Bachelor's Thesis (UAS)

Degree Program in Information Technology

Information Technology

2014

Yini Xu

DATA MINING AND ITS APPLICATION TO POWER MARKET

– The programming and application of decision tree algorithm to predict the price of electricity

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR’S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology

2014 | 38

Instructor: Patric Granholm

Yini Xu

DATA MINING AND ITS APPLICATION TO POWER MARKET

The increasing amount of data has taken a significant space of our expensive hard-disks. People are looking for a solution to deal with these data, from which they can extract useful information and put it into further analysis. Apparently, data mining seems to be the best solution at this moment. After being processed by analysts, the hidden patterns emerging from the huge amount of data can be used to predict future trends. Data mining techniques can be used in many fields, such as stock market, chemical constituent analysis and power market.

As a mainstream of data mining algorithms, decision tree (DT) has its advantages in many aspects, such as uncertainty manageability, computational efficiency and interpretability (visualization). In particular, the models of binary tree have a more descriptive name, Classification and Regression Trees (CART).

The thesis first introduces the definition of DT and analyzes the algorithms of CART in detail. Then it presents several typical applications using CART, such as medical diagnosis and identification of radar waves, followed by a case study based on the analysis of annual power market data of Danish power system in 2011. In addition, a MATLAB implementation of the DT algorithm is attached in the appendix.

This project has successfully achieved the DT program in MATLAB. By using this program, the relationship of electricity price and power flow between countries can be discovered. More importantly, the future electricity price of western Denmark can be forecast to some extent, which is beneficial to the electricity consumers by guiding them in optimizing the electricity consumption.

KEYWORDS:

CART, data mining, decision tree, MATLAB

CONTENTS

FIGURES

TABLES

1 Introduction

History is the ship carrying living memories to the future, so ares the historical data. Nowadays, more and more industries rely on digging historical data in order to predict the future trends.

1.1 Background

As a new powerful technology, data mining has great potential in predicting the future behaviors of physical nature. It is frequently used in many fields, such as finance, engineering, marketing, and medicine. For instance, by collecting and analyzing some variables from heart attack patients, including age, heartbeat, blood pressure, and other 16 binary variables, the high risk patients can be recognized. We have every reason to believe that if the doctor could recognize the high risk patients at the initial 24 hours, it might save their lives. Another example, according to customers' age, gender, income, hobbies, and some other variables, companies can identify potential clients. In this case, if companies pay more attention to those customers, more profits can be gained.

1.2 Overview

1.2.1 Data Mining

Data mining, the technology, art and science of delving into complex and large bodies of data in order to ascertain useful patterns, is a part of the general process of knowledge discovery in databases (KDD). Practitioners and theoreticians are incessantly seeking ameliorated techniques to make the process more accurate, cost-effective and efficient.

Data mining uses modern highly automated, computer intensive methods which are based on machine learning, database systems, statistics and artificial intelligence as seen in Figure 1.

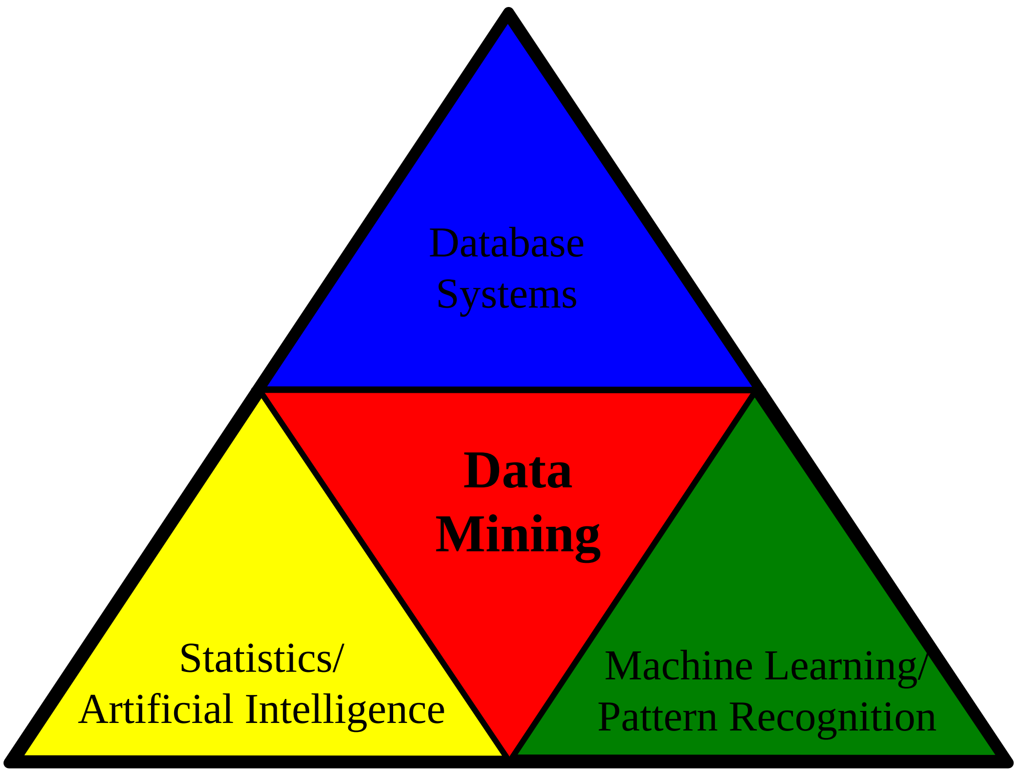

Database Systems
Data Mining
Statistics/ Artificial Intelligence
Machine Learning/ Pattern Recognition

Figure 1. Intersection methods of data mining.

1.2.2 Decision Tree

Tree methodology, unlike other statistical procedures, was developed from pencil to calculators and finally to computers. It is a child of the computer age and was unthinkable before computers. Binary trees use an absorbing and often irradiating way to analyze datasets in classification and regression problems.

The technique now known as Classification and Regression Trees (CART) was initially described in a technical monograph and was only available as a set of hard-coded FORTRAN algorithms. With rising interest in data mining and accompanying exponential growth in dataset size and complexity, the key advantages of CART have become transparent. Over the years CART has become known as the fastest and most versatile predictive modeling algorithm available to analyst, it is also used as a foundation for many modern data mining approaches based on bagging and boosting (Rokach & Maimon, 2008).

Figure 2 summarizes the general learning approach shared by many modern data mining methods, including CART.

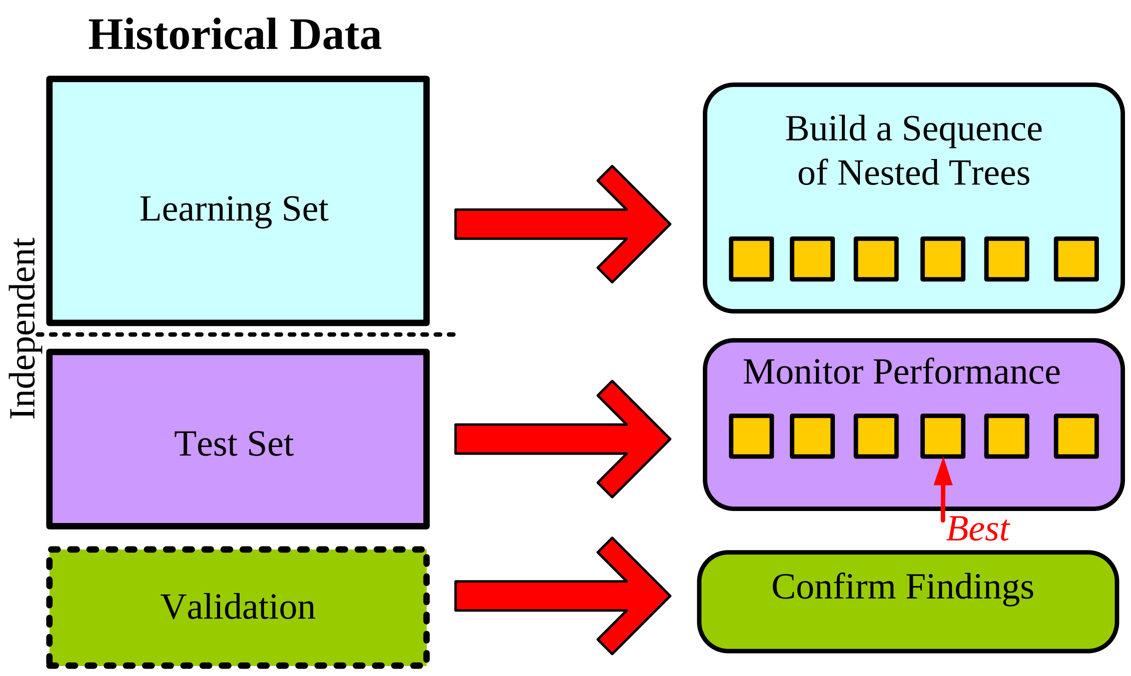


Figure 2. General learning approach.

1.3 Applications

The use of Regression Trees dates back to the Automatic Interaction Detection (AID) program, which was developed at Institute for Social Research, University of Michigan, by Morgan and Sonquist in the early 1960s. In early 1970s, Morgan and Messenger developed the initial classification program which is named THAID. In 1980, the conception of CART (Classification and Regression Trees) emerged. Now CART has been widely applied in many industry fields such as financial analysis, medical diagnostics, power system, and chemical constituent identification.

1.3.1 Standard Structure

With standard datasets, the most interesting applications have been to medical data. This is best illustrated with a famous example – the UCSD (University of California, San Diego Medical Center) Heart Disease study (Breiman, et al., 1984). The study contributed to the initial spread of CART among the medical community researchers back when the technique was still in relative obscurity. The aim of the study is to identify patients who are at risk of dying and who have survived at least 24 hours past admission to the UCSD.

Diagnosis of a heart attack is based on Chest pain, Indicative electrocardiograms (EKGs), Elevation of enzymes typically released by damaged heart muscle, and so on. The medical center has recorded 100 variables for each patient, which include demographics, medical history, lab results, and 19 noninvasive variables were used in the analysis.

