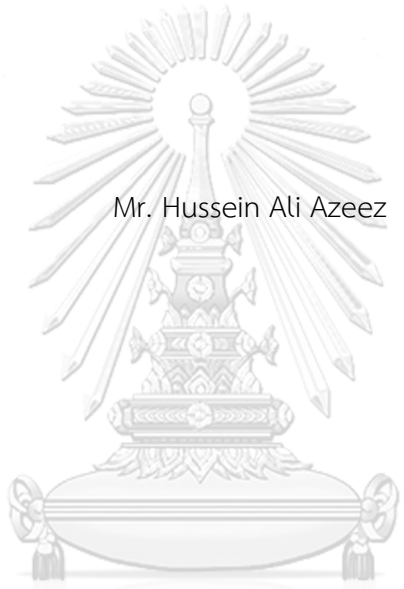
# RULE-BASED RECOMMENDATION MODEL FOR SELECTING CLASSIFIERS IN WEKA BASED ON USER SPECIFICATIONS

Mr. Hussein Ali Azeez





A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Computer Science and Information Technology

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2017

ตัวแบบการแนะนำด้วยกฎสำหรับการเลือกตัวจำแนกประเภทในโปรแกรมวีก้าบนข้อกำหนดของผู้ใช้

นายฮุซเซ็น อาลี อาซีซ



วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2560

| | |
|---|---|
| Thesis Title | RULE-BASED RECOMMENDATION MODEL FOR SELECTING CLASSIFIERS IN WEKA BASED ON USER SPECIFICATIONS |
| By | Mr. Hussein Ali Azeez |
| Field of Study | Computer Science and Information Technology |
| Thesis Advisor | Professor Chidchanok Lursinsap, Ph.D. |

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

.....Dean of the Faculty of Science

(Professor Polkit Sangvanich, Ph.D.)

THESIS COMMITTEE

.....Chairman

(Assistant Professor Suphakant Phimoltares, Ph.D.)

.....Thesis Advisor

(Professor Chidchanok Lursinsap, Ph.D.)

.....External Examiner

(Assistant Professor Saichon Jaiyen, Ph.D.)

ฮุซเซ็น อาลี อาซีซ : ตัวแบบการแนะนำด้วยกฎสำหรับการเลือกตัวจำแนกประเภทในโปรแกรมวีก้าบนข้อกำหนดของผู้ใช้ (RULE-BASED RECOMMENDATION MODEL FOR SELECTING CLASSIFIERS IN WEKA BASED ON USER SPECIFICATIONS) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ศ. ดร. ชิดชนก เหลือสินทรัพย์, หน้า.

ในวิทยาการการเรียนรู้ของเครื่องหรือแมทชีน เลินนิ่ง มีชุดคำสั่งที่หลากหลายแตกต่างกันไป ดังนั้นจึงเป็นเรื่องยากสำหรับผู้ใช้ที่ไม่มีความเชี่ยวชาญในการเลือกชุดคำสั่งและมุ่งเน้นไปที่การเพิ่มประสิทธิภาพเชิงประจักษ์ วิทยานิพนธ์เล่มนี้ได้พิจารณาปัญหาในการเลือกชุดคำสั่งบนข้อกำหนดของผู้ใช้ ตัวอย่างเช่น ความเร็วในการฝึกฝน หน่วยความจำ และการตีความชุดคำสั่ง โดยเฉพาะอย่างยิ่งจะพิจารณาการจำแนกข้อมูลของตัวแบบ 20 ตัวแบบ ประกอบด้วยวิธีการเมตา 2 วิธี และเทคนิคการจำแนกข้อมูล 18 แบบ ซึ่งมาจากกลุ่มเทคนิคการจำแนกข้อมูลที่แตกต่างกัน 10 กลุ่ม ได้แก่ การจำแนกข้อมูลด้วยเบย์เซียน การสร้างต้นไม้ตัดสินใจ การจำแนกข้อมูลด้วยกฎ การค้นหาเพื่อนบ้านใกล้สุด การถดถอยโลจิสติกส์ การถดถอยพหุกลุ่ม การใช้โครงข่ายประสาทเทียม ซัพพอร์ตเวกเตอร์แมชชีน เทคนิคบูสติ้ง เทคนิคแบ็กกิ้ง และกลุ่มเทคนิคอื่นๆ ซึ่งทั้งหมดนี้จะปฏิบัติการบนโปรแกรมวีก้า ลักษณะเฉพาะที่แตกต่างกันจะถูกรวบรวมสำหรับแต่ละตัวแบบ เช่น ความเร็วในการฝึกฝน และหน่วยความจำที่มี จากนั้นชุดของกฎต่างๆ จะถูกกำหนดตามลักษณะเฉพาะดังกล่าวโดยใช้สถาปัตยกรรมข้อมูลแบบต้นไม้เพื่อที่จะได้เลือกตัวแบบหรือเพื่อตอบสนองความต้องการของผู้ใช้ท้ายที่สุด ตัวแบบนั้นจะถูกประเมินผลบนชุดข้อมูลจำนวน 10 ชุดจากคลังเก็บการเรียนรู้ของเครื่องยูซีไอ ซึ่งผลการจำแนกประเภทพบว่าดีกว่าหรือใกล้เคียงกับการทำงานก่อนหน้าที่มีปัญหาเดียวกัน



| | | |
|---|---|---|
| ภาควิชา | คณิตศาสตร์และวิทยาการคอมพิวเตอร์ | ลายมือชื่อนิสิต ..... |
| | | ลายมือชื่อ อ.ที่ปรึกษาหลัก ..... |
| สาขาวิชา | วิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ | |
| ปีการศึกษา | 2560 | |

# # 5972621823 : MAJOR COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
KEYWORDS: WEKA, MODEL SELECTION, CLASSIFICATION, RULE-BASED MODELS

HUSSEIN ALI AZEEZ: RULE-BASED RECOMMENDATION MODEL FOR SELECTING CLASSIFIERS IN WEKA BASED ON USER SPECIFICATIONS. ADVISOR: PROF. CHIDCHANOK LURSINSAP, Ph.D., pp.

Machine learning field has many different algorithms; selecting an algorithm for non-expert user and aiming for maximizing empirical performance could be a tough task. This thesis considers the problem of selecting an algorithm based on the user specifications, for instance, training speed, memory, and interpretation. Specifically, considers the classification problem in the range of 20 models (2 meta-methods, 18 based classifiers) arising from 10 different families (Bayesian, Decision trees, Rule-based methods, Nearest neighbor methods, Logistic, multinomial regression, Neural networks, Support vector machines, Boosting, Bagging, and other ensembles), all implemented in WEKA. Different characteristics have been gathered for each model such as training speed and the available memory, then a set of rules have been defined based on these characteristics by using a tree architecture in order to choose one or for given user requirements. Finally, the model evaluated on 10 datasets from the UCI repository, the classification results show a better than or close to a previous work that addressed the similar problem.



| | | |
|---|---|---|
| Department: | Mathematics and Computer Science | Student's Signature ..... |
| | | Advisor's Signature ..... |
| Field of Study: | Computer Science and Information Technology | |
| Academic Year: | 2017 | |

# ACKNOWLEDGEMENTS

I would like to thank all of those who made it possible for me to complete this thesis.

First, I am very much obliged and grateful to my research advisor, Professor Dr Chidchanok Lursinsap for suggesting and helping me understand the process of research and checking and correcting the thesis.

Also, I would like to thank the committee chair, Assistant Professor Dr Suphakant Phimoltares and external examiner, Assistant Professor Dr Saichon Jaiyen for their valuable suggestions and comments for my thesis.

Finally, thanks to my family and friends for their unceasing encouragement.

**CONTENTS**

Page

Page

Page

# List of Tables

## List of Figures



# Chapter 1. Introduction

In last decade, machine learning in specific and artificial intelligence, in general, gained a lot potential to get involved in a variety of applications, for instances, object detection, speech recognition, machine translation, etc. Machine learning also started to grab newcomers of users from either technical and non-technical backgrounds, for such users they required off-the-shelf software packages that help them implement, validate, and evaluate their models. Machine learning community has developed multiple such applications built-in with a variation of advanced machine learning models, feature selection techniques, and analytics tools through open source packages. Such as WEKA [1], RapidMiner, and PyBrain [2]. These applications usually ask the user for two main tasks: firstly, to select a learning algorithm and secondly to tune its hyperparameters. Often a user may lack an in-depth understanding of each learning algorithm and its hyperparameters. However, the users usually choose an algorithm based on reputation or background knowledge. This result in selecting an algorithm that might give a worse performance than the optimal one.

