes graphical representation

Further graphical analysis on a sample from the attributes to observe the distribution of values in each attribute. Two graphical representation were used: histogram in Figure 10, and impulse in figure 11.

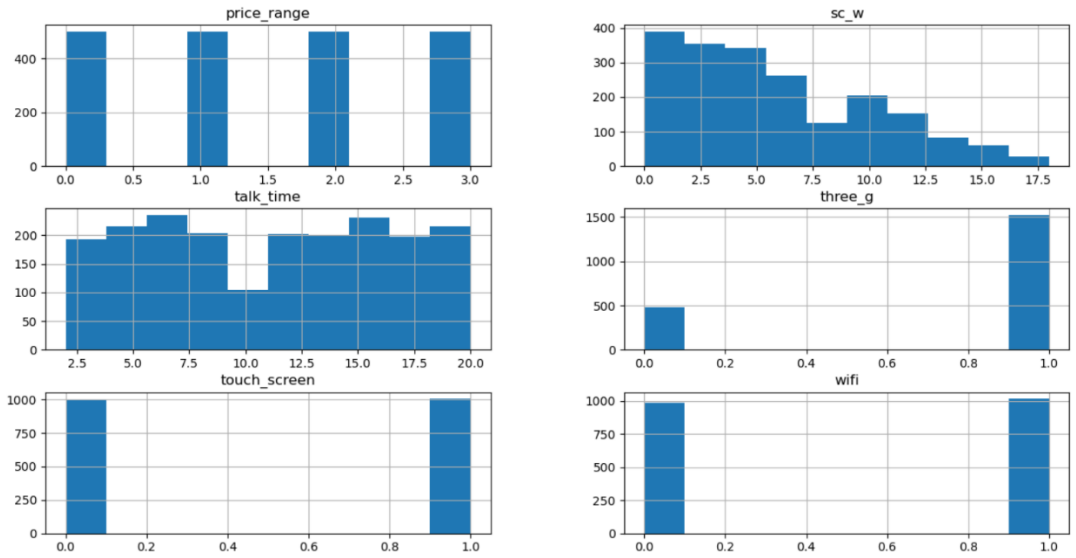


Figure 10 histogram for selected attributes

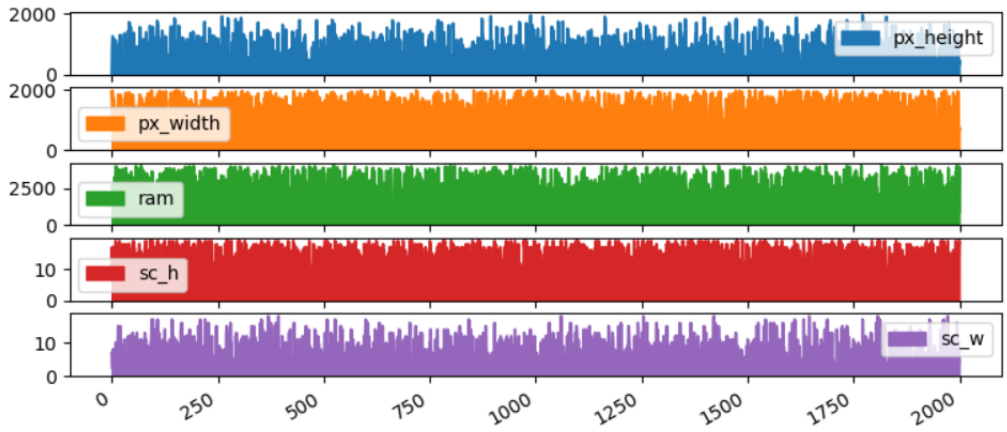


Figure 11 impulse for selected attributes

Also, a boxplot was plotted for selected attributes before and after removing outliers, so it would be easy to observe the difference of removing them. The outliers are represented in the circles found above each box. As we can deduce from figure 12 and 13, the number of circles decreased as some of the outliers have been removed.

