Ontology-based Semantics vs Meta-learning for Predictive Big Data Analytics

by

Mustafa Veysi Nural

(Under the Direction of John A. Miller)

Abstract

Predictive analytics in the big data era is taking on an ever increasingly important role. Issues related to choice on modeling technique, estimation procedure (or algorithm) and efficient execution can present significant challenges. For example, the selection of appropriate and most predictive models (*i.e.*, the models that maximize the chosen performance criteria such as lowest error) for big data analytics often requires careful investigation and considerable expertise which might not always be readily available. In this thesis, we propose two alternative methods to assist data analysts and data scientists in selecting appropriate modeling techniques and building specific models as well as the rationale for the techniques and models selected.

The first approach uses ontology-based semantics to assist selecting the most predictive model for a given dataset. To formally describe the modeling techniques, models, and results, we developed the Analytics Ontology that supports inferencing for semi-automated model selection. The ScalaTion framework, which currently supports over sixty modeling techniques for big data analytics, is used as a testbed for evaluating the use of semantic technology.

In the second approach, we present a meta-learning system for selecting the most predictive regression algorithm in a predictive big data analytics setting. The meta-learning system uses meta-features characterizing the aspects of the dataset to select most predictive modeling techniques for that dataset. We show that our meta-learning system provides promising performance in predicting top performing modeling techniques for a given dataset. In addition to evaluating the system against existing baseline approaches, we also compare meta-learning approach with the ontology-assisted suggestion engine.

Finally, we present detailed performance analysis of the regression algorithms, namely Lasso and Ridge Regression, that we have implemented in ScalaTion and show that they provide robust performance compared to R, both in terms of training time and error.

Index words: predictive big data analytics; automated modeling; meta-learning; ontology-based semantics; machine learning.

Ontology-based Semantics vs Meta-learning for Predictive Big Data Analytics

by

Mustafa Veysi Nural

B.S., Fatih University, Turkey, 2008

A Dissertation Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2017

Ontology-based Semantics vs Meta-learning for Predictive Big Data Analytics

by

Mustafa Veysi Nural

To my wife..

Acknowledgments

This dissertation would not come to existence without the continuous support and prayers from my family and especially my beloved wife Esra. She has generously sacrificed so much while patiently allowing me spend countless nights and weekends in the lab. She has been the greatest source of motivation during the times I needed the most.

Additionally, my advisor, Dr. John Miller has graciously extended his support in the most critical moments during my studies and never hesitated to sit together for countless hours in the lab (sometimes even through late weekend nights) to overcome obstacles encountered along the way. I will forever be indebted for his never-ending support and confidence in me as this thesis came together.

I would like to specially thank my supervisor and committee member Dr. Jessie Kissinger and my colleagues for supporting me and providing great flexibility at work as I have been preparing my dissertation.

I would like to thank my committee members Drs. Arabnia, Kissinger, and Rasheed for their valuable feedback and early guidance during my candidacy. I am also grateful for Dr. Budak Arpinar for his treasured mentorship in the very early days of my graduate studies and for supporting me while I was taking my baby steps into research.

Finally, I would like to thank my lab mates Hao Peng and Michael Cotterell for their willingness to extend their help and expertise without hesitation anytime I've needed them. Their friendly faces will always be missed.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Data scientists are becoming increasingly overwhelmed by dealing with the quantities of data they are being exposed to. As technology to collect and store data becomes cheaper, the amount of data readily available for analysis and decision making grows steadily. In a data-driven environment, quicker turnaround times are required to keep up with the constant flow of data. As there are hundreds of popular modeling techniques in the fields of Statistics and Machine Learning (ScalaTion already supports over sixty), how can one decide? With smaller data sets and high expertise on the part of the analyst, one practice is to try all possible models for a set of preferred techniques. For big data and less experienced analysts, this practice cannot be relied upon. Thus, leveraging automation for predictive big data analytics becomes tremendously important.

Predictive analytics can be defined as the process of building a statistical model to capture the relationships between variables in order to make sense of or to predict future outcomes from data. Although classification is similar to prediction, it tries to model a binary/categorical response as the outcome.

Predictive big data analytics is a non-trivial and a highly iterative task requiring domain knowledge, expertise in Statistics, and often times, familiarity with programming and big data frameworks. In this thesis, we explore leveraging the success of ontology-based semantics and meta-learning in other problem domains to help fully or partially automate predictive big data analytics workflows, providing assistance to data scientists, analysts, or even domain experts regardless of their background.

