SELECTING AN EFFICIENT CLASSIFIER FOR THE GIVEN DATASET

Prepared By

Riya S. Shah

11MCEC17

Guided by

Dr. Ketan Kotecha

Department of Computer Science & Engineering

Institute of Technology,

Nirma University

Ahmedabad-382481

May 2013

SELECTING AN EFFICIENT CLASSIFIER FOR THE GIVEN DATASET

Major Project - Part II

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Science & Engineering

By

Riya S. Shah

11MCEC17

Department of Computer Science & Engineering

Institute of Technology,

Nirma University

Ahmedabad-382481

May 2013

Certificate

This is to certify that the Major Project entitled **"Selecting an Efficient Classifier for the Given Dataset"** submitted by **Ms. Riya S. Shah (Roll No: 11MCEC17)**, towards the partial fulfillment of the requirement for the degree of Master of Technology in Computer Science & Engineering of Instutite of Technology, Nirma university, Ahmedabad is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The result embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

.....

Project Guide:
Dr. Ketan Kotecha
Director,
Institute of Technology,
Nirma University,
Ahmedabad.

Prof. Vijay Ukani
MTech(CSE) Coordinator,
Department of C.S.E.,
Institute of Technology,
Nirma University, Ahmedabad

Dr Sanjay Garg
HOD,
Department of C.S.E.,
Institute of Technology,
Nirma University, Ahmedabad

Acknowledgements

It is indeed a pleasure for me to express my sincere gratitude to those who have always helped me throughout my project work.

First of all I like to thank my internal project guide Dr. Ketan Kotecha (Director, Institute of Technology,Nirma University) for his keen interest, constant encouragement and valuable guidance at all stages of this dissertation work. I am sincerely thankful for his valuable guidance and help to enhance my presentation skills.

I am very much indebted to Dr. Sanjay Garg (HOD, Dept. of Computer Science & Engineering), Prof. Vijay Ukani (PG course coordinator, Dept. of Computer Science & Engineering) and Prof. Priyank Thakkar (Project Coordinator, Dept. of Computer Science & Engineering) for their valuable support, guidance and constant encouragement throughout my study.

Specially, I would like to thank Prof. Matthias Reif, Multimedia Analysis and Data Mining group(MADM) of German Research Center for Artificial Intelligence and Prof. Christophe Giraud-Carrier, Dept. of Computer Science at Brigham Young University, Provo, UT for their kind and warm support throught out the project work.

Last but not the least, I would like to thank God Almighty, My parents, My family members and friends for their love, support and excellent co-operation to build my moral during the work.

- Riya S. Shah

11MCEC17

Abstract

Over the past two decades, Machine Learning applications has spread to almost every fields and also the methods used for machine learning has converged from many sources. Current machine learning tools contain excessive set of these methods but are lacking in providing assistance to select a suitable classifier because of the variations in the development and the disciplines. The main goal of meta learning is to provide this assistance to the end user who may not be expert in the field of machine learning but has to deal with the algorithms for the classification process. Here, in this paper, the idea to design an assistant model using the meta learning techniques has been discussed. The survey of various meta learning techniques is done and have been explained in brief in this paper.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning is one of the most active research field for last two decades. As the applications of this field spread over almost every field, researchers are trying to develop the learning techniques for each application domain varying from classification to recognition. As a result, there are hundreds of methods with variations in different parameters and their application domains developed till now and hence if a user wants to use any applications which have learning technique embedded in it, the user ought to have the knowledge of that technique. The researchers have developed several tools specifically developed for machine learning techniques.

Machine learning tools are the softwares that incorporate various machine learning algorithms and provides an user interface which is easy-to-use. Machine Learning researchers have already proposed many different types of classification algorithms, including nearest neighbor methods, decision tree induction, error back-propagation, reinforcement learning, lazy learning, rule based learning, and relatively new addition is statistical learning. Hence, the user need to have the knowledge of these algorithms to use them properly for their dataset. As every dataset has a different characteristics, one particular algorithm which works well with one dataset, may not work efficiently with another dataset. Thus, the user need to have some expertise about the algo-

rithms as well as the purpose for which the dataset has been used. In general, the user may not have the specific knowledge about the algorithms or the classification process to which the user has to deal, which creates a barrier in using the machine learning tool efficiently by the end user. One possible solution to this problem would be by evaluating all possible candidate algorithms using a well known cross-validation (CV) procedure. All the candidate algorithms then can be reordered using the mean success rate and pick the first algorithm in the ordering. However, this trial-and-error method could be very cost effective as the number of algorithms available is very large.

The lack of these guidelines to select the right method according to the nature of the given dataset, leads to development of an assistance system which can help the end user in getting the better results. The primary focus of meta learning is to define or to learn the relationship between the classification method and the tasks of the domain (datasets and final class or prediction), which results in appropriate selection of a suitable classifier(predictive model) or combination of classifiers. The solution is to construct a meta learning system which provides automatic assistance to the user in selecting the classifier for the particular domain/dataset. The focus is to exploiting the cumulative knowledge of previous experiences in the current process [16].

The goal of this project is to develop a meta learning system which provides recommendations to the non-experts who wish to use the machine learning approaches for their pattern recognition applications. The report is organized as follows: First section gives a brief look on the machine learning field. Next section provides an overview to meta learning and how it works. It also explains existing meta learning systems. Subsequent section explains different approaches explored for meta learning process. Next two sections give the proposed work and the partly implementation results as well as the future work.

1.1 Background

1.1.1 Overview to Machine Learning

Machine learning can be defined as the computer program that improves its performance at some task through experience. It has been grew out of Artificial Intelligence. A Machine learns whenever it changes structure, program or data and improves its future performance with record of each change. Machine learning deals with the design of prediction algorithms or models that takes the input dataset and gives the output patterns using the features provided in the dataset.There are several reasons behind the need of machines which can learn. For example, an AI application, where the machine derives the relationship between the input-output pairs and takes the decision accordingly. Other applications of machine learning includes extracting the relationships and correlations hidden among large pile of data (mining), for job improvement of existing machine designs according to the environment in which they are working (adopting the changing environment) [25].

Types of Learning

In any learning method, there are three parameters: input data, function and output class. Now, in machine learning there are two phase: 1) training phase and 2) testing phase. During training phase, the input data, whose output class is known, is provided to the learning algorithm. The learning algorithm derives the relationship between the data and its output class. The function is the implementation of these relationships only.

Learning techniques can be categorized based on the way the function is implemented: 1) Supervised Learning 2) Unsupervised Learning 3) Semi-supervised Learning.Basically, they differ in the way in which the program need to learn. It can learn

with and/or without supervisor.

a. Supervised Learning:

   In this technique, the input to the learning technique is the training dataset from which the learner derives the relationships between the data and the output class. The values of function f for the input data X is derived from the m samples of X only. The output class of samples m is already given, the learner designs the function f from that. When the testing data Y is provided to f, it gives the output class of that data, which is matched with the original class. The accuracy of the learner is calculated based on the number of class predicted correctly and the number of class predicted incorrectly. Here the learning is done under the supervision of pre-determined classifications. Examples of supervised learning are concept learning, decision tree learning, bayesian learning, ANN, genetic algorithm, ant colony optimization, etc.). Few of this learning techniques have been explained below [25]:

b. Unsupervised Learning: In unsupervised learning, the machine simply receives inputs X but does not receive any supervised target outputs. Hence, here the learner develops a pattern from the dataset. Two widely used examples of unsupervised learning are clustering and dimensionality reduction. In this type of learning, the class labels or the output classes are unknown. So, if the dataset has been provided, then the function of the learner is to establish a group or cluster of the data, and hence the data has to be explored to find patterns in it.

c. semi-supervised learning: In many domains, there is a large amount of unlabeled data but limited labeled data, which can be expensive to generate, e.g. text processing, video-indexing, bioinformatics, etc. In semi-supervised learning, a learner learns from a combination of both labeled and unlabeled data. As seen earlier, that supervised learning requires large amount of training data to get the higher accuracy, while unsupervised learning methods are employed to

discover structure in unlabeled data. Semi-supervised learning takes both the techniques and gets the advantage out of it. Unlabeled data in semi-supervised learning helps to identify data structure and to make cluster assumption. Two important notations for this learning: 1) Local consistency: Nearby points are likely to have the same label. 2) Global consistency: Points on the same structure are likely to have the same label [26].

The goal of semi-supervised learning is to understand how combining labeled and unlabeled data may change the learning behavior. Semi-supervised learning is of great interest in machine learning and data mining because it can use readily available unlabeled data to improve supervised learning tasks when the labeled data are scarce or expensive. Semi-supervised learning also shows potential as a quantitative tool to understand human category learning, where most of the input is self-evidently unlabeled.