This introduces an essential problem in machine learning: given a data set and some input settings, automatically recommend a learning algorithm to optimize empirical performance. This work provides a semi-automated tool which requires minimal inputs from its user, it searches and recommends the learning algorithm with the best performance.

There has been considerable past work address the problem of model selection and hyperparameters optimization, in 2013 the authors have combined these two problems together and solve it using Bayesian optimization technique and named it CASH [3], the technique performs well in most of the data sets epically the large once, however, the running time for this technique is quite long despite the data set size been relatively small. Last year Google published another work in automated

machine learning which addresses the problem of optimizing a neural network using another one [4], the work considers the optimization problem of a large data sets using complex models such as recurrent neural network [5], however, which this is not a starting point for non-expert users. Also, there have been considerable past works separately addressing model selection [6] [7] [8] and hyperparameters optimization [9] [10] [11].

## 1.1 Objective

In order to build a high-performance recommendation model, there are three objectives as follows.

1. Define the characteristics such as training speed, interpretation, and memory-usage for the considered learning algorithms.
2. Define a set of rules to select a proper learning model to achieve the accuracy as highest as possible from the user-specified parameters.
3. Evaluate the performance of the model compared with a previous work that addresses the same problem.



## 1.2 The scope of the thesis and constraints

The proposed model use rule-based technique, there are some advantages of using this technique over another technique used in recent works. Here is a list of these advantages:

1. **Training speed:** rule-based models often faster to run due to the complexity of the model which is conditions against predefined rules.
2. **Easy to understand:** the output of a rule-based model often easy to understand by its user and that helps a lot in this work since it targeting non-expert users.

3. **Modularity:** Each rule can be designed independently, that allows to for easy modification such as adding or deleting rules later.

This research faced two important challenges as the following.

1. The number of learning algorithms is considerably large, the full group of algorithms in WEKA reaches 39 classifiers, however, in this study, only 20 classifiers have been considered.
2. Most of the previous work addresses the problem of optimizing the hyperparameters for the selected algorithm, so a result of that it was quite challenging to compare the proposed model results to those approaches.

## 1.3 Expected outcome

Rule-based recommendation model that searches and selects a learning algorithm based on user inputs such as memory size, training speed, and the interpretation complexity of the desired algorithm.

# Chapter 2. Theoretical backgrounds

## 2.1 WEKA data mining software

"Waikato Environment for Knowledge Analysis" [1], it is a software package for data mining models implemented using Java. The first development started back in 1997 at the University of Waikato in New Zealand. It is free software licensed under the GNU General Public License. The current version of WEKA (Weka 3) contains a variety of visualization tools, machine learning algorithms, feature selection techniques for data analysis and machine learning, wrapped up using graphical user interfaces to ease of use.

There are some reasons for considering WEKA in this work. Here is a list of these reasons:

1. Easy to use for non-expert users which it makes it quite popular in the academic area.
2. WEKA is open-source software, which makes it simple to build new packages and deploy them using WEKA's package manager.
3. Completely portable, because it implemented in Java.
4. Contains a variety of visualization graphs, machine learning algorithms, and feature selection techniques.

WEKA offers a variety of data mining techniques, precisely, data preprocessing techniques, clustering algorithms, classification models, regression, visualization, and feature selection. All of these techniques run on the supposition that the data is stored and imported from one flat file or a relational database. This proposed approach can be extended and implemented using a newer application such as PyBrain [2] or RapidMiner.

## 2.2 The Classification in machine learning

Since the proposed model is built on assumption that all the recommended models belong to the classification problem, is essential to define the classification in machine learning. The following is a brief description.

In machine learning, there are different tasks and techniques. The challenge of estimating the desired hypothesis (h) from input variables (X) to discrete output variables (y) is called classification. The output called labels or classes. The hypothesis predicts the class or label for a given input. Concretely, mapping an email to spam or non-spam or an object to a category are instances of the classification. If the output (y) contains only two classes, this kind of task called binary classification, otherwise if (y) is greater than two the problem called multi-class classification.

### 2.2.1 Binary class classification

Here an example of a binary class classification. Suppose you have purchasing history data (see Table 1) for a group of customers. You try to make a prediction about whether a new customer going to buy your product or not based on two attributes (age, estimated salary).

*Table 1 A sample from purchasing history data set*

| Age | Estimated Salary | Purchased |
|---|---|---|
| 19 | 19000 | 0 |
| 35 | 20000 | 0 |
| 26 | 43000 | 0 |
| 27 | 57000 | 1 |
| 19 | 76000 | 0 |
| 27 | 58000 | 1 |
| 27 | 84000 | 0 |
| 32 | 150000 | 1 |
| 25 | 33000 | 0 |
| 35 | 65000 | 0 |

Since the data set contains only two features it is possible to plot the data set in 2-dimensional space (see Figure 1). As the plot shows, the output labels (y) consists of only two types of true or false. A standard machine learning approach will try to separate these two classes into two regions.

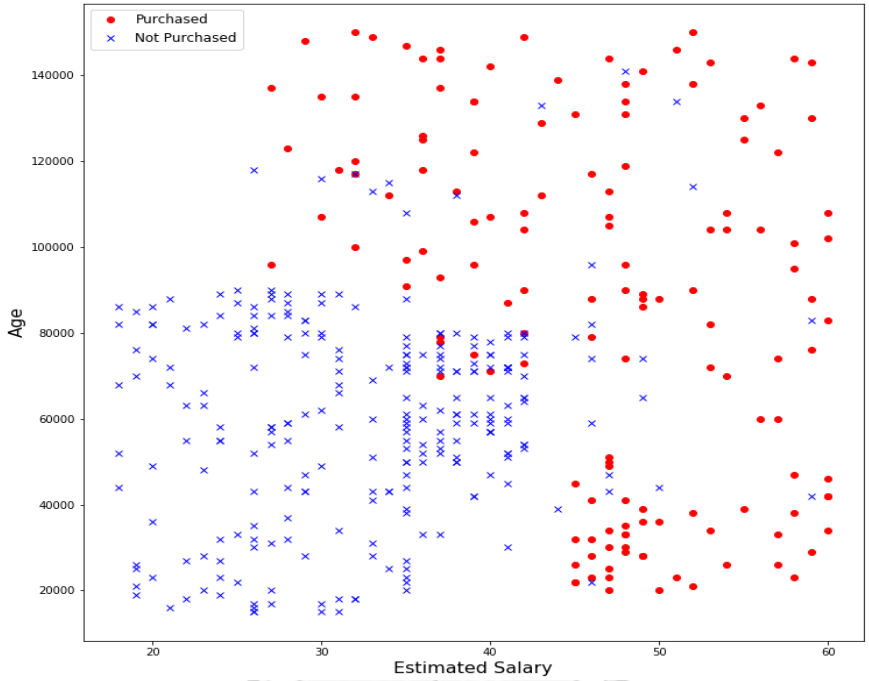



*Figure 1 Binary class classification.*



### 2.2.2 Multi-class classification

Another type of classification is the multi-class classification when the output labels are more than two categories. Let's take the Iris dataset. The dataset represents three types of Iris flowers, and the task again to classify a new unseen flower to one of these types based on some features. The dataset (see Table 2) contains four features corresponding to the width and the length of sepal and petal correspondingly, the data set also contains the output labels that correspond to three types of Iris flowers (Setosa, Versicolour, Virginica).

*Table 2 A sample from the Iris data set.*

| Sepal length | Sepal width | Petal length | Petal width | Class |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3 | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 7 | 3.2 | 4.7 | 1.4 | versicolor |
| 6.4 | 3.2 | 4.5 | 1.5 | versicolor |
| 6.9 | 3.1 | 4.9 | 1.5 | versicolor |
| 6.3 | 3.3 | 6 | 2.5 | virginica |
| 5.8 | 2.7 | 5.1 | 1.9 | virginica |

For the purpose of demonstration, only two features (Petal length, Petal width) have used in the below figure. As the figure show, there are clearly three different classes, the machine learning algorithm then learning how to separate these three regions based on the flower features.