We use the ScalaTion [Miller et al., 2013] framework to develop, test, and integrate the solutions presented in our work. We have developed the Analytics Ontology in Web Ontology Language (OWL) to formally describe modeling techniques, capture domain expertise which allows using logical reasoning to perform inference for automated model selection. The ontology covers popular predictive modeling techniques including Generalized Linear Models, Regularization models such as Lasso, Ridge and Partial Least Squares Regression among others such as time series analysis models. Using the ontology backend, we were able to build `scala-dash`, a graphical tool to load a dataset, to automatically extract the key characteristics and to suggest suitable regression algorithms to build and analyze predictive models. As the suggestions were generated by an ontological reasoner, the justifications for each suggestion are provided as well.

We have also built a machine learning based meta-learning system for performing automated model selection. A meta-learner is built using quality-of-fit metrics and 21 meta-features from 114 datasets to automatically predict the best performing algorithm among 15 regression algorithms. Evaluation of the system suggests promising results for using meta-learning in the context of predictive analytics (*i.e.,* regression algorithms).

The rest of the thesis is organized as follows.

Chapter 2 provides an introduction to predictive analytics and presents the current state-of-the-art approaches for automated predictive analytics.

Chapter 3 presents our manuscript published in IEEE International Journal of Big Data [Nural et al., 2015] which introduces ontology-based semantics for assisting predictive big data analytics workflows.

Chapter 4 covers our work on meta-learning based suggestion of appropriate modeling techniques. The work presented in this chapter has been partially published in the Proceedings of IEEE Big Data Conference 2017, Special Session on Intelligent Data Mining.

Finally, **Chapter 5** concludes this thesis and provides a summary of the presented work.

Chapter 2

Background

A predictive analytics problem may generally be modeled as

$$y \;=\; f(\mathbf{x}; \boldsymbol{\beta}) + \epsilon$$

where the response variable y is modeled as a function of the vector of predictor variables $\mathbf{x}$ (also commonly referred as *features* or *attributes*) and the $\boldsymbol{\beta}$ parameter/coefficient vector plus the ϵ error/residual term representing what the model does not account for. The parameter/coefficient vector $\boldsymbol{\beta}$ can be estimated by collecting training data (*e.g.*, a response vector $\mathbf{y}$ and a data matrix X) and minimizing some norm of the error/residual vector $\|\boldsymbol{\epsilon}\|$. For a linear model and L_2 norms, this involves using matrix factorization to solve the normal equations, $X^t X\boldsymbol{\beta} = X^t\mathbf{y}$. More general optimization techniques are required in other cases.

In almost all real life scenarios, predictive analytics involve many steps and are highly iterative exercises. Figure 2.1 shows a typical workflow for performing predictive big data analytics.