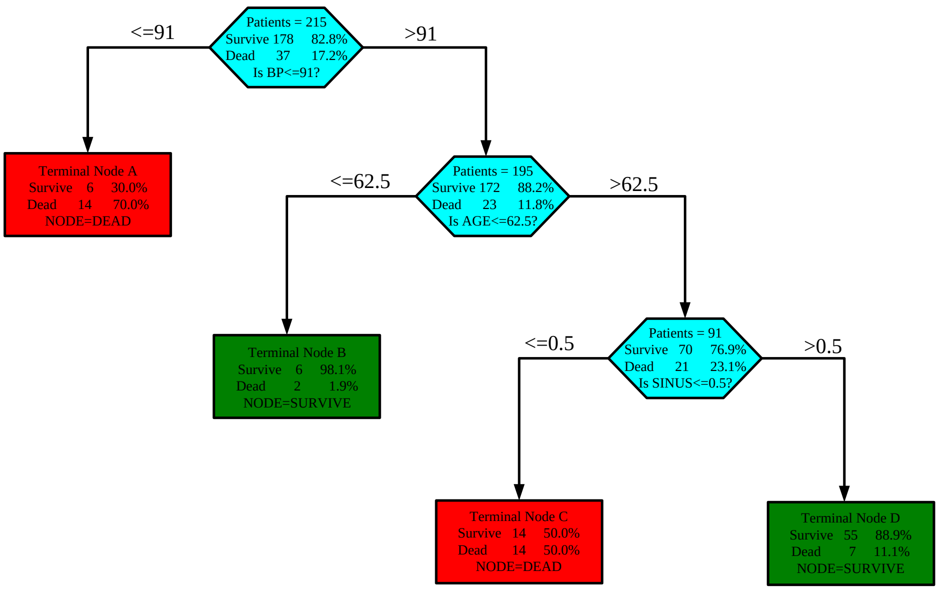


Figure 3. Typical CART solution.

The solution is presented in form of a decision tree is called Binary Recursive Partitioning, which mimics a thought process usually done by a physician.

1.3.2 General Structure

With general structured datasets, there is the example of ship classification problem. The project was implemented by Hooper and Lucero in 1976 (Breiman, et al., 1984), which involved identify six ship classes through their radar range profiles. An airplane flying around the ship in a large circle is used to gather the data in six structural types. The intensity returned by the radar depends on the smoothness of the shape of the ship: smoother shapes result in smaller intensities. In this case, the reflection from the ocean could be ignored, however the shape of the ship could be figured distinctly.

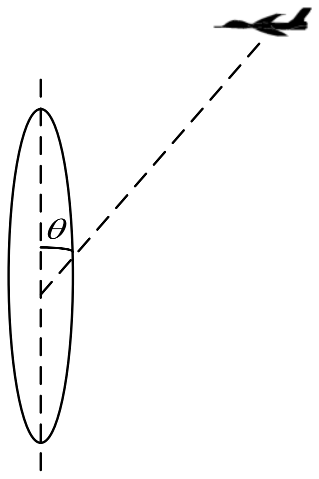


Figure 4. Angle between ship and airplane.

The number of profiles which were taken at angles of approximately 20 degrees around the compass (see Figure 4 and Figure 5) for different ship classes could be expressed in a plot.

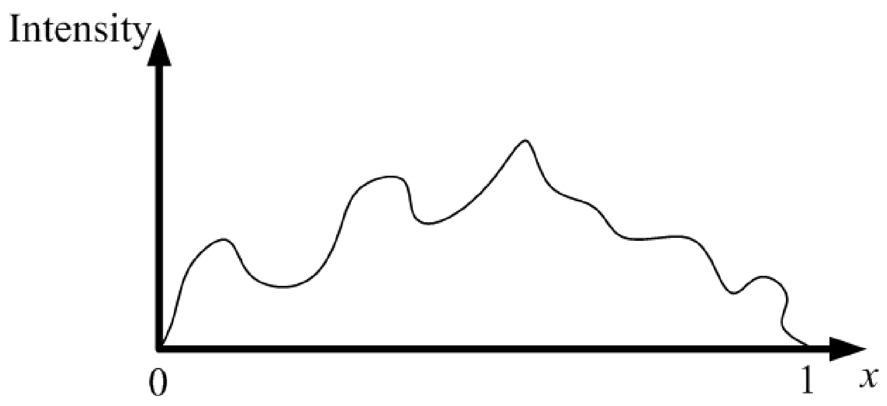


Figure 5. Typical range profile.

According to Figure 6, the corresponding x of each peak has been recorded from x1 to x15, where the x1 is the position of the first local maximum.

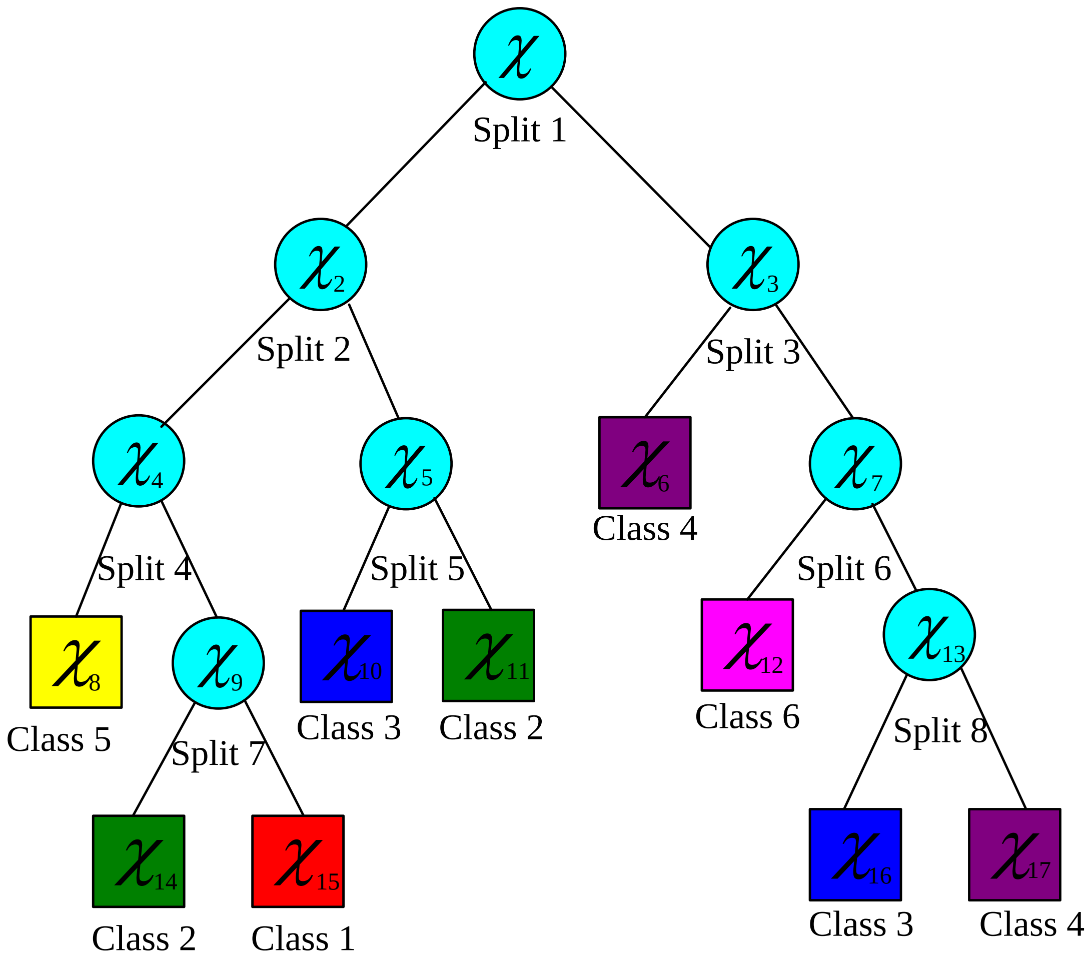


Figure 6. Tree model.

It is shown that the model of the tree is more complicated than Binary Recursive Partitioning~~,~~ which has more than two targets.

2 Decision Tree Algorithm

2.1 Essence of Decision Tree

A decision tree is a classifier expressed as a recursive partition of the instance space. An entire tree represents a complete analysis or model. The decision tree consists of nodes that form a rooted tree, which means it is a directed tree with a node called a “root” that has no incoming edges. The root of the inverted tree contains all data, which it can create child nodes, and the child nodes can in turn create child nodes of their own. The path through the tree governed by the answers to QUESTIONS or RULES is formulated so that only two answers possible are called binary partitioning. In CART the YES answer always goes left. Each node is assigned to one class which represents the most appropriate target value. Alternatively, the node may hold an affinity vector (probability vector) which indicates the probability of the target attribute having a certain value. At some point, a given path ends in a terminal node and all records in the node are assigned to a single class. The implied classification threshold is constant across all nodes and depends on the currently set priors, costs, and observation weights.

Figure 7 shows the principle of data mining algorithms. Sufficient amounts of data are collected from the dataset population, used as the learning database, including input variables as predictors and output variables as target. The data mining engine plays a significant role to discover important patterns that are hidden in the learning database followed by the evaluation and interpretation of the created model. Thereafter, a new dataset is used as the input to predict the classification or the target value of the output.

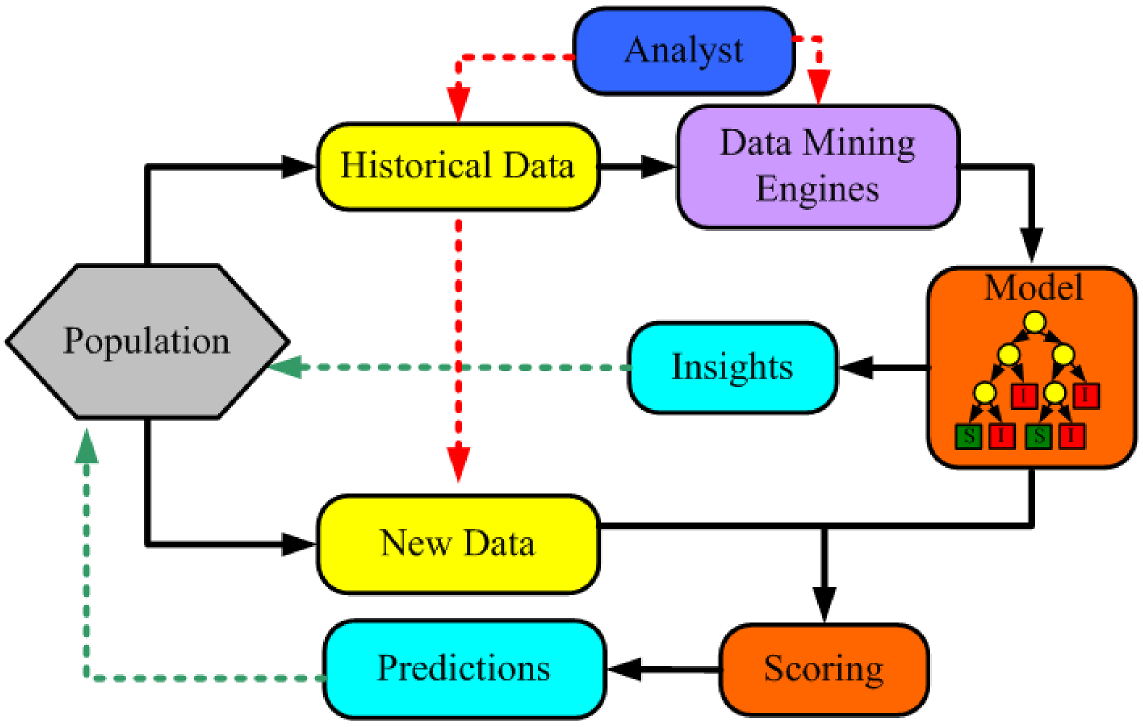


Figure 7. Essence of Machine learning.

