A comparison of meta-learning strategies

by

John Liddell

A THESIS

Submitted to the faculty of Delaware State University in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in the Division of Physical and Computational Sciences

DOVER, DELAWARE
November 2019

This thesis is approved by the following members of the Final Oral Review Committee:

Dr. Tomasz Smolinski, Committee Chairperson, Division of Physical and Computational Sciences, Delaware State University

Dr. Gary F. Holness, Committee Member, Division of Physical and Computational Sciences, Delaware State University

Dr. Marwan Rasamny, Committee Member, Division of Physical and Computational Sciences, Delaware State University

Dr. Yuriy Markushin , External Committee Member, Physics and Pre-Engineering Department, Delaware State University

ACKNOWLEDGEMENTS

I would like to acknowledge those who enabled the completion of this thesis, Dr. Smolinski, Dr. Holness, the staff of the computer science department and my committee group. Without you all I could not have completed this thesis. Thank you all for the guidance and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

A comparison of several meta-learning strategies in order to ascertain the truth of the "no free lunch theorem," as it applies to metalearning strategies, was performed. The data needed in order to validate the performance of each meta-learning strategy were gathered from the UCI Irvine dataset repository, parsed, then stored in a database. Collections of datasets were then created within this database, and the meta-learning strategies were compared in terms of their accuracies with relation to these collections. t score analysis was then performed in order to determine whether the relative accuracies of the meta-learning algorithms were within expectation.

CHAPTER 1: INTRODUCTION

Determining what algorithm to use when analyzing a dataset is a problem as old as machine learning itself. In “No free lunch theorems for optimization” Wolpert and Macready demonstrate that the relative performance of any two given machine learning algorithms will be uniform across all datasets, that is to say a machine learning algorithms performance is contingent on the problem space in which the algorithm is operating. As such, the decision of algorithm is not arbitrary and some strategy must be employed in order to decide on an algorithm. In some cases, the individuals wishing to perform an analysis have access to an expert, possibly themselves, that can simply tell them which algorithm is best in the given situation. In other situations, the individuals wishing to perform analysis may not have the budget necessary to acquire access to such an expert, in which case the usage of a meta-learner becomes appropriate. For instance, at Walmart Labs, meta learning algorithms are used to decide how best to detect placeholders and to identify fraudulent transactions, all without the use of manual parameter tunning or even direct algorithm selection [1]. With a meta-learner, one feeds the meta-learner a dataset, and it returns to the user what it thinks is the most appropriate machine with which to perform analysis. To get to the point wherein a decision can be made on new datasets, the meta-learner itself must first be trained, and this training requires some sort of learning strategy. This fact suggests that the decision of what meta-learning strategy to use for some given body of datasets should be susceptible to the previously mentioned no free lunch theorem, that is to say that some meta-learning strategies will work better on some given set of databases than others. The confirmation or denial of this theorem in this context is the goal of this thesis.

There are many factors that may be considered when attempting to optimize the performance of a machine learning algorithm. One can choose to take special effort to ensure the parameters of the model are fine tuned to the type of data one is running the algorithm on. Another option to

ensure optimum performance with a machine learning algorithm is to carefully select the features that one will extract from a given dataset with respect to the desired algorithm. In the case of this experiment, neither of those is addressed at this point in time, the work of this thesis is entirely in the comparison of the effectiveness with which the meta learning strategies utilize a given meta base. Guaging the effects of base learner parameter tunning and feature alteration is left to future work as it is beyond the scope of this experiment.

Including the current one, this thesis is comprised of five chapters. In Chapter 2, a review of the base machine and meta learning strategies used within the experiment is done. Chapter 3 describes the structure of the experiments code at a high level. Chapter 4 analyzes the results table in order to determine whether or not one meta-learning strategy strictly dominates. Chapter 5 presents a summary of the document and presents possible future work that can be performed to follow up on the results presented.

CHAPTER 2: REVIEW OF THE LITERATURE

SECTION 2.1: NO FREE LUNCH THEOREM

Introduced in Wolpert and Macready's "No Free Lunch Theorems for Optimization" 1997 paper, the No Free Lunch theorem states that the performance of all algorithms, when averaged out across all datasets, should be the same; that is to say there is no one algorithm that is universally the best. The root cause of this phenomenon is in that differing algorithms make different assumptions about the distributions from which the data the algorithms work with arises. A learning algorithm with an implicit assumption of a random distribution will have a far lower test case classification accuracy than an algorithm that assumes a Gaussian distribution if the distribution from which the set of observed samples derives is truly normal and vice versa, if the distribution is truly random, the Gaussian classifier's accuracy will suffer relative to the classifier with a random assumption.

SECTION 2.2: ON META LEARNING

Before describing each of the individual meta learners compared within this experiment, it is neccessary to describe what a meta learning algorithm is. Much like regular machine learning algorithms, meta learning algorithms process inputs and make predictions on new instances from experience. Where meta learners differ from base learners is in the scope of the level of their adaptation; whereas learning at the base level is focued on accumulating experience on a specific learning task (e.g., credit rating, medical diagnosis, mine-rock discrimination, fraud detection, etc.), learning at the meta level is concerned with accumulating experience on the performance of multiple applications of a learning system [2]. As such, base learners attempt to optimize performance with respect to a specific task or problem domain, while meta learners attempt to predict what the best base learner would be for a given task or problem domain. This allows users wishing to use machine learning in their problem domain to skip the struggle associated with descovering the best algorithm for the specific problem; they need only gain access to a trained meta learner, present the

dataset they wish to analyze, then make use of the base learner suggested by the meta learner.

All meta learners operate in two phases: Acquisition mode and advisory mode. During the knowledge acquisition mode, the main goal is to learn about the learning process itself [2]. The meta learner recieves as its input during this phase a set of database sets (hereafter refered to as a "metabase") from which dataset characteristics and run statistics are extracted. The goal of the extraction of these characteristics and statistics is to gather information that can be used to generalize the run results from this specific metabase to other unproccessed distributions.

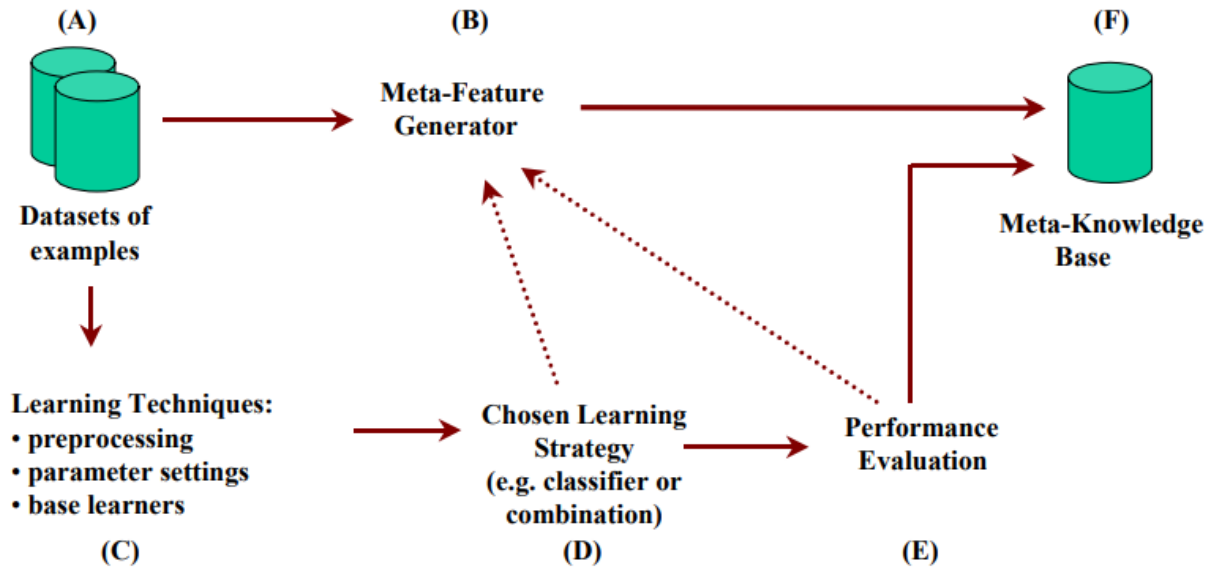

Figure 2.1: An Example of Meta Learner Knowledge Acquisition
Image borrowed from [2]
Parameter setting not applicable in this experiment

In advisory mode, the meta learning system makes use of the knowledge gathered in the acquisition phase in order to suggest a best learning algorithm for a new dataset. Meta features extracted from the dataset are "matched" with the meta knowledge base to produce a recommendation regarding the best available learning strategy [2].

What this entails is a mapping between meta features and an optimum base learning strategy.