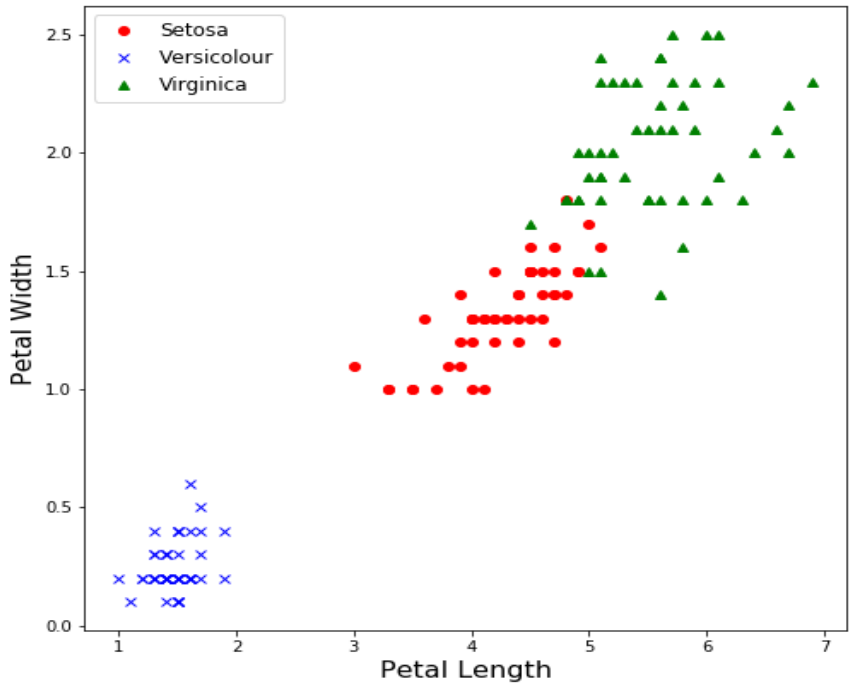



*Figure 2 Multi-class classification*

## 2.3 Machine learning algorithms in WEKA

WEKA has a variation of classification and regression models [1]. These models by default and non-native packages the ones can be downloaded and used. In this thesis, a range of 20 algorithms has used to build the model. The reasons for using these models are:

1. These 20 models are the most used once among the other learning algorithms in the area of machine learning.
2. Since the proposed model consider the classification problem it was necessary to choose models that built for the purpose of classification.
3. The proposed model required a set of characteristics for each model such as training speed, interpretation, and memory-usage, for that reason it necessary to select most general learning algorithms that have available public information.

In chapter 3 all the 20 learning algorithms will be discussed. Here is the full list of the used algorithms with a brief description.



1. **Bayes Net:** Bayes Network architecture, the network learns by using various search algorithms.
2. **J48:** A leaning model for producing a C4.5 tree classifier.
3. **IBK:** K-nearest neighbours classifier.
4. **Decision Stump:** Classifier that uses decision stump method for building models.
5. **Logistic:** A machine learning use ridge function for building a multinomial logistic regression model.
6. **SMO:** A classifier use for Implementing a sequential minimal optimization algorithm for training a support vector classifier.

7. **MultilayerPerceptron:** A classifier that uses the famous backpropagation in this model to classify instances.
8. **Random Forest:** A classifier for building a forest of random trees.
9. **AdaBoostM1:** A Classifier for improving a nominal class classifier using the AdaboostM1 technique.
10. **Bagging:** A classifier for catching classifiers to reduce variance.
11. **KStar:** A classifier build using the instance-based (IB), is predicting the output value by calculating some similarity function with similar instances.
12. **NaïveBayes:** A classifier that uses estimator classes for making a prediction.
13. **PART:** A classifier for producing a PART decision tree.
14. **SimpleLogistic:** A Classifier uses linear logistic regression for prediction.
15. **JRip:** A classifier that implements a rule learner model.
16. **OneR:** 1R classifier uses a technique called minimum-error attribute for making a prediction.
17. **ZeroR:** A classifier for building and using a 0-R classifier.
18. **REPTree:** Fast decision tree learner.
19. **DecisionTable:** A classifier that uses a decision table model for making a prediction.
20. **RandomTree:** A tree-like model uses a predefined random number of attributes K in each node.



These 20 learning algorithms are built for the classification problem which can be used in a variety of applications such as image classification, object detection, recommendation systems, etc.

# Chapter 3. Related works

This chapter reviews several regarding the related works that address the problem of model selection and all the considered classifiers in this research.

## 3.1 Learning algorithms

### 3.1.1 BayesNet classifier

BayesNet is a probabilistic model used in machine learning for solving different problems such as classification, regression, etc. BayesNet build based the Bayes's theorem which works on the assumption that one feature is independent of the others. Let's say we have $U$ which is a set of variables. Note that $U = \{x_1, \dots, x_n\}\, \mathrm{n} > 1$. $B_S$ is a network structure defined by a Bayesian network $B$ across a group of variables $U$, which is creates a directed acyclic graph (DAG) over $U$ and a set of probability tables $B_p = \{p(u|pa(u))|u \in U\}$ where $pa(u)$ is the set of parents of $u$ in $B_S$.

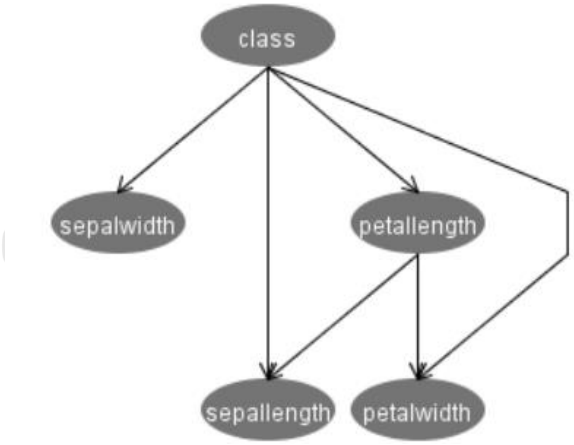



*Figure 3 network structure of the BayesNet*

Figure 3 shows the BayesNet learning algorithm network structure, for each of the nodes, if the node's parents are given then the network can specify a probability distribution for the node. For instances, in the BayesNet above (see Figure 3) there is a conditional distribution for petal length given the value of the class.

Bayesian networks require additional functionality in order to be used as a classifier. This can be done by adding inference algorithms to the network. One simply calculates the maximum probability $argmaxyP(y|x)$ by using the distribution $P(U)$ that given by the Bayesian network. Now note that

$$P(y|x) = P(U)/P(x)$$

$$\propto P(U)$$

$$= \prod_{u \in U} p\big(u|pa(u)\big) \qquad (3.1)$$

Complex inference function is not required in this case since all variables in $x$ are known, calculate (3.1) for all class values will be enough for making predications.

### 3.1.2 J48 classifier

J48 is a classifier built on the ID3 learning algorithm. J48 has more supplementary features such as considering for missing values, pruning the decision trees. WEKA data mining application contains an open-source implementation of the J48 classifier. There are some options available to the user in case of pruning the tree. J48 classifies the data set by generating rules for each feature. The objective is to build a decision tree that generalizes well and progressively obtains a balance of flexibility and accuracy.

**J48 steps**

1. Create the root node with the different available attributes in case this case class type, memory-usage, data set size, etc.
2. Calculate the entropy for the attributes and select the one with the highest entropy and assign it to the current node.
3. Keep this process until there are not attributes left.

**Counting the Gain**

Entropy is the process of measuring the data disorder. The Entropy can be calculated using the following equation:

$$Entropy(\vec{y}) = -\sum_{j=1}^{n} \frac{|y_i|}{|\vec{y}|} \log\left(\frac{|y_i|}{|\vec{y}|}\right)$$

The intention is maximizing the Gain. the Gain can be calculated using the equation below:

$$Gain(\vec{y}, j) = Entropy(\vec{y} - Entropy\ (j|\vec{y})\ ) \qquad (3.2)$$

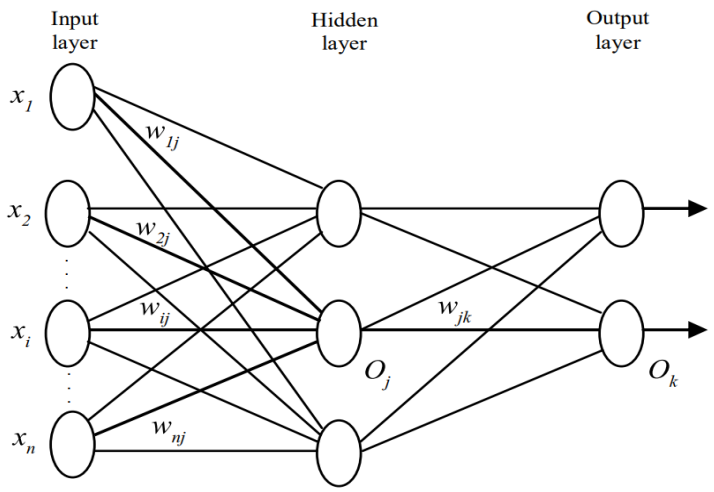


### 3.1.3 MultilayerPerceptron classifier

Multilayer Perceptron learning model is built based on the famous backpropagation algorithm to classify data. The network is constructed using a Multilayer Perceptron (MLP) algorithm. Even though sometimes it a challenging task but the network still can be monitored such as keep tracking of the learning curves and tune the network hyperparameters during the training time. The nodes in the hidden layers are all using the sigmoid activation function for non-linearity. The backpropagation neural network is basically a combination of multi simple processing units that work together to predict a complex output. The backpropagation algorithm uses a multilayer feed-forward neural network for learning the best fitting model that will correctly predict the desired output. It learns a vector of weights for each feature

in the data set in order to make a prediction of the class label. The simplest neural network constructed using one input layer, one hidden layer, and one output layer. Neural networks can be much more complicated and that can be done by adding more hidden layers, and it can also call a deep neural network. An example of a multilayer network is shown in the figure below.