Here, some of the machine learning techniques have been explained, which are used for different domains of applications. [25] [26]

a. **Decision Trees:**

C4.5 algorithm of this decision tree, derives the results simply using divide-and-conquer method. Like, in numeric attributes, the possibilities are restricted to a binary split. When creating decision trees using the divide-and-conquer method, once the first attribute to split on has been selected, a top level tree node is created that splits on that attribute, and the algorithm proceeds recursively on each of the child nodes. For each numeric attribute, it appears that the subset of instances at each child node must be re-sorted according to that attributes values. This algorithm is enhanced using the concept of missing values. There are several ways to tackle this. One is to handle them as another attribute value, another is to ignore those dataset instances which con-

tain missing values. But a good solution is that the instances for which the values are missing, are split into pieces, one piece is provided to each branch. Another problem with decision tree is that the fully expanded decision trees can be too long and hence some of the part of the tree is pruned. This can be done using pre-pruning (forward pruning) and the post-pruning (backward pruning) techniques. The entropy is calculated using the following formula:

$$entropy(D) = -\sum_{j=1}^{|C|} Pr(c_j) log_2 Pr(c_j)$$

This entropy is used as the measure of impurity or disorder of dataset. The attribute with highest information gain is chosen for the split. The information gain is calculated using

$$gain(D, A_i) = entropy(D) - entropy_{A_i}(D)$$

b. **Bayesian Learning:**

In bayesian learning, explicit probabilities of the dataset is calculated. Each training example can incrementally increase/decrease the probability that a dataset instance is classified correctly. Hypothesis are created, weighted by their probabilities. Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured. The Naïve Bayes Theorem can find the probability of a certain class given mutually exclusive events. The Naïve Bayes classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and where the target function *f(x)* can take on any value from some finite set V. A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values $< x_1, x_2, , x_n >$. The learner is asked to predict the target value, or

classification, for this new instance.

c. **Clustering:**

A group of similar data is created, which is called cluster. There are different types of clustering algorithms based on the method using which the similarity of the data has been checked i.e. K-means clustering, hierarchical clustering. The k-means clustering algorithm partitions the dataset into k clusters. Each cluster has a center point called a centroid, from which the distance of the data instance is measured. Data instances which are nearer in distance are grouped together. For categorical data, centroid is represented by most frequent values and it is sensitive to outliers. Hierarchical clustering produces a nested sequence of clusters. At the beginning each data point is considered as a centroid of the cluster. Then, the clusters who have least distance between them are merged. This step is followed recursively until when the desired number of clusters is not achieved.

d. **Generative Models:**

Generative models are perhaps the oldest semi-supervised learning method. It assumes a model p(x, y) = p(y)p(x—y) where p(x—y) is an identifiable mixture distribution, for example Gaussian mixture models. With large amount of unlabeled data, the mixture components can be identified; then ideally we only need one labeled example per component to fully determine the mixture distribution. The mixture model ideally should be identifiable. If the mixture model assumption is correct, unlabeled data is guaranteed to improve accuracy. However if the model is wrong, unlabeled data may actually hurt accuracy. Even if the mixture model assumption is correct, in practice mixture components are identi?ed by the Expectation-Maximization (EM) algorithm.

e. **Self-Training:**

Self-training is a commonly used technique for semi-supervised learning. In self-training a classifier is first trained with the small amount of labeled data. The classifier is then used to classify the unlabeled data. Typically the most confident unlabeled points, together with their predicted labels, are added to the training set. The classifier is re-trained and the procedure repeated. Note the classifier uses its own predictions to teach itself. The procedure is also called self-teaching or bootstrapping. Self-training has been applied to several natural language processing tasks. Self-training has also been applied to parsing and machine translation. Self-training is a wrapper algorithm, and is hard to analyze in general.

f. **Co-training:**

Co-training assumes that (i) features can be split into two sets; (ii) each sub-feature set is sufficient to train a good classifier; (iii) the two sets are conditionally independent given the class. Initially two separate classifiers are trained with the labeled data, on the two sub-feature sets respectively. Each classifier then classifies the unlabeled data, and teaches the other classifier with the few unlabeled examples (and the predicted labels) they feel most confident. Each classifier is retrained with the additional training examples given by the other classifier, and the process repeats. In co-training, unlabeled data helps by reducing the version space size. In other words, the two classifiers (or hypotheses) must agree on the much larger unlabeled data as well as the labeled data.

g. **Artificial Neural Networks:**

Artificial neural networks are among the most powerful learning models. They have the versatility to approximate a wide range of complex functions representing multi-dimensional input-output maps. An Artificial Neural Network (ANN)

is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. It is composed of a large number of highly interconnected simple processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Artificial neural networks are represented by a set of nodes, often arranged in layers, and a set of weighted directed links connecting them. The nodes are equivalent to neurons, while the links denote synapses. The nodes are the information processing units and the links acts as communicating media.

h. **Genetic Algorithms:**

Genetic Algorithm is robust and probabilistic search algorithm based on the mechanics of natural selection and genetics. Genetic Algorithm follows the principle of Survival of the Fittest laid down by Charles Darwin. The basic components of genetic algorithms are chromosomes (encoding a solution), population (set of solutions), fitness function (quality of a solution), genetic operator (reproduction, crossover, mutation), and generations. Genetic algorithm starts with creating random population of the given genes. Then the fitness function evaluates fitness of each individual of the population. By selecting some of the best individuals from the population, they are operated with the genetic operators one by one. At last, the best chromosome is assigned as solution and the process is repeated.

i. **IBK:**

IBk is an implementation of the k-nearest-neighbors classifier that employs the distance metric. Here number of nearest neighbor can be specified manually. If more than one neighbor is selected, the predictions of the neighbors can be

weighted according to their distance to the test instance. The time taken to classify a test instance with nearest-neighbor classifier increases linearly with the number of training instances.

j. **OneR:**

One-attribute-Rule (OneR) is an algorithm for finding a rule, such that the minimum error attribute is chosen for prediction. A rule is created for each attribute and the one with the lowest error is chosen. A single node decision tree consisting of the chosen attribute as root is then built as a classification model. which produces simple rules based on one attribute only. It takes a single parameter: the minimum number of instances that must be covered by each rule that is generated (default value 6).

k. **ZeroR:**

ZeroR simply predicts the majority class in the training data if the class is categorical and the average class value if it is numeric.

l. **Decision Table:**

Decision Table produces a decision table using the wrapper method to find a good subset of attributes for inclusion in the table. This is done using a bestfirst search. Usually, a decision table assigns the majority class from the training data to a test instance if it does not match any entry in the table.

m. **Decision Stump:**

Decision Stump, builds binary decision stumpsone-level decision treesfor datasets with either a categorical or a numeric class. It copes with missing values by extending a third branch from the stump; in other words, by treating missing

as a separate attribute value.

n. **Random Forest:**

It is based on learning ensemble consisting of a bagging of un-pruned decision tree learners with a randomized slection of features at each split. Random forest applies randomness by selecting a random subset of the predictors for each split. It bulds a randomized decision tree in each iteration of the bagging algorithm, and often produces excellent predictors.

o. **Support Vector Machines:**

SVM is a supervised learning models which uses a non-probabilistic binary linear classifier. An SVM model can be viewed as a space where input examples are plotted according to their categories. It selects a small number of critical boundary instances called support vectors from each class and build a linear discriminant function that separates them as widely as possible.

p. **Ensemble Methods:**

This type of methods uses combination of models to increase accuracy. It combines a series of k learned models, M_1, M_2, ..., M_k, with the aim of creating an improved model M^*. Bagging and Boosting are popular ensemble methods. In bagging the final prediction is average of all the predictions made by each classifier from the collection of the classifiers. In Boosting, Each prediction is given a weight and final prediction is done through weighted vote. Detailed description of these methods are given in later section.

1.1.2 Machine Learning Tools

This section gives a brief introduction to some of the machine learning tools which are widely used. At the end of the section a comparison of several other machine learning tools are also given.

- WEKA (Waikato Environment for Knowledge Analysis) is a open source software in Java. It is having GPL license. [27] [28]It is a portable tool having a comprehensive collection of data processing and modeling techniques. It is divided into three interfaces:
  1) Explorer: main user interface providing the functionalities in GUI form.
  2) CLI: Provides all the functionalities from command line.
  3) Experimenter: provides comparison of the various machine learning algorithms provided in Weka.

- RapidMiner (previously known as YALE) supports machine learning, data mining, text mining, and analysis(predictive, business, etc.) [9] [17] It is written in Java and has used many algorithms of Weka.

- KNIME (Konstanz Information Miner) has integrated many machine learning algorithms with the mining algorithms for data analysis. [17]It has been used mainly in pharmaceutical research. It is also written in Java and also provides fragments calling for Pearl and Python.

The comparison of these three tools with other very well known tools are summarized in the following two tables. The Table I describes the comparative study of the tools based on the usability aspects in the terms of programming languages in which they are developed, human interaction (how much human intervention is needed to complete the process), interoperability (whether the system is internally operable or it supports PMML(the predictive modelling markup language)), and extensibility. Table II describes the comparison in terms of various known classifiers provided by the tools. First column 'pre-processing' rates the tool on pre-processing on the scale

of 0 to 3. Next few columns shows the availability of the classifier in the tool. Last two columns rates the tool on Data Visualization and Model Visualization on the scale of 0 to 3. [17]

Tool	**Language**	**Human Interaction**	**Interoperability**	**Extensibility**
Weka	Java	Manual	Self	Excellent
RapidMiner	Java	Manual	Self	Excellent
KNIME	Java	Manual	PMML	Excellent
ADAM	Python	Autonomous	Self	Simple
AlphaMiner	Java	Manual	PMML	Excellent
MLC++	C++	Guided	Self	Simple
Orange	C++ Python	Manual	Self	Excellent

Table 1.1: Comparison based on Usability Aspects [29]

Tool	**Pre-proc.**	**Bayes**	**DTree**	**NNet**	**SVM**	**Boosting**	**KMeans**	**Associations**	**Evaluation**	**Data Vis.**	**Model Vis.**
Weka	3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	3	3
RapidMiner	3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	3	3
KNIME	3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	3	3
ADAM	3	Yes	Yes	Yes	No	No	Yes	Yes	Yes	3	3
AlphaMiner	3	Yes	Yes	No	No	No	No	No	No	1	1
MLC++	3	Yes	Yes	Yes	No	No	No	No	Yes	3	3
Orange	3	Yes	Yes	No	Yes	Yes	No	Yes	Yes	3	3

Table 1.2: Comparison based on the Supported Classifiers [29]

Chapter 2

Literature Survey

2.1 Meta Learning

As we saw, there are numerous learning techniques developed for various domains and if a user wants to embed any technique in the application, he ought to learn the domain of that application. It may possible that the user does not have adequate knowledge of the technique and might choose the technique which is not suitable for the application for which he is working. As a solution to this, meta learning concept is developed under supervised learning with keeping a goal of providing an assistance in selecting the classifier based on performance. [16]