In the case of this thesis, this mapping is accomplished via the k-means algorithm; with the meta features being a set of standard statistical measures in the case of the brute force and active strategies (these features are described in section 3.3) and the meta features being "learning curves" in the case of the curve comparison strategy (these curves are described in section 2.3.3).

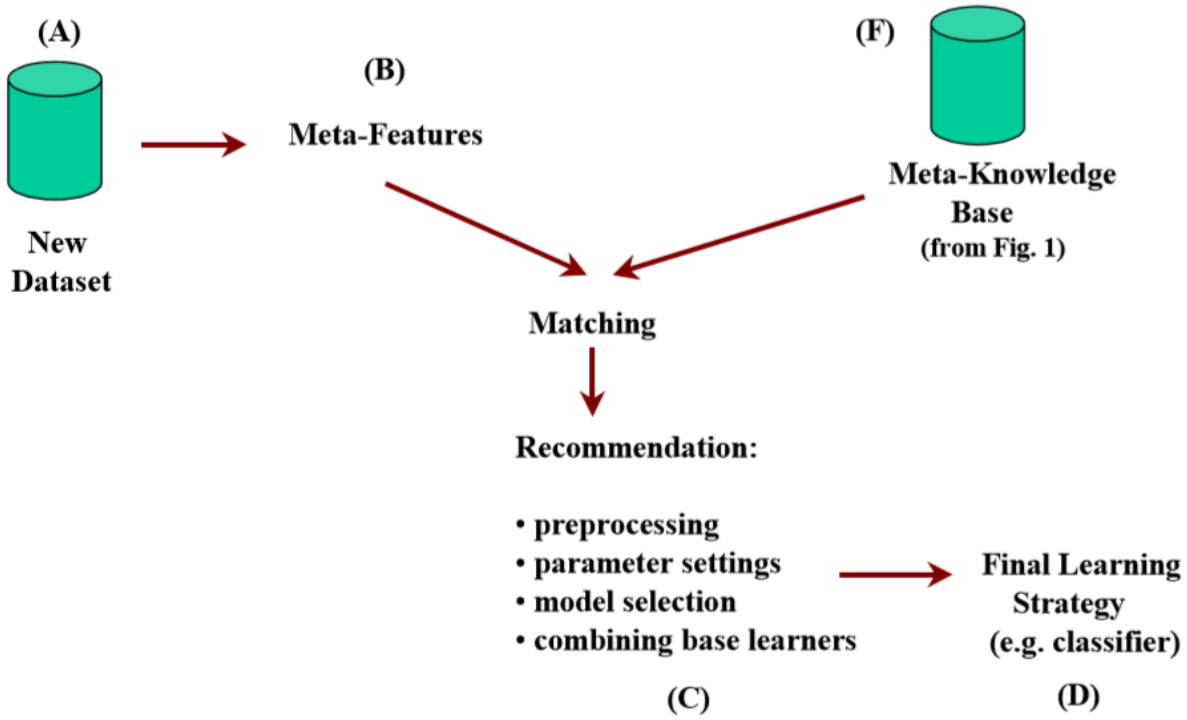

Figure 2.2: An Example of Meta Learner Advisory Mode
Image borrowed from [2]
Parameter setting not applicable in this experiment

SECTION 2.3: SUMMARY OF THE COMPARED META LEARNING STRATEGIES

In order to ascertain whether or not the NFL theorem applies to meta learning strategies, we require a set of meta learning strategies with which to make run comparisons. The meta learning strategies used in the experiment that comprises this thesis are described within this section.

2.3.1 Brute Force Metabase

The most basic meta-machine learning algorithm. The accuracies of the meta-learners producible machines for some metabase are gathered. To classify a new dataset d_n, a clustering algorithm (K-Means in the case of this experiment) is used to find d_m, the dataset within the metabase with which the new dataset d_n is most similar. The algorithm which had the greatest classification accuracy for the metabase dataset d_m will then be returned by the meta-learner.

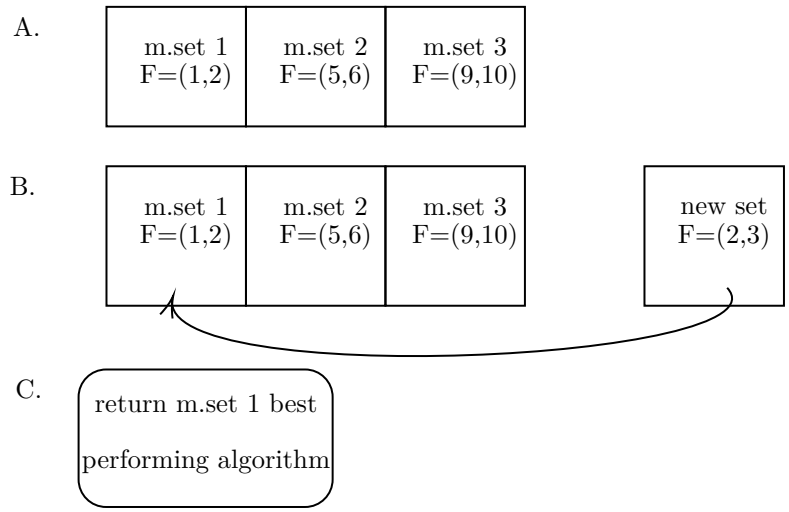

Figure 2.3: Example of base meta learning

A. Metabase sets with given meta feature vectors
B. Classify new dataset by meta feature vector comparison
C. Return best algorithm of associated meta base set

2.3.2 Active Meta Learning

The second of the meta-learning strategies implemented within the study, Active Meta Learning is a meta learning technique "that reduces the cost of generating Meta examples by selecting relevant Meta examples" [3]. What this entails is a decision on what datasets to allow into a meta-learner's metabase. Rather than analyze every candidate meta base dataset, an active meta-learner will ana-

lyze the next dataset with the highest uncertainty. The relative uncertainty between two datasets is defined to be:

$$\delta(V_x, d_i, V_x, d_j) = \frac{|V_x, d_i - V_x, d_j|}{Max_{k \neq i}(V_x, d_k) - Min_{k \neq i}(V_x, d_k)}$$

where V_x, d_k is the value of some metaparameter V_x for dataset d_k, $Max_{k \neq i}(V_x, d_k)$ is the maximum value of V_x, d_k when dataset i is removed and $Min_{k \neq i}(V_x, d_k)$ is its corresponding minimum. Determining which dataset has the overall highest uncertainty can be done via the following procedure. First, sum the relative uncertainties for each dataset and meta-parameter combination. Then, rank the uncertainty scores of the datasets within each meta-parameter. After obtaining the uncertainty ranks within each parameter for each dataset, sum the parameter ranks in order to obtain an overall uncertainty rank for each dataset. Finally, select the parameter with the highest rank for inclusion in the metabase. The equation representing the overall uncertainty score in a specific metaparameter V_x for dataset d_i is

$$\delta(V_x, d_i) = \frac{\sum_{j \neq i} |V_x, d_i - V_x, d_j|}{Max_{k \neq i}(V_x, d_k) - Min_{k \neq i}(V_x, d_k)}$$

2.3.3 Predicting Relative Performance of Classifiers from Samples

The third of the meta-learning strategies implemented within this study is one in which a representative subsection of the metabase is trained with each algorithm entirely, after which point the rest of the metabase undergoes curve sampling analysis; the accuracies of the base algorithms are predicted from run curve similarity rather than directly ran [4]. As with the other two meta learning strategies, the label for new datsets is then determined via clustering with the datasets contained within the metabase.

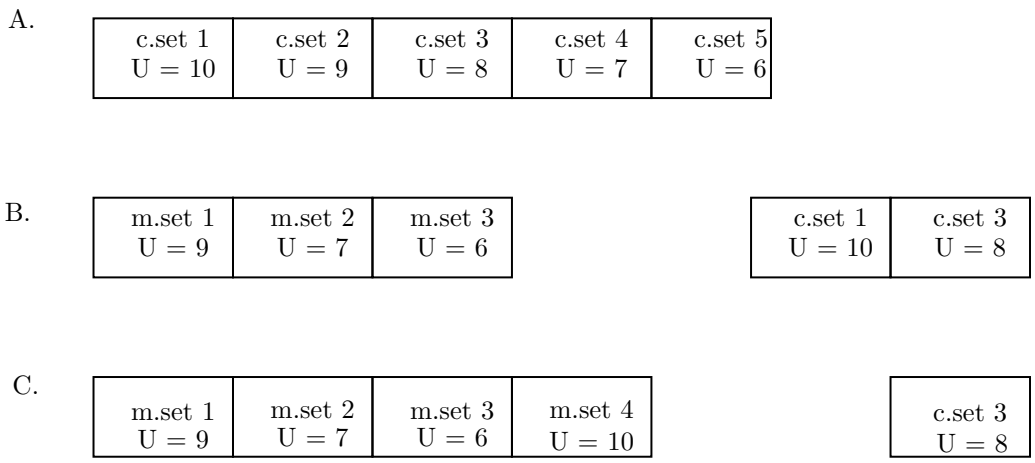

Figure 2.4: Example of active meta learning

A. Set of candidate metabase sets
B. Random selection of half the original candidates
C. Inclusion of half remaining candidates by uncertainty comparison

SECTION 2.4: SUMMARY OF PRODUCIBLE MACHINES

The strategies mentioned in the previous section all consume a vector representation of some dataset, and then make a guess as to what algorithm would best be able to classify its data. The machines that these strategies can choose from are the K-means clustering algorithm, a neural network, a naive Bayes classifier, the support vector machine, and regression; with the results coming from the regression machine being cast into classificatory bins from the real valued result that it would produce. An in depth description of each of these different learning algorithms will comprise the rest of this chapter.