As illustrated in Figure 8, given a set of attributes (i.e., A,B,C,…) as input predictors of case *n*, the target value (i.e. Discrete or Continuous) of the case can be predicted by dropping the case downward along a path of “if-then” questions from the root node to a terminal node of a DT. The vector of predictors can be composed of both numerical variables (e.g., A) and categorical variables (e.g., B). Variables are called numerical variables if the predictors are real numbers, but are called categorical variables if they take values in a finite set (e.g., S) which may not have any natural ordering. The DT is a classification tree (CT) if the target is a discrete class, while it is a regression tree (RT) if the target is a continuous value.

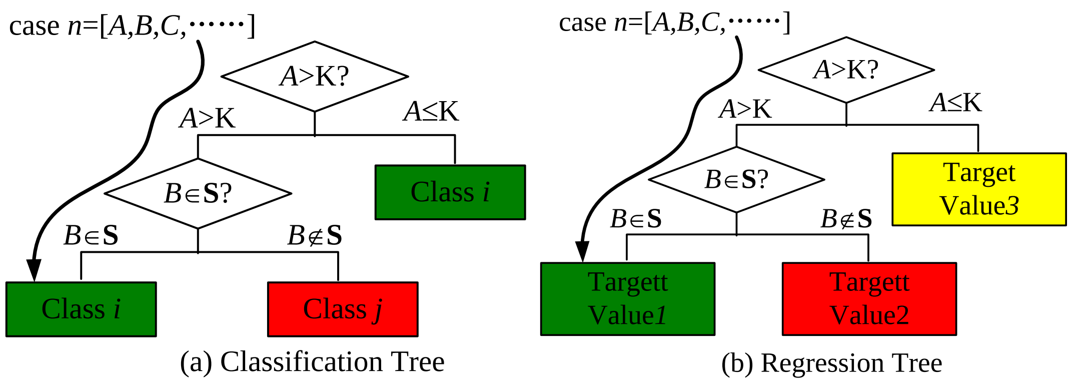


Figure 8. A simple (a) classification tree and (b) regression tree.

The following are the general procedures when using a decision tree:

- Tree growing

  Consider all possibilities of the splits of a node

  Find the best split according to the given improvement function

  Continue sequentially until the largest tree is grown

- Tree pruning – create a sequence of nested trees (pruning sequence) by systematic removal of the weakest branches

- Optimal Tree selection – using test sample or cross-validation find the best tree in the pruning sequence

2.2 Splitting Rules

2.2.1 Optimal Splits

A database composed of a number of cases is necessary for training the DTs. Each case in the database consists of a vector of predictors and a target, the predictors usually serve as the input attributes, and the target is the output. The cases in the database are divided into a learning set (LS) and a test set (TS). The LS is used to grow a series of DTs with increasing sizes, while the TS is used to evaluate their accuracies to decide the optimal one. As demonstrated by a 2-class problem in a 2-D space in Fig. 9, the training process of the DT is to iteratively split the dataset into 2 subsets. The fundamental idea to select each splitting rule is such that to make the cases in each of the divided subset as pure as possible.

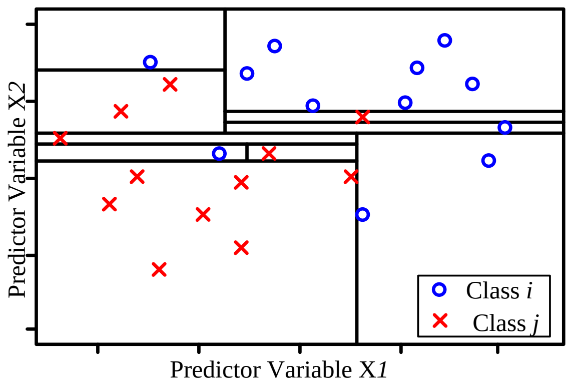


Figure 9. Optimal splitting rules in 2-D space.

More generally, for a J class problem in the LS, let N_j be the number of cases in class j ($j = 1,\ldots, J$). In a node t, the total number of cases in LS which $x_n \in t$ is denoted as $N(t)$, and the number of class j cases in node t is denoted $N_j(t)$. The proportion of class j cases in LS falls into node t is $N_j(t)/N_j$. Therefore, for a given set of prior probabilities $\pi(j)$ that class j will be presented in the new dataset, the joint probability that a case will both be class j and fall into node t can be defined by Equation 2.1.

$$p(j,t) = \pi(j)N_j(t)/N_j \qquad \textbf{(2.1)}$$

Hence, the probability that any case falls in to node t is defined by Equation 2.2,

$$p(t) = \sum_j p(j,t) \qquad \textbf{(2.2)}$$

According to Bayes' Theorem, the conditional probability that a case is class *j* given that it falls into node *t* is given by Equation 2.3,

$$p(j \mid t) = p(j,t)/p(t) \qquad \textbf{(2.3)}$$

which satisfies the equation in Equation 2.4.

$$\sum_j p(j \mid t) = 1 \qquad \textbf{(2.4)}$$

Many impurity indices are adopted to find optimal splits of a DT, such as GINI index and entropy index, as defined in Equation 2.5 and Equation 2.6,

$$I_{GINI}(t) = \sum_{i \neq j} p(i \mid t) p(j \mid t) \qquad \textbf{(2.5)}$$

$$I_{ENTROPY}(t) = -\sum_j p(j \mid t) \log p(j \mid t) \qquad \textbf{(2.6)}$$

If a split δ of node t is predicted to send a proportion p_L of the data cases to left descendant node t_L and proportion p_R to right descendant node t_R, the decrease of impurity is defined by $\Delta I(\delta,t)$, as given in Equation 2.7. The optimal selection of splitting rules for each node can be calculated by repeated attempts to maximize the decrease of impurity $\Delta I(\delta, t)$ which is equivalent to select those splits that minimize the overall impurity of the whole tree $I(T)$, as defined in Equation 2.8, in which Tt are terminal nodes of the DT.

$$\max.\Delta I(\delta,t) = \max.\left\{\left[I(t) - p_L(t)I(t_L) - p_R(t)I(t_R)\right] p(t)\right\} \qquad \textbf{(2.7)}$$

$$I(T) = \sum_{t \in Tt} I(t) = \sum_{t \in Tt} i(t)p(t) \quad \textbf{(2.8)}$$

Initially, the stop-splitting rule was either that the improvement of splitting is below a threshold β or the number of cases in the terminal node is less than a given number N_{Tt}, as defined in Equation 2.9 and Equation 2.10.

$$\max_{t \in T_t} .\Delta I(\delta, t) \leq \beta \quad \textbf{(2.9)}$$

$$N(t) \leq N_{Tt} \quad \textbf{(2.10)}$$

2.2.2 Competitor

In DT, competitors are the next best splits in the given node ordered by improvement values. It means that, by considering the impurity improvement, the winner will become the main optimal splitter; the top runner-ups will become competitors.

2.2.3 Association Calculation

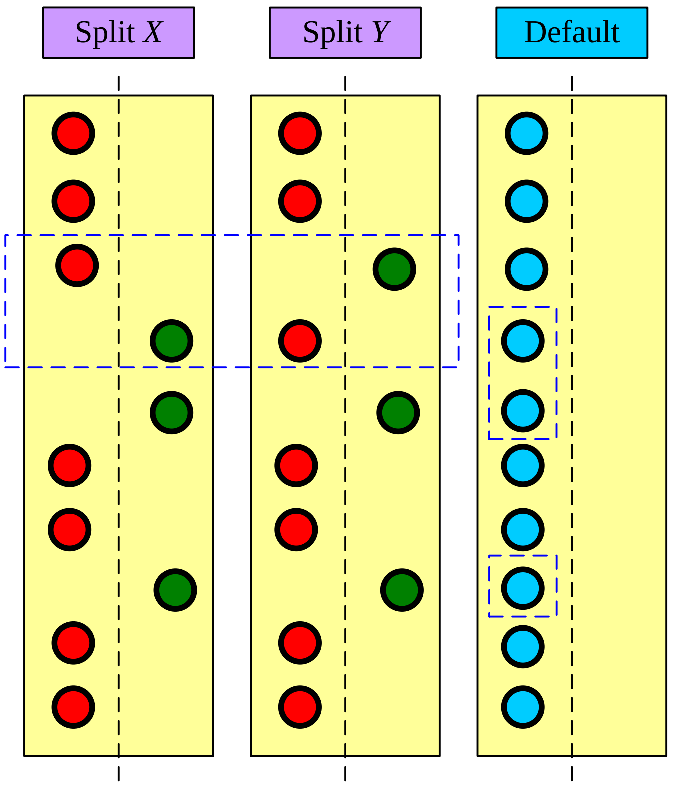


Figure 10. Association = (Default – Split Y) / Default = (3-2)/3 = 1/3.

Association measures the degree of resemblance. There are ten observations in a node, in which X is the main splitter. However, splitter Y mismatches on 2 observations. The default rule sends all cases to the dominate side mismatching 3 observations. As Figure 10 shows, the Split Y is better than the default and therefore has a positive association value.

2.2.4 Surrogate

Surrogates are the splits most resembling (case by case) the main split. Surrogates represent powerful means of handling missing values. Suppose that the main splitter is INCOME reported on a survey from people with very low or very high income and it is likely to be blank. The splitter itself wants to separate low income on the left from high income on the right. Treating missing INCOME as a separate value will do no good. This is where using a surrogate split will help to resolve ambiguity and redistribute missing income records between the two sides. For example, all low education will join the low income side while all high education subjects will join the high income side.

CART is the only data mining engine known to us that has universal handling of all missing value situations. This includes the difficult cases of having missing data in the TEST sample with clean LEARN sample and the other way around.