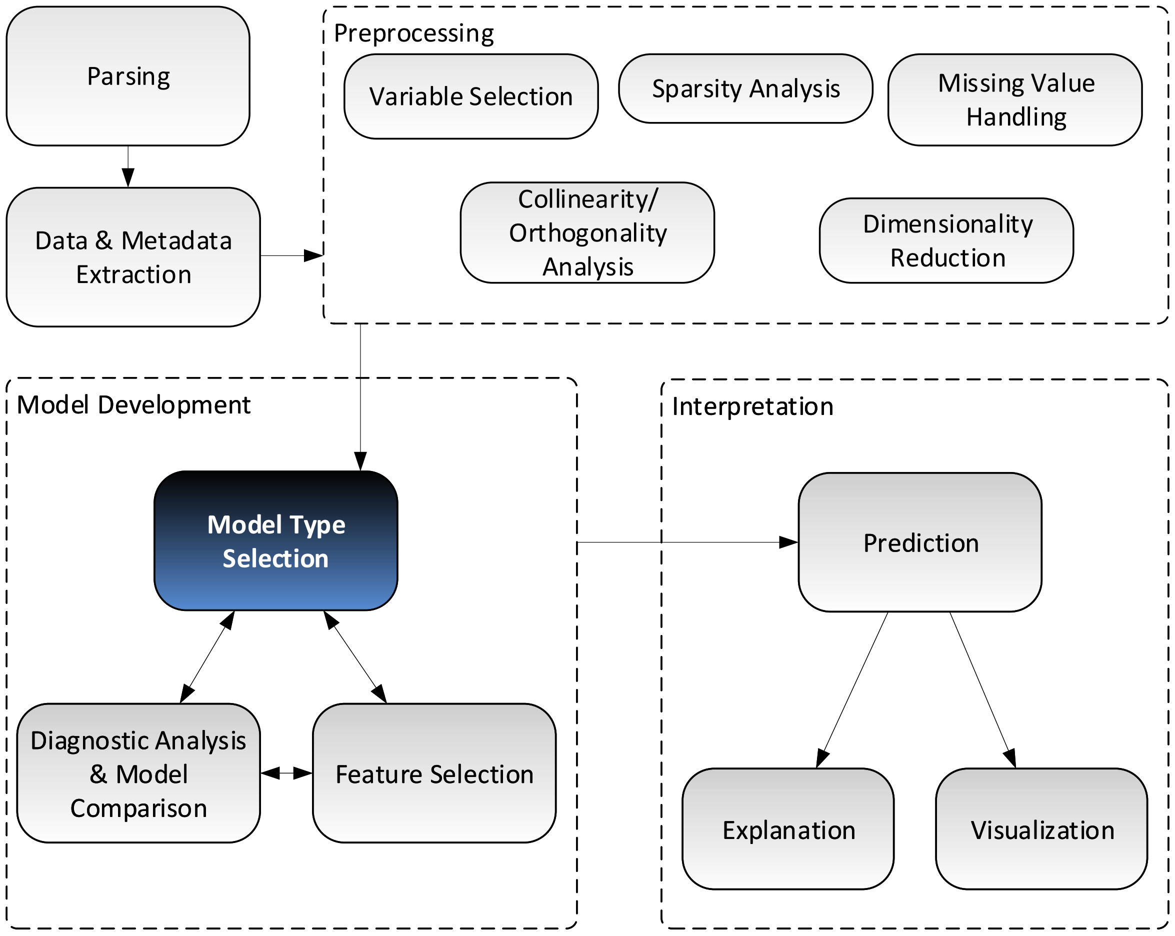


Figure 2.1: Sample Predictive Analytics Workflow

Preprocessing

Preprocessing plays a significant role for building successful predictive models. At the very least, it involves preparing the training dataset for the input requirements of the target algorithms/techniques.

For example, most algorithms cannot recognize dates, thus, treating them as strings. For that reason, a date needs to be properly encoded. One method is to split the date into three variables: year, month, and day. If the ordering is important then the date might be encoded as the day of the year (*e.g.*, *January 15th* becomes 15 whereas *July 27th* becomes 207). The choice of which encoding will be used usually depends on the inherent meaning that the date carries with respect to the other variables in the dataset. If the data includes an ID string, a variable that uniquely identifies each instance of the dataset, it should be removed.

Missing values are another challenge that must be handled before running any modeling algorithm, as they would fail. There are a number of different techniques for dealing with missing values. These range from naive approaches such as removing instances/variables containing missing values to more advance methods such as utilizing an imputation function based on the characteristics of a variable to replace the missing values.

In addition to meeting the input requirements of a target modeling algorithm, preprocessing can also be performed to increase model performance. A few of the most commonly used approaches include feature selection, dimensionality reduction, outlier analysis and removal, sparsity analysis and etc.

Model Development

Model development is a highly iterative process involving selecting a suitable model-type (*i.e.,* modeling technique/algorithm), building and performing model diagnostics, and feature selection. In most use cases, many models are built and later on compared.

Selecting a suitable modeling algorithm is a challenging task as there are literally hundreds of different algorithms for performing predictive analytics. Different modeling techniques have certain assumptions that the dataset must satisfy (*e.g.,* OLS requires a tall input matrix $m \geq k$) in order to make effective predictions. For a given dataset, certain modeling techniques may be automatically eliminated due to the violations of their assumptions. However, it is still a daunting task to choose an appropriate modeling technique among the remaining as this often times involves building a model for each candidate technique. Afterwards, metrics such as the R^2 values and cross-validated errors may be used to select the best or top modeling techniques. We discuss different approaches for automating this task in more detail in Section 2.1.

Feature selection is usually performed during model development to improve predictive power of a model by eliminating unnecessary features and hence reducing the risk of overfitting. This procedure is often carried out during the process of model comparisons but may also be done during preprocessing or model-type selection via selecting a modeling technique with implicit feature selection (*e.g.,* Lasso Regression).

Interpretation

Once the most predictive (or sufficiently good) model is found during the model development phase, it is deployed to production for prediction. The explanation and visualization of predictions are equally important in many contexts. For example, recent regulations introduce mandates such as "right to explanation" [Goodman and Flaxman, 2016] that leverage individuals' rights to ask for an explanation when they are affected by an algorithmic decision-making process (*e.g.,* credit approval based on user-level predictors). Some modeling techniques such as regression provide inherent explanatory capabilities and therefore are preferred when explanatory power is a requirement. On the other hand, there are efforts [Ribeiro et al., 2016] to provide explanation for prediction models even when the model itself is opaque, that is, being inherently difficult to interpret how the model predicts. Similarly, visualization helps interpreting prediction results and quickly identify insights from data.