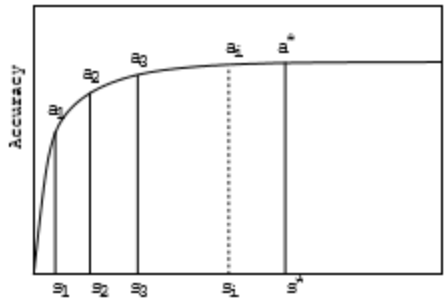

Figure 2.5: Example of a learning curve

where:
A. Horizontal axis: Fractions of training set
B. Verticle axis: Test case accuracy for given set fraction

Image borrowed from [4]

2.4.1 Linear Regression

Linear regression is one of the most common and oldest machine learning techniques. It asserts that the response is a linear function of the inputs [5]. This relation takes the following form:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \epsilon = \sum_{j=1}^{D} w_j x_j + \epsilon$$

where $w^T x$ represents the inner or scalar product between the input vector x and the model's weight vector w^T, and ϵ is the residual error between our linear predictions and the true response.

To fit a linear regression model, the least squares approach is usually used. Given some "overdetermined" linear system (that is to say a system in which there are more data points than parameters), one can write an expression for the sum of squares of the system

$$S(\beta) = (y_1 - \beta x_1)^2 + (y_2 - \beta x_2)^2 + ...(y_n - \beta x_n)^n$$

and then take the partial derivative of this sum of squares deviations with respect to each of the

components of β, set them to zero, then solve the resulting equations to directly determine the values of the parameters that minimize the sum of the squared errors of the system. With linear regression in two dimensions (one dimension in the Independent variable and one dimension in the Dependent variable, we see a system with two parameters $\beta_0 = yintercept$ and $\beta_1 = slope$. If we had, for example, 3 data points (2,1),(3,7), and (4,5) we would have the equations $\beta_0 + 2 * \beta_1 = 1$, $\beta_0 + 3 * \beta_1 = 7$, and $\beta_0 + 4 * \beta_1 = 5$. The sum of squared errors would then be $S(\beta_0, \beta_1) = [1 - (\beta_0 + 2 * \beta_1)]^2 + [7 - (\beta_0 + 3 * \beta_1)]^2 + [5 - (\beta_0 + 4 * \beta_1)]^2$,which we could then differentiate with respect to β_0 and β_1 then directly solve the resulting set of linear equations directly for the minimum of the summed squares.

2.4.2 Naive Bayes

The Naive Bayes classifier algorithm fits a set of data to Bayes' Theorem with a strong assumption of feature independence. Given a set of discrete-valued features $x \in 1, ..., K^D$, we can calculate the class conditional density for each feature, then, with our assumption of independence, generate a guess at what the class should be for a new input by multiplying the conditional likelihood values for each of the new inputs features times the prior on the desired to be known class, that is to say $p(y|\mathbf{x}) \propto p(y) \sum_j^D p(x_j|y)$. The calculation of the posterior probability for a new example can be done manually, or can be derived from distributions that are inferred from the provided data. Consider, for example, a collection of data listing individuals that did or did not purchase a house from a real estate agent, where, for some reason or another, the only data remaining pertaining to these individuals is what their income level was, what their age was, and how far they have to or would have had to drive to work from their new home.

Say we get a new datapoint: income: \$25,000, age: 30, distance: 10. In this case the conditional likelihood of this data given a yes for each of the individual features is 2/9, 1/9, and 2/9 respectively. The prior on yes is 1/3. The marginal likelihoods of each the individual features are

2/9, 1/9 and 2/9 respectively. As such, the posteriors for our new datapoint are $p(y = yes|x) = \frac{(2/9)*(1/9)*(2/9)*(9/27)}{(6/27)*(3/27)*(6/27)} = \frac{0.00182}{0.00548} = 0.33$ and $p(y = no|x) = \frac{(4/18)(2/18)(4/18)(18/27)}{(6/27)(3/27)(6/27)} = \frac{0.0036}{0.00548} = 0.66$.

Note that $0.66 > 0.33$ and as such our classifier would label this datapoint with a no, this individual is not likely to purchase a house.

2.4.3 Support Vector Machine

The support vector machine (svm) is a two-group classification algorithm that attempts to find a hyperplane that separates the inputs within a given input space with a maximum margin of separation between the hyperplane and the "support vectors," those vectors on either side of the hyperplane that are closest to it. To arrive at a form of the support vector machine that can be used to classify new inputs, one first needs a representation of the potential separating hyperplane

$$y_i(\mathbf{w} * \mathbf{x_i} + b) > 1$$

where y_i is the truth label of given training input x_i, w is a vector normal to our candidate separating hyperplane that represents how much "weight" is to be applied to an input, and b is a bias constant representing the threshold the weight/input product needs to pass before it is considered classified. The distance of a given hyperplane can be determined by calculating the difference between these previously mentioned "support vectors" in the direction normal to this hyperplane. This difference can be calculated via the following equation:

$$(\mathbf{x_{s+}} - \mathbf{x_{s-}})\frac{\mathbf{w}}{||\mathbf{w}||}$$

where $\mathbf{x_{s+}}$ and $\mathbf{x_{s-}}$ are respectively positive and negative support vectors and $\frac{\mathbf{w}}{||\mathbf{w}||}$ is the unit vector in the direction normal to the hyper plane towards the positive examples. The size of the margin is $2/||w||$. As such, the discovery of a working svm can be accomplished by solving a constrained

optimization problem in which the thing to be minimized is $1/2||w||^2$, subject to the constraints $y_i(\mathbf{w} * \mathbf{x_i} + b) = 1$ for support vectors. Crafting an expression of this constrained optimization that can be solved by a computer can be done by taking the Lagrangian:

$$L(\mathbf{W}, , \mathbf{Y}) = 1/2||\mathbf{w}||^2 \sum_{i=1}^{n} y_i(\mathbf{w} * \mathbf{x_i} + b) - 1)$$

then taking care of the fact that the vector w_o that determines the optimal hyperplane can be written as a linear combination of the training vectors: $w_0 = \sum_{i=1}^{n} y_i \alpha_i^0 \mathbf{x_i}$ [6]. Swapping this equation into the Lagrangian yields:

$$L = \sum \alpha_i - 1/2 \sum_i \sum_j \alpha_i \alpha_j y_i y_j (\mathbf{x_i} \cdot \mathbf{x_j})$$

at which point one should consult their closest computer so that it can maximize this expression.

This final form of the Lagrangian reveals the support vector machines most powerful attribute: the kernel. The optimization of the hyperplane within the inputs depends only on their dot product of pairs of inputs; they do not appear anywhere else in the Lagrangian other than the very end and then only so as pairs of dot products. This fact allows the writing of a decision function on new inputs:

$$f(\mathbf{u}) = \sum_{i}^{N} \alpha_i y_i (\mathbf{x_i} \cdot \mathbf{u}) + b$$

where **u** is a vector whose label we do not know. The support vector machine can use kernels to map input vectors into non-linear high-dimensional feature space without actually calculating the position of the vectors within that feature space [6]. The kernel accomplishes this by calculating the distance between (or similarity) of its two input vectors in this space without reference to their exact position within this higher space. This then allows the computation of a linear separation between the points in this higher dimensional space which translates into a non-linear separation for the vectors in their original lower dimensional space where a separation might otherwise not have been discoverable.