2.3 Accuracy Estimate

The accuracy of DTs can be evaluated by an internal or external test. An internal test uses the TS selected from the same database to evaluate the accuracy, while an external test tests the accuracy of the created DT by using the cases of another dataset. There are 3 methods of internal tests to estimate the accuracy of a DT.

2.3.1 Resubstitution Estimate

The first, overoptimistic and least accurate, is the resubstitution estimates, which is computed using the same data as LS used to train the DT.

Define the function $X(S)$ to be 1 if the statement S inside the parentheses is true, otherwise zero, as defined by Equation 2.11.

$$X(\mathrm{S}) = \begin{cases} 0 & \text{statement S is true} \\ 1 & \text{statement S is false} \end{cases} \tag{2.11}$$

The accuracy of DT can be evaluated by resubstitution estimate as defined by Equation 2.12.

$$R(d) = \frac{1}{N} \sum_{n=1}^{N} X\left(d(x_n) \neq j_n\right) \tag{2.12}$$

As an exaggerated example, by using a dataset as LS to grow a tree model, then with the same dataset as TS to estimate the misclassification cost of the model. If the tree is over fitting, as each leaf has only one pattern, the TS will match the tree model entirely, which means the value X is all 0. Then $R(d)$ = 0, but it is hard to believe that the misclassification cost is anywhere near zero.

2.3.2 Test Sample Estimate

The second is the test sample estimates. The cases in the database are randomly divided into a LS and TS. The LS is used to train the DT, while the TS is used to evaluate the accuracy of the created DT.

The accuracy of test sample estimate is given by Equation 2.13.

$$R^{TS}(d) = \frac{1}{N_{TS}} \sum_{(x_n, j_n) \in TS} X\left(d(x_n) \neq j_n\right) \tag{2.13}$$

Frequently datasets are split into 2/3 and 1/3, where 2/3 of the data are LS which used to construct tree model, and only 1/3 used in TS which could estimate the real misclassification cost. If the size of the sample is large, for example in a mass spectra problem, this is a lesser difficulty. The test sample estimation is efficient and reliable.

2.3.3 Cross Validate Estimate

The third method, called cross-validation, is parsimonious with data and preferred for databases with small sizes. The case in the database L is randomly divided to V

subsets of equal size (i.e. $L_1, L_2 \ldots L_V$), For every i iteration ($i = 1,2,\ldots,V$), L-L_i is used as the LS, and L_i is used as TS. The accuracy of DT is the average estimates of all V iterations.

The accuracy of test sample estimate is defined by Equation 2.14.

$$R^{CV}(d^{(V)}) = \frac{1}{N_V} \sum_{(x_n, j_n) \in L_V} X\left(d^{(V)}(x_n) \neq j_n\right) \qquad \textbf{(2.14)}$$

where N_{TS} is the number of cases in TS, and $N_V=N/V$ is the number of cases in L_V.

The N-fold cross validation is known as “leave one out” estimate. It is commonly used in tenfold. Every case in L is used in LS to construct the tree model, and every case could be used exactly once in TS as well.

Among these tree methods, the second one, namely, test sample estimate, is the most popular and acceptable one.

2.3.4 Misclassification Cost

The misclassification cost is used to find the optimal DT, which is defined by the proportion of misclassified cases with cost as its weighing factor, as defined in Equation 2.15,

$$C(d) = \frac{1}{N} \sum_{i,j} \left[c(i \mid j) \cdot X\left(d_j(x_n) \neq j_n\right) \right] \qquad \textbf{(2.15)}$$

where N is the number of test cases, $c(i|j)$ is the cost of misclassifying a class j case as a class i case.

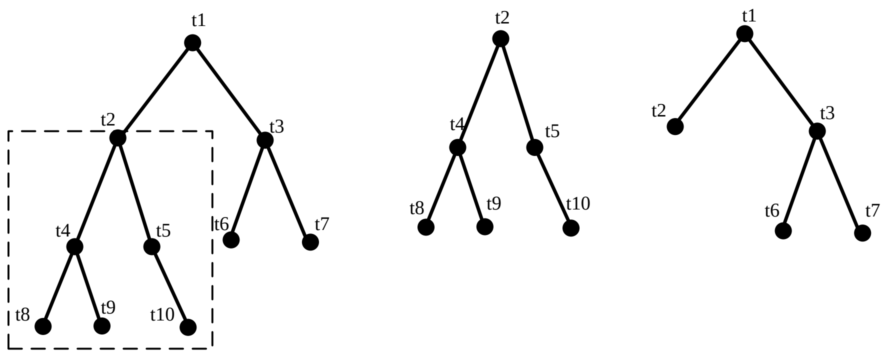
The misclassification cost is used case by case. For example, as mentioned in the introduction chapter, people use airplanes to identify the type of the ship. When misclassification cost is put into practice, as an external estimate, to assume the weight of the mistake which distinguishes a warship from a cruise is 3 and conversely is 1.

The weight of the mistake is considered by how heavy the effect is. Obviously, the former mistake is much more serious than the later one. Hence, the misclassification cost is high if there are mistakes with high weight.

2.4 Pruning of Decision Tree to the Right Size

As mentioned before, the key criterion of a successful DT is the high accuracy, in other words, low misclassification rate/cost of the cases in TS. Too large a tree will have a high misclassification rate/cost than the right sized one. The reason is that the over-fitting of LS is likely to result in the high misclassification rate/cost of the cases in TS. On the other hand, too small a tree will not use some of the classification information available in LS, again resulting in a higher misclassification rate/cost than the one with right size.

As shown in Fig. 11, the pruning process of a tree T consists of deleting a branch T_b from T, that is, cutting of all T_b expect its root node t. The pruned substree is denoted as $T_k = T_{max} - T_b$ and T_k p $T_{\max}$, where k is the number of nodes in the pruned tree.



(a) unpruned tree T_{max} (b) pruned brance T_b (c) pruned subtree T_k

Figure 11. Process of pruning a DT.

There are two criteria to select the optimal DT. The first one is to use the accuracy estimates to select the optimal substree T_{opt} with the right size for all a series of pruned substrees by the rule defined in Equation 2.16.

$$R(T_{opt}) = \min_{k \in [1,\max]} .R^{ts}(T_k) \quad \textbf{(2.16)}$$

The second one considers statistical error. The standard error estimate for $R^{TS}(T_k)$ is denoted by $\Delta R^{TS}(T_k)$ calculated by Equation 2.17.

$$\Delta R^{TS}(T_k) = \sqrt{\frac{R^{TS}(T_k)\left[1 - R^{TS}(T_k)\right]}{N^{TS}}} \quad \textbf{(2.17)}$$

Two DTs whose misclassification rate is smaller than the standard error estimate of either one have almost the same performance, and the optimal DT is selected as the smaller sized one within the range of $R(T_{opt}) \pm \Delta R^{TS}(d)$, defined in Equation 2.18.

$$R(T_{opt}) = \min_{k \in [1,\max]} .\left[R^{ts}(T_k)\right] \pm \Delta R^{TS}(T_k) \quad \textbf{(2.18)}$$

2.5 Prior Probabilities

In some studies, the cases in the dataset are very unbalanced with classes, so prior probabilities are a useful set of parameters to assist in constructing a more reliable DT. Prior probabilities can be adjusted to effectively control the splitting rules over the tradeoff between the class purity and tree accuracy. Combined with Equation 2.1 and Equation 2.2, the probability that any case falls in to node t can be defined by Equation 2.19,

$$p(t) = \sum_j \left[\pi(j) N_j(t) / N_j \right] \tag{2.19}$$

where π_i ,N_i ,n_i $(i = 1,\dots,J)$ are the prior probabilities, number of cases in LS, and number of cases contained in node t for class i.

From Equation 2.1 and Equation 2.3, the conditional probability that a case is class j given that it falls into node t can be given by Equation 2.20,

$$p(j \mid t) = \left(\pi(j) N_j(t) / N_j \right) / p(t) \tag{2.20}$$

so the probabilities $p_L(t)$, $p_R(t)$ that the cases in node t going to left descendent node t_L and right descendent node t_R are defined as Equation 2.21 and Equation 2.22 respectively.

$$p_L(t) = p_t^{left} / p_t^{parent} \tag{2.21}$$

$$p_R(t) = p_t^{right} / p_t^{parent} \tag{2.22}$$

Therefore, by adjusting prior probabilies π_i $(i = 1,\dots,J)$, one can find the overlapping region between class for each split, as shown in the yellow region in Fig. 12. Accordingly, the regions in red and blue are identified for different classes, with probabilities of exceptions lower than π_b and π_r, respectively.

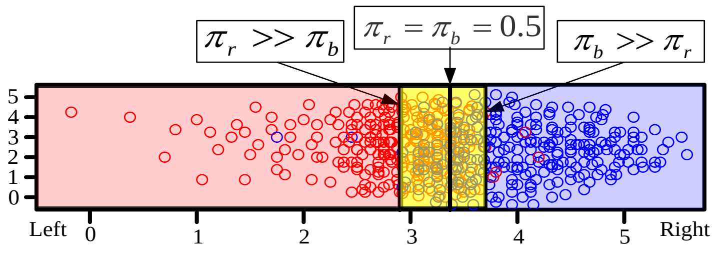


Figure 12. Thresholds of DTs with respect to prior probability adjustment.

3 Case Study

The principles of data mining and its associated applications have been introduced in Chapter 1. Furthermore, the details of DT algorithms have been presented and analyzed in Chapter 2. In this Chapter, DT algorithms are used to analyze the annual power market data of Danish power system in 2011 using the code programmed in MATLAB 2013a. The essential part of the DT code is attached in the Appendix. The DT model created by m.file MATLAB 2013a is able to not only discover the relationship between the electricity price in Denmark and other real-time data (e.g., as inter-state power exchange, wind speed, power generation etc.), but also predict the electricity price of Denmark using these real-time data. The electricity price prediction models can provide a simple but practical approach for industrial and residential consumers to reduce the cost of power consumption if the mechanism of adjustable electricity charges will be adopted in Nordic countries in the future. The performance of this approach demonstrates that this application has great practical significance, not only in Denmark but also in all other countries.