Meta learning can be defined as learning to learn. As from the No Free Lunch Theorem, there is no single learning technique which can be applied to all type of domains/datasets and can deliver the better results, as the process presents us with learning problems induces a non-uniform probability distribution over the possible functions. Cross-validation is regularly used as a mechanism to select among competing learning algorithms. Fig-01 shows a general CV method. Even, NFL holds for cross validation technique too and hence it can not be used for each domain, which

leads to other solution to make individual algorithm for each application, which is not viable. The researchers used trial-and-error method- that performs well on the target domains and others that are to be evaluated. But, as the number of algorithms have increased, this approach is not suitable. As the solution to this a new knowledge driven strategy, meta learning found. Meta learning assumes only a bias for learning the estimate of the function distribution. [18]

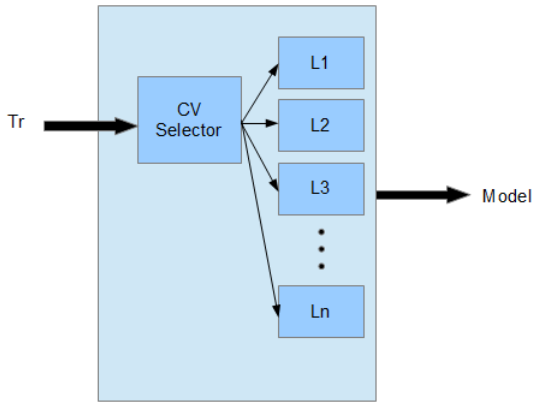

Figure 2.1: Cross Validation Meta Learner [17]

Given a new dataset, we wish to produce a recommendation of an algorithm from the set of candidate algorithms, that would be related to their actual performance on that dataset, without actually executing them. According to No Free Lunch theorem, it is not possible to determine one algorithms which performs well on all the types of datasets. And hence, in meta learning, the process of choosing the datasets from previously executed ones are done. This selection process is based on the relevance of the training datasets to the query dataset. It is expected that if two datasets are quite similar, so should be the corresponding performance of a given candidate algorithm. [1]

This process is done through training of the meta model. Now, the amount of training data available is really small for the meta learning process. The process of building the meta model is explained below.

2.2 Basic Architecture

The meta learning is a two phase process:

1) Knowledge Acquisition Phase

2) Advisory Phase

a. **Knowledge Acquisition Phase**

The input to the knowledge acquisition phase is the datasets or the samples with the known classifiers appropriate for them. The meta learner tries to set a relationship between datasets and the classifiers through a set of meta features. This process in detail is shown in the Fig. 02. [19]

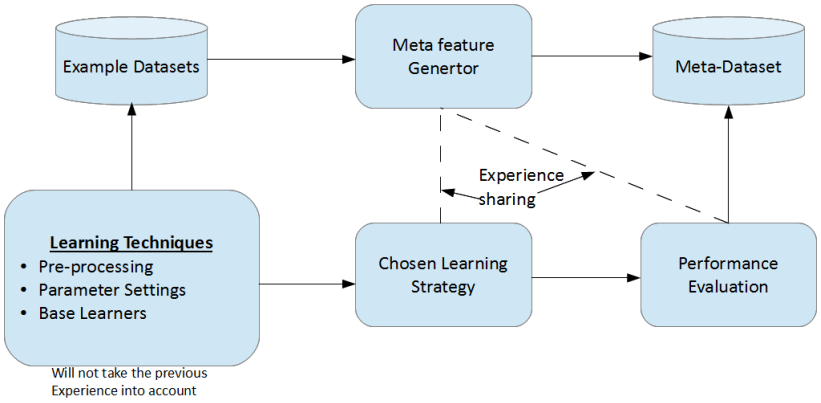

Figure 2.2: Knowledge Acquisition Phase

In knowledge acquisition phase, the example datasets are provided to the meta feature generator, which creates meta-dataset. Before using these example datasets for the feature generation, the data pre-processing can be done. Another module of this phase is basic learner, which applies the basic learning techniques to the example dataset. These learning techniques include data pre-processing, parameter settings and base learners. This is an independent module which does not take any previous experience into account. After apply-

ing the learning techniques to the example dataset, the user selects a learning strategy best known to him. The machine learning tool then evaluates the performance of the strategy on the given dataset and generates a report. The modules 'chosen learning strategy' and 'performance evaluation' pass their results to the meta feature which at the end exploit this knowledge to choose the classifier for another dataset. Finally, the results of 'meta feature generator' and 'performance evaluation' are combined and an instance of meta-dataset is generated.

During this phase, the meta learner does not exploit the knowledge of previous results. Hence, this phase can be seen as training phase of the meta learner. The next phase is advisory phase, which accumulates the current results with the known previous results.

b. **Advisory Phase**

The input to the advisory phase is also the new dataset. The meta learner now process on it to recommend the classifier. The modules of this phase is shown in Fig. 03.

Firstly, the 'meta feature generator' processes the arrived dataset as it does in advisory mode. This module processes on the dataset and generates the meta features, which is sent to the 'meta feature'. The inputs to the 'meta feature' come from the 'meat feature generator' and 'meta-dataset'. Both inputs are meta features, which is compared in this module. There are various interpretations developed for the matching process of the meta features. Using the classification problem, this module generates the list of best suitable classifiers for the given dataset. It may also provide the ranking to these classifiers according to their performance. This list is provided to the user using the

'recommendation' module.

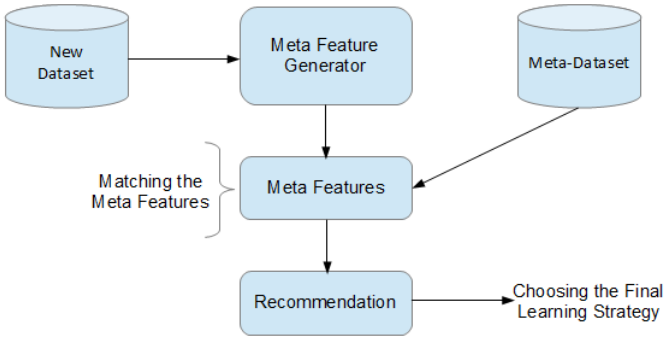

Figure 2.3: Advisory Phase

It has been observed that effectiveness of the meta learner increases as increase of the accumulation of meta knowledge in the classification process. Meta learning can be seen as the classification process of the classifiers themselves. As we are accumulating the meta knowledge with each new dataset, it is also called experience learning and the meta learner itself becomes an expert after a certain point.

2.2.1 Meta Feature Generation

As discussed earlier, the goal behind meta-learning is to learn about the learning algorithms themselves. The prediction of choosing the best algorithm for the given dataset is based on extraction of meta features- Features that describe the dataset itself. In other words, meta features are the characteristics of the datasets. These features are used to train the meta model on training data (characteristics of the datasets whose classifiers are known). Thereafter, this model is applied on the meta-features of a new dataset (whose classifier need to be learn). The model should be able to predict the appropriate classifier for the target dataset based on the performance of the previously known datasets and classifiers pairs. Meta features construct the feature space in which each dataset is represented as a point. The training dataset is

already plotted points in this feature space and this distribution is used to infer the knowledge about the new dataset.

Meta features are the measures which plays important role in computing the relationship among the classifiers and the datasets. A major focus of research has been on defining the proper meta feature for the datasets. These meta features should characterize the dataset in each and every aspect to keep comparison between the datasets as simple as possible. The number of meta features to describe the dataset should be optimized.

The development of meta features for meta learning should take the following issues into account:

- **Discriminative Power:**
  The set of meta features should contain information that distinguishes between the base-algorithms in terms of their performance. Therefore they should be carefully selected and represented in an adequate way.

- **Computational Complexity:**
  The meta features should not be too computationally complex. If this is not the case, the savings obtained by not executing all the candidate algorithms may not compensate for the cost of computing the measures used to characterize datasets.

- **Dimensionality:**
  The number of meta features should not be too large compared to the amount of available metadata; otherwise overfitting may occur.

Many different meta features has been defined by many researchers till now. Following are the techniques to define the meta features for any dataset:

a. **Direct Measures: (Statistical and Information Theoretic**

This is the simplest way to generate the meta features of the dataset. They are directly derived from the dataset. Here, the assumption made by the researchers is that learning algorithms can be separated from each other with respect to this dimensions offered by this information.

(I) Genotype of dataset: generates the measures based on statistical and informational propertiess of the dataset. These measures are directly visible in the dataset.

- Simple Characteristics
  i. No. of instances
  ii. No. of attributes
  iii. No. of classes
  iv. No. of binary attributes
- Statistical Characteristics
  i. Standard deviation ratio
  ii. Mean absolute correlation attributes
  iii. First canonical correlation
  iv. Mean skewness of the attributes
  v. Mean kurtosis of attributes

b. **Landmarking**

In this technique, the performance evaluation is done using a set of simple learners which are fast in execution. [5] [9] This method follows the concept that every dataset has certain characteristics which relates to the classifier which have expertise in that area. The simple learners are called landmarkers. Later, the

landmarkers or the combination of these landmarkers can estimate the performance of any given dataset. Here, the assumption made by the researchers is that While choosing the landmarkers following criteria should be satisfied:

(1) The learner should have minimal execution time, having less complex calculations.

(2) The learners chosen in the landmarkers set, should differ in the computational mechanism, parameter measurements, and bias.