2.4.4 K-Means Clustering

The objective of the k-means algorithm is to partition a dataset into k groups such that the points within some group are all closest to the mean of that group than they are to any other group. A clear informal explanation of the work that the k-means algorithm performs was given by James MacQueen in 1967: "...the k-means procedure consists of simply starting with k groups each of which consists of a single random point, and thereafter adding each new point to the group whose mean the new point is nearest. After a point is added to a group, the mean of that group is adjusted in order to take account of the new point. Thus at each stage the k-means are, in fact, the means of the groups they represent" [7]. Formally stated, given an integer k and a set of n data points in $\mathbb{R}^d$ the K-means algorithm seeks to minimize Φ, the over all total summed in class distance between each point and its closest center such that $\lneqq = \sum_{x \in X} min_{c \in C} x - c^2$ [8].

The k-means model is a type of Gaussian mixture model that is trained with a procedure called expectation maximization. Given a set of distributions with missing data, mixture models tend to have derivatives that are either difficult to define or are entirely undefinable. On the other hand, the calculation of some ML/MAP (maximum likelihood/maximum a posteriori) estimates for some set of models can generally be calculated with little difficulty if every point within the distributions is known (at which point our learner would obviously have nothing to do) and thus calculus would be entirely unnecessary ($i.e.$, it would not matter that the derivative cannot be defined). Expectation maximization uses this fact in order to obtain an estimation of the ML/MAP indirectly. The algorithm consists of two steps. First, an estimate as to what the expected value of the hidden data is based off the current guess for the parameters is made. Then the likelihood function for the parameters is maximized under the assumption that the data discovered in the previous step is complete, $i.e.$, that there is no longer any hidden data. These steps are then repeated until some convergence criteria is met. The k-means is exactly this type of algorithm, but with the covariance matrix $\Sigma_k = \rho^2 * I_D$ and the mixing weights $\Pi_k = 1/K$ all being fixed, such that the only free

parameters are the cluster centers $\mu_k \in \mathbb{R}^D$, and such that the hidden data that is the ground truth label of the data points.

2.4.5 Neural Networks

A neural network is a type of machine learning algorithm that mimics the interconnectivity of animal brains in order to automatically discover rules to classify given inputs. The neural network is one of the most flexible learning algorithms within literature, so flexible in fact that it is capable of approximating any continuous function [9]. As such, its inclusion within a metalearning system is almost mandatory. Genrally, a neural network system works by first being presented with a set of classified or unclassified inputs. Said system will then attempt to make a decision on these inputs on which an error value will then be assigned. The system will then see some kind of correction function applied to it. This process will continue until the system has exhausted its supply of training data, at which point it will hopefully have discovered a strong set of rules for peforming whatever work it is that it was designed to perform.

The type of neural network that will be used within this thesis is what is called the feed-forward neural network (multilayer perceptron, a.k.a. MLP). The feed forward neural network is essentially a series of logistic regression models stacked on top of each other, with the final layer being either another logicstic regression or linear regression model depending on whether or not a classification or regression problem is being solved [5]. The leftmost layer of this stack is called the input layer and consists of a set of neurons $x_i|x_1, x_2, x_3, ..., x_m$ representing the input's features. Each neuron in the hidden layer transforms the values from the previous layer via weighted linear summation $w_1X_1 + w_2x_2 + ... + w_mx_m$ which is then passed into a non linear-action function $g()$, such as the logistic function or the hyperbolic tangent function. It is important to note that g must be non-linear, otherwise the entire model will collapse into a large linear regression model of the form $y = w^T(Vx)$ [5].

The multi-layer perceptrons created in this experiment will be trained used in by an error propagation/training technique called backpropagation. Backpropagation is a procedure that repeatedly adjusts the weights of the connections in a neural network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector [10]. In order to accomplish this, the algorithm adjusts the weights of the nerual network by considering the error of the outputs then minimizes this error via gradient decent with respect to each of the weights within the network. Specifically, the gradient vector of the negative log likelihood error on the output neurons is computed by use of the chain rule of calculus.[5]. Say we have a one layer neural network in which the hidden layer is described by $\alpha^L = \sigma(w^L \alpha^{L-1} + b^L) = \sigma(z^L)$, where L superscript refers to the hidden layer and $L-1$ refers to the input layer of the network. The parameters of this network can be said to be $\Theta = (V, W)$ where V is the weight vector for the input layer and W is the weight vector for the hidden layer. The error (or more specifically the costs function) of a such a network is given by:

$$J(\Theta) = -\sum_n \sum_k (\hat{y}_{nk}(\Theta) - y_{nk})^2$$

in the case of regression, and via cross entropy

$$J(\Theta) = -\sum_n \sum_k y_{nk} log \hat{y}_{nk}(\Theta)$$

in the case of classification. The gradient of this error $\nabla_\Theta J$ is found via the chain rule of calculus:

$$\frac{\partial C}{\partial w^L} = \frac{\partial z^L}{\partial w^L} \frac{\partial \alpha}{\partial z^L} \frac{\partial C}{\partial \alpha^L}$$

This equation is easiest to understand if read from right to left. Notice how in each stage the rates being compared are between nearest elements, first the error to the output that produced it, then the output to the element to which the non-linearity is applied, then finally the the non-linerity recieving value to the weight vector. The result of this calculation easily gives us the direction of

the gradient, the negative of which we will use to modify w^L in a direction that will reduce the output error. Reduction of the error of a multilayer perceptron with more than one neuron in each layer works mostly the same way. Once again, the chain rule of calculus is used in order to find the derivative of the output error with respect to the weights of the connections between the hidden layer before the output neurons and the output neurons $\frac{\partial C}{\partial w^L_{n-j}}$ where L is the target neurons layer (the last layer in this case), n is the index of a neuron within this layer, and j is the index of the neuron in the previous layer $L-1$ from which neuron n is recieving input (note that in this case the hyphen does not mean subtract, but rather indicates that there is a connection between these neurons). For an output neuron, its error is then given by:

$$\frac{\partial C}{\partial w^L_{n-j}} = \frac{\partial z^L_j}{\partial w^L_{n-j}} \frac{\partial \alpha^L_j}{\partial z^L_j} \frac{\partial C}{\partial \alpha^L_j}$$

with the error relative to neurons earlier in the network being calculable by continuing usage of the chain rule.

CHAPTER 3: RESEARCH DESIGN

SECTION 3.1: GENERAL PLAN

The overall goal of this experiment is to determine whether one metalearning strategy within a given set can strictly dominate the others. The core elements required in order to determine this are: (1) a set of metalearning strategies to be compared, (2) sets of metabase datasets on which to apply the different strategies, and (3) a means to evaluate the results so as to determine the relative performance of the metalearners. The general flow of the program that constitutes this experiment begins with a set of unprocessed datasets, and then extracts the metafeatures that are required to perform dataset clustering and to run the active metalearning strategy. The program then constructs 10 metalearning bases with the elements in these sets choosen at random. A run with every machine algorithm and dataset combination is then performed, with the results being stored in the experiments database. Learning curves for each dataset/algorithm combination are then crafted. Finally, enough information now exists within the the database to run the metalearning strategies and extract results. This process is then repeated 30 times, with each run constituting an individual sample. This produces enough information in order to perform statistical analysis techniques to test the null hypothesis and obtain a margin of error of 5 percent. A detailed explanation of the experiment steps follows now, with a description of the statistical analysis to follow in Chapter 4.

SECTION 3.2: DATA PARSING

The data used in this experiment comes from the UCI Irvine Machine learning repository, the obtainment of which was accomplished via the use of a bash shell script that allowed the downloading of every dataset in the repository all at once. To make use of a dataset from the repository, the algorithms used in this experiment required a vector representation of the dataset currently being analyzed; the data could not be used without first translating it into this form. As such, I needed to do two things with the data before making use of them: ensure that the data could be parsed into a

vector via the use of program, then write a program in order to do this. The strategy I employed in order to accomplish these goals was was twofold: I first went through the set of candidate datasets and ensured non of them was in a format so exotic that they could not be parsed programatically. Manual examination of the files revealed that those of either the .data, .svm, or .dat format were agreeable to formating and so it is these that were processed by the parser. These files were then inspected by the parser, with each dataset's column vectors being inspected one by one. Those column vectors containing only numerical data were left as is, those with any non-numerical data were assumed to be categorical, with the categories of said vector being translated to numbers with a unique number being assigned to each unique string. Rather than storing the numerical representation of each dataset within the database, the parsed form of a given dataset is crafted when it is needed, saving an enourmous amount of disk space.

SECTION 3.3: METAFEATURE EXTRACTION

The existance of a parser allows us to craft the first table needed for this experiment which is one containing the meta-features of those datasets that are parsable. Being as how the datasets used within this project have vastly differing structures with respect to metrics such as the number of features and the maximum and minimum values of these features, the project requires a set of normalized meta-features that are applicable to any possible individual distribution or set of probability distribution(s). A set of features that meet this criteria are weighted mean, coefficient of variation, skewness, kurtosis, and entropy. The vector that represents a given dataset is crafted by taking the value of each of these attributes for each of said datasets features then normalizing them by dividing by the total number of features within that dataset, that is to say

$$F_{ad} = \frac{\sum_{c=i}^{N} f_{ai}}{N}$$

is the metafeature value a for dataset d, c is an iterator across columns for dataset d, f is the value of metafeature F applied to individual column i, and N is the number of columns within dataset d. The vector that represents a given dataset is then determined to be $V_d = (F_{1d}, F_{2d}, ..., F_{ad})$.