3.1 Analysis Tools

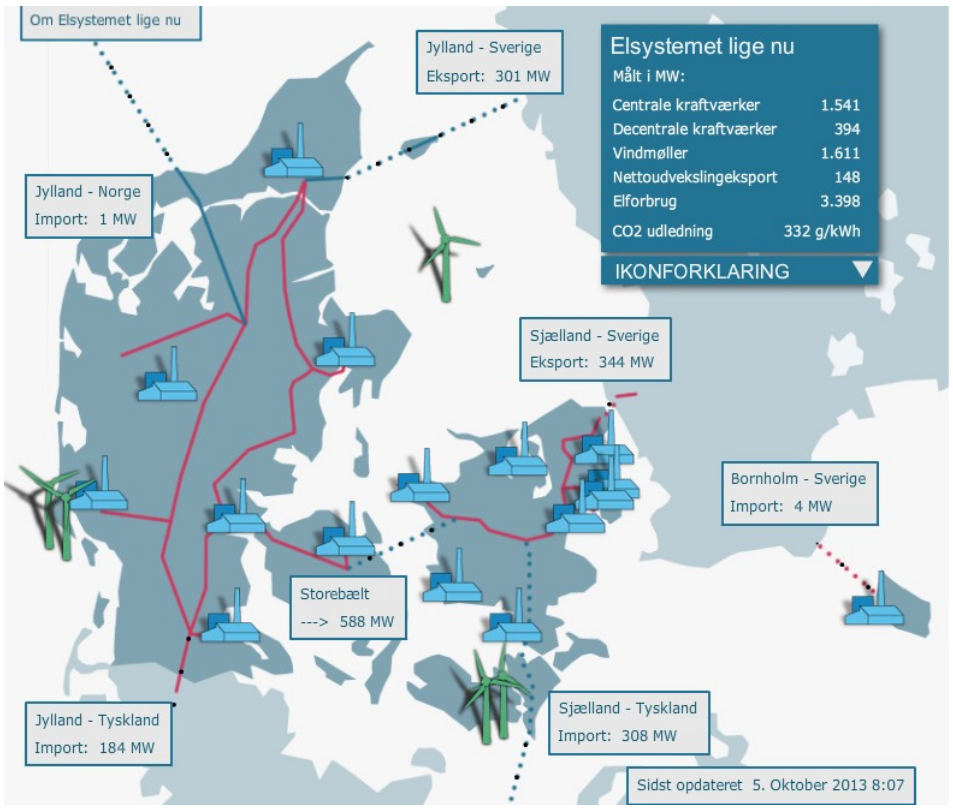
In this work, three tools were used to develop the proposed approach: *Excel* was necessary for collecting and processing the data downloaded from the website of Danish national power grid – Energinet.dk; *MATLAB* was used to program the DT algorithm and generate the result; *Microsoft Visio* was adopted to visualize the DT models.

3.2 Introduction of the Danish Power System

Figure 13 shows the real time power flow pattern in Denmark, which can be found in the official website [energynet.dk]. The Danish power grid constists of two non-synchronous systems, i.e., Western Danish power system (DK1) and Eastern Danish power system (DK2). The Western Danish power system in Jylland Peninsula and Fyn Island is synchronous with the European Network of Transmission System Operators for Electricity (ENTSO-E) RG Continental Europe via Germany, while the Eastern Danish power system on the Zealand Island is synchronous to Nordic system via Sweden. Besides, another island situated in Baltic Sea, Bornholm Island, is connected to the Swedish power system.

Figure 14 shows the real time electricity prices of Nordic countries, which can be found in official website [energynet.dk]. The electricity price of DK1 and DK2 are sometimes different. In this study, our objective was to predict the electricity price of DK1 only. The wind power generation level is much higher in DK1 than that in DK2, so the wind power generation has a much higher impact on electricity price of DK1.

As shown in Figure 14, the interconnection of DK1 to the external grid is strong. To the north, DK1 is connected to Norway and Sweden via HVDC links, with capacities of around 1000MW and 750MW respectively. To the south, DK1 is connected to Germany two 400kV and two 220kV HVAC transmission lines. To the east, the HVDC link “Great Belt” (with a capacity of 600MW) interconnects western DK1 and DK2. The abundance of hydro power generation in Norwegian and Swedish power systems can compensate the fluctuating wind power generation in Denmark and Germany.


Om Elsystemet lige nu
Jylland - Sverige Eksport: 301 MW
Jylland - Norge Import: 1 MW
Sjælland - Sverige Eksport: 344 MW
Bornholm - Sverige Import: 4 MW
Storebælt ---> 588 MW
Jylland - Tyskland Import: 184 MW
Sjælland - Tyskland Import: 308 MW
Sidst opdateret 5. Oktober 2013 8:07

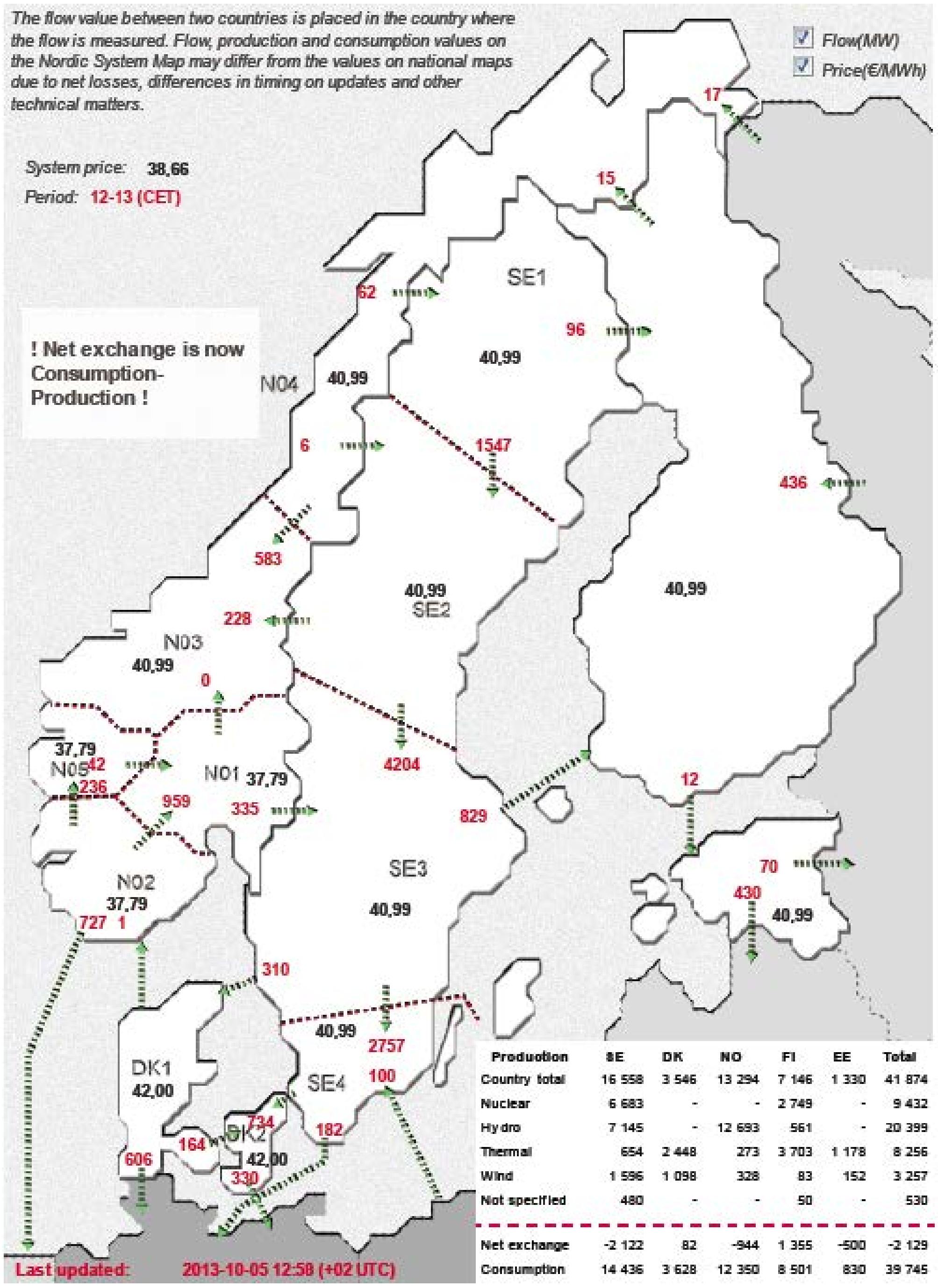
Elsystemet lige nu

Målt i MW:

Centrale kraftværker	1.541
Decentrale kraftværker	394
Vindmøller	1.611
Nettoudvekslingeksport	148
Elforbrug	3.398
CO2 udledning	332 g/kWh

IKONFORKLARING

Figure 13. Real time power flow pattern in Denmark.



The flow value between two countries is placed in the country where the flow is measured. Flow, production and consumption values on the Nordic System Map may differ from the values on national maps due to net losses, differences in timing on updates and other technical matters.

! Net exchange is now Consumption-Production !

Production	SE	DK	NO	FI	EE	Total
Country total	16 558	3 546	13 294	7 146	1 330	41 874
Nuclear	6 683	-	-	2 749	-	9 432
Hydro	7 145	-	12 693	561	-	20 399
Thermal	654	2 448	273	3 703	1 178	8 256
Wind	1 596	1 098	328	83	152	3 257
Not specified	480	-	-	50	-	530
Net exchange	-2 122	82	-944	1 355	-500	-2 129
Consumption	14 436	3 628	12 350	8 501	830	39 745

Figure 14. Real time power flow pattern in Denmark.

3.3 Annual Power Market Data of Danish Power System

The annual power market data of Danish power system are broadcasted periodically in the website of Danish national transmission system——Energinet.dk, which can also be downloaded from http://energinet.dk/EN/El/Engrosmarked/Udtraek-af-markedsdata/Sider/default.aspx.