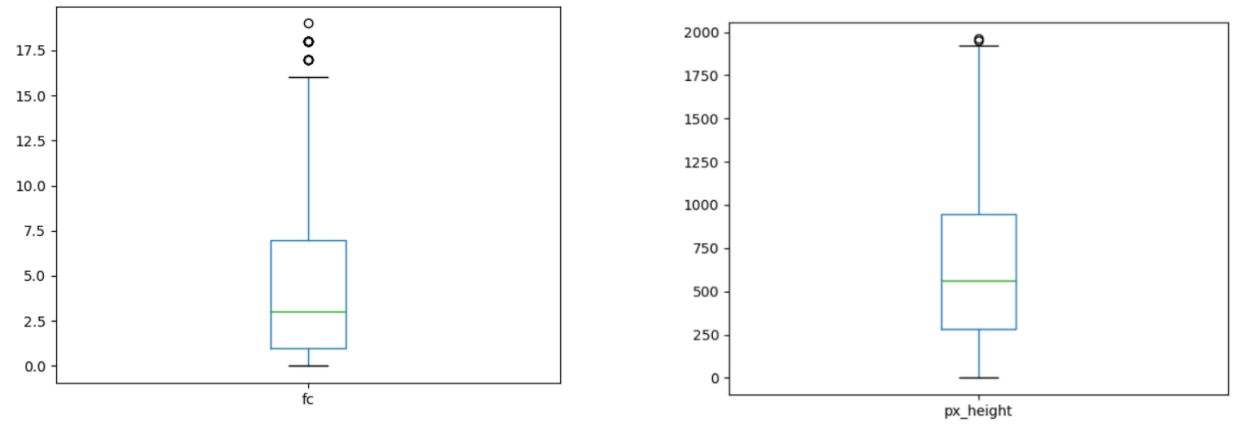


Figure 12 boxplot before removing outliers

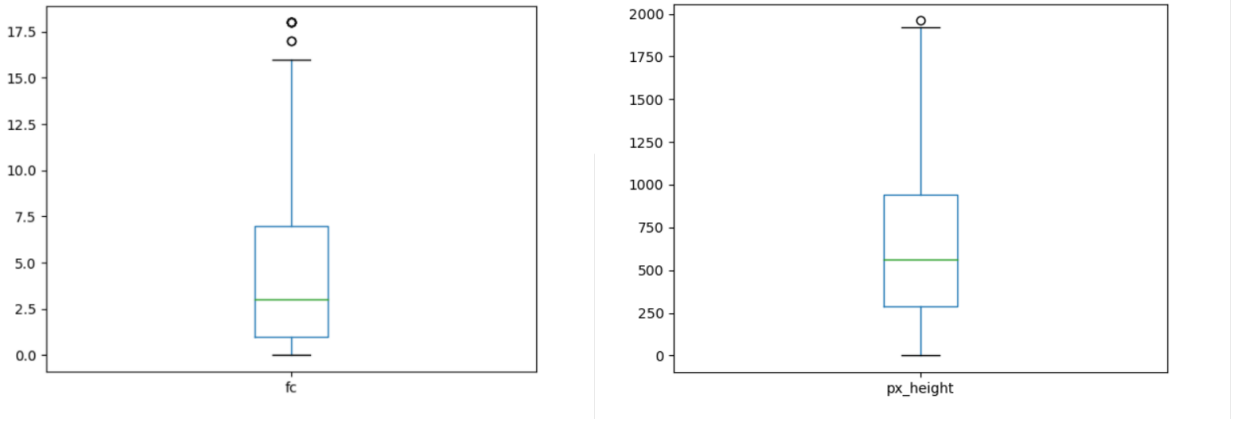
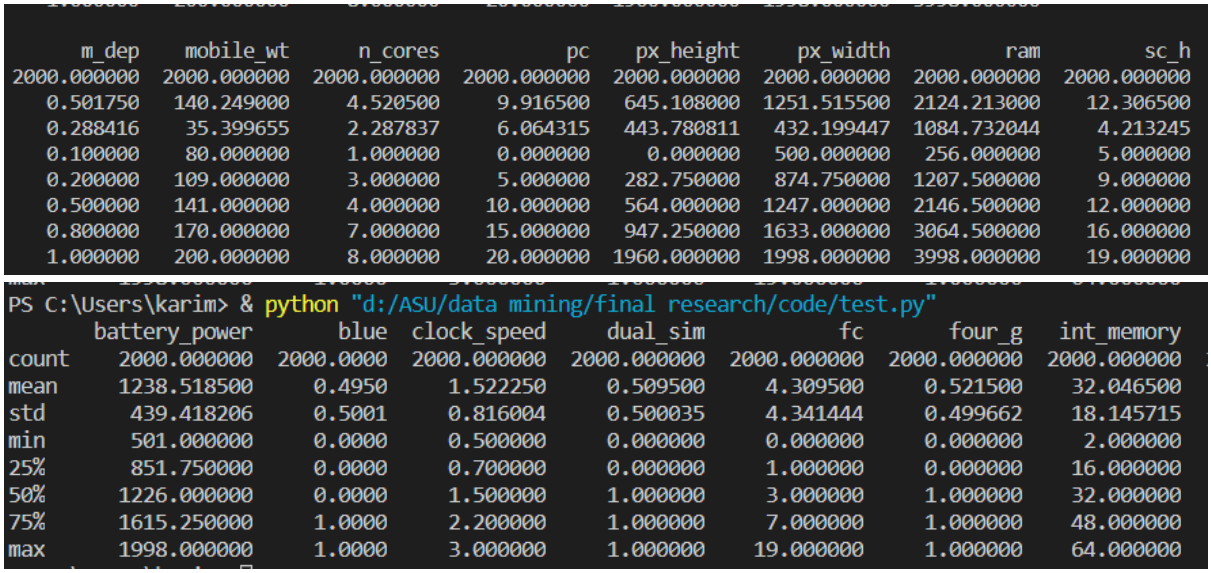