SECTION 3.4: METABASE CONSTRUCTION AND PERFOMANCE TESTING

The work of a metalearning algorithm is essentially the applying the things known from a specific set of datasets towards a new, unlabeled dataset. That initial set of datasets is known as a meta database, which I shorten to metabase for convience. The core premise of this experiment is the determination as to whether or not one metalearning strategy may or may not dominate other metalearning strategies across a set of of different metabases. As such, this experiment requires multiple sets of metabases in order to produce samples that can be used to test our hypothesis. The datasets in a given metabase are randomly choosen from the set of all datasets then stored in the database. There are 10 of these per sample, each being a fifth the size of the entire set of datasets. Testing the performance of a metalearner is done by using the metalearning strategy with some given metabase and applying it to every other dataset within the set of datasets. The guesses a given metalearning strategy makes with some given metabase are then stored within a database table for later analysis.

SECTION 3.5: COMPILING RESULTS

Once the guess tables are populated, it is finally possible to compile a table of results. Each entry in the table notes the metalearning strategy being evaluated, the metabase collection ($i.e.$, the sample), the metabase within that sample that the strategy used in order to analyze its test datasets, and the accuracy, training time, and rate correct score as its performance metrics, where the rate correct time measures how often the metalearner makes the correct guess given the time spent to train it in units of correct guesses per second.

CHAPTER 4: RESEARCH FINDINGS

SECTION 4.1: RUN RESULTS AND ANALYSIS TOOLS

If the no free lunch theorem applies to meta-learning strategies, we should see near equal performance across a variety of meta-set collections. We will call this assertion the null hypothesis, that is to say our null hypothesis is that the meta-learning strategies used in this experiment are equal.

In order to test the null hypothesis, 30 such samples of the kind described in Chapter 3 were collected. The metrics of interest on these samples are contained in the tables within this chapter. How often the algorithms placed first, second, or third can be seen in table 4.1. The average of these placements across all samples, i.e the algorithms average placements, can be seen in table 4.2. Table 4.3 contains the proportion of probability for the results contained in table 4.1. Table 4.4 contains the average of the proportion of probabilities contained in table 4.3 across all samples. Table 4.5 contains the standard deviations of the placement values for the meta-algorithms. Table 4.6 contains the t scores of each of the values present in table 4.1. Table 4.7 contains the average of the t scores contained in table 4.6 across all samples and it is this table that I use later to draw the conclusion that the null hypothesis may be safely rejected.

Each of the meta-set collections contains 10 basesets and the experiment compares the performance of 3 meta-learning algorithms. As such, the expected average number of first, second, and third place finishes given that the meta-learning algorithms are equal is 3.3. The values seen within the placement counts table given equal meta-learning algorithms should more often than not be either 3 or 4 and the averages of the placements across all samples should all be near 3.3. Instead, it appears that the sampler performed the best, with an average number of first place finshes of 4.5. Moreover most of the averages present in table 4.2 seem to be farther away from the expected value of 3.3 than one would intuitively expect if the meta-learning algorithms were truly equal. Whether

	GuessesActive			GuessesEx			GuessesSamp		
	First	Second	Third	First	Second	Third	First	Second	Third
sample 1	1	4	5	6	2	2	3	4	3
sample 2	1	4	5	5	2	3	4	4	2
sample 3	1	3	6	7	3	0	2	4	4
sample 4	1	5	4	6	3	1	3	2	5
sample 5	0	6	4	8	2	0	2	2	6
sample 6	3	3	4	5	4	1	2	3	5
sample 7	4	3	3	4	4	2	2	3	5
sample 8	2	3	5	7	2	1	1	5	4
sample 9	1	3	6	3	5	2	6	2	2
sample 10	0	4	6	7	3	0	3	3	4
sample 11	0	6	4	7	3	0	3	1	6
sample 12	1	5	4	7	2	1	2	3	5
sample 13	3	3	4	5	4	1	2	3	5
sample 14	2	5	3	6	3	1	2	2	6
sample 15	2	1	7	4	6	0	4	3	3
sample 16	1	5	4	6	0	4	3	5	2
sample 17	1	4	5	6	4	0	3	2	5
sample 18	1	3	6	8	1	1	1	6	3
sample 19	1	4	5	7	3	0	2	3	5
sample 20	2	4	4	6	2	2	2	4	4
sample 21	1	2	7	4	6	0	5	2	3
sample 22	3	3	4	2	7	1	5	0	5
sample 23	3	4	3	6	4	0	1	2	7
sample 24	3	3	4	4	4	2	3	3	4
sample 25	2	6	2	7	3	0	1	1	8
sample 26	1	3	6	6	2	2	3	5	2
sample 27	7	2	1	3	5	2	0	3	7
sample 28	0	5	5	7	2	1	3	3	4
sample 29	1	2	7	4	5	1	5	3	2
sample 30	2	6	2	4	3	3	4	1	5

Table 4.1: Placement results

	GuessesActive	GuessesEx	GuessesSamp
First	1.70	3.8	4.50
Second	5.57	3.3	1.13
Third	2.73	2.9	4.37

Table 4.2: Average placement results across all samples

or not these results fall far enough outside expectation in order to reject the null hypothesis requires analysis with the machinary of classical statistics. Three well established hypothesis testing measures are the method of calculating sampling distribution probabilities, analysis of variance (anova) and t score analysis. A brief description of each of these statistical methods and the result from their usage follows.

4.1.1 Exact Sampling Distribution

The following description losely follows the procedure described in [11]. In it, the author asks the reader to imagine testing a coin to see whether or not it is fair, flipping the coin 1,2,..N times. He then asks the reader to consider whether some proportion of heads is actually fair from 0/N, 1/N.., N/N heads. The propability that some proportion of heads p = i/N is fair can be calculated exactly with the binomial distribution

$$\frac{N!}{i!(N-i)!}r^i(1-r)^{N-i}$$

This situation is analogous to the number of first, second, or third place finishes some meta-algorithm obtained in this thesis experiment. The probabilty of proportions for each of the meta-learning algorithms can be seen in table 4.3 and the average of these proportions across all samples can be seen in table 4.4.

We can calculate the probability of drawing either of the values closest to expectation, 3 or 4, by use of the previously mentioned binomial distribution, with $N = 10$, $r = 0.33$, and i being either 3 or 4. The values we get for the propabilities of the most expected values are then 0.26 and 0.22 respectively. The average of all values within this table is 0.15, significantly lower than the probability of the expected value. Still, this is not enough to reject the null hypothesis as proportion probability analysis does not come with a rejection criteria.

	GuessesActive			GuessesEx			GuessesSamp		
	First	Second	Third	First	Second	Third	First	Second	Third
sample 1	0.09	0.23	0.14	0.06	0.20	0.20	0.26	0.23	0.26
sample 2	0.09	0.23	0.14	0.14	0.20	0.26	0.23	0.23	0.20
sample 3	0.09	0.26	0.06	0.02	0.26	0.02	0.20	0.23	0.23
sample 4	0.09	0.14	0.23	0.06	0.26	0.09	0.26	0.20	0.14
sample 5	0.02	0.06	0.23	0.00	0.20	0.02	0.20	0.20	0.06
sample 6	0.26	0.26	0.23	0.14	0.23	0.09	0.20	0.26	0.14
sample 7	0.23	0.26	0.26	0.23	0.23	0.20	0.20	0.26	0.14
sample 8	0.20	0.26	0.14	0.02	0.20	0.09	0.09	0.14	0.23
sample 9	0.09	0.26	0.06	0.26	0.14	0.20	0.06	0.20	0.20
sample 10	0.02	0.23	0.06	0.02	0.26	0.02	0.26	0.26	0.23
sample 11	0.02	0.06	0.23	0.02	0.26	0.02	0.26	0.09	0.06
sample 12	0.09	0.14	0.23	0.02	0.20	0.09	0.20	0.26	0.14
sample 13	0.26	0.26	0.23	0.14	0.23	0.09	0.20	0.26	0.14
sample 14	0.20	0.14	0.26	0.06	0.26	0.09	0.20	0.20	0.06
sample 15	0.20	0.09	0.02	0.23	0.06	0.02	0.23	0.26	0.26
sample 16	0.09	0.14	0.23	0.06	0.02	0.23	0.26	0.14	0.20
sample 17	0.09	0.23	0.14	0.06	0.23	0.02	0.26	0.20	0.14
sample 18	0.09	0.26	0.06	0.00	0.09	0.09	0.09	0.06	0.26
sample 19	0.09	0.23	0.14	0.02	0.26	0.02	0.20	0.26	0.14
sample 20	0.20	0.23	0.23	0.06	0.20	0.20	0.20	0.23	0.23
sample 21	0.09	0.20	0.02	0.23	0.06	0.02	0.14	0.20	0.26
sample 22	0.26	0.26	0.23	0.20	0.02	0.09	0.14	0.02	0.14
sample 23	0.26	0.23	0.26	0.06	0.23	0.02	0.09	0.20	0.02
sample 24	0.26	0.26	0.23	0.23	0.23	0.20	0.26	0.26	0.23
sample 25	0.20	0.06	0.20	0.02	0.26	0.02	0.09	0.09	0.00
sample 26	0.09	0.26	0.06	0.06	0.20	0.20	0.26	0.14	0.20
sample 27	0.02	0.20	0.09	0.26	0.14	0.20	0.02	0.26	0.02
sample 28	0.02	0.14	0.14	0.02	0.20	0.09	0.26	0.26	0.23
sample 29	0.09	0.20	0.02	0.23	0.14	0.09	0.14	0.26	0.20
sample 30	0.20	0.06	0.20	0.23	0.26	0.26	0.23	0.09	0.14