Figure 4 The architecture of a multilayer neural network

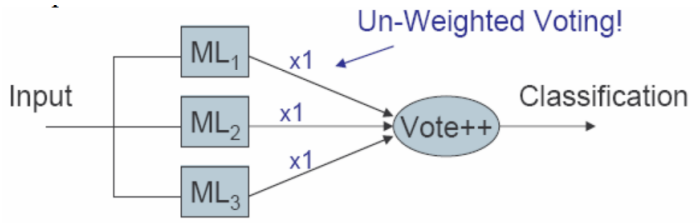





### 3.1.4 RandomForest classifier

The random forest machine learner algorithm is a meta-learner. That constructed using many individual learners such as decision trees. The random forest is a combination of multiple random trees that make a prediction by voting on a particular outcome. Random forest algorithm assigns an equal weight for each vote. The algorithm chooses the classification that has the maximum votes. Figure 5 shows the architecture of the meta-learner [5].

Figure 5 The architecture of the Meta-learner model




### 3.1.5 OneR classifier

OneR stands for (One Rule), is a rule-based model classification algorithm. The classifier produces a rule for every feature in the data set, As the name suggests "One Rule" it selects the rule that minimizes total error. To create a rule for a feature, a table containing the frequency count should be built for each feature in contradiction of the target. OneR algorithm shows in many experiments the ability to produce performance only slightly less accurate than the state-of-the-art models, but the generated output is much easier for a human to interpret.

### 3.1.6 SMO classifier

SMO stands for Sequential minimal optimization is an algorithm build to solve the problem of optimization of the cost function during the training of support vector machine. In 1998 SMO invented by John Platt at Microsoft Research. SMO [6] is commonly used for the purpose of training the support vector machine. SMO is an iterative algorithm for solving the optimization problem of the support vector machine. SMO divides the problem of optimization into a set of smallest possible sub-problems, then it solves them analytically. The optimization function is given by the following equation:

$$max_a\, W(\alpha) = \sum_{i=1}^{m} a_i \; - \tfrac{1}{2} \sum_{i=1}^{m} * \sum_{j=1}^{m} y^{(i)}\, y^{(j)} a_i a_j \langle x^{(i)}, x^{(j)} \rangle \quad (3.3)$$

$$subject\ to \quad 0 \leq a_i \leq C, \qquad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} a_i y^{(i)} = 0$$

### 3.1.7 NaïveBayes classifier

NaïveBayes classifier is a powerful probabilistic model, and their hypothesis [7] or features. The NaïveBayes classifier is built on the architecture of the Bayesian network the difference is that NaïveBayes contains a class without parents and each feature has the class as its single parent. The NaïveBayes classifier is based on Bayes' theorem, which assumes that the features are independent of each other.

A NaïveBayes model is easy to build, with no complex hyperparameters optimization which makes it useful especially for very large datasets. Even though it is quite simple, NaïveBayes is widely used because it outperforms more sophisticated classification models. Bayes theorem [8] provides a way of calculating the succeeding probability, $P(c|x)$, from $P(c),\ P(x),\ and\ P(x|c)$. NaïveBayes classifier works on the assumption of feature independency, meaning the effect of the value of a feature $(x)$ on a given class $(c)$ is independent of the values of other features. This behavior is called class conditional independence.

$$P(c|x) = \frac{P(x|c)\,P(x)}{P\,(x)}$$

$$P(c|x) = P(x_1|c) * P(x_2|c) * \ldots * P(x_n|c) * P(x) \qquad (3.4)$$

$P(c|x)$ the succeeding probability of target class given a feature value.
$P(c)$ the preceding probability of class.
$P(x|c)$ the likelihood which is the probability of feature given class.
$P(x)$ the preceding probability of a feature.



### 3.1.8 ZeroR classifier

ZeroR is a simple classification model that relies on the target and discord all given features. The ZeroR only predicts the majority class correctly. Although there is no impressive performance in ZeroR. But yet it is still useful for benchmarking the average performance for other classification models. ZeroR classifies the instances by building a table containing a frequency count for the target output and then select the output with the maximum frequency.

### 3.1.9 PART classifier

Part is a classifier that uses the rule-based technique for classification problem. It generates the rules from the tree [9] built using the J48 decision tree classifier. This is the reason why usually PART and J48 produce the same result for a particular data set.

### 3.1.10 K-Nearest Neighbor (IBK)

The nearest neighbours algorithm is a statistical algorithm. It is quite simple to implement. The training phase of K-Nearest Neighbor store the data set in the memory, and in the prediction phase when the model gets an unclassified data point, then the model will try to search for the closest data point to the unclassified point and predicts the class label of that training point accordingly to some distance metric [10]. Although is it possible to use any distance metric it common to use Euclidean distance. For numerical feature.

### 3.1.11 DecisionStump classifier

A classifier that uses a decision tree with only one level to classifies the dataset, the model split at the root level based on a specific attribute or value.

### 3.1.12 REPTree classifier



REPTree is a machine leering algorithm that uses tree logic to creates multiple trees in different steps. After that, it selects the best one of all created trees. The REPTree classifier uses the mean square error for pruning the tree and for making predictions [11]. REPTree is fast decision tree learning model and it builds a decision tree based on the information gain.

### 3.1.13 AdaBoostM1 classifier

It is the most famous boosting algorithm. AdaBoostM1 uses a combination of base learners and keep iterating them over a given data set for a predefined number of iterations.

### 3.1.14 Bagging classifier

Bagging is an ensemble technique used to classify the data sets with decent performance. The algorithm first takes the whole data set $D$ and make $n$ a number of samples $D1, D2, .., Dn$ where is $n$ is the number of classifiers $c$, second, it builds all the classifiers $c1, c2, \dots, cn$ on these samples. Finally, it selects the best combination of these classifiers using the voting technique. Bagging the technique can be applied on any classifier such as RandomForest, C4.5, REPTree and J48.

### 3.1.15 RandomTree classifier

This classifier constructs a decision tree using $K$ features at each node, but these features should be random. RandomTree does not prune the tree. RandomTree allows us to approximately estimate the class probability [12].

### 3.1.16 KStar classifier

KStar is Instance-based (IB) learning algorithm, it similar to the Nearest Neighbor algorithm**.** In classification tasks, for each new instance the distance between itself and closest instance Is measured by using some distance metric e.g. Euclidean distance, then the new instance then will get assigned to the same class that the closest data point belongs.



### 3.1.17 JRip classifier

JRip (RIPPER) is a machine learning algorithm; it has a different implementation which allows it to work well with the large size of data set. JRip uses reduced error JRip to produce a set of rules for each class. It uses the training set to find a set of rules that covers it sub-classes. It keeps repeating the same process until it covers all the class in the data set.

### 3.1.18 Logistic Regression classifier

Logistic Regression is a machine learning model, which is extremally powerful classifier if the output label used as a categorical variable. The output class variable has two cases like buy/or not. logistic regression model can be used to predict nonlinear function, and the non-linearity can be done by adding polynomial terms to the hypothesis function.

Logistic regression can be used to solve two different types of target variables:

1. The target as a categorical variable which consists of two binary categories that can be represented by $\mathbf{0}$ or $\mathbf{1}$.
2. The target as a continuous variable which consists of values of a range from $\mathbf{0.0}$ to $\mathbf{1.0}$.

Logistic regression can predict a binary class classification which has values of 0, 1 by using the following formula:

$$P = \frac{1}{2}\left(1 + exp\left(-(B0 + B1 * X1 + B2 * X2 + \cdots .. Bk * Xk)\right)\right) \qquad (3.5)$$

$B0$ A constant.