Figure 13 boxplot after removing outliers

Moreover, Statistical analysis has been performed on all attributes to analyse them. It is shown in figure 14.

```
PS C:\Users\karim> & python "d:/ASU/data mining/final research/code/test.py"
              sc_w    talk_time      three_g  touch_screen         wifi  price_range
count  2000.000000  2000.000000  2000.000000   2000.000000  2000.000000  2000.000000
mean      5.767000    11.011000     0.761500      0.503000     0.507000     1.500000
std       4.356398     5.463955     0.426273      0.500116     0.500076     1.118314
min       0.000000     2.000000     0.000000      0.000000     0.000000     0.000000
25%       2.000000     6.000000     1.000000      0.000000     0.000000     0.750000
50%       5.000000    11.000000     1.000000      1.000000     1.000000     1.500000
75%       9.000000    16.000000     1.000000      1.000000     1.000000     2.250000
max      18.000000    20.000000     1.000000      1.000000     1.000000     3.000000
```



```
      m_dep    mobile_wt     n_cores          pc    px_height     px_width          ram         sc_h
2000.000000  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000
   0.501750   140.249000     4.520500     9.916500   645.108000  1251.515500  2124.213000    12.306500
   0.288416    35.399655     2.287837     6.064315   443.780811   432.199447  1084.732044     4.213245
   0.100000    80.000000     1.000000     0.000000     0.000000   500.000000   256.000000     5.000000
   0.200000   109.000000     3.000000     5.000000   282.750000   874.750000  1207.500000     9.000000
   0.500000   141.000000     4.000000    10.000000   564.000000  1247.000000  2146.500000    12.000000
   0.800000   170.000000     7.000000    15.000000   947.250000  1633.000000  3064.500000    16.000000
   1.000000   200.000000     8.000000    20.000000  1960.000000  1998.000000  3998.000000    19.000000
PS C:\Users\karim> & python "d:/ASU/data mining/final research/code/test.py"
       battery_power         blue   clock_speed     dual_sim           fc       four_g   int_memory
count    2000.000000  2000.0000  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000
mean     1238.518500     0.4950     1.522250     0.509500     4.309500     0.521500    32.046500
std       439.418206     0.5001     0.816004     0.500035     4.341444     0.499662    18.145715
min       501.000000     0.0000     0.500000     0.000000     0.000000     0.000000     2.000000
25%       851.750000     0.0000     0.700000     0.000000     1.000000     0.000000    16.000000
50%      1226.000000     0.0000     1.500000     1.000000     3.000000     1.000000    32.000000
75%      1615.250000     1.0000     2.200000     1.000000     7.000000     1.000000    48.000000
max      1998.000000     1.0000     3.000000     1.000000    19.000000     1.000000    64.000000
```