2.1 Model Type Selection

Finding the most predictive (or sufficiently good) modeling algorithm is a non-trivial task as it often times involves building and running models with each of the different modeling algorithms. Automating this process has been studied extensively and resulted in a multitude of approaches that will be covered below.

Exhaustive Approach

As the name suggests, one of the simplest ways to find the most predictive modeling algorithm for a given dataset is to exhaustively try building a model for each technique and pick the best performing one(s). In fact, many analytics software packages and frameworks (both commercial and open-source) provide tools to automate this process. Typically, the user chooses an evaluation metric and the techniques of interest (or simply all). The tool then builds a model for every possible technique, evaluates each model by the metric of choice (*e.g.,* root mean squared error) and returns the model with the highest performance to the user as a result. As the dataset size increases however, training each model becomes much more expensive. It is not uncommon to encounter a model taking hours and even days for training in practice. Considering the fact that some tools have hundreds of modeling algorithms available[1], it is clear how this approach would quickly become impractical.

[1]The `caret` package lists 240 modeling algorithms that are available in R ecosystem. See `http://topepo.github.io/caret/available-models.html`

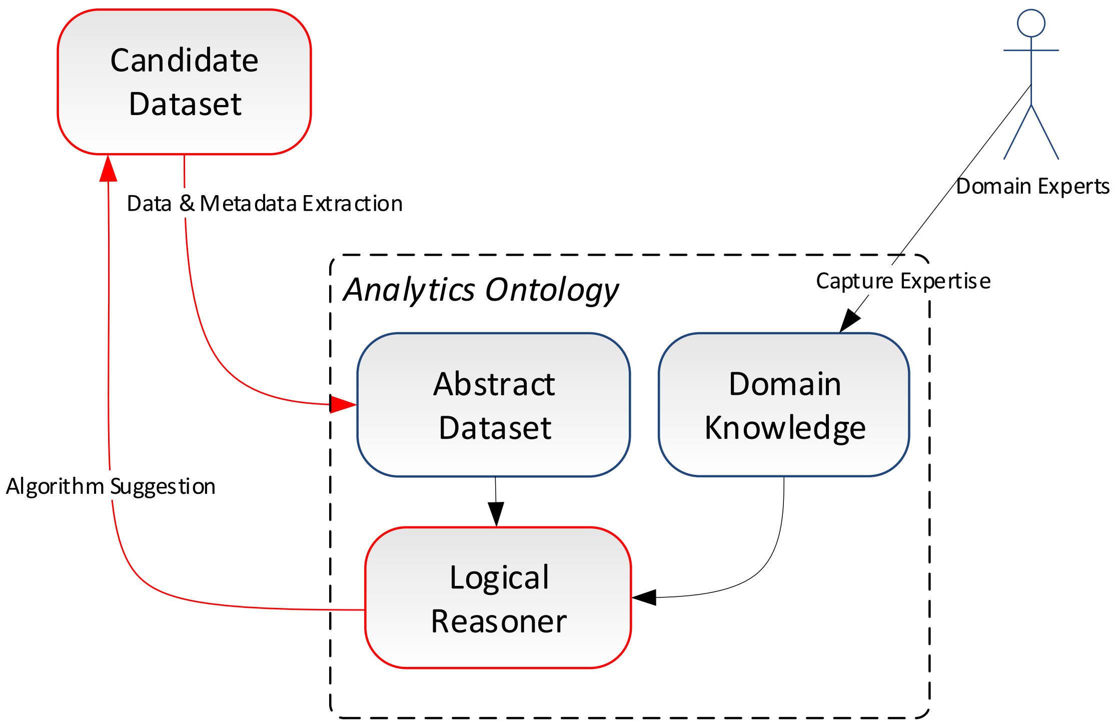


Figure 2.2: Setup for Ontology-based Model Suggestion

Ontology-based Approach

A well-designed ontology can assist with the model selection problem in a number of ways. By capturing domain expertise expressed in a formal structure, one can use logical reasoning to reduce search space for finding the most predictive model type for a given dataset. Additionally, using a logical reasoner also makes it possible to provide justifications for the suggestions provided.

Figure 2.2 shows the basic setup for how we have used ontology-based semantics to provide model type suggestions.

First, domain knowledge from the experts has to be captured formally. Next, a dataset