$Bi$ coefficients of feature variables.

$P$ probability which ranges from $\mathbf{0}$ to $\mathbf{1}$.



### 3.1.19 DecisionTable classifier

A classifier uses a simple decision table classifier to build the classification model. This classifier uses best-first search to evaluates feature subsets, also it is possible to use cross-validation for evaluation.

### 3.1.20 SimpleLogistic classifier

A classifier that uses linear logistic regression for building models. In this classifier LogitBoost with simple regression hypothesis use as base learners for fitting the logistic models. In order to come up with the optimal number of LogitBoost

iterations, a cross-validation should be used, which will automatically select the attributes.

In the last two decades, a significant amount of past works have published that separately addressing model selection [13] [14] [15] and hyperparameters optimization [16] [17] [18]. Here are some of these works.

## 3.2 Learning algorithm selection

Learning algorithm selection, also called model selection, it is the process of selecting a model for a given training data. There have been many studies to address this problem. The simplest and most common approach is given a group of learning algorithms $\mathbf{A}$ and a set of training data $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, the objective of model selection is to find the algorithm $A^* \in A$ with the best performance. The way of measuring the generalization is by splitting $D$ into a constant number of subsets between training and validation sets $D_{train}^{(i)}$ and $D_{valid}^{(i)}$ for $i = 1, \ldots, k$ and then learning function $f_i$ by applying $A^*$ to $D_{train}^{(i)}$, and evaluate the result on $D_{valid}^{(i)}$. This process can be given using the following equation.

$$A^* \in \underset{A \in \mathbf{A}}{argmin} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A, D_{train}^{(i)}, D_{valid}^{(i)}) \qquad (3.1)$$



Where $\mathcal{L}(A, D_{train}^{(i)}, D_{valid}^{(i)})$ is the loss achieved by $\mathbf{A}$ when trained $D_{train}^{(i)}$ on and evaluated $D_{valid}^{(i)}$ .

Another way to solve this problem is by using Meta-learning, which uses machine learning to make predictions about a data set as a whole, rather than a specific element in the data set. One example of Meta-learning technique is landmarking. On a group of data sets, a vector of features of the data set is computed, for instances, the number of categorical or numeric features, the number of prediction labels or the size of the dataset. A meta-learner is then trained on these dataset

features to either predict the best algorithm for a particular data set or provide a ranking over algorithms that should be used on the data set.

### 3.3 Hyperparameters optimization

optimizing the hyperparameters $\lambda \epsilon \Lambda$ of a given learning algorithm $A$ is by far similar to that of model selection. Both of these problems optimizing for the best performance for a given data set. The main difference is instead of selecting from many different algorithms the optimizations considers a single algorithm's hyperparameters. The optimization function can be written as follows:

$$\lambda^* \in \underset{\lambda \in \Lambda}{argmin} \frac{1}{k} \sum_{i=1}^{k} \iota \left(A_\lambda, D_{train}^{(i)}, D_{valid}^{(i)}\right) \tag{3.2}$$

There are different techniques used in hyperparameters optimization problems such as grid search [10], random search [11], evolutionary techniques [12], and Bayesian optimization [13], however, in this work does not consider the problem of hyperparameters optimization. The decision was to use the default hyperparameters during the training phase, this because the intention of this work is to build a recommendation model that selects a learning algorithm with minimizing the training time.

### 3.4 Auto-WEKA

It is an open-source full automated tool build for WEKA application [3]. It combines the problems of model selection and hyperparameters optimization into one single hierarchy and dubs it as the combined algorithm selection and hyperparameters optimization problem (CASH). Combining the model selection and hyperparameters problems can be given by the following equation:

$$A^*_{\lambda^*} \in \underset{A^{(j)} \in A,\ \lambda \in \Lambda^{(j)}}{argmin} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}\left(A_{\lambda}^{(j)}, D_{train}^{(i)}, D_{valid}^{(i)}\right) \qquad (3.3)$$

Even though the proposed model does not consider the problem of hyperparameters optimization, the results have compared to Auto-WEKA technique for benchmarking the performance of the proposed model.

# Chapter 4. The proposed model

This work proposed a semi-automated technique built based on WEKA data mining application for selecting a learning algorithm based on the user inputs, such as training speed, the amount of memory the user's machine contains, and the interpretation of the selected algorithm. This chapter demonstrates the techniques that used to build the recommendation model by firstly, collect a set of features for the considered learning algorithms, secondly, eliminate the duplicated patterns, thirdly, create the rules based on the optimal patterns. The model built using the rule-based technique which has some advantages over the other techniques used in previous works such as Bayesian optimization, random search, and grid search. Some of these advantages are the speed especially for the small number of variables which the case in this work, simplicity, and easy to understand by a non-expert user.

## 4.1 The characteristics of the models

In this thesis 20 learning algorithms have been used to build the recommendation model, for each model, characteristics have been collected into a single table as shown in

. These characteristics represent the average training speed of the model, how complex is the output to interpret, and the dataset size. In WEKA Graphical User Interface (GUI) there is a limitation that classifiers with nominal class will throw an error if it trains on a dataset with numeric class. Since the model does not perform any preprocessing technique it was required to add another feature to the properties table to indicate whether that classifier accepts numeric class values or nominal class values.

The table contains a column for the classifier name and four features. The first feature is the class type which indicates whether that classifier accepts numeric class value or nominal class value. The second feature is the data set size which has two values representing two types of models. The first value is more than 10K samples, which represents models that perform better with large amounts of data, and the second value is less than 10K samples, which represents models that perform better

with small amounts of data. To make it more general any data set with more than 10K samples is considered a large data set. And any data set with less than 10K samples considered a small dataset [19] [20]. Note that data sets with more than 1.5M samples tend to throw out-of-memory an exception using WEKA's graphical user interface tool.

For the memory column, experiments have done to come up with average estimation of the required memory for each classifier, and this column indicates the total amount of memory should be available. Note that classifiers are still able to run if the available memory less than the stated once. Interpretation column indicates whether the output of the classifier is hard or easy to understand. The last feature indicates whether the model fast during the training phase or slow which has measured during the memory experiments. The following table shows the classifiers with their properties.

*Table 3 Learning algorithms characteristics*

| Classifier | Class | Dataset Size | Memory | Interpretation | Training Speed |
|---|---|---|---|---|---|
| Bayes Net | Nominal | > 10K | 4GB | Easy | Fast |
| J48 | Nominal | > 10K | 2GB | Easy | Fast |
| Decision Stump | Numeric | > 10K | 2GB | Easy | Fast |
| IBK | Numeric | < 10K | 4GB | Easy | Fast |
| Logistic | Nominal | > 10K | 4GB | Easy | Fast |
| SMO | Nominal | > 10K | 4GB | Hard | Slow |
| Multilayer Perceptron | Numeric | > 10K | 4GB | Hard | Slow |
| Random Forest | Nominal | > 10K | 4GB | Hard | Slow |
| AdaBoostM1 | Nominal | > 10K | 2GB | Hard | Fast |
| Bagging_J48 | Numeric | > 10K | 4GB | Hard | Slow |
| KStar | Numeric | < 10K | 2GB | Easy | Fast |
| Naïve Bayes | Nominal | > 10K | 2GB | Easy | Fast |
| PART | Nominal | > 10K | 2GB | Easy | Fast |
| Simple Logistic | Nominal | > 10K | 4GB | Easy | Fast |
| JRip | Nominal | < 10K | 2GB | Easy | Fast |
| OneR | Nominal | < 10K | 2GB | Easy | Fast |
| ZeroR | Numeric | < 10K | 2GB | Easy | Fast |
| REP Tree | Numeric | > 10K | 2GB | Easy | Fast |
| Decision Table | Numeric | < 10K | 2GB | Easy | Fast |
| Random Tree | Numeric | > 10K | 2GB | Easy | Fast |

## 4.2 Pattern optimization

As the above table shows, there are many algorithms have the same characteristics, it is important to combine these classifiers with the same features into distinct patterns. Each pattern represents one or a group of classifiers. For this reason, classifiers with same properties have been combined together. The following table shows the combined similar learning algorithms into nine distinct patterns.