Table 4.3: Placement results proportion probabilities

	GuessesActive	GuessesEx	GuessesSamp
First	0.13	0.19	0.16
Second	0.10	0.19	0.11
Third	0.19	0.20	0.16

Table 4.4: Average of proportion probabilities across all samples

4.1.2 Analysis of Variance

Analysis of variance is a test that decomposes the variance in data and shows how much of it is due to random variations and how much is due to the influence of a factor [11]. It allows one to measure the probability that the means of a group of distributions is the same. It is often used as an exploratory tool to explain observations, and it is in this spirit that it is used here.

To perform anova, one requires the means of the groups under consideration, the variance of the groups under consideration, the "grand mean" of the groups under consideration (the mean of each group mean), and the "grand variance" of the groups under consideration (the normalized sum of the variances of the group). The grand mean and grand variance are then used to find the mean square value of the deviations of the variances within and between the groups. The test is then comprised of taking the ration between these values, with rejection or acceptance of the null hypothesis (in the case of anova almost always that the means are equal) then depending on how many "degrees of freedom" were involved in the calculation of the mean squared values.

Inspecting the representative equations clarifies the procedure. Consider first, the deviation of a value from its grand mean:

$$e_{j,k} = (x_{j,k} - \overline{x}_G),$$

where x is an observed value, e is its variance, j is a groups label and k is the within group label of the given value. Taking the sum of the squares of all of these across each of the groups gives us the sample grand variance:

$$s_G^2 = \frac{\sum_j \sum_k (x_{j,k} - \overline{x}_G)^2}{N-1} = \frac{SS_{total}}{df_{total}},$$

where the total degrees of freedom df is equal to the total number of items in all the groups N minus the one way that the variances are not free to vary 1 (the grand variance itself). The total sum

of squares SS_{total} can be broken up into two components: $SS_{total} = SS_{between} + SS_{within}$ where the sum of squares between the groups is equal to:

$$SS_{between} = \sum_j n_j(\overline{x}_j - \overline{x}_G)^2,$$

and the sum of squares within the groups is equal to:

$$SS_{within} = \sum_j \sum_k (x_{j,k} - \overline{x}_j)^2.$$

By taking the degrees of freedom between groups and within groups, we can obtain the mean square deviations between and within the groups via division. The degrees of freedom between the groups is equal to the number of groups j minus 1 and the number of degrees of freedom within the groups is equal to the total number of items N minus the number of groups those items are contained within j. The mean square deviations are then as follows:

$$MS_{between} = \frac{SS_{between}}{df_{between}} = \frac{\sum_j n_j(\overline{x}_j - \overline{x}_G)^2}{j-1};$$

$$MS_{within} = \frac{SS_{within}}{df_{within}} = \frac{\sum_j \sum_k (x_{j,k} - \overline{x}_j)^2}{N-j}.$$

The value then used to score the sample is the F statistic

$$F = \frac{MS_{between}}{MS_{within}}.$$

Notice that $MS_{between}$ and MS_{within} are variances - sums of squares divided by degrees of freedom - so F is a ratio of variances, and under H_0 (the null hypothesis) these variances are equal [11]. The distribution with which on compares this results is the F distribution and the values that determine the critical point within this distribution are $j-1$ and $N-j$, which are the degrees of

freedom of the F values numerator and denominator.

An analysis of variance of the results of this experiment was conducted by passing the results into an excel spread sheet then running excels native anova tool on the items. The results where as follows:

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
GA First	30	51	1.7	2.079310345
GE First	30	167	5.566666667	2.46091954
GS First	30	82	2.733333333	1.926436782

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	240.4666667	2	120.2333333	55.77835052	2.57697E-16	3.101295757
Within Groups	187.5333333	87	2.155555556			
Total	428	89				

Figure 4.1: ANOVA analysis of first placements

The probability of the "F" values of the first and second placement columns is increadibly low, leading one to suspect that we can reject the null hypothesis. This suspicion will be confirmed in formalized in the next section after we apply the t test.

4.1.3 t score

t score analysis is a form of hypothesis testing that allows one to determine whether or not some result emerged from some given distribution via consideration of how many standard deviations the result deviates from the mean of said given distribution. Its equation has the form:

$$t = \frac{\overline{x} - \mu}{\hat{\sigma}_{\overline{x}}} = \frac{\overline{x} - \mu}{\frac{s}{\sqrt{N}}}$$

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
GA Second	30	114	3.8	1.751724
GE Second	30	99	3.3	2.424138
GS Second	30	87	2.9	1.817241

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	12.2	2	6.1	3.05351	0.052282	3.101295757
Within Groups	173.8	87	1.997701			
Total	186	89				

Figure 4.2: ANOVA analysis of second placements

where s is the sample standard deviation, N is the number of samples, $\overline{x}$ is an individual samples mean/calculated value, and μ is the mean of the distribution of comparison.

The decision as to whether or not a specific t score value implies a result is from a different distribution depends on how many samples were used in the calculation of the sample mean and on what the desired confidence interval is. The critical threshold used in order to make this decision is gained by the use of t distribution table. In the case where 30 samples are used and the desired margin of error is 5%, the critical thresholds for a two tailed t test are -2.042 and 2.042. If the averaged values of the t scores falls outside of these bounds then we can reject the null hypothesis with a 5 % margin of error. The standard deviations for each of the samples can be seen in table 4.5. Taking these with the average placement results available in table 4.2 allows the calculating of the desired t scores, which can be seen in table 4.6. Taking the average of the absolute value of each of these t scores yeilds 5.11. We can thus comfortably reject the null hypothesis.

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
GA Third	30	135	4.5	2.258621
GE Third	30	34	1.133333	1.154023
GS Third	30	131	4.366667	2.585057

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	218.0666667	2	109.0333	**54.53756**	**4.45E-16**	**3.101296**
Within Groups	173.9333333	87	1.999234			
Total	392	89				

Figure 4.3: ANOVA analysis of third placements

	GuessesActive	GuessesEx	GuessesSamp
First	1.42	1.30	1.48
Second	1.54	1.53	1.06
Third	1.36	1.33	1.58

Table 4.5: Placement results standard deviations across all samples

	GuessesActive	GuessesEx	GuessesSamp
First	-6.29	1.98	4.32
Second	7.97	-0.11	-11.36
Third	-2.42	-1.77	3.61

Table 4.6: t scores of placement averages

CHAPTER 5: CONCLUSION AND FUTURE WORK

In this thesis, I proposed that the no free lunch hypotheis might not apply to meta learning algorithms. In order to test this hypothesis, I first built a system to determine the accuracy of three meta learning strategies: Exhaustive, Active, and Sampling. To use these strategies, a base of datasets would first be randomly choosen from the collection of all available datasets that had been gathered from the UCI Irvine data repository. Each strategy would then be carried out on the metabase and make an estimate as to what algorithm would result in the highest classification accuracy. Each algorithm would make this guess for each dataset in the collection of available datasets excluding the datasets within the current metabase. A new metabase would then be choosen at random and the process would be repeated 9 more times, giving a number between 0 and 10 for how many times each algorithm got the First, second, or third most correct guesses. This process was repeated 30 times, resulting in 30 samples. t test analysis was then performed, giving an an average among the absolute values of each of the position t scores of 5.11, allowing us to reject the null hypothesis at a 5 percent margin of error.