The learners which have beenthe learning algorithms' space can be designed into areas of expertise and that the performances of the simple learners give an indication of the performance of more complex learning algorithms. The excessively used as landmarkers: Naive Bayes Learner, Linear Discriminant Learner, One Nearest Neighbor Learner, Decision Node Learner, Randomly Chosen Node Learner, Worst Node Learner, Average Node Learner [6].

c. **Model Based Features**

The model based approach creates a model from the data and uses its properties as feature values. The used model in this context is typically a decision tree as it is relatively simple to characterize a tree. In this technique, a decision tree is generated from the given dataset. [8] The measures of this decision tree is used as the features for meta learner. In this techniques, the assumption is made for the characteristics and that is the characteristics of the dataset should be strongly dependent on the task/domain.The following is the list of the measures generally derived from decision trees [7]:

- Nodes per attribute
- Nodes per instance
- Average leaf corroboration

- Average gain-ratio difference
- Maximum depth
- Number of repeated nodes
- Shape
- Homogeneity
- Imbalance
- Internal Symmetry

2.3 Meta Learning Tools and Applications

This section describes several tools developed for meta learning process. These tools are being used in the industry to provide the recommendations for the classifiers. Some tools are extensions to the machine learning tools and some are developed independently [20].

a. **PaREn: A RapidMiner Extension for Meta Learning**

The PaREn Automatic System Construction Wizard is a tool supporting meta learning in RapidMiner. This extension has been developed at DFKI (German Research Center for Artificial Intelligence) as part of the BMBF funded project. The user just need to provide the dataset, this PaREn extension automatically recommends and constructs a classification process based on certain characteristics of the dataset. The user need to provide the dataset in the RapidMiner environment. The PaREn wizard generates the meta features of the given dataset and gives the predicted accuracies of the classifiers. Execution time of this process depends on the size of the dataset. At the end of the

process, the wizard displays the list of classifiers with their ranking based on their accuracies and RMSE (root Mean Squared Error) value. The user can now select any of the classifier from the list. RapidMiner then starts it normal path for the data pre-processing for the selected classifier and the classification process. This extension uses a predefined set of classifiers for landmarking process which may limit its performance.

b. **WekaMetal: Weka Meta Learning Environment**

WekaMetal is a meta learning extension to the Weka. It provides assistance in selection of classification algorithms based on expected accuracy and time performance. Like PaREn, it also provides ranking to the classifiers. The training phase is performed on the datasets known as benchmark datasets and the ranker characterizes these datasets to generate the meta knowledge. Whenever a new dataset arrives, ranker compares the characteristics of the new dataset with the previously gained knowledge and creates the list of the classifiers predicting the performance for the given dataset. There are several limitations of this system in the ranking and timing. This extension needs more benchmarks for characterization and more waiting time is experienced. Caching of the characteristics also causes problems.

c. **Metal DM: DMA (Data Mining Assistant)**

It is prototype for web enabled assistance system, based on meta learning, that helps the user in selecting the classifier. It is developed by European ESPRIT Framework IV's METAL project. It uses a set of 10 classifiers containing C5.0 Trees, C5.0 Rules, C5.0 with boosting, Naive Bayes, IB1, MLP, RBF Network, Ripper, Linear Discriminant and LTree. The meta learner algorithm is k-NN

with k=3. The recommendation takes the form of the 10 algorithms in L, obtained by aggregating the ranking of the query task's nearest neighbors.

d. **IDEA: Intelligent Discovery Electronic Assistant**

An IDA interacts with the user to obtain dataset and the desired goal. Then it generates the set of features based on all valid KD (Knowledge Discovery) processes according to the user's requirements. This involves the pre-processing and post-processing modules and the induction algorithms. Next, the IDA will rank suitable processes into a suggested order based on the desired results. The IDA basically works on machine learning operators which is divided into three categories:

1) **Pre-processing:** This includes processes like categorical attribute transformation (e.g. categorical to binary, dual scaling etc.), Continuous attribute transformation (e.g. class-based discretization, fixed-bin discretization, principle component analysis), record sampling(e.g. progressive sampling, random sampling, stratified sampling) and selecting features (e.g. sequential forward selection, correlation based selection).

2) **Induction Algorithm:** This include classifier (e.g. Decision trees, rule learner), CPE (e.g. Naive Bayes, logistic regression) and regressor (e.g. Neural network, linear regression).

3) **Post-processing:** This includes pruning (e.g. rule-set pruning, tree pruning), thresholding (e.g. CPE thresholding, regression thresholding) and logical model transformation (e.g. decision trees to rules).

Main advantage of IDAs is that they take user requirements into account while ranking the algorithms. Still it is lacking in composition of the algorithms.

e. **Consultant and Selecting Classification Algorithms**

The European ESPRIT research project MLT was one of the first formal attempts at addressing the practice of machine learning. To facilitate such practice, MLT produced a rich toolbox consisting of a number of symbolic learning algorithms for classification, datasets, and standards. Considerable insight into many important machine learning issues was gained during the project, much of which was translated into rules that form the basis of Consultant-2, the user guidance system of MLT.

Consultant-2 is a kind of expert system for algorithm selection. It functions by means of interactive question-answer sessions with the user. Its questions are intended to elicit information about the data, the domain and user preferences. Answers provided by the user then serve to fire applicable rules that lead to either additional questions or, eventually, a classification algorithm recommendation. Several extensions to Consultant-2, including user guidance in data preprocessing, were suggested and reflected in the specification of a next version called Consultant-3. To the best of our knowledge, however, Consultant-3 has never been implemented.

Although its knowledge base is built through expert-driven knowledge engineering rather than via metalearning, Consultant-2 stands out as the first automatic tool that systematically relates application and data characteristics to classification learning algorithms.

f. **MiningMart and Preprocessing**

MiningMart, another large European research project , focused its attention on

algorithm selection for preprocessing rather than for model building. Preprocessing generally consists of nontrivial sequences of operations or data transformations, and is widely recognized as the most time consuming part of the KDD process, accounting for up to 80% of the overall effort. Hence, automatic guidance in this area can indeed greatly benefit users. The goal of MiningMart is to enable the reuse of successful preprocessing phases across applications through case-based reasoning. A model for metadata, called M4, is used to capture information about both data and operator chains through a user-friendly computer interface. The complete description of a preprocessing phase in M4 makes up a case, which can be added to MiningMarts case base.

Given a new mining task, the user may search through MiningMarts case base for the case that seems most appropriate for the task at hand. M4 supports a kind of business level, at which connections between cases and business goals may be established. Its more informal descriptions are intended to help decision makers to find a case tailored for their specific domain and problem. Once a useful case has been located, its conceptual data can be downloaded. The local version of the system then generates preprocessing steps that can be executed automatically for the current task. MiningMarts case base is publicly available on the Internet.

2.4 Approaches

2.4.1 Common Terms and Evaluation Parameters in Meta Learning

Confusion Matrix

A confusion matrix contains information about actual and predicted classifications done by different classifiers. Performance of such classifiers is commonly evaluated using the data in the matrix. The confusion matrix is shown below:

	Classified Positive	**Classified Negative**
Actual Positive	TP	FN
Actual Negative	FP	TN

Table 2.1: Confusion Matrix

where,
TP: the number of correct predictions of the positive examples (True Positive), FN: the number of incorrect predictions of the negative examples (False Negative), FP: the number of incorrect predictions of the positive examples (False Positive), TN: the number of correct predictions of the negative examples (True Negative).

Precision

Precision p is the number of correctly classified positive examples divided by the total number of examples that are classified as positive.

$$p = \frac{TP}{TP + FP}$$

Recall

Recall r is the number of correctly classified positive examples divided by the total number of actual positive examples in the test set.

$$r = \frac{TP}{TP + FN}$$

F_1 value or F_1 score

It combines precision and recall into one measure. It is harmonic mean of precision and recall.

$$F_1 = \frac{2pr}{p + r}$$

$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{q}}$$

Accuracy

The accuracy (AC) is the proportion of the total number of predictions that were correct. Accuracy is related to the degree of bias in the measurements. Accuracy is used to represent the correct answer or percentage of accurate classification.

2.4.2 Meta Learning Approaches

Stacked Generalization

Originally proposed by Wolpert, this is the common approach to meta learning. Here, a set of q base learning algorithms are applied to a training dataset T_{train}

: $\{(\tilde{X}_i, c_i)\}_{i=1}^{m}$ to produce q hypothesis, $\{h_j\}_{q}^{j=1}$, also called level-0 generalization. Meta learning takes place when training set T_{train} is redefined into a new set T'_{train}. The redefinition replaces each vector $\tilde{X}_i$ with the class predicted by each of the q hypothesis on $\tilde{X}_i$:

$$T'_{train} = \{(\tilde{X}_i, c_i)\} = \{((h_1(\tilde{X}_i), h_2(\tilde{X}_i), ..., h_q(\tilde{X}_i)), c_i)\}$$

The new training set T'_{train} serves as input to a set of meta learners, which produce a new set of hypothesis or level-1 generalization. The redefinition of T_{train} into T'_{train} is done via k-fold cross validation. Stacked Generalization is considered a form of meta learning because the transformation of the training set conveys information about the predictions of the base learners (i.e. conveys meta knowledge). But it has severe limitation in that both base learning algorithms and meta learning algorithms have a fixed form of bias, i.e. no dynamic selection of bias. The dominant region of the meta learning algorithm may be different from the base learning algorithm, but ultimately fixed.

Research in stacked generalization paradigm investigates what base learning algorithms and meta learning algorithms produce best empirical results. After transforming the original training dataset, each example contains the predictions of the base learning algorithms, but it may also contain the original features.

Sampling Landmarks

This method, also known as Significant wins(SW), relies on relatively fast pairwise comparisons involving two algorithms. This method exploits performance information of sampling landmarks representing accuracy estimates on simplified versions of the data. A series of sampling landmarks represents in effect a partial learning curve. [2] The characterization involves conducting experiments on new dataset. The

plan of these experiments is built up gradually, by taking into account the results of all previous experiments. This whole process is divided into two steps, described as follows:

Step-01: Determining the Partial Order of Classification algorithms

a. Pick a pair of classification algorithms and determine which one of the two is better.

b. Repeat this for all pairs of algorithms.

c. Process the partial order obtained in order to identify the best elements.