Figure 14 statistical analysis for attributes

Weka

As mentioned before, Weka software was used for clustering and classification. For Clustering, two algorithms were used: DBSCAN, K-means. In DBSCAN, the dataset was divided into 8 clusters, and 404 instances were unclustered which could be considered as noise in the dataset. The parameters used in the DBSCAN was epsilon=0.9, minPoints=4.

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

Clustered Instances

0       410 ( 26%)
1       396 ( 25%)
2       371 ( 23%)
3       400 ( 25%)
4         4 (  0%)
5         7 (  0%)
6         4 (  0%)
7         4 (  0%)

Unclustered instances : 404
```

Figure 15 Output for DBSCAN clustering

As for K-means algorithm, the algorithm was used with K=4, so 4 clusters were obtained. The output is shown in figure 15.

```
Time taken to build model (full training data) : 0.08 seconds

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

Clustered Instances

0       473 ( 24%)
1       484 ( 24%)
2       510 ( 26%)
3       533 ( 27%)
```

Figure 16 Output for K-means clustering

Finally, Decision trees were trained and used to classify mobile price range according to the given attributes. The dataset has been split so 66% would be used for training the tree and remaining used for testing the accuracy of the trained tree. The output tree and accuracy results are shown in the appendix [2].

Conclusion

Decision trees directly respond to the issue of prejudice is one of the only approaches that can be brought forward easily enough for a non-specialist audience to interpret data without being stuck in conceptual formulas that are challenging to grasp. In this research, we illustrated the various uses of decision trees, along with the mathematical formulation of it. Also, ID3 and C4.5 were presented and how are they related to probability. Furthermore, numerical examples for building and classifying decision trees were presented and explained. Finally, a real data set was used to perform statistical analysis, clustering and classification using libraries and software for datamining uses.

References

1. Hssina, Badr & MERBOUHA, Abdelkarim & Ezzikouri, Hanane & Erritali, Mohammed. (2014). A comparative study of decision tree ID3 and C4.5. (IJACSA) International Journal of Advanced Computer Science and Applications. Special Issue on Advances in Vehicular Ad Hoc Networking and Applications. 10.14569/SpecialIssue.2014.040203.
2. Rokach L, Maimon OZ (2014) Data mining with decision trees: theory and applications, 2nd ed. World Scientific, New Jersey
3. Han, J., “Data mining: Concepts and techniques”, third edition, 2012, Waltham, Mass.: Morgan Kaufmann Publishers.
4. Gareth James, et al., “An Introduction to Statistical Learning with Applications in R”, Springer.
5. https://www.kaggle.com/iabhishekofficial/mobile-price-classification?select=test.csv
6. Maimon Oded, Rokach L (2005) Data mining and Knowledge Discovery. springer

Appendix

1.

```
import pandas as pd
import matplotlib.pyplot as plt
import sys
import numpy as np
from scipy import stats

np.set_printoptions(threshold=sys.maxsize)
# read dataset file
df = pd.read_csv('D:/ASU/data mining/final research/code/dataset_train.c
sv')
#  take part of the data to be used for plotting
part = df.iloc[:, 4]
# display quick summary for data
df.info()
# plot all attributes in one graph
df.plot()
# plot a certain range
part.plot.box()
axs = part.plot.area(figsize=(9, 4), subplots=True)
pd.DataFrame.hist(part)
part.boxplot()
plt.show()

# perform stats on all attrbiutes
print(part.describe())
# remove outliers
new_df = df
for i in df:
    z_scores = stats.zscore(df[i])
    abs_z_scores = np.abs(z_scores)
    new_df = df[abs_z_scores <= 0.5]