Future work could involve repeat runs of the system described in this thesis with more datasets, which would test the results in a wider variety of problem domains, therefore strengthening the results. Other interesting variations of this experiment could come from testing the results with diffrent meta features, testing the results with noisy datasets, and even testing the results with artificially generated datasets. One could also test the results of this procedure with an intermediary parameter tuning step, that is to say we could tune the parameters of the various base algorithms to fit the domain of the dataset currently being analyzed. Also, a future researcher might also choose to vary the algorithm used to classify the meta features, as here we make use of the k-means clustring algorithm to make our choice from the meta features of our modified meta bases and the selection of a different algorithm could change the results.

REFERENCES

[1] A. Gupta, "An introduction to meta-learning," *medium*, [Online]. Available: https://medium.com/walmartlabs/an-introduction-to-meta-learning-ced7072b80e7.

[2] P. B. Ricardo Vilalta Christophe Giraurd-Carrier and C. Soares, "Using meta-learning to support data mining," *International Journal of Computer Science and Applications*, vol. 1, no. 1, pp. 31–45, 2004.

[3] T. Bhatt and G. Bhatt, "Ranking of classifiers based on dataset characteristics using active meta learning," *International Journal of Computer Applications*, vol. 69, no. 20, pp. 31–36, 2013. [Online]. Available: https://pdfs.semanticscholar.org/93ed/31bf211362c7b77f254d0fb8d5f2d1a101a8pdf.

[4] R. Leite and P. Brazdil, "Predicting relative performance of classifiers from samples," *ICML 05 Preceedings of the 22nd International conference on Machine learning*, pp. 497–503, Aug. 2005. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.74.641&rep=rep1&type=pdf.

[5] K. P. Murphy, *Machine Learning: A probabilistic Perspective*.

[6] V. Vapnik and C. Cortes, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995. [Online]. Available: https://link.springer.com/article/10.1007/BF00994018.

[7] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Fifth Berkeley Symp on Math Statistics and Probability*, vol. 1, pp. 281–197, 1967. [Online]. Available: https://projecteuclid.org/euclid.bsmsp/1200512992.

[8] D. Arthur and S. Vassilvitskii, "K-means ++: The advantages of careful seeding," *SODA 07 Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms*, pp. 1027–1035, Jan. 2007. [Online]. Available: http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf.

[9] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, pp. 251–257, 1991. [Online]. Available: http://zmjones.com/static/statistical-learning/hornik-nn-1991.pdf.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[11] P. R. Cohen, *Empirical Methods for Artificial Intelligence*.

APPENDIX

The tables in this appendix describe the datasets used in this experiment. For a more thorough description of the datasets described here, feel free to visit UCI Irvine's dataset repository at https://archive.ics.uci.edu.

	data_name	weighted_mean	coefficient_variation	fpskew	kurtosis	entropy	metric_time
1	diagnosis.data	0.455596	1.081471	0.209499	-1.630211	4.029086	0.004
2	adult.data	0.249230	1.759203	2.333778	17.031884	6.565645	0.038
3	anneal.data	0.111052	4.215940	4.141710	50.030589	3.403991	0.052
4	arrhythmia.data	0.086959	11.152305	3.879379	77.969921	2.415850	0.272
5	imports-85.data	0.436747	1.075070	1.133555	3.276080	4.564727	0.017
6	auto-mpg.data	0.573606	0.360857	0.491209	-0.599355	5.904698	0.007
7	balance-scale.data	0.618240	0.467817	-0.100944	-1.169594	6.314346	0.005
8	transfusion.data	0.187368	1.093471	2.052989	8.021527	6.053162	0.008
9	breast-cancer-wisconsin.data	0.306162	0.887600	2.588329	24.762927	6.190649	0.011
10	wdbc.data	0.330615	0.608651	1.812627	8.505057	6.142504	0.033
11	wpbc.data	0.426710	0.450751	1.121284	2.439887	5.149098	0.025
12	bridgesVer1.data	0.458340	0.852633	0.184317	-0.334123	4.174636	0.009
13	bridgesVer2.data	0.436147	0.879386	0.145149	-0.337846	4.170606	0.006
14	car.data	0.448440	0.925115	0.284561	-0.709155	6.908229	0.015
15	census-income.data	0.248851	1.801047	2.210776	12.786901	6.541483	0.037
16	cmc.data	0.609695	0.726334	-0.106305	0.663389	6.925342	0.020
17	analysis.data	0.235192	1.239534	1.984866	14.175971	4.636233	0.011
18	eval.data	0.358205	0.872081	0.509958	6.643169	4.522067	0.008
19	results.data	0.279459	1.189090	1.288467	7.489623	4.288263	0.010
20	communities.data	0.314464	1.040879	1.659675	6.856834	7.137863	0.366
21	connect-4.data	0.141390	4.750136	4.556722	102.786091	3.472028	0.107
22	covtype.data	0.132469	7.777535	7.563910	206.785361	2.977847	0.148
23	machine.data	0.192442	1.221299	2.632259	11.643295	4.808715	0.006
24	crx.data	0.248056	1.626685	2.141604	18.265535	5.536108	0.019
25	dermatology.data	0.261512	1.715460	1.373823	2.344857	4.670429	0.029
26	ecoli.data	0.493172	0.424508	2.888568	39.583253	5.667176	0.008
27	CalIt2.data	0.495292	0.729640	0.521587	0.166213	7.233784	0.011
28	Dodgers.data	0.435739	0.709887	0.093486	-1.188387	4.904294	0.007
29	flag.data	0.284432	1.772677	1.929493	10.050680	4.105736	0.020
30	glass.data	0.494146	0.753661	1.443271	7.759524	4.954772	0.010
31	haberman.data	0.563177	0.597221	1.064999	2.234696	5.430365	0.002
32	hayes-roth.data	0.541667	0.474571	0.415503	-0.847602	4.766142	0.003
33	hepatitis.data	0.366565	1.052046	0.658949	0.505284	4.357252	0.014
34	Hill_Valley_without_noise_Testing.data	0.502528	0.575874	-0.000808	-1.206860	6.215904	0.123
35	Hill_Valley_without_noise_Training.data	0.502447	0.575911	-0.000156	-1.206924	6.215895	0.127
36	Hill_Valley_with_noise_Testing.data	0.497953	0.581528	0.021738	-1.210553	6.212843	0.133
37	Hill_Valley_with_noise_Training.data	0.498707	0.580824	0.016880	-1.208480	6.213012	0.132
38	housing.data	0.462123	0.904626	0.903890	4.078481	5.744754	0.016
39	ad.data	0.008945	14.061083	13.985260	292.134957	2.138439	5.157
40	ionosphere.data	0.250876	14.014291	-0.531918	0.186610	0.302357	0.029
41	bezdekIris.data	0.591776	0.440286	0.050692	-0.924839	4.841930	0.003
42	iris.data	0.591591	0.440329	0.053360	-0.912546	4.841726	0.003
43	lenses.data	0.709491	0.378707	-0.164451	-1.548223	3.096876	0.004
44	letter-recognition.data	0.413347	0.443411	0.299376	0.576037	7.477820	0.047