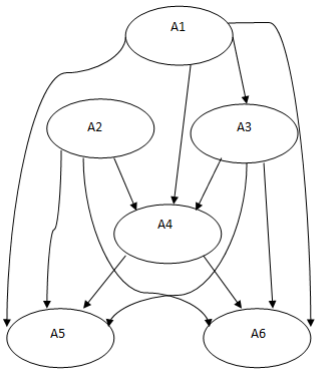

Figure 2.4: Example of partial order of 6 classification algorithms [2]

The aim is to analyze the partially ordered set of items and identify the top most elements. Consider an example which involves three elements as A1, A2, and A3. After completing first two process of this step, there are three possible outcomes:

In case 1, it is visible that A1 is the best algorithm, where in case 2, A1 and A2 are indistinguishable. In case 3, more consideration for placing the elements are needed which includes following strategies:

a. Considering losses, L.

b. Considering wins, W.

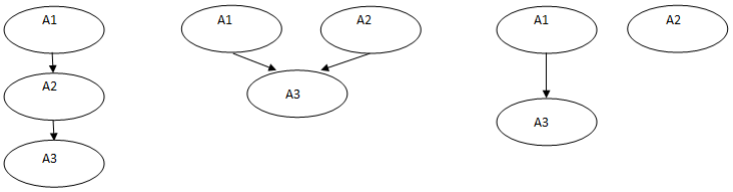

Figure 2.5: Several possible outcomes [2]

c. Considering both, LW.

Step-02: Predicting the Relative Performance of a Pair of Algorithms This step relies on two kinds of information:

- Information about various datasets used in the past. (i) Data characteristics (ii) accuracies of the two classification algorithms on samples of data from these datasets. This information is known as metaknowledge.

- Information about the new dataset. This is again of two kinds as described in above point.

This method now builds a ranking on the basis of results of pairwise hypothesis tests concerning the performance of pairs of algorithms. Evaluation starts by testing the significance of the differences in performance between each pair of algorithms. This is done for all datasets. It is denoted as:
Algorithm j is significantly better than algorithm k on dataset i as $\mathrm{E}R_j^i << \mathrm{ER}_k^i$. Then, a win table is constructed for each of the dataset as follows. The value of each cell, $W_{j,k}^i$, indicates whether algorithm j wins over algorithm k on dataset i at a given significance level and is determined in the following way:

$$W_{j,k}^i = \begin{cases} 1 & \text{iff } ER_j^i << ER_k^i \\ -1 & \text{iff } ER_k^i << ER_j^i \\ 0 & \text{otherwise} \end{cases}$$

Note that, $W^i_{j,k}$= - $W^i_{j,k}$ by definition. Next, the pairwise estimate of the probability of winning for each pair of algorithms, is calculated which is denoted by $pw_{j,k}$. This is calculated by dividing the number of datasets where algorithm j is significantly better than algorithm k by the number of datasets, n. This value estimates the probability that algorithm j is significantly better than algorithm k.

Zoomed Ranking [13]

This technique employs the k-nearest neighbor algorithm with distance function based on a set of statistical, information theoretic, and other dataset characterization measures to identify datasets that are similar to the one at hand. The first phase is to define these features. In the second phase, a ranking is constructed on the basis of the performance information of the candidate algorithms on the selected datasets. Authors have presented adjusted ratio of ratios method. This method processes performance information on accuracy and total execution time. It is based on the ratio of success rate ratio and an adjusted time ratio:

$$ARR^{d_i}_{a_p,a_q} = \frac{\frac{SR^{d_i}_{a_p}}{SR^{d_i}_{a_q}}}{1 + \frac{log(\frac{T^{d_i}_{a_p}}{T^{d_i}_{a_q}})}{K_T}}$$

where $SR^{d_i}_{a_p}$ and $T^{d_i}_{a_p}$ are the success rate and time of algorithm a_p on dataset d_i respectively, and K^l_t is a user defined value that determines the relative, importance of time. The ratio of success rates, $SR^{d_i}_{a_p}/SR^{d_i}_{a_q}$ can be seen as a measure of advantage, and the ratio of times, $T^{d_i}_{a_p}/T^{d_i}_{a_q}$, as a ,easure of the disadvantage of algorithm a_p relative to algorithm a_q on dataset d_i. The former can be considered a benefit while the latter a cost. Thus, by dividing a measure of benefit by a measure of the cost, the overall quality of the algorithm can be assessed.

Instance Based Learning(IBL) :

In meta learning the amount of data available is usually small. Hence the task including models that are general is hard, especially with algorithms that generate crisp thresholds, like decision trees and rule learning. IBL has advantage that the system is extensible, once a new experimental result becomes available, it can be easily integrated into the existing results without the need to reinitiate complex re-learning. The features that are used for this approach are explained below:

- The number of examples: it discriminates algorithms according to how scalable they are with respect to this measure.
- The proportion of symbolic attributes is indicative of the aptitude or inadequacy of the algorithm to deal with symbolic or numeric attributes.
- The proportion of missing values (prop.missing.values) discriminates algorithms according to how robust they are with respect to incomplete data. This measure was later eliminated, as explained below.
- The proportion of attributes with outliers (prop.attr.outliers) discriminates algorithms according to how robust they are to outlying values in numeric attributes, which are possibly due to noise.2 An attribute is considered to have outliers if the ratio of the variances of mean value and the -trimmed mean is smaller than 0.7. We have used $\alpha = 0.05$.
- The entropy of classes (class.entropy) combines information about the number of classes and their frequency, measuring one aspect of problem difficulty.
- The average mutual information of class and attributes (mut.info) indicates the amount of useful information contained in the symbolic attributes.
- The canonical correlation of the most discriminating single linear combination of numeric attributes and the class distribution (can.cor) indicates the amount of useful information contained in groups of numeric attributes.

Next, a visual analysis of this set of measures is performed with the aim of identifying measures that seem to provide little useful information. This is done by analyzing the correlation between values of a particular meta-attribute chosen and the performance of each algorithm. For each meta-attribute and algorithm pair, the values of the given meta-attribute and the ranks of the algorithm for all the datasets considered are plotted. For example, a plot for C4.5 algorithm is shown in the figure.

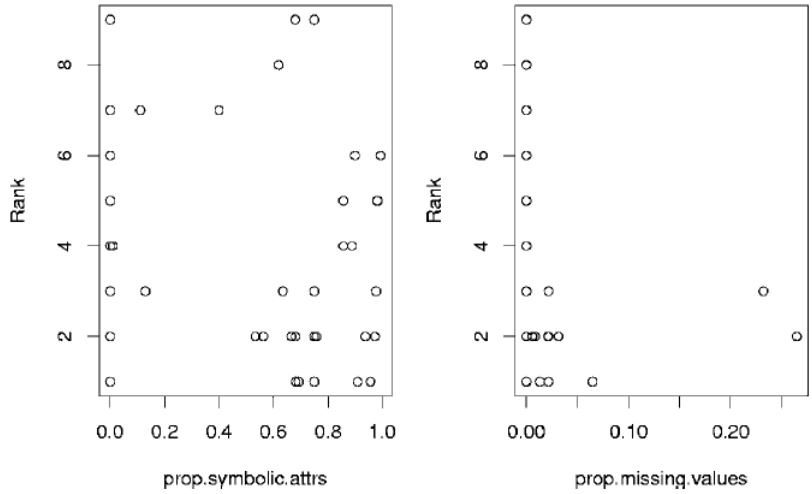

Figure 2.6: Plots of two meta attributes values against the rank of C4.5 for 53 datasets [1]

The distance function used here is the unweighted L_1 norm.

$$dist(d_i, d_j) = \sum \frac{|v_x, d_i - v_x, d_j|}{max(v_x, d_k) - min(v_x, d_k)}$$

where d_i and d_j are datasets, and v_x,d_i is the value of meta-attribute x for dataset d_i . The distance is divided by the range to normalize the values, so that all meta-attributes have the same range of values.

It may be the case that a meta-attribute is not applicable to a particular dataset. If dataset d_i has no numeric attributes, then it makes no sense to calculate, for instance, the canonical correlation meta-attribute. For such an attribute, dataset d_j

can be considered close to dataset d_i if it also does not have any numeric attributes and their distance is 0. On the other hand, if dataset d_j does have numeric attributes, the dataset is considered to be different from d_i in this respect. So, the distance is the maximum possible, i.e., 1, after normalization.

Meta Learning using Regression:

One of the objectives of the METAL project was to extend the approach to consider regression problems in addition to classification problems. As soon as we consider a new problem domain, however, the most important question becomes: What are the suitable features that can adequately characterize a dataset? While the StatLog features are suitable for classification problems (e.g., measuring the distributions between classes, entropy of classes, etc.), they need to be adapted for regression problems that model a relationship between inputs and a continuous single output or target variable. These included coefficient of variation, sparcity, and stationarity of the target variable; presence of outliers in the target; R^2 coefficient of a linear regression model; and average absolute correlations between the variables themselves, and between the variables and target. These measures were in addition to the suitable StatLog measures.

Active Meta Learning:

Machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns. An active learner may pose queries, usually in the form of unlabeled data instances to be labeled by an oracle (e.g., a human annotator).Active learning is well-motivated in many modern machine learning problems, where unlabeled data may be abundant or easily obtained, but labels are difficult, time-consuming, or expensive to obtain. In the traditional supervised learning, the learner is trained by taking input as a set of randomly chosen

training examples. This approach is referred in the literatures passive learning, and it has been broadly applied in several applications. However, both empirical and theoretical studies have shown that the learning performance can be improved when the learner is allowed to ask questions to the teacher, i.e. to decide which data will be included in the training set.