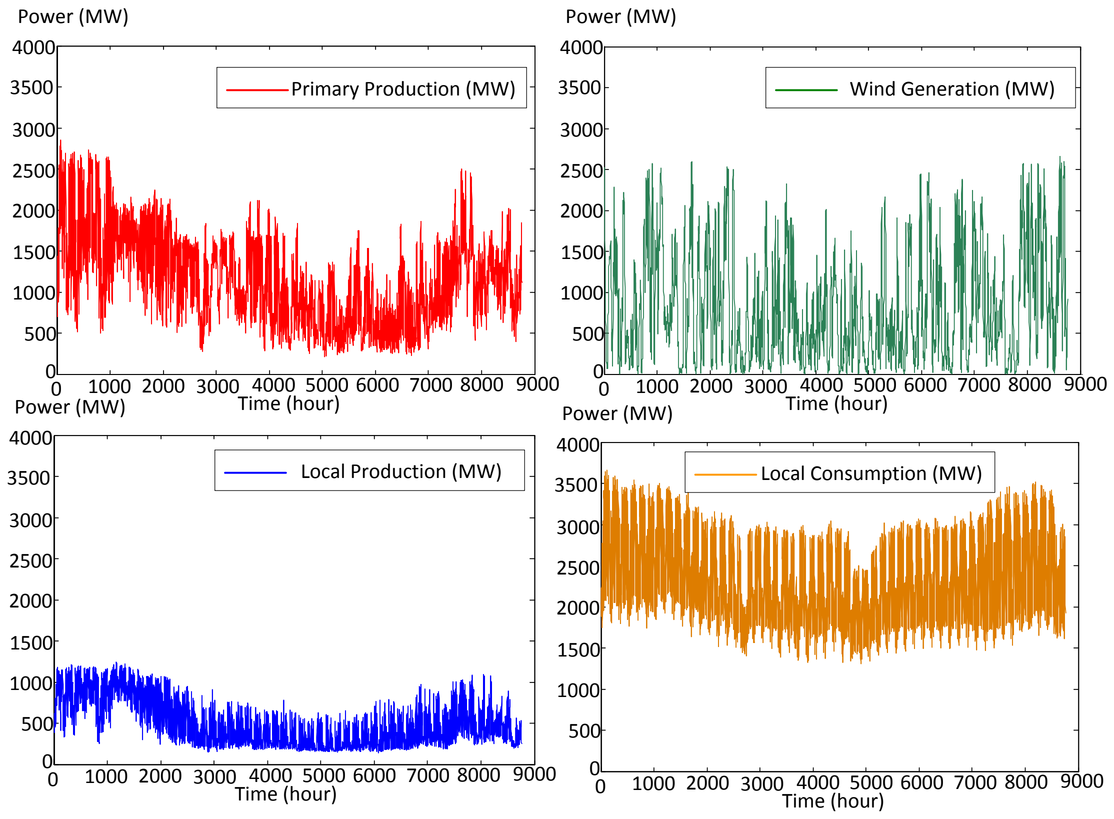


Figure 15. Hourly data of primary production, wind production, local production and load consumption in 2011.

Figure 15 shows the hourly data of primary generation, wind power generation, local generation (i.e., part of dispersed generation) and load consumption in 2011 (8760 hours per year) respectively, whilst Figure 16 shows the hourly data of physical power exchange with Norway, Sweden and Germany in 2011 respectively.

Power balance should be kept to maintain the frequency of electricity as the nominal value of 50Hz, which can be expressed in Equation 3.1,

$$G_{pri,k} + G_{loc,k} + G_{wind,k} + P_{exNO,k} + P_{exSE,k} + P_{exDE,k} + P_{exDK2,k} - P_{load,k} = 0 \qquad \textbf{(3.1)}$$

where $G_{pri,k}$, $G_{loc,k}$ and $G_{wind,k}$ represent the primary production, local production and wind power production at hour k in DK1 respectively; $P_{exNO,k}$, $P_{exSE,k}$, $P_{exDE,k}$ and $P_{exDK2,k}$ stand for the power exchange with Norway, Sweden, Germany and Eastern Denmark (DK2) at hour k, respectively ("+" for import, "-" for export); $P_{load,k}$ represents the gross load consumption at hour k in DK1, respectively.

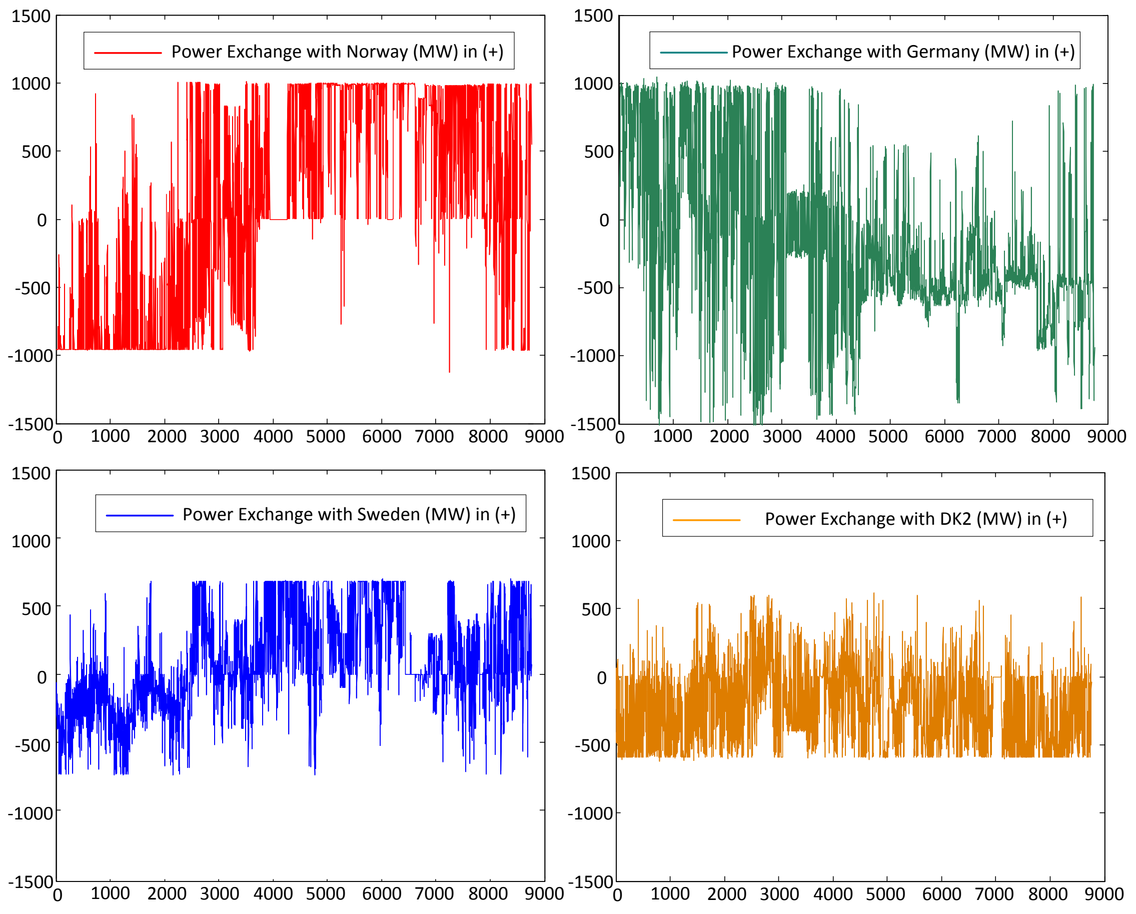


Figure 16. Hourly data of power exchange with Norway, Sweden, Germany and Eastern Denmark in 2011.

In this case study, the objective was to create a DT model to find the relationship between power generation, power flow and the price of DK1. The future class of electricity price can be predicted based on the model using the values of predictors, which can be obtained on the website of Energinet.dk.

Figure 17(a) shows the distribution of electricity prices in DK1 in the whole year of 2011, which can be classified into four classes. As shown in Figure 17(b), the red area stands for the low price area represent by "*Q1*" (below 296.88DKK/MW); the blue area stands for medium low price area represented by "*Q2*" (between 296.88DKK/MW and 361.68DKK/MW); the green area stands for medium high price area represented by "*Q3*" (between 361.68DKK/MW and 423.50DKK/MW); while the yellow area stands for high price area represented by "*Q4*" (between above 423.50DKK/MW) respectively.

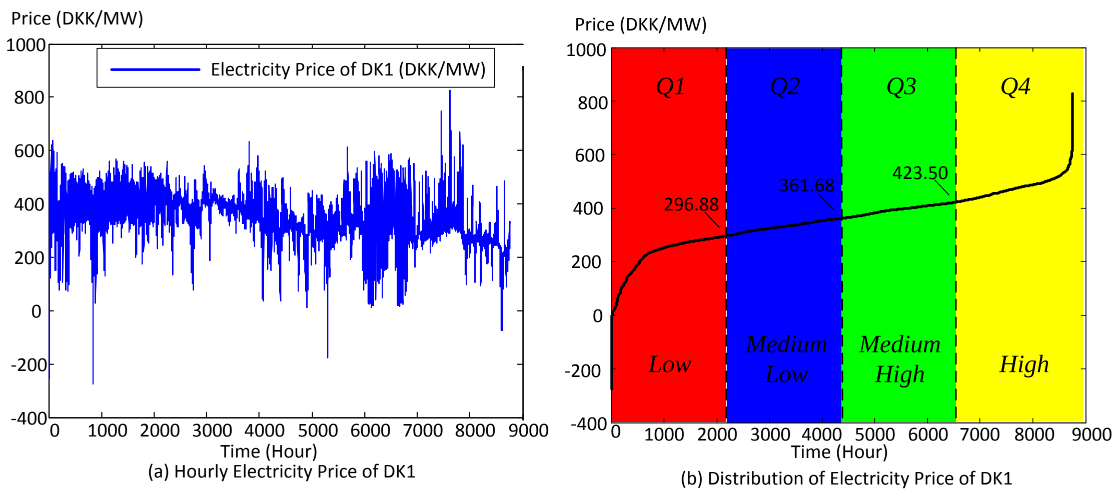


Figure 17. Hourly data and distribution of electricity price of DK1.

3.4 Creation of Multi-class Decision Tree

In order to better configure the parameters of decision tree creation, the code is programmed with an m. file in MATLAB. Part of the core program for DT creation can be found in the Appendix.

Table 1 shows the information of predictors used to train a multi-class decision tree. *DATE* and *TIME* are categorical predictors and the rest are numerical predictors. Eighty percent of the 8760 hourly data are randomly selected as the learning set, while the rest 20% cases serve as the test set. The multi-class DT was created by the MATLAB program as shown in Figure 18. The details of each node in Figure 18 are shown in Table 2.

Table 1. Predictors of Multi-class Decision Tree.

No.	Name	Type	Description
C1	*DATE*	Categorical	Date in 2011 (Jan-01~Dec-30).
C2	*TIME*		Hour in each day.
N1	*DK1_GroLod*	Numerical	Gross load consumption.
N2	*DK1_NetLod*		Net load consumption.
N3	*DK1_PriPro*		Primary generation.
N4	*DK1_LocPro*		Dispersed generation
N5	*DK1_WindPro*		Wind power generation.
N6	*Elspot_DE_EPEX*		SPOT market price of power exchange with Germany.
N7	*Elspot_NO*		SPOT market price of power exchange with Norway.
N8	*Elspot_SE*		SPOT market price of power exchange with Sweden.
N9	*Elspot_SysPri*		SPOT market price of NordPool.
N10	*Elspot_DiffDK1-EPEX*		SPOT market price difference between western Denmark and Germany
N11	*Sch_DE->DK1*		Scheduled power exchange with Germany.
N12	*Sch_NO->DK1*		Scheduled power exchange with Norway.
N13	*Sch_SE->DK1*		Scheduled power exchange with Sweden.
N14	*Sch_Nordic->DK1*		Scheduled power exchange with Nordic countries (Norway + Sweden).
N15	*Sch_DK2->DK1*		Scheduled power exchange with Eastern Denmark.
N16	*Phy_DE->DK1*		Physical power exchange with Germany.
N17	*Phy_NO->DK1*		Physical power exchange with Norway.
N18	*Phy_SE->DK1*		Physical power exchange with Sweden.
N19	*Phy_DK2->DK1*		Physical power exchange with Eastern Denmark.