# plot after removing outliers
part = new_df.iloc[:, 4]
part.plot.box()
plt.show()
```

2. === Run information ===

Scheme: weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Relation: dataset_train

Instances: 2000

Attributes: 21

battery_power

blue

clock_speed

dual_sim

fc

four_g

int_memory

m_dep

mobile_wt

n_cores

pc

px_height

px_width

ram

sc_h

sc_w

talk_time

three_g

touch_screen

wifi

price_range

Test mode: split 66.0% train, remainder test

=== Classifier model (full training set) ===

REPTree

==========

ram < 2113.5

| ram < 1235

| | px_height < 1441.5

| | | ram < 1052.5

| | | | battery_power < 1816.5 : 0.03 (221/0.03) [136/0.02]

| | | | battery_power >= 1816.5

| | | | | px_width < 1180

| | | | | | ram < 900.5 : 0 (18/0) [4/0]

| | | | | | ram >= 900.5 : 0.67 (3/0.22) [0/0]

| | | | | px_width >= 1180

| | | | | | ram < 513 : 0 (4/0) [3/0]

| | | | | | ram >= 513

| | | | | | | px_height < 384 : 0 (2/0) [0/0]

| | | | | | | px_height >= 384 : 1 (10/0) [2/0]

| | | ram >= 1052.5

| | | | battery_power < 1125.5 : 0.1 (30/0.03) [12/0.23]

| | | | battery_power >= 1125.5 : 0.8 (29/0.21) [17/0.18]

| | px_height >= 1441.5

| | | ram < 565.5 : 0.2 (4/0.19) [1/0.06]

| | | ram >= 565.5 : 0.96 (14/0) [12/0.25]

| ram >= 1235

| | battery_power < 1086.5

| | | ram < 1490.5 : 0.22 (36/0.19) [22/0.15]

| | | ram >= 1490.5

| | | | px_width < 698.5 : 0.42 (11/0.25) [1/0.21]

| | | | px_width >= 698.5

| | | | | sc_w < 0.5 : 0.42 (6/0.22) [6/0.39]

```
|   |   |   |   sc_w >= 0.5
|   |   |   |   |   px_height < 1455
|   |   |   |   |   |   px_width < 908.5 : 0.88 (8/0.11) [8/0.11]
|   |   |   |   |   |   px_width >= 908.5 : 1.01 (59/0) [24/0.04]
|   |   |   |   |   px_height >= 1455 : 1.33 (4/0.19) [2/0.31]
|   |   battery_power >= 1086.5
|   |   |   px_height < 954
|   |   |   |   ram < 1896.5
|   |   |   |   |   px_width < 1545.5
|   |   |   |   |   |   battery_power < 1243 : 0.85 (18/0.1) [8/0.21]
|   |   |   |   |   |   battery_power >= 1243 : 1.02 (64/0) [39/0.1]
|   |   |   |   |   px_width >= 1545.5 : 1.19 (25/0.16) [11/0.15]
|   |   |   |   ram >= 1896.5
|   |   |   |   |   px_width < 1083.5 : 1.11 (17/0.06) [2/0.44]
|   |   |   |   |   px_width >= 1083.5 : 1.95 (13/0.07) [6/0.01]
|   |   |   px_height >= 954
|   |   |   |   ram < 1461.5 : 1.22 (13/0.07) [10/0.34]
|   |   |   |   ram >= 1461.5
|   |   |   |   |   battery_power < 1113 : 1 (3/0) [0/0]
|   |   |   |   |   battery_power >= 1113
|   |   |   |   |   |   battery_power < 1570.5
|   |   |   |   |   |   |   px_height < 1298.5
|   |   |   |   |   |   |   |   ram < 1807.5 : 1 (4/0) [2/0]
|   |   |   |   |   |   |   |   ram >= 1807.5 : 1.78 (5/0.16) [4/0.19]
|   |   |   |   |   |   |   px_height >= 1298.5 : 2 (9/0) [2/0]
|   |   |   |   |   |   battery_power >= 1570.5 : 2 (18/0) [5/0]
ram >= 2113.5
|   ram < 3013.5
|   |   battery_power < 1370.5
|   |   |   ram < 2421
|   |   |   |   px_width < 1389
|   |   |   |   |   battery_power < 952.5 : 1 (24/0) [11/0]
```