*Table 4 The combined learning algorithms*

| Classifier | Class | Dataset Size | Memory | Interpretation | Training Speed |
|---|---|---|---|---|---|
| Bayes Net \|\| Simple Logistic, Logistic | Nominal | > 10K | 4GB | Easy | Fast |
| J48 \|\| Naïve Bayes \|\| PART | Nominal | > 10K | 2GB | Easy | Fast |
| Random Tree \|\| REP Tree \|\| Decision Stump | Numeric | > 10K | 2GB | Easy | Fast |
| IBK | Numeric | < 10K | 4GB | Easy | Fast |
| SMO \|\| Random Forest \|\| Bagging | Nominal | > 10K | 4GB | Hard | Slow |
| Multilayer Perceptron | Numeric | > 10K | 4GB | Hard | Slow |
| AdaBoostM1 | Nominal | > 10K | 2GB | Hard | Fast |
| JRip \|\| OneR | Nominal | < 10K | 2GB | Easy | Fast |
| ZeroR \|\| KStar \|\| Decision Table | Numeric | < 10K | 2GB | Easy | Fast |

## 4.3 The proposed algorithm

The proposed model constructed from the table above as a hierarchy problem. In this thesis, the decision tree technique used to come up with the rules for the model. The decision tree splits the classifiers into nine distinct nodes. Each node contains one or a group of classifiers.

First, let's discuss the steps that have done in order to prune the tree in the way that shown in Figure 6.

1. Create the root node with the different available attributes in case this case class type, memory-usage, data set size, etc.
2. Calculate the entropy for the attributes and select the one with the highest entropy and assign it to the current node.
3. Keep this process until there are not attributes left.

Entropy is the process of measuring the disorder of the data. In this task, the model calculated the entropy in order to get the value of the Gain. The Entropy is calculated by:

$$Entropy(\vec{y}) = -\sum_{j=1}^{n} \frac{|y_i|}{|\vec{y}|} \log\left(\frac{|y_i|}{|\vec{y}|}\right)$$

$$Entropy(j|\vec{y}) = \frac{|y|}{|\vec{y}|} \log\left(\frac{|y_i|}{|\vec{y}|}\right)$$



The intention is to maximize the Gain. the Gain can be calculated using the following equation:

$$Gain(\vec{y}, j) = Entropy(\vec{y} - Entropy\ (j|\vec{y})\ ) \quad (3.2)$$

The following steps will demonstrate that:

1. First split the classifiers into two groups based on the class value, however, classifiers with numeric class value are still applicable to nominal class data sets.
2. Divide the two groups from step 1 based on data set size. This will create two groups with more than 10K samples and less than 10K samples.
3. Then perform another split based on memory usage.
4. Finally, use the interpretation feature to split the classifiers with a similar pattern.

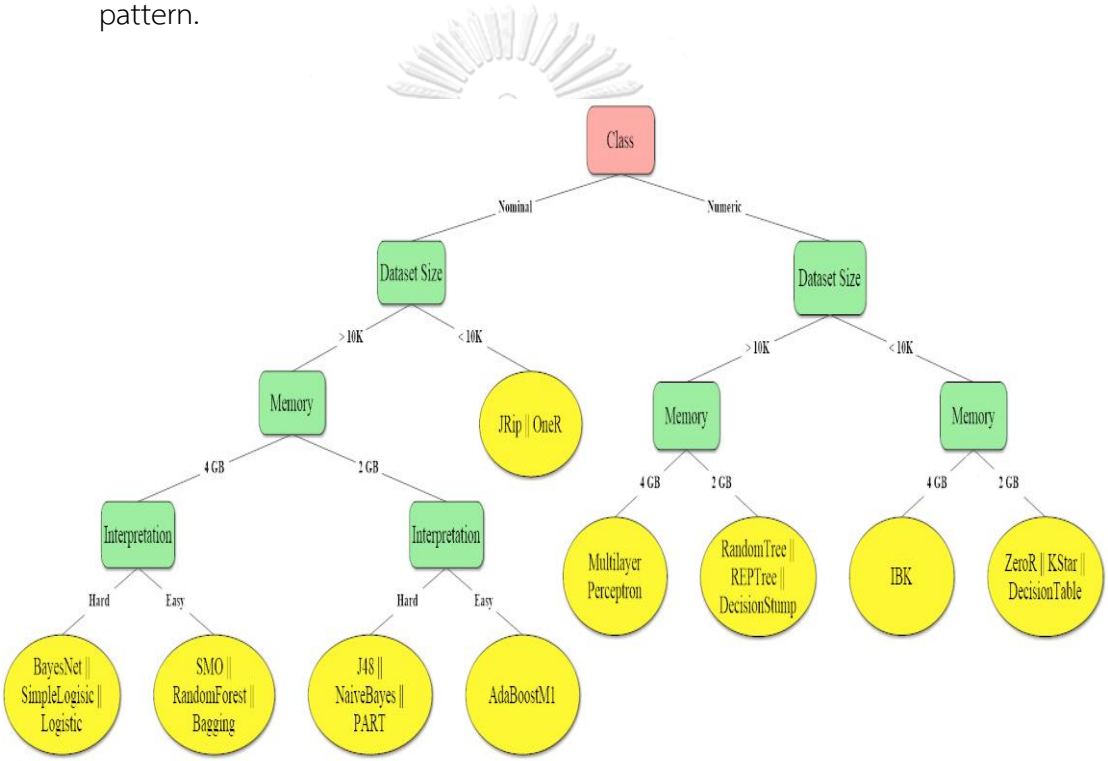



*Figure 6 The architecture of the proposed model*

In this approach, the size of the dataset and the type of the class are automatically selected during the algorithm running time by the system. For the rest of the specification, practically, memory, training speed, and interpretation are manually determined by the user. Now after defining the required rules for selecting a model, let's demonstrate the steps that the model and the user perform in order to come up with a recommended classifier for a given data set.

- **Step 1.** Data set selected by the user.
- **Step 2.** Automatically the model detects the size (> 10K or < 10K) of the data set and the class type (Nominal or Numeric).
- **Step 3.** These three properties of memory, interpretation, and training speed should get selected by the user.
- **Step 4.** For patterns with multi classifiers, the model will train the classifiers on the data set and pick the one with the highest accuracy.
- **Step 5.** The model will recommend one classifier based on all the parameters from the previous steps.

# Chapter 5. Experiments and Results

## 5.1 Experimental setup

All of the experiments run on a Linux machine, having dual Intel Xeon Intel Core i5-7500 processors with 8GB of RAM. WEKA version 3.8 used to perform all the experiments. 2 GB of RAM have enforced the training of the learning algorithm. In this experiment Auto-WEKA [10] tool used for result comparison, the time limit for Auto-WEKA set to 15 minutes and the memory limit set to 1GB of RAM. Finally, the reported memory usage for each recommended classifier measured using JConsole application that comes by default with the Java Virtual Machine (JVM).

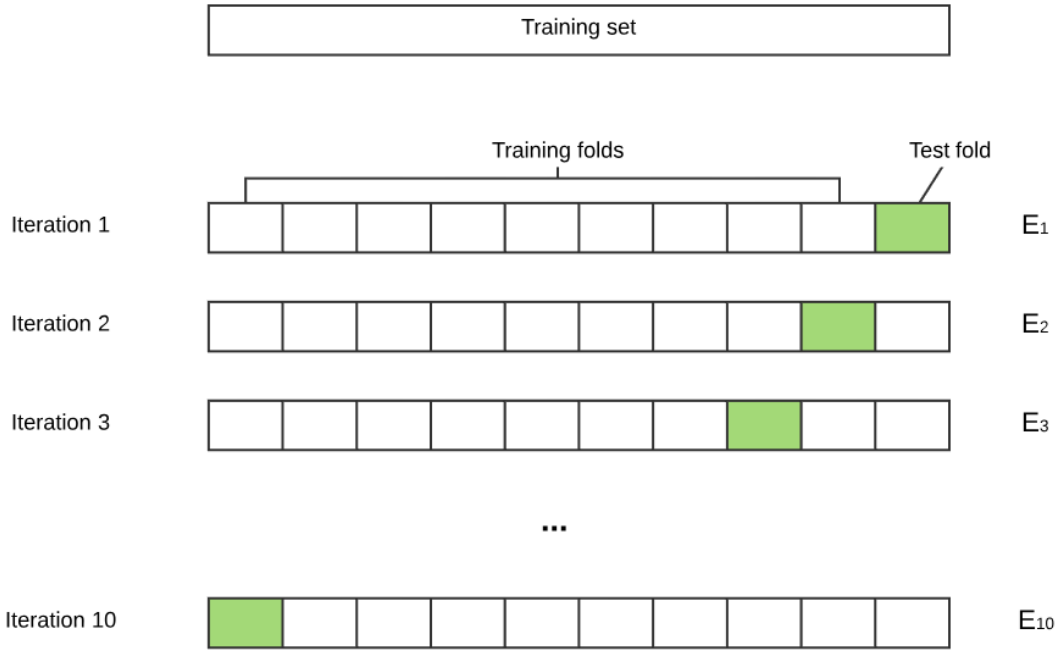
## 5.2 K-fold cross-validation

Cross-validation [25] is a simple but effective technique used a tremendous amount of time for testing in the machine learning literature and real-world applications. The approach works simply by splitting the full data set into a training set for training the model, and a test set for testing it. k-fold cross-validation means the full data set is randomly splitting into k equal size subsets. In the training phase, K-1 subsets will be used for training the model, and just one subset will be used for testing the model.