Active Leaning is a paradigm of Machine Learning in which the learning algorithm has some control over the inputs on which it trains. The main objective of this paradigm is to reduce the number of training examples, at same time maintaining, or even improving, the performance of the learning algorithm. Active Learning is ideal for learning domains in which the acquisition of labeled examples is a costly process, such as image recognition, text classification and information filtering. [41]

The cost of acquiring labels for Meta Learning is computationally expensive as it is necessary to execute candidate algorithms on the datasets used for training. This makes Meta learning a good candidate problem for active learning techniques. So here evaluation of the candidate algorithm is performed only in a selected subset of the available datasets. Figure shows active learning process.

Active Learning starts with a small set of one or more labeled examples and a large set of unlabeled ones. Labeled examples in the context of meta-learning are generated from datasets for which the candidate algorithms were already evaluated. The unlabeled examples in turn correspond to datasets which are only candidates for metaexample generation. An active learning module receives these two sets as input and selects, from the later, the next example to be labeled. The selection of unlabeled meta-examples is performed based on pre-defined criteria which take into account the meta-features of the problems and current set of labeled examples. Labeling is done by evaluating the candidate algorithms on the selected problems and the best algorithm on each of them becomes the label of the corresponding meta-example.

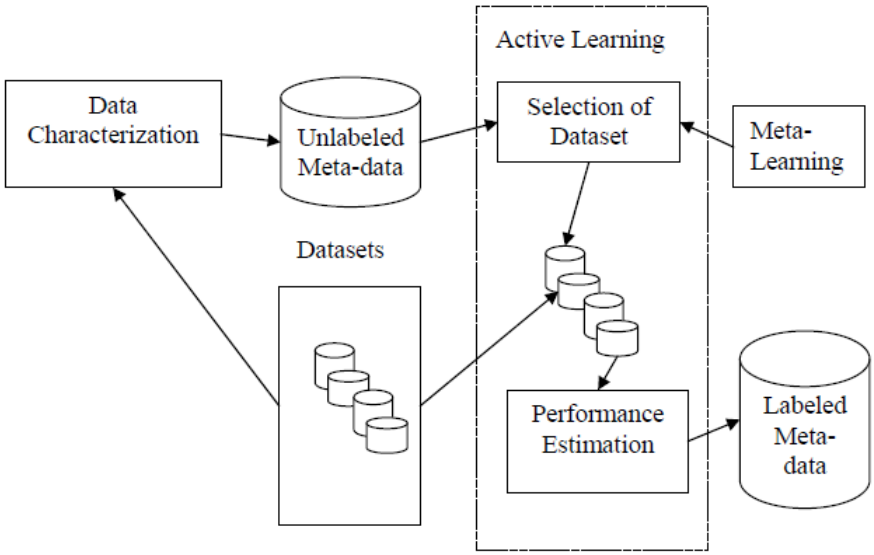

Figure 2.7: Active Learning Process [22]

Other Ranking Methods

Average Ranks Ranking Method [3]:

This is a simple ranking method, inspired by Friedman's M Statistics (explained later). For each dataset the algorithms are ordered according to the measured error rates and ranks are assigned accordingly. The best algorithm will be assigned rank 1, the runner-up 2, and so-on. Let r_j^i be the rank of algorithm j on dataset i. The avarage rank of each algorithm is $\bar{r}_j = (\sum_i r_j^i)/n$, where n is the number of datasets. The final ranking is obtained by ordering the average ranks and assigning ranks to the algorithms accordingly.

Success Rate Ratios Ranking Method [3]:

This method employs ratios of success rates between pairs of algorithms. This method starts by creating a success rate table for each of the datasets. Each slot of this table is filled with $SRR_{j,k}^i = (1 - ER_j^i)/(1 - ER_k^i)$, where ER_j^i is the measured error rate of algorithm j on dataset i. Next, a pairwise mean success rate ratio is calculated, $SRR_{j,k} = (\sum_i SRR_{j,k}^i)/n$, for each pair of algorithms j and k, where n is the number

of datasets. This is an estimate of the general advantage/disadvantage of algorithm j over algorithm k. Finally, the overall mean success rate ratio for each algorithm, $SRR_j = (\sum_k SRR_{j,k})/(m-1)$, where m is the number of algorithms. The ranking is derived directly from this measure.

2.4.3 Statistical Methods

This section is based on simple statistics. As here, we need to compare the statistics derived from the datasets. Here, the comparison between two datasamples from two different populations are compared. If two independent random variables y_1 and y_2 are normally distributed with means and variances (μ_1, σ_1^2) and (μ_2, σ_2^2), respectively, the difference between the random variables is normally distributed with mean equal to $(\mu_1 - \mu_2)$ and variance equal to $(\sigma_1^2 + \sigma_2^2)$. Similarly, the sum $(y_1 + y_2)$ of the random variables is also normally distributed with mean $(\mu_1 + \mu_2)$ and variance $(\sigma_1^2 + \sigma_2^2)$.

Wilcoxon Signed Ranks Test: Nonparametric Analysis for Two Populations

Steps:

a. For each item in a sample of n items, compute a difference score, D_i between the two paired values.

b. Neglect the + and - signs and list the set of n absolute differences, $|D_i|$.

c. Omit any absolute difference score of zero from further analysis, thereby yielding a set of n' nonzero absolute difference scores, where n'$\leq$ n. After you remove values with absolute difference scores of zero, n' becomes the actual sample size.

d. Assign ranks R_i from 1 to n' to each of the $|D_i|$ such that the smallest absolute difference score gets rank 1 and the largest gets rank n'. If two or more $|D_i|$ are equal, assign each of them the mean of the ranks they would have been assigned individually had ties in the data not occurred.

e. Reassign the symbol + or - to each of the n' ranks, R_i, depending on whether D_i, was originally positive or negative.

f. Compute the Wilcoxon test statistics, W, as the sum of the positive ranks as shown below.

$$W = \sum_{i=1}^{n'} R_i{}^{(+)}$$

Chapter 3

Proposed Work

Two approaches are proposed in this project. The first approach is based on the meta learning as well as some statistical concepts. The second proposal is combination of ensemble methods, reinforcement learning and meta learning. In the following section these three concepts are explained. In the subsequent sections, the proposed algorithm and the partial implementation of the first proposed work is given. Following section is concluded with future work.

3.1 Background

Ensemble Methods [23]

Ensembles of models are sets of (simple) models whose outputs are combined, for instance with majority voting, into a single output or prediction. As it is often the case thatpredictive accuracy of ensembles is better than that of their constituent (base) models, they are quite popular in the machine learning field. The ensemble method depends on two steps. The first is to generate the base models and the second step is to combine these models to get the final prediction results. The majority of ensemble

research has focused on methods from the first group, i. e., methods that use different learning data sets. Such data sets can be obtained by resampling techniques such as bootstrapping where learning sets are drawn randomly with replacement from the initial learning data set. Once a sufficiently diverse set of base models is generated, they are combined so that a single prediction can be obtained from the ensemble. Generally, it is done by two methods: Model Selection or Model Fusion. In model selection, performance of the all base models is evaluated, and simply the predictions of the best one is chosen. In model fusion, predictions of all the base learners are combined to get the final result. Some of the very popular ensemble methods, which are going to be used in this proposed work are discussed below:

Voting

Strictly speaking, voting is not an ensemble method, but a method for combining base models, i. e., it is not concerned with the generation of the base models. Still, it is included in this selection of ensemble methods because it can be used for combining models regardless of how these models have been constructed. As mentioned before, voting combines the predictions of base models according to a static voting scheme, which does not depend on the learning data or on the base models. It corresponds to taking a linear combination of the models. The simplest type of voting is the plurality vote (also called majority vote), where each base model casts a vote for its prediction. The prediction that collects most votes is the final prediction of the ensemble. If we are predicting a numeric value, the ensemble prediction is the average of the predictions of the base models.

A more general voting scheme is weighted voting, where different base models can have different influence on the final prediction. Assuming we have some information

on the quality of the base models predictions (provided by the models themselves or through some background knowledge), we can put more weight on the predictions coming from more trustworthy models. Weighted voting predicting nominal values simply means that vote of each base model is multiplied by its weight and the value with the most weighted votes becomes the final prediction of the ensemble. For predicting numeric values we use a weighted average. If d_i and w_i are the prediction of the ith model and its weight, the final prediction is calculated as $Y = \sum_{i=1}^{b} w_i d_i$.

Bagging

Bagging (short for bootstrap aggregation) is a voting method where base models are learned on different variants of the learning data set which are generated with bootstrapping (bootstrap sampling). Bootstrapping is a technique for sampling with replacement; from the initial learning data set we randomly select examples for a new learning (sub)set, where each example can be selected more than once. If we generate a set with the same number of examples as the original learning set, the new one will on average contain only 63.2% different examples from the original set, while the remaining 36.8% will be multiple copies.

Using these sampled sets, a collection of base models is learned and their predictions are combined by simple majority voting. Such an ensemble often gives better results than its individual base models because it combines the advantages of individual models. Bagging has to be used together with an unstable learning algorithm (e. g., decision trees or neural networks), where small changes in the learning set result in largely different classifiers. Another benefit of the sampling technique is that it is less likely that (many) outliers in the learning set show up also in the bootstrap sample. As a result, base models and the ensemble as a whole should be less sensitive to data outliers.

Boosting

Boosting comprises a whole family of similar methods that, just as bagging, use voting to combine the predictions of base models learned by a single learning algorithm. The difference between the two approaches is that in bagging the complementarity of the constructed base models is left to chance, while in boosting we try to generate complementary base models by learning subsequent models, taking into account the mistakes of previous models. The procedure starts by learning the first base model on the entire learning set with equally weighted examples. For the next base models, we want them to correctly predict the examples that have not been correctly predicted by previous base models. Therefore, we increase the weights of these examples (or decrease the weights of the correctly predicted examples) and learn a new base model.We stop learning new base models when some stopping criterion is satisfied (like when the accuracy of the new base model is less then or equal to 0.5). The prediction of the ensemble is obtained by weighted voting, where more weight is given to more accurate base models; the weights of all classifiers that vote for a specific class are summed and the class with the highest total vote is predicted.