Table 2. Details of the created DT.

Node No.	Conditions	Predicted Class
1	*N6<=424.06; N9<294.95; N12<848.30; N6<375.75*	Q1
2	*N6<=424.06; N9<294.95; N12<848.30; N6>=375.75; N10<79.25*	Q1
3	*N6<=424.06; N9<294.95; N12<848.30; N6>=375.75; N10>=79.25*	Q3
4	*N6<=424.06; N9<294.95; N12>=848.30; N10<-15.13; N14<1628.75*	Q1
5	*N6<=424.06; N9<294.95; N12>=848.30; N10<-15.13; N14>=1628.75*	Q2
6	*N6<=424.06; N9<294.95; N12>=848.30; N10>=-15.13; N6<296.55*	Q1
7	*N6<424.06; N9<294.95; N12>=848.30; N10>=-15.13; 296.55=<N6<362.55*	Q2
8	*N6<=424.06; N9<294.95; N12>=848.30; N10>=-15.13; N6>=362.55*	Q3
9	*N6<=424.06; N9>=294.95; N6<296.93; N10<24.26*	Q1
10	*N6<=424.06; N9>=294.95; N6<296.93; N10>=24.26; N3<981.95*	Q2
11	*N6<=424.06; N9>=294.95; N6<296.93; N10>=24.26; N3>=981.95*	Q3
12	*N6<=424.06; N9>=294.95; 296.93<=N6<361.63; N10<17.17*	Q2
13	*N6<=424.06; N9>=294.95; 296.93<=N6<361.63; 17.17<=N10<112.83*	Q3
14	*N6<=424.06; N9>=294.95; 296.93<=N6<361.63; N10>=112.83*	Q4
15	*N6<=424.06; N9>=294.95; N6>=361.63; N10<-17.56; N8<361.67*	Q2
16	*N6<=424.06; N9>=294.95; N6>=361.63; N10<-17.56; N8>=361.67*	Q3
17	*N6<=424.06; N9>=294.95; N6>=361.63; -17.56<=N10<53.33*	Q3
18	*N6<=424.06; N9>=294.95; N8<312.70; N10>53.33*	Q4
19	*N6>=424.06; N10<-29.03; N8<296.92*	Q1
20	*N6>=424.06; N10<-29.03; N8>=296.92; N10<-68.89*	Q2
21	*N6>=424.06; N10<-29.03; N8>=296.92; N10>=-68.89; N6<488.02*	Q3
22	*N6>=424.06; N10<-29.03; N8>=296.92; N10>=-68.89; N6>=488.02*	Q4
23	*N6>=424.06; N10<-29.03; 361.70<=N8<423.32*	Q3
24	*N6>=424.06; N10<-29.03; N8>=423.32*	Q4
25	*N6>=424.06; N10>=-29.03*	Q4

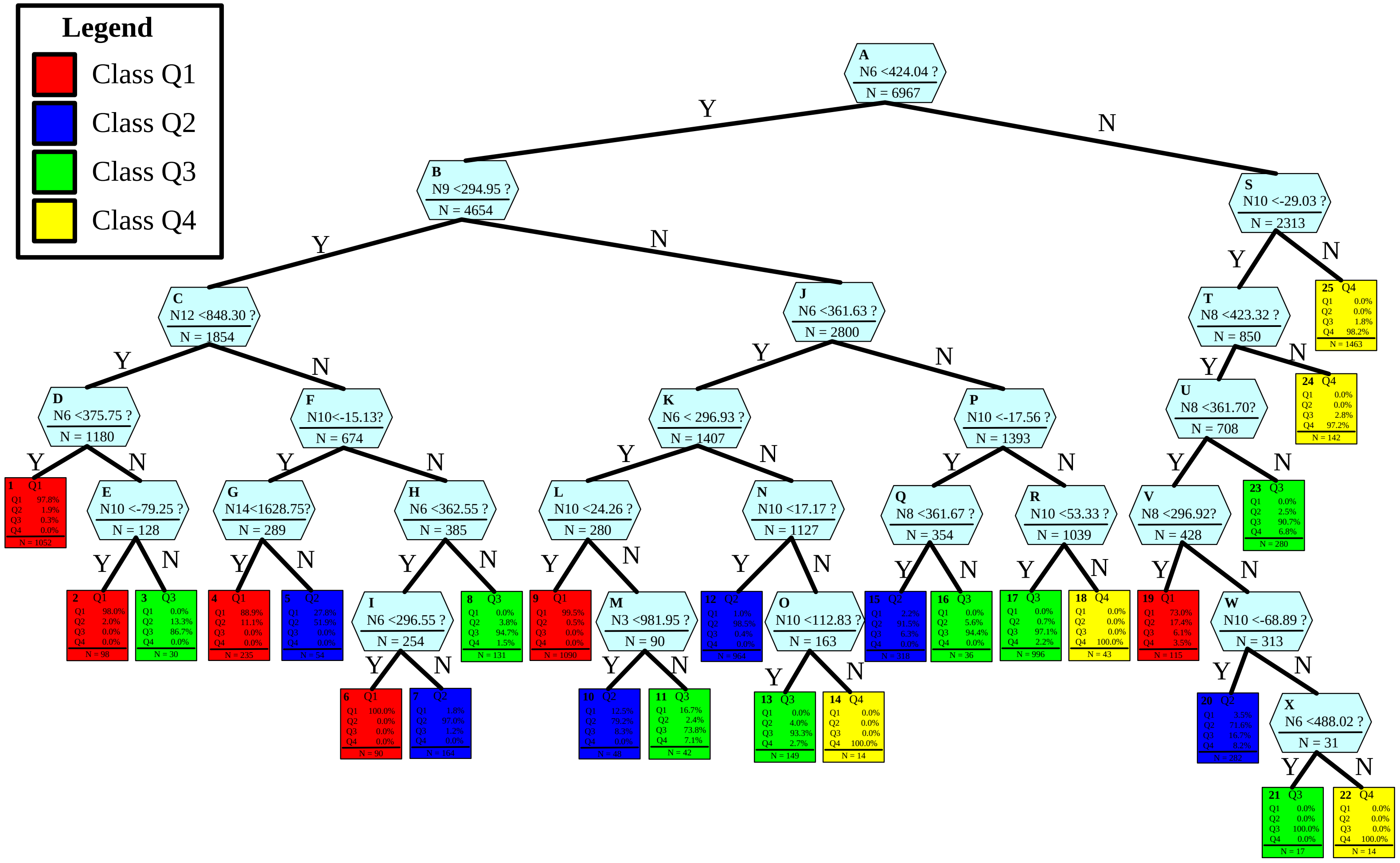


Figure 18. Multi-class decision tree.

The performance of learning set and test set in the multi-class Decision Trees are shown in Table 3 and Table 4 respectively. The overall correct rate of learning set data in multi-class DT is 95.76%, and the overall correct rate of test set data in multi-class DT is 92.69%. These high accuracy values can fulfill the requirement of discovering the information underneath the annual power market data.

Table 3. Performance of Learning Set in Multi-class Decision Tree.

Actual Class	Total Class	Percent Correct (%)	Q1 N=1780	Q2 N=1830	Q3 N=1681	Q4 N=1676
Q1	1755	96.70	1697	51	7	0
Q2	1769	94.29	69	1668	32	0
Q3	1721	92.50	10	88	1592	31
Q4	1722	95.53	4	23	50	1645
Total	6967	Average Correct Rate 94.75% Overall Correct Rate 95.76%				

Table 4. Performance of Test Set in Multi-class Decision Tree.

Actual Class	Total Class	Percent Correct (%)	Q1 N=1780	Q2 N=1830	Q3 N=1681	Q4 N=1676
Q1	435	94.71	412	21	2	0
Q2	421	94.21	18	384	19	0
Q3	469	89.34	8	26	419	16
Q4	468	95.51	0	5	16	447
Total	1793	Average Correct Rate 92.69% Overall Correct Rate 92.69%				

Table 5. Performance of Validation Set in Multi-class Decision Tree.

Actual Class	Total Class	Percent Correct (%)	Q1 N=1780	Q2 N=1830	Q3 N=1681	Q4 N=1676
Q1	2180	88.72	1934	218	28	0
Q2	2200	89.05	201	1959	40	0
Q3	2175	90.94	8	184	1978	5
Q4	2205	87.07	0	67	218	1920
Total	8760	Average Correct Rate 88.94% Overall Correct Rate 88.94%				

In order to validate the DT model, the annual data of the next year (2012) are dropped into the created DT to test the performance of the DT model. Table 5 shows the performance of validation set in multi-class DT. The overall accuracy is 88.94%.

4 Conclusion

This thesis first introduced the data mining techniques and its associated algorithms. Then a comprehensive overview of data mining and decision tree applications was provided. Furthermore, a new methodology using DT algorithms to predict the electricity prices was proposed in the case study. The annual data of electricity prices were divided into 4 classes. The hidden relationship between power production, power exchange and electricity price can be discovered by the DT model. Both the internal test and external validation show that the proposed method can predict the electricity prices with high accuracy. This DT model can provide a reference for industrial and residential power consumers to reduce the cost of electricity if adjustable electricity charges are adopted in the future.

REFERENCES

Breiman, L., Friedman, J. H., Olshen, R. A. & Stone , C. J., 1984. In: *Classification and Regression Trees.* s.l.:Chapman&Hall/CRC, pp. 8-171.

energinet.dk, 2011. *Hourly Danish power system data in 2011.* [Online]
Available at: http://www.energinet.dk/EN/El/Engrosmarked/Udtraek-afmarkedsdata/Sider/default.aspx
[Accessed 10 2013].

energinet.dk, n.d. *Real-time electricity prices in Nordic countries.* [Online]
Available at: http://driftsdata.statnett.no/snps/
[Accessed 10 2013].

energinet.dk, n.d. *Real-time power patterns of Danish power system..* [Online]
Available at: http://www.energinet.dk/Flash/Forside/index.html
[Accessed 10 2013].