In this experiment, a 10-fold cross-validation used to partition the 10 data sets (see Table 15), 9-fold used to train the models and 1-fold to test it. The reported accuracy is the average performance for each fold.

*Figure 7 10-fold cross-validation.*




Where $E_i$ the performance result for each fold. The final accuracy can be given using the following equation:

$$E = \frac{1}{k}\sum_{i=1}^{k} E_i \qquad (5.1)$$

## 5.3 Performance measurements

### 5.3.1 The accuracy

The classification results measured using the accuracy matrix, which is one of the built-in performance matrices in WEKA. The accuracy matrix can be calculated using the following equation.

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (5.2)$$

- **True Positive (TP):** # of instances correctly classified into the positive class.
- **True Negative (TN):** # of instances correctly classified into the negative class.
- **False Positive (FP):** # of instances classified by the model as a positive class, while they belong to the positive class.
- **False Negative (FN):** # of instances classified by the model as a negative class, while they belong to the positive class.

### 5.3.2 The memory

The memory usage for the selected classifiers by either the proposed model or Auto-WEKA measured using JConsole application which comes by default with the Java Virtual Machine (JVM). JConsole is a monitoring system allows the user for monitoring Java applications both on a local or remote server by using a graphical user interface [21].

## 5.4 Experimental data sets

In this experiment, 10 data sets used for testing the proposed model. All the data sets are selected from the UCI repository [22]. In order to perform an exhausted testing for all the model characteristics, a variety of the data set have been selected. The data sets vary in the number of samples that each data set has, the class type, and the number of features. 5 of the dataset have more than 10K samples and 5 have

less than 10K samples. There are three reasons why these data sets have chosen for the testing purpose:

1. The proposed model is built for non-expert users, so a typical user will try out most popular data sets in the UCI repository.
2. There is a limitation in WEKA's graphical user interface that data sets with more than 1 million samples will stop running and throw an out-of-memory exception.
3. Since there are two values for data set size in the model table see Bank the decision was to select data sets that satisfy those two values.

The following content briefly describes the ten data sets and list the characteristics of each one:

### 5.4.1 Iris data set

This data set includes information about the Iris flower. The objective is to predict the new flower belongs to which type. The data set has 150 instances and 4 features [22].

*Table 5 The characteristics of the Iris data set*



| | | | |
|---|---|---|---|
| **Data Set Type** | Multivariate | **# of Instances** | 150 |
| **Attribute Type** | Real | **# of Attributes** | 4 |
| **Task** | Classification | **Missing Data** | No |

### 5.4.2 Breast Cancer data set

This data set is quite famous in the machine learning. The objective is to predict whether the patient has a breast cancer or not. This data set contains 286 instances and 9 features [22].

*Table 6 The characteristics of the Breast Cancer data set*

| | | | |
|---|---|---|---|
| **Data Set Type** | Multivariate | **# of Instances** | 286 |
| **Attribute Type** | Categorical | **# of Attributes** | 9 |
| **Task** | Classification | **Missing Data** | Yes |

### 5.4.3 Bank data set

The data set classifies the bank customers by a set of features as buy PEP or not. The data set contains 600 instances and 11 attributes with the output label which has two binary values 0 or 1.

*Table 7 The characteristics of Bank data set*

| | | | |
|---|---|---|---|
| **Data Set Type** | Multivariate | **# of Instances** | 600 |
| **Attribute Type** | Categorical, Real, Integer | **# of Attributes** | 11 |
| **Task** | Classification | **Missing Data** | No |

### 5.4.4 German Credit data set

This data set includes information about credit card holders. The objective is to correctly classify whether there will be a credit risk or not. The data contains 1000 instances and 20 features [22].

*Table 8 The characteristics of German Credit data set*

| | | | |
|---|---|---|---|
| **Data Set Type** | Multivariate | **# of Instances** | 1000 |
| **Attribute Type** | Categorical, Integer | **# of Attributes** | 20 |
| **Task** | Classification | **Missing Data** | N/A |

### 5.4.5 Abalone data set

This data set includes information about the abalone. The objective is to predict the age of the abalone form different features repressing the physical measurements. The data set has 4177 instances and 8 features [22].

*Table 9 The characteristics of Abalone data set*

| | | | |
|---|---|---|---|
| **Data Set Type** | Multivariate | **# of Instances** | 4177 |
| **Attribute Type** | Categorical, Integer, Real | **# of Attributes** | 8 |
| **Task** | Classification | **Missing Data** | No |

### 5.4.6 Letter Recognition data set

The data set includes information about images representing the 26 English letters. The objective is to predict a large number of pixels grouped together to make a shape of one in the image. The data set contains 20000 instances and 16 features [22].

*Table 10 The characteristics of Letter Recognition data set*

| | | | |
|---|---|---|---|
| **Data Set Type** | Multivariate | **# of Instances** | 20000 |
| **Attribute Type** | Integer | **# of Attributes** | 16 |
| **Task** | Classification | **Missing Data** | No |

### 5.4.7 Credit Card Clients data set

This data set includes information about payments for a group of customers based in Taiwan. The objective is to predict the default client's payment. The data set has 30000 instances and 24 features. [22].

*Table 11 The characteristics of Credit Card data set*

| | | | |
|---|---|---|---|
| **Data Set Type** | Multivariate | **# of Instances** | 30000 |
| **Attribute Type** | Integer, Real | **# of Attributes** | 24 |
| **Task** | Classification | **Missing Data** | N/A |

### 5.4.8 Shuttle data set

This data set includes information about the shuttles. The data set is unbalanced almost 80% of the data belongs to class 1. The data set has 58000 instances and 9 features. [22].

*Table 12 The characteristics of Shuttle data set*

| | | | |
|---|---|---|---|
| **Data Set Type** | Multivariate | **# of Instances** | 58000 |
| **Attribute Type** | Integer | **# of Attributes** | 9 |
| **Task** | Classification | **Missing Data** | N/A |

### 5.4.9 Bank Marketing data set

The data set includes information about marketing campaigns for a banking institution based in Portuguese. The objective is to predict whether the customer will subscribe or not [22].

*Table 13 The characteristics of Bank marketing data set*

| | | | |
|---|---|---|---|
| **Data Set Type** | Multivariate | **# of Instances** | 45211 |
| **Attribute Type** | Real | **# of Attributes** | 17 |
| **Task** | Classification | **Missing Data** | N/A |

### 5.4.10 Adult data set

This data set includes information about a different group of people. The objective is to predict whether a person will make over fifteen thousand a year or not. The data set has 48842 instances and 14 features [22].

*Table 14 The characteristics of the Adult data set*

| | | | |
|---|---|---|---|
| **Data Set Type** | Multivariate | **# of Instances** | 48842 |
| **Attribute Type** | Real, Integer | **# of Attributes** | 14 |
| **Task** | Classification | **Missing Data** | Yes |

The below Table 15 shows all the 10 data sets including the number of instances, features, and classes for each one.

*Table 15 The characteristics of all data sets used in this experiment*

| Dataset Name | Number of Instances | Number of Features | Number of Classes |
|---|---|---|---|
| Iris | 150 | 4 | 3 |
| Breast Cancer | 286 | 9 | 2 |
| Bank | 600 | 11 | 2 |
| German Credit | 1000 | 20 | 2 |
| Abalone | 4176 | 8 | 28 |
| Letter Recognition | 20000 | 16 | 26 |
| Credit Card Clients | 30000 | 24 | 2 |
| Bank Marketing | 45211 | 17 | 2 |
| Adult | 48842 | 14 | 2 |
| Shuttle | 58000 | 9 | 7 |

## 5.5 Classification results

In this work, the proposed model evaluated on 10 famous benchmark classification datasets (see Table 15): all the data sets have collected from the UCI repository [27]; the data sets have a various type of class as well as the number of features. In the proposed model all the selected classifiers run on the default hyperparameters settings, and this is not the case with Auto-WEKA classifiers since it performs hyperparameters optimization for the selected classifier.