Table A.1: Dataset metrics

	data_name	weighted_mean	coefficient_variation	fpskew	kurtosis	entropy	metric_time
45	movement_libras.data	0.573426	0.346463	-0.184549	-0.720449	5.819943	0.085
46	movement_libras_1.data	0.628648	0.320631	-0.213415	-0.511945	3.749292	0.047
47	movement_libras_10.data	0.588077	0.343811	-0.217365	-0.676963	5.532706	0.077
48	movement_libras_5.data	0.651979	0.314020	-0.450573	-0.622841	4.442975	0.047
49	movement_libras_8.data	0.622662	0.321201	-0.390150	-0.552621	4.846960	0.068
50	movement_libras_9.data	0.629232	0.328592	-0.139975	-0.693618	3.747229	0.044
51	bupa.data	0.424263	0.523013	1.394949	5.265461	5.685315	0.005
52	lungCancer.data	0.632584	0.485952	0.179886	1.245987	3.274193	0.030
53	magic04.data	0.238056	8.002667	0.618631	2.052942	4.736776	0.029
54	mammographic_masses.data	0.320325	1.147384	1.232022	3.040999	6.186567	0.008
55	meta.data	0.304250	1.460049	2.092136	13.126212	5.565504	0.023
56	agaricusLepiota.data	0.295192	1.101141	0.496554	0.453252	5.538493	0.062
57	nursery.data	0.422444	0.720314	0.170619	-1.499947	6.349519	0.022
58	eighthr.data	0.411486	0.673489	0.266232	-0.749786	7.335710	0.195
59	onehr.data	0.418501	0.688123	0.257433	-0.644747	7.328529	0.198
60	parkinsons.data	0.491494	0.575462	0.106011	-1.049867	5.086222	0.016
61	poker-hand-testing.data	0.537438	0.561783	0.160573	-0.652672	7.404570	0.029
62	poker-hand-training-true.data	0.536591	0.577614	0.380960	1.559063	7.403507	0.028
63	post-operative.data	0.322428	1.418410	1.098058	2.215768	3.521178	0.008
64	semeion.data	0.320030	1.572189	0.878478	-0.735656	6.158535	0.736
65	servo.data	0.484915	0.652535	0.539930	-0.468594	4.865617	0.003
66	backup-large.data	0.253132	9.368737	-0.027662	1.757700	0.309845	0.031
67	soybean-large.data	0.286443	1.482283	1.023199	1.995048	4.713094	0.029
68	soybean-small.data	0.418637	0.664157	0.199064	-1.632732	2.441425	0.019
69	o-ring-erosion-only.data	0.658722	0.577226	0.054172	-0.782238	2.799334	0.002
70	o-ring-erosion-or-blowby.data	0.667418	0.546196	0.009550	-0.976233	2.827592	0.003
71	spambase.data	0.041449	6.175640	10.681459	209.998626	4.958273	0.188
72	sponge.data	0.255833	2.218192	1.841016	7.699819	2.911164	0.024
73	synthetic_control.data	0.578865	0.324330	0.033920	-0.562755	5.497560	0.069
74	tae.data	0.630134	0.440100	-0.364775	-0.228237	4.894267	0.003
75	allbp.data	0.120862	3.955484	3.825354	28.037094	4.693295	0.080
76	allhyper.data	0.121666	5.231252	5.096495	68.004142	4.688492	0.078
77	allhypo.data	0.121380	5.210236	5.082337	89.276533	4.738580	0.076
78	allrep.data	0.119879	5.584221	5.451824	97.279147	4.659462	0.079
79	ann-test.data	0.164714	4.130476	4.422860	37.150889	4.989043	0.054
80	ann-train.data	0.155563	6.196652	6.756284	104.155015	4.888621	0.073
81	new-thyroid.data	0.326651	0.971352	2.418092	11.780620	4.260200	0.003
82	tic-tac-toe.data	0.435647	0.952777	0.237814	-1.389979	6.265111	0.014
83	house_votes_84.data	0.375727	0.989249	0.266405	-1.044629	5.420079	0.016
84	water-treatment.data	0.409096	0.661534	0.634270	0.703112	6.000097	0.042
85	wine.data	0.568132	0.301297	0.330019	-0.119284	5.127177	0.012
86	yeast.data	0.398247	1.341340	2.446721	22.775247	6.751428	0.023
87	zoo.data	0.432885	1.280205	0.398740	-0.212447	3.715472	0.013
88	abalone.data	0.420168	0.495720	0.272209	0.204785	7.435145	0.027

Table A.2: Dataset metrics cont.

	data_name	description
1	diagnosis.data	Dataset used to test for Acute nephritis
2	adult.data	Dataset used to predict if someone makes more than 50k a year
3	anneal.data	Description not given by authors
4	arrhythmia.data	Dataset used to determine presence of cardiac arrhythmia
5	imports-85.data	Dataset used for auto insurance risk analysis
6	auto-mpg.data	Dataset used to determine city-cycle fuel consumption
7	balance-scale.data	Dataset used to model psychological experimental results
8	transfusion.data	Dataset used to predict who donates blood
9	breast-cancer-wisconsin.data	Dataset used to predict whether breast cancer is benign or malignant
10	wdbc.data	Dataset used to predict whether breast cancer is benign or malignant
11	wpbc.data	Dataset used to predict whether breast cancer is benign or malignant
12	bridgesVer1.data	Dataset used to predict bridge parameters
13	bridgesVer2.data	Dataset used to predict bridge parameters
14	car.data	Dataset used to predict car safety
15	census-income.data	Dataset used to predict if someone makes more than 50k a year
16	cmc.data	Dataset used to predict womens choice of contraception
17	analysis.data	Dataset used to predict water quality
18	eval.data	Dataset used to predict water quality
19	results.data	Dataset used to predict water quality
20	communities.data	Dataset used to predict crime rates
21	connect-4.data	Contains all possible 8-ply connect 4 positions
22	covtype.data	Description not provided by author
23	machine.data	Dataset used to predict relative cpu performance
24	crx.data	A dataset of credit card information
25	dermatology.data	Dataset contains information needed to predict erythemato-squamous disease
26	ecoli.data	Dataset used to predict the cellular localization sites of proteins
27	CalIt2.data	Dataset used to predict unique events in UCI conference building
28	Dodgers.data	Dataset used to predict unique events at Dodgers stadium
29	flag.data	Dataset contain metrics on various countries flags
30	glass.data	Dataset used to categorize glass type
31	haberman.data	Dataset detailing breast cancer survival rates
32	hayes-roth.data	Dataset used to identify hobbies
33	hepatitis.data	Dataset used to diagnose hepatitis
34	Hill_Valley_without_noise_Testing.data	Dataset used to represent hills and valleys
35	Hill_Valley_without_noise_Training.data	Dataset used to represent hills and valleys
36	Hill_Valley_with_noise_Testing.data	Dataset used to represent hills and valleys
37	Hill_Valley_with_noise_Training.data	Dataset used to represent hills and valleys
38	housing.data	Dataset used to predict value of homes
39	ad.data	Description not provided by authors
40	ionosphere.data	Dataset used to determine presense of structures in ionosphere
41	bezdekIris.data	Dataset used to classify flowers
42	iris.data	Dataset used to classify flowers
43	lenses.data	Dataset for fitting contact lenses
44	letter-recognition.data	Dataset used to identify letters and numbers

Table A.3: Dataset descriptions

	data_name	description
45	movement_libras.data	Dataset used to map hand movements to brazilian sign language
46	movement_libras_1.data	Dataset used to map hand movements to brazilian sign language
47	movement_libras_10.data	Dataset used to map hand movements to brazilian sign languag
48	movement_libras_5.data	Dataset used to map hand movements to brazilian sign languag
49	movement_libras_8.data	Dataset used to map hand movements to brazilian sign languag
50	movement_libras_9.data	Dataset used to map hand movements to brazilian sign languag
51	bupa.data	Dataset used to identify liver disorders
52	lungCancer.data	Dataset used to identify types of lung cancer
53	magic04.data	Dataset used to classify radiation spectra
54	mammographic_masses.data	Dataset used to classify breast tumors as benign or malignant
55	meta.data	Dataset used for meta machine learning experiment at University of Porto
56	agaricusLepiota.data	Dataset used to classify edibility of mushrooms
57	nursery.data	Dataset used to predict health of infants
58	eighthr.data	Dataset used to predict windspeeds
59	onehr.data	Dataset used to predict windspeeds
60	parkinsons.data	Dataset used to identify parkinson's disease from voice signatures
61	poker-hand-testing.data	Dataset of poker hands
62	poker-hand-training-true.data	Dataset of poker hands
63	post-operative.data	Dataset used to predict leaving patients level of care
64	semeion.data	Handwritten digit dataset
65	servo.data	Dataset used to predict rise time of a servo mechanism
66	backup-large.data	Dataset used to predict disease in soybeans
67	soybean-large.data	Dataset used to predict disease in soybean
68	soybean-small.data	Dataset used to predict disease in soybean
69	o-ring-erosion-only.data	Dataset used to predict o-ring distress in spacecraft
70	o-ring-erosion-or-blowby.data	Dataset used to predict o-ring distress in spacecraft
71	spambase.data	Dataset used to classify spam emails
72	sponge.data	Dataset used to classify marine sponges
73	synthetic_control.data	Description not provided by authors
74	tae.data	Dataset used to classify english speaking ability of teacher's assistants
75	allbp.data	Dataset used to predict thyroid disease
76	allhyper.data	Dataset used to predict thyroid disease
77	allhypo.data	Dataset used to predict thyroid disease
78	allrep.data	Dataset used to predict thyroid disease
79	ann-test.data	Dataset used to predict thyroid disease
80	ann-train.data	Dataset used to predict thyroid disease
81	new-thyroid.data	Dataset used to predict thyroid disease
82	tic-tac-toe.data	Dataset containing all possible tic tac toe positions and who wins
83	house_votes_84.data	Dataset containing voting records of US congressmen
84	water-treatment.data	Dataset used to predict water treatment plant faults
85	wine.data	Dataset used to predict wine type
86	yeast.data	Dataset used to predict protein localization sites
87	zoo.data	Dataset used to classify animals
88	abalone.data	Predicting the age of abalone from physical measurements

Table A.4: Dataset descriptions cont.