Palace, B., 1996. *Data Mining: What is data mining.* [Online]
Available
at: http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/datamining.htm
[Accessed 8 2013].

Rokach, L. & Maimon, O., 2008. DATA MINING WITH DECISION TREES: Theory and Applications. *Series in Machine Perception Artificial Intelligence,* Volume 69, pp. 5-68,101-109.

Witten, I. H., Frank, E. & Hall, M. A., 2011. In: *Data Mining - Practical Machine Learning Tools and Techniques (3rd Ed).* Burlington,MA: Morgan Kaufmann Publishers, pp. 3-9,192-273.

APPENDIX

```
clc
clear all
load DatabaseV1.mat % Load database, M cases | N-1 predictors 1 target
t1 = clock;
[M,N] = size(OptDT); % M = Row N = Col

%%%%%%%%%%%%%%%%% Marco, defined by the user%%%%%%%%%%%%%%%%%%%%%%%%%
DEPTH = 8; % The maximum depth of DT
WIDTH = 1000;  % The maximum width of DT Width >= 2^Depth
MINCASE = 10;  % The minimum cases in the terminal node
MAXPUR = 0.99; % The maximum purity of the terminal node
PERCLS = 0.8;  % The percentage of Learning Set
MINDECIMP = 0.005; % The minimum decrease of impurity in the terminal node

%%%%%%%%%%%%%%% Variable Definition%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
NumLS = round(M*PERCLS) % Number of cases in the Learning Set, nearest integer
NumTS = M - NumLS  % Number of cases in the Test Set
randindex = randperm(M); % Randomly permutaion of integers from 1 to M
OptDTTS = OptDT(randindex(1:NumTS),:); % The first NumTS rows are used as Test
Set
OptDTLS = OptDT(randindex(NumTS+1:M),:); % The latter M-NumTS rows are used as
Learning Set
[M,N] = size(OptDTLS); % Size of Maxtrix for LS, M = Row N = Col
DataBase = zeros(M,N,WIDTH); % M for rows of the subset, N for columns of
subset, WIDTH for terminal nodes
DataBaseSon = zeros(M,N,WIDTH);
DataBase(:,:,1) = OptDTLS; % Store the LS in the Database 1st node
Count=ones(DEPTH,WIDTH); % For number of cases in subset
Purity=zeros(DEPTH,WIDTH); % For purity of node
CriAtr=zeros(DEPTH,WIDTH); % Store the critical attribute of Critical
Splitting Rule
Thresh=zeros(DEPTH,WIDTH); % Store the thresholds of Critical Splitting Rule
Count(1,1) = M; % The root node has M cases
d = 1;
w = 1;
MaxImpDec = 0;
NumClassI = 0;
NumClassS = 0;

%%%%%%%%%% Count the Insecure/Secure cases in the LS%%%%%%%%%%%%%%%%%
for m = 1:1:M
    if(OptDTLS(m,N)==1)
        NumClassI = NumClassI + 1;
    end
    if(OptDTLS(m,N)==0)
        NumClassS = NumClassS + 1;
    end
end
Purity(1,1)=max(NumClassI/M,NumClassS/M);% The original purity in the LS

%%%%%%%%%%%%%% Main Splitting Function%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[MaxImpDec,LeftNode,RightNode,mL,mR,mLS,mLI,mRS,mRI,cols,SplitRule] =
split(OptDTLS,Count(1,1));
%Input: The parent node, the number of cases in the parent node
%Output:
%MaxImpDec: Maximum impurity decrease
%LeftNode: The matrix of the left node
%RightNode: The matrix of the right node
%mL: The number of cases in the left node
%mR: The number of cases in the right node
%mLS,mLI,mRS,mRI: The number of cases left stable
%cols,SplitRule: The CA, and CSR
```

```
cols;
SplitRule;
CriAtr(1,1) = cols; % Store the first CA
Thresh(1,1) = (max(LeftNode(:,cols)) +    min(RightNode(:,cols)))/2; %
Calculate and store the first CSR
[ML,NL] = size(LeftNode);% Rows and Colums
[MR,NR] = size(RightNode);
DataBase(1:ML,1:NL,2*w-1) = LeftNode; % Store the Left Node into 1st
DataBase(1:MR,1:NR,2*w) = RightNode; % Store the Right Node into 2nd
Count(d+1,2*w-1)=ML;
Count(d+1,2*w)=MR;
Purity(d+1,2*w-1)=max(mLS/mL,mLI/mL);
Purity(d+1,2*w)=max(mRS/mR,mRI/mR);

 for d=2:1:DEPTH % From the 2nd level of DT
     for w = 1:2:2^(d-1)
        if(Count(d,w)<MINCASE||Purity(d,w)>MAXPUR)% If the number is less than
MINCASE or pure enough
            Count(d+1,2*w-1)=0;
            Count(d+1,2*w)=0;
        end
        if(Count(d,w+1)<MINCASE||Purity(d,w+1)>MAXPUR)
            Count(d+1,2*(w+1)-1)=0;
            Count(d+1,2*(w+1))=0;
        end
        if(Count(d,w)>=MINCASE&&Purity(d,w)<=MAXPUR) % Else, Split the odd
node

[MaxImpDec1,LeftNode1,RightNode1,mL1,mR1,mLS1,mLI1,mRS1,mRI1,cols1,SplitRule1]
= split(DataBase(1:Count(d,w),:,w),Count(d,w));
            if(MaxImpDec1>=MINDECIMP)
                cols1;
                SplitRule1;
                CriAtr(d,w) = cols1;
                Thresh(d,w) = (max(LeftNode1(:,cols1)) +
min(RightNode1(:,cols1)))/2;
                [ML1,NL1] = size(LeftNode1);
                [MR1,NR1] = size(RightNode1);
                Count(d+1,2*w-1)=ML1;
                Count(d+1,2*w)=MR1;
                Purity(d+1,2*w-1)=max(mLS1/mL1,mLI1/mL1);
                Purity(d+1,2*w)=max(mRS1/mR1,mRI1/mR1);
            end
        end
        if(Count(d,w+1)>=MINCASE&&Purity(d,w+1)<=MAXPUR) % Else, Split the
even node

[MaxImpDec2,LeftNode2,RightNode2,mL2,mR2,mLS2,mLI2,mRS2,mRI2,cols2,SplitRule2]
= split(DataBase(1:Count(d,w+1),:,w+1),Count(d,w+1));
            if(MaxImpDec2>=MINDECIMP)
                cols2;
                SplitRule2;
                CriAtr(d,w+1) = cols2;
                Thresh(d,w+1) = (max(LeftNode2(:,cols2)) +
min(RightNode2(:,cols2)))/2;
                [ML2,NL2] = size(LeftNode2);
                [MR2,NR2] = size(RightNode2);
                Count(d+1,2*(w+1)-1)=ML2;
                Count(d+1,2*(w+1))=MR2;
                Purity(d+1,2*(w+1)-1)=max(mLS2/mL2,mLI2/mL2);
                Purity(d+1,2*(w+1))=max(mRS2/mR2,mRI2/mR2);
            end
        end
            if(MaxImpDec1>=MINDECIMP)% If the split result has enough impurity
improvement
```

```
                DataBaseSon(1:ML1,1:NL1,2*w-1) = LeftNode1; % Store the 
LeftNode
                DataBaseSon(1:MR1,1:NR1,2*w) = RightNode1;
            end
            if(MaxImpDec2>=MINDECIMP)
                DataBaseSon(1:ML2,1:NL2,2*(w+1)-1) = LeftNode2;
                DataBaseSon(1:MR2,1:NR2,2*(w+1)) = RightNode2;
            end
     end
     DataBase = DataBaseSon;% For every level, store the splitted database
     d = d % Print the Progress
     w = w
 end
 
 %%%%%%%%%%%%%%%%%%%%Print the decision tree%%%%%%%%%%%%%%%%%%%%%%%%%
 i = 0;
 for d=1:1:DEPTH
     for w = 1:1:2^(d-1)
         if(Thresh(d,w)~=0)
             i = i + 1;
             if((Count(d+1,2*w-1)<MINCASE||Purity(d+1,2*w-
1)>MAXPUR)&&(Count(d+1,2*w)<MINCASE||Purity(d+1,2*w)>MAXPUR))
                 fprintf('%d Node%d,%d CaseNo.%d if X%d <= %f then Terminal 
Node%d,%d CaseNo.%d else Terminal Node%d,%d 
CaseNo.%d.\n',i,d,w,Count(d,w),CriAtr(d,w),Thresh(d,w),d+1,2*w-
1,Count(d+1,2*w-1),d+1,2*w,Count(d+1,2*w));
             end
             if(~(Count(d+1,2*w-1)<MINCASE||Purity(d+1,2*w-
1)>MAXPUR)&&(Count(d+1,2*w)<MINCASE||Purity(d+1,2*w)>MAXPUR))
                 fprintf('%d Node%d,%d CaseNo.%d if X%d <= %f then Parent 
Node%d,%d CaseNo.%d else Terminal Node%d,%d 
CaseNo.%d.\n',i,d,w,Count(d,w),CriAtr(d,w),Thresh(d,w),d+1,2*w-
1,Count(d+1,2*w-1),d+1,2*w,Count(d+1,2*w));
             end
             if((Count(d+1,2*w-1)<MINCASE||Purity(d+1,2*w-
1)>MAXPUR)&&~(Count(d+1,2*w)<MINCASE||Purity(d+1,2*w)>MAXPUR))
                 fprintf('%d Node%d,%d CaseNo.%d if X%d <= %f then Terminal 
Node%d,%d CaseNo.%d else Parent Node%d,%d 
CaseNo.%d.\n',i,d,w,Count(d,w),CriAtr(d,w),Thresh(d,w),d+1,2*w-
1,Count(d+1,2*w-1),d+1,2*w,Count(d+1,2*w));
             end
             if(~(Count(d+1,2*w-1)<MINCASE||Purity(d+1,2*w-
1)>MAXPUR)&&~(Count(d+1,2*w)<MINCASE||Purity(d+1,2*w)>MAXPUR))
                 fprintf('%d Node%d,%d CaseNo.%d if X%d <= %f then Parent 
Node%d,%d CaseNo.%d else Parent Node%d,%d 
CaseNo.%d.\n',i,d,w,Count(d,w),CriAtr(d,w),Thresh(d,w),d+1,2*w-
1,Count(d+1,2*w-1),d+1,2*w,Count(d+1,2*w));
             end
         end
     end
 end
t2 = clock;
fprintf('Elapsed Time = %f\n',etime(t2,t1));
```