*Table 16 The Experiment results for the proposed model compared to Auto-WEKA*

| Dataset | Our Model | | | | Auto-WEKA | | | | SD |
|---|---|---|---|---|---|---|---|---|---|
| | Classifier | Accuracy (%) | Training Time (Seconds) | Memory Usage (MB) | Classifier | Accuracy (%) | Training Time (Seconds) | Memory Usage (MB) | |
| Bank Marketing | RandomForest | 90.3 | 6.4 | 470 | RandomForest | **99.7** | 6.4 | 470 | 6.65 |
| Adult | J48 | **85.4** | 0.6 | 385 | BayesNet | 83.5 | 0.1 | 390 | 1.34 |
| Abalone | JRip | **56** | 0.24 | 550 | RandomSubSpace | 54.2 | 0.26 | 115 | 1.27 |
| Iris | IBK | 95.3 | 0.001 | 50 | SMO | **95.9** | 0.02 | 360 | 0.42 |
| German-Credit | IBK | **72** | 0.001 | 65 | LWL | 70 | 0.001 | 600 | 1.41 |
| Breast Cancer | KStar | 73.4 | 0.001 | 100 | J48 | **75.5** | 0.01 | 65 | 1.48 |
| Bank | JRip | **90.6** | 0.05 | 100 | Bagging | 54.3 | 0.11 | 290 | 25.67 |
| Shuttle | RandomForest | **99.9** | 5.3 | 140 | RandomForest | 99.8 | 5.3 | 130 | 0.07 |
| Letter Recognition | RandomForest | 96.4 | 6.9 | 558 | AdaBoostM1 | **99.9** | 22 | 785 | 2.47 |
| Credit Card Clients | RandomForest | 81.8 | 16.29 | 370 | MLP | **82** | 9.437 | 115 | 0.14 |

In order to fully interpret the experimental result of the classification, a statistical test performed. A paired t-test was performed on the accuracy column (see Table 16) to determine if the proposed model was effective.

*Table 17 T-Test: Paired Two Sample for Means*

| | *The proposed model* | *Auto-WEKA* |
|---|---|---|
| Mean | 84.11 | 81.48 |
| Variance | 184.9498889 | 318.8528889 |
| Observations | 10 | 10 |
| Standard Deviation | 12.31440439 | |
| Hypothesized Mean Difference | 0 | |
| df | 9 | |
| t Stat | 0.675370889 | |
| P(T<=t) one-tail | 0.258207447 | |
| t Critical one-tail | 1.833112933 | |
| P(T<=t) two-tail | 0.516414895 | |
| t Critical two-tail | 2.262157163 | |
| Confidence Level (95.0%) | 8.809194225 | |



## 5.6 Discussion

The experimental results of the classification tasks have shown a good performance in the problem of model selection, however, the accuracy of each selected model still not quite high same as the other approach and there are two main reasons for that:

I. Auto-WEKA runs some techniques in order to select the best fitting features provided in the data set, while the proposed model train the selected models on all provided features.

II. Auto-WEKA preforms hyperparameters optimization technique to select the best-optimized set of hyperparameters that provide the highest accuracy, while the proposed model train the selected models with the default hyperparameters.

A paired t-test was performed on the accuracy column (see Table 16) to determine if the proposed model was effective. The mean weight loss (M=2.63, SD =12.31, N= 10) was meaningfully greater than zero, t (10) = 0.675, two-tail p = 0.516, providing evidence that the proposed model is effective in selecting learning algorithms that have a significant performance even though it does not perform feature selection or hyperparameters optimization techniques.

# Chapter 6. Conclusion

In this thesis, a semi-automated model proposed for the purpose of selecting a classifier in WEKA data mining application based on user specification. This model shows that the problem of model selection can be solved by a building a recommendation model. It is possible by using a rule-based approach and tree architecture. The model demonstrates how to build a recommendation model just by using a simple approach, which takes the user inputs and matches them to several learning algorithms patterns. The proposed model shows how to design a tool that covers a variety of machine learning algorithms implemented all in WEKA and create a simple approach in order to help the user especially the non-experts to select high-performance models for their application. An exhausted comparison of 10 famous benchmark classification data sets from the UCI repository showed that the proposed model often outperformed Auto-WEKA even though it does not perform hyperparameters optimization or feature selection techniques. This work can be extended by adding more sophisticated hyperparameters optimization techniques after the model selection step in order to select better classifiers. This model also can be implemented with another data mining applications such as PyBrain [2] or SNNS [23] since it lightweight and portable.

## REFERENCES


[1] E. F. Mark Hall, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H., "The WEKA Data Mining Software: An Update," *SIGKDD Explor. Newsl.,* vol. 11, no. 1, pp. 10--18, 2009.

[2] J. B. T. Schaul, D. Wierstra, Y. Sun and M. Felder, "PyBrain," *Journal of Machine Learning Research,* vol. 11, pp. 743-746, 2010.

[3] C. T. Leyton-Brown, F. Hutter and H. Hoos, "Auto-WEKA : Combined Selection and Hyperparameter Optimization of Classification Algorithms," 2013.

[4] V. V. Barret Zoph, Jonathon Shlens, and Quoc V. Le. Arxiv, "Learning Transferable Architectures for Scalable Image Recognition," in *Google AI Blog*, ed: Google Brain, 2017.

[5] F. Livingston, "Implementing Breiman's Random Forest Algorithm into Weka," in *ECE591Q Machine Learning Conference Papers*, 2005.

[6] J. Platt. (2018). *Sequential minimal optimization (SMO)*. Available: http://en.wikipedia.org/wiki/Sequential_Minimal_Optimization

[7] D. G. a. P. Domingos, "Learning Bayesian Network Classifiers by Maximizing Conditional Likelihood," 2004.

[8] *Naive Bayesian*. Available: http://chem-eng.utoronto.ca/~datamining/dmc/naive_bayesian.htm

[9] A. l. H. W. E. frank, "Data Mining Practical Machine Learning Tools and techniques," p. 404, 2005.

[10] T. D. a. P. I. a. G. Shakhnarovich, "Nearest Neighbor Methods in Learning and Vision: Theory and Practice," *MIT Press,* 2006.

[11] S. Kalmegh, "Analysis of WEKA Data Mining Algorithm REPTree, Simple Cart and RandomTree for Classification of Indian News," *IJISET - International Journal of Innovative Science, Engineering & Technology,* vol. 2, no. 2, 2015.

[12] M. K. P.Yasodha, "Analysis of a Population of Diabetic PatientsDatabases in Weka Tool," *International Journal of Scientific & Engineering Research,* vol. 2, no. 5, 2011.

[13] A. Biem, "A Model Selection Criterion for Classification: Application to HMM Topology," in *International Conference on Document Analysis and Recognition*, 2003.

[14] M. Adankon, M. Mathias and Cheriet, "Model Selection for the LS-SVM. Application to Handwriting Recognition," *Pattern Recogn,* vol. 42, pp. 3264--3270, 2009.

[15] V. V. O. Chapelle, and Y. Bengio, "Model selection for small sample regression," in *Machine Learning*, 2001.

[16] Y. Bengio, "Gradient-based optimization of hyperparameters," *Neural Computation,* vol. 12, no. 8, pp. 1889–1900, 2000.

[17] R. B. J. Bergstra, Y. Bengio and B. Kegl, "Algorithms for Hyper-Parameter Optimization," in *NIPS*, 2011.

[18] J. B. a. Y. Bengio, "Random search for hyperparameter optimization," *JMLR,* no. 13, pp. 281–305, 2012.

[19] M. Y. Kiang, "A comparative assessment of classification methods," *Decision Support Systems,* vol. 35, pp. 441– 454, 2002.

[20] A. R. Reza Entezari-Maleki, and Behrouz Minaei-Bidgoli, "Comparison of Classification Methods Based on the Type of Attributes and Sample Size," *Journal of Convergence Information Technology* vol. 4, no. 3, 2009.

[21] Wikipedia. (2015). *JConsole*. Available: https://en.wikipedia.org/wiki/JConsole

[22] A. F. a. A. Asuncion. (2010). *UCI*.

[23] Wikipedia. (2008). *Stuttgart Neural Network Simulator*. Available: https://en.wikipedia.org/wiki/SNNS

# APPENDIX

# VITA

Name: Hussein Azeez

Affiliation: Advanced Virtual and Intelligent Computing (AVIC) Center, Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University.

Country: Thailand

Biography: Mr Hussein Azeez was born on March 23, 1991, in Baghdad, Iraq.

He received a Bachelor's Degree in Computer Science and Information Technology from Al-Mansour University College.

Now he is a Master's degree student in Computer Science and Information Technology Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University.