Stacking

Stacking or stacked generalization is a method for combining heterogeneous base models, i. e., models learned with different learning algorithms such as the nearest neighbor method, decision trees, naive Bayes, etc. Base models are not combined with a fixed scheme such as voting, but rather an additional model called meta (or level 1) model is learned and used for combining base (or level 0) models. The procedure has two steps. First, we generate the meta learning data set using the predictions of the base models. Second, using the meta learning set we learn the meta model which can combine predictions of base models into a final prediction.

3.2 Proposed Design

Here, in this project, we are proposing a meta learning approach with hierarchical training to choose the good classification algorithm for the given dataset. For this a meta dataset is designed, consisting of 26 meta features of 85 datasets. Then, for testing, we have chosen 45 datasets from random sources(mainly from USI machine learning repository).

Meta Dataset Generation

The datasets chosen for the training purpose, go through the meta dataset generator. This module(java based) generates the meta dataset by calculating the meta features. Here, we have used 26 meta features combining the direct measures, information theoretic measures and statistical measures. The set of meta features used in this project are listen in Appendix-A. The meta feature generator module calculates all these measures for each 85 datasets and puts them as an instance in the metadataset. After calculating the meta dataset, to assign the class to each data instances of this meta dataset, a module is designed which builds the classification models of each classification algorithm selected in the algorithms' pool. Results are compared of all built models and a best classification algorithm is chosen based on the values of accuracy and RMSE(Root mean squared error). (High in accuracy, low RMSE). This phase returns a complete meta dataset which is used for the training purpose.

Training Phase

The Neural Network algorithm given in the Weka Machine Learning Tool with default parameter settings, is trained using this meta dataset.Here, we have 16 classification algorithms hence 16 number of classes and the number of instances are only 85. If we directly apply the classifier's name as class, then the accuracy of this model

is very less(18%). As in the classifier's pool, there are 16 classification algorithms, comprising of Neural network (MLP), J48, Decision Stump, Random Forest, Random Tree, One-R, Zero-R, PART, Conjuctive Rule, Decision Table, SVM, Bayes Network, Naive Bayes, KStar, LWL, and IBk. The neural network is not getting the enough data for the training purpose.

Hence, here, we have followed hierarchical approach in training phase. Note that here, class is the name of the classification algorithm itself which is well suited for the particular dataset. These classification algorithms are categorized into five catogories which are neural based learning algorithms, rule based learning algorithms, tree based learning algorithms, statistics based learning algorithms, and Instance based learning algorithms. Following is the list of categorized algorithms:

- **Neural based learning algorithms:** Multi-layer Perceptron(MLP)
- **Rule based learning algorithms:** One-R, Zero-R, PART, Conjuctive Rule, Decision Table
- textbfTree based learning algorithms: J48, Decision Stump, Random Forest, Random Tree
- **Statistics based learning algorithms:** Support Vector Machine (SVM), Bayes Network, Naive Bayes
- **Instance based learning algorithms:** KStar, LWL, IBk

In the first phase, a NN is trained using the original meta dataset where the class attribute has the 5 values, Neural, Rule, Tree, Statistics, and Instance. Using cross validation, we are getting the accuracy of this classification around 77%. When the new dataset comes (whose well suited classification algorithm is to be determined), it goes to meta feature generator first, where 26 meta features of the dataset is calculated. This 26 meta features make an instance and it is presented to the trained model, which predicts the algorithms' category for the given dataset. After comple-

tion of this first phase, we will be having the instance for the new dataset and its class i.e. algorithm's category which is well suited for the dataset.

Now, as the category is determined, all the instances from the training meta dataset containing the same category of the class is extracted, which forms another dataset which is subset of the original meta dataset. Now in this subdataset, the class attribute is replaced with the classification algorithms' name, which is already determined in the first stage. Another NN is trained and the test instance is presented to this trained model which now determines the classification algorithms' name which is well suited for the dataset.

3.3 Experimental Results

The following table shows some accuracy results of 5 algorithms over 17 datasets.

The following graphs show the average performance of each classification algorithm on the 85 training datasets.

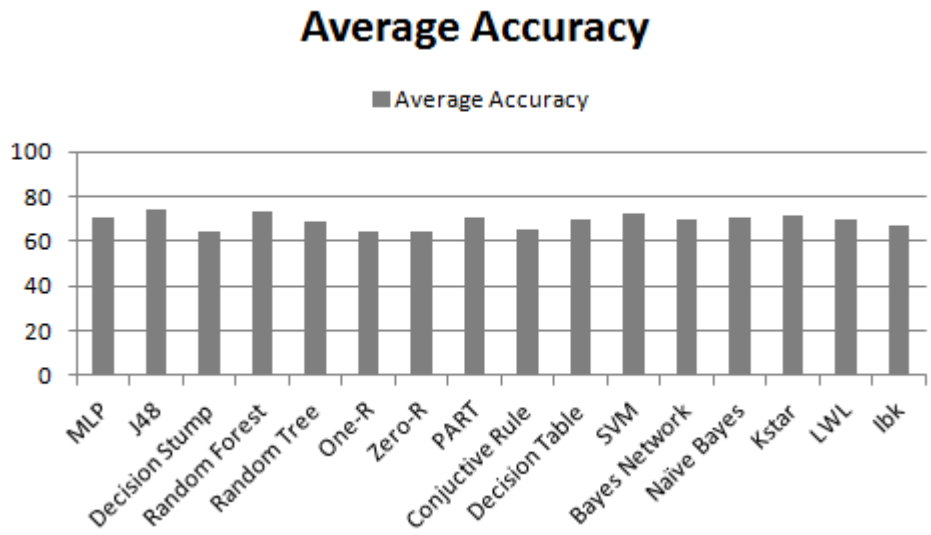

Figure 3.1: Average Performance (Accuracy)

Dataset	Base Learners				
	BN	J48	NN	IBK	KS
Glass	46.38	63.81	61.96	67.33	50.38
Breast	95.8	94.38	96.03	95.74	96.14
Diabetes	75.13	70.53	75.42	70.74	77.68
Spirals	47.79	42.84	48.56	79.47	49.05
Segment	80.01	94.51	95.7	96.94	88.35
Wine	97.65	92.35	97.65	96.12	98.71
Heart	83.04	73.19	82.01	77.26	78.96
Ecoli	80.18	82.36	84.18	79.58	85.88
Flags	43.79	54.93	75.38	71.69	67.69
Balance	90.35	41.35	93.3	78	86.25
Ionosphere	89.37	92.46	89.71	86.8	84.17
Iris	55.37	43.16	54.52	39.26	54.21
Lenses	87.87	89.49	98.44	97.36	94.85
Mushroom	22.05	70.04	43.21	60.01	22.22
Ozone	82.07	90.55	91.25	91.16	90.26
Soybean	76.52	72.14	79.63	82.79	80.19
Yeasy	85.69	93.59	99.37	94.6	93.79

Table 3.1: Results: Accuracies of 5 Base Learners over 17 selected Datasets

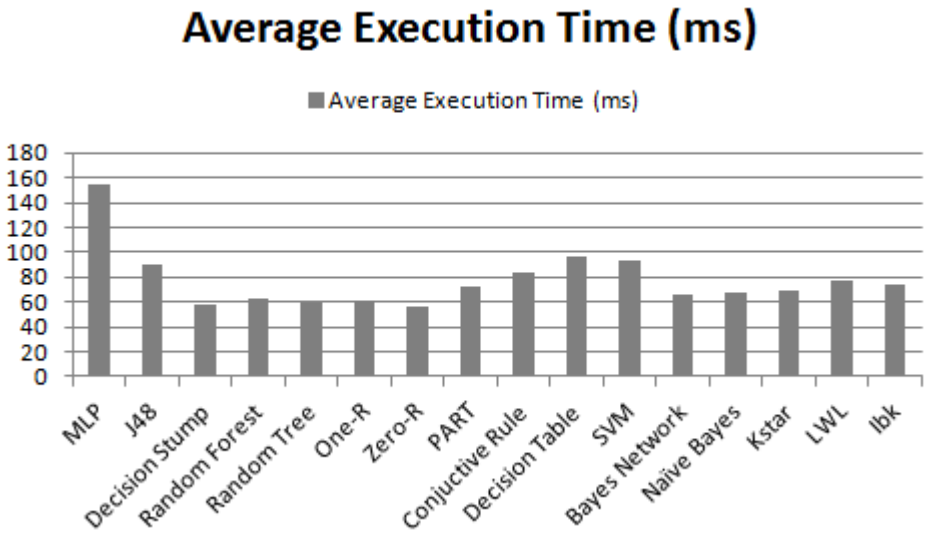

Figure 3.2: Average Performance (Time)

From the above table and graphs, it is clearly visible that there is no particular algorithm which works well for each dataset. The average performance is also similar in most of the classification algorithm.

After assigning the class value, we trained the neural networks in a hierarchy as explained above. After this training, we selected other 35 datasets from different domains (mainly from UCI machine learning repository), and calculated their meta features. The trained neural network predicted the classification algorithm for the given datasets with 84% of accuracy with 10-fold cross validation. While evaluating the prediction accuracy for the new 35 datasets, we are getting the accuracy of 81% and the confidence factor for each individual instance higher than 77%(The minimum confidence factor is 76.98%). The confidence factor for each test instance is calculated using the following equation:

$$CF_i = \frac{n_i}{m_i}$$

where,

i is the particular test instance for which the CF is calculated.

n_i = number of datasets for which (class value = predicted class value$_i$)

m_i = number of datasets that are similar to dataset$_i$

Here, the similarity measure is calculated using Euclidean distance between two dataset instances.