```
|   |   |   |   battery_power >= 952.5
|   |   |   |   |   px_height < 486 : 1.11 (11/0.08) [8/0.11]
|   |   |   |   |   px_height >= 486 : 1.67 (7/0.2) [5/0.25]
|   |   |   |   px_width >= 1389
|   |   |   |   |   m_dep < 0.95
|   |   |   |   |   |   pc < 5 : 1.55 (6/0.22) [5/0.31]
|   |   |   |   |   |   pc >= 5 : 1.96 (19/0) [7/0.14]
|   |   |   |   |   m_dep >= 0.95 : 1 (2/0) [0/0]
|   |   |   ram >= 2421
|   |   |   |   px_height < 988
|   |   |   |   |   battery_power < 631
|   |   |   |   |   |   px_width < 859.5 : 1.1 (7/0.12) [3/0.02]
|   |   |   |   |   |   px_width >= 859.5
|   |   |   |   |   |   |   px_height < 202 : 1.2 (2/0) [3/0.33]
|   |   |   |   |   |   |   px_height >= 202 : 1.91 (16/0.06) [6/0.15]
|   |   |   |   |   battery_power >= 631
|   |   |   |   |   |   ram < 2641 : 1.83 (34/0.13) [12/0.2]
|   |   |   |   |   |   ram >= 2641 : 2.01 (57/0) [15/0.2]
|   |   |   |   px_height >= 988 : 2.26 (26/0.18) [12/0.23]
|   |   battery_power >= 1370.5
|   |   |   px_width < 1199.5
|   |   |   |   ram < 2834
|   |   |   |   |   ram < 2294
|   |   |   |   |   |   battery_power < 1567 : 1.33 (4/0.19) [5/0.26]
|   |   |   |   |   |   battery_power >= 1567 : 2 (6/0) [3/0]
|   |   |   |   |   ram >= 2294 : 2.01 (44/0) [24/0.04]
|   |   |   |   ram >= 2834 : 2.29 (12/0.22) [5/0.18]
|   |   |   px_width >= 1199.5
|   |   |   |   ram < 2499.5 : 2.23 (30/0.16) [13/0.22]
|   |   |   |   ram >= 2499.5 : 2.82 (30/0.12) [19/0.21]
|   ram >= 3013.5
|   |   battery_power < 651.5
```

```
|   |   |   px_height < 831.5 : 2.18 (27/0.13) [11/0.21]
|   |   |   px_height >= 831.5 : 2.89 (7/0.12) [11/0.09]
|   |   battery_power >= 651.5
|   |   |   ram < 3314.5
|   |   |   |   battery_power < 1040.5
|   |   |   |   |   px_height < 887.5
|   |   |   |   |   |   px_width < 1851.5
|   |   |   |   |   |   |   ram < 3273 : 2 (19/0) [5/0]
|   |   |   |   |   |   |   ram >= 3273 : 2.5 (2/0.25) [0/0]
|   |   |   |   |   |   px_width >= 1851.5 : 3 (3/0) [0/0]
|   |   |   |   |   px_height >= 887.5 : 3 (4/0) [3/0]
|   |   |   |   battery_power >= 1040.5
|   |   |   |   |   battery_power < 1394
|   |   |   |   |   |   px_width < 1025 : 2 (7/0) [3/0]
|   |   |   |   |   |   px_width >= 1025 : 2.87 (19/0.09) [11/0.15]
|   |   |   |   |   battery_power >= 1394 : 2.95 (42/0.02) [14/0.14]
|   |   |   ram >= 3314.5
|   |   |   |   battery_power < 731
|   |   |   |   |   px_width < 1142.5 : 2.38 (6/0.22) [2/0.28]
|   |   |   |   |   px_width >= 1142.5 : 3 (9/0) [5/0]
|   |   |   |   battery_power >= 731 : 2.97 (203/0.01) [107/0.05]

Size of the tree : 113

Time taken to build model: 0.1 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.01 seconds

=== Summary ===
```

Correlation coefficient	0.936
Mean absolute error	0.2204
Root mean squared error	0.3954
Relative absolute error	22.1624 %
Root relative squared error	35.5173 %
Total Number of Instances	680