3.4 Conclusion and Future Work

As shown in the table, and as per No Free Lunch Theorem, there is no such one learning algorithm which can perform well on every dataset. The results of the learning algorithm is totally based on its area of expertise as well as the data characteristics. As, the classifiers selected in the classification pool are predefined, introducing the categories and the hierarchical approach improves the results significantly. Still, there are few issues which are untouched in this project work. The pool of the classification algorithms contains 16 algorithms currently. With this approach, by increasing the number of algorithms, the performance of the module is affected in terms of the building time of the model for the meta dataset. This issue can be solved by parallelizing the models used to create the metadataset. Another issue is feature selection. Here, we have selected 28 meta features of three types (Direct, Statistical, and Information theoretic). The metadatasets for all these individual meta feature types are available but there is no such metadataset available where all the types have been used. Here, we have applied feature selection on each individual metadataset and have selected the most affecting features from those metadatasets. Though, feature selection can be applied to our meta dataset. Another issue which has not been included in this project work is parameter tuning of the classification algorithms. Even if the number of algorithms is not increased in the pool, with the same pool, there can be different variations of all the algorithms with different parameter settings. Here, we have considered all the classification algorithms with their default parameter settings as given

in the WEKA machine learning tool. As shown in many papers, parameter tuning can improve the classification algorithm's performace significantly.

References

[1] Pavel B. Brazdil, Carlos Soarse, Joaquim Pinto Da Costa, Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results, Proceedings of International Conference on Machine Learning, 50, 251-277, 2008.

[2] Pavel Brazdil, Rui Leite, Determining the Best Classification Algorithm with Recourse to Sampling and Meta-Learning, Advances in Machine Learning I, SCI 262, pp. 173-188, 2010.

[3] Pavel B. Brazdil, Carlos Soarse, A Comparison of Ranking Methods for Classification Algorithm Selection, In proceedings of the European Conference on Machine Learning ECML2008, pp. 63-74, 2008.

[4] Matthias Reif, Faisal Shafait, Markus Goldstein, Thomas M. Breuel, Andreas Dengel, Automatic Classifier Selection for Non-Experts, Journal of Pattern Analysis and Applications, July-2012.

[5] Johannes Furnkranz, Johann Petrak, An Evaluation of Landmarking Variants, Proceedings of the ECML/PKDD Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning (IDDM-2001), 57-68, 2001.

[6] Hilan Bensusan, Christophe Giraud-Carrier, Discovering task neighbourhoods thorugh landmark learning performances, Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD-00), 325-330, 2000.

[7] H. Bensusan, C. Giraud-Carrier, C.J. Kennedy, A Higher order approach to meta-learning, A Technical Report, University of Bristol, UK, 2000.

[8] Hilan Bensusan, Odd bites into bananas don't make you blind - learning about simplicity and attribute addition, Proceedings of the ECML'98 workshop on upgrading learning to the meta-level: model selection and data transformation, April 1998, 30-42.

[9] Bernhard Pfahringer, Hilan Bensusan, Christophe Giraud-Carrier, Tell me who can learn you and I can tell you who you are: Landmarking various Learning Algorithms, Proceedings of the Seventeenth International Conference on Machine Learning (ICML2000) (pp. 743-750). San Mateo, CA: Morgan Kaufmann.

[10] Yonghong Peng, Peter A. Flach, Carlos Soarse, Pavel Brazdil, Improved Dataset Characterization for Meta-learning, Proceedings of the 5th International Conference on Discovery Science, pp. 141-152, 2002.

[11] Matthias Reif, Faisal Shafait, Andreas Dengel, Meta-Features: Providing Meta-Learners More Information, Poster and Demo track of 35th German Conference on Artificial Intelligence (KI-2012), pp. 74-77, 2012.

[12] Shawkat Ali, Kate A. Smith, On learning algorithm selection for classification, Applied Soft Computing 6, pp. 119-138, 2006.

[13] Pavel B. Brazdil, Carlos Soarse, Zoomed Ranking: Selection of Classification Algorithms based on Relevant Performance Information, Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'00), pp. 126-135, 2000.

[14] Robert Engels, Christiane Theusinger, Using a Data Metric for Preprocessing Advice for Data Mining Applications, 13th European Conference on Artificial Intelligence, 1998.

[15] Meta Learning, Cognitive Technologies, pp. 31-59, 2009.

[16] Ricardo Vilalta, Christophe Giraud-Carrier, Pavel Brazdil, Carlos Soares, Using Meta Learning to Support Data Mining, International Journal of Computer Science and Applications, Vol. I, No. 1, pp. 31-45.

[17] Christophe Giraud-Carrier, Metalearning-A Tutorial, December 2008

[18] Ricardo Vilalta, Youssef Drissi, A Perspective View and Survey of Meta-Learning, Journal of Artificial Intelligence, Vol. 18, 77-95, 2002.

[19] Ricardo Vilalta, Christophe Giraud-Carrier, Pavel Brazdil, Meta-Learning: Concepts and Techniques, International Journal of Soft Computing and Engineering, Vol. I, pp. 31-45, 2006.

[20] Xiaojun Chen, Yunming Ye, Graham Williams, Xiaofei XuA, Survey of Open Source Data Mining Systems, Proceedings of the 2007 international conference on Emerging technologies in knowledge discovery and data mining (PAKDD'07), 3-14, 2007.

[21] Meta Learning, Cognitive Technologies, pp. 61-72, 2009.

[22] Nikita Bhatt, Amit Thakkar, Ranking of Classifiers based on Dataset Characteristics Using Active Meta Learning, 2012.

[23] Saso Dzeroski, Pance Panov, Bernard Zenko, Ensemble methods in Machine Learning, Report, Jozef Stefan Institute, Slovenia, 2010.

[24] Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[25] Tom Mitchell, Machine Learning, McGraw Hill, 1997.

[26] Xiaojin Zhu, Semisupervised Learning Literature Survey, 2006.

[27] Data Pre-processing in WEKA, www.cs.waikato.ac.nz/ml/weka .

[28] WEKA- Experiences with Java Open Source Project,

[29] Janez Demsar, Statistical Comparisons of Classifiers over Multiple Data Sets, The journal of Machine Learning Research, January 2006, Vol. 7, 1-30.

[30] Matthias Reif, A Comprehensive Dataset for Evaluating Approaches of Various Meta Learning Tasks, ICPRAM (2012), p. 273-276.

[31] Matthias Reif, Faisal Shafait, and Andreas Dengel, Dataset Generation for Meta Learning, 35th German Conference on Artificial Intelligence (KI-2012) pp. 69-73, 2012.

[32] Pavel Kordik, Jan Cerny, On Performance of Meta Learning Templates on Different Datasets, IEEE World Congress on Computational Intelligence, June 10-15, 2012, Brisbane, Australia.

[33] Janez Demsar, Statistical Comparisons of Classifiers over Multiple Datasets, Journal of Machine Learning Research, 7, 1-30, 2006.

[34] Silviu Cacoveanu, Camelia Vidrighin, Rodica Potolea, Evolutional Meta-learning Framework for Automatic Classifier Selection, IEEE 5th International Conference on Intelligent Computer Communication and Processing, 2009, 27-30.

[35] Marcilio C. P. de Souto, Ricardo B. C. Prudencio, Rodrigo G. F. Soarse, Daniel S. A. de Araujo, Ivan G. Costa, Teresa B. Ludermir, Alexander Schliep, Ranking and Selecting Clustering Algorithms Using a Meta-learning Approach, IEEE International Joint Conference on Neural Networks, 2008. (IJCNN 2008), 3729-3735.

[36] M. A. Torsello, C. Castiello, A. M. Fanelli, A Meta Classification Framework, IEEE International Conference on Fuzzy Systems, July 16-21, 2006, 401-407.

[37] Abraham Bernstein, Foster Provost, An Intelligent Assistant for the Knowledge Discovery Process, 2001.

[38] Troy Raeder, Nitesh V. Chawla, Model Monitor (M2) : Evaluating, Comparing and Monitoring Models, The journal of Machine Learning Research, January 2009, Vol. 10, 1387-1390.

[39] D. Michie, D. J. Spiegelhalter, C.C. Taylor,Machine Learning, Neural and Statistical Classification, Ellis Horwood Series in Artificial Intelligence, February 1994, Prentice Hall.

[40] Xiaojun Chen, Yunming Ye, Graham Williams, Xiaofei XuA, Survey of Open Source Data Mining Systems, Proceedings of the 2007 international conference on Emerging technologies in knowledge discovery and data mining (PAKDD'07), 3-14.

[41] Ricardo B. C. Prudencio, Teresa B. Ludermir, Active Meta Learning with Uncertainty Sampling and Outlier Detection, IEEE International Joint Conference on Neural Networks (IJCNN 2008), pp. 346-351, 2008.

Appendix A

Appendix

Meta Features

Here is the list of meta features which have been used in this project:

- **Simple(Direct) Measures:**

  Number of Classes

  Number of Attributes

  No. of Nominal Attributes

  No. of Numeric Attributes

  Samples

  Dimensionality

  Fraction of Nominal Attributes

  Fraction of Numeric Attributes

  Minimum Symbols

  Maximum Symbols

  Mean Symbols

  Sum of Symbols

No. of Missing Values

- **Information Theoretic Measures:**

  Minimum Class Probability

  Maximum Class Probability

  Mean Class Probability

  Class Entropy

  Attribute Entropy

  Joint Entropy

  Mutual Information

  Equivalent Attributes

  Noise-to-Signal Ratio

- **Statistical Measures:** Skewness

  Kurtosis

  Absolute Correlation

  Canonical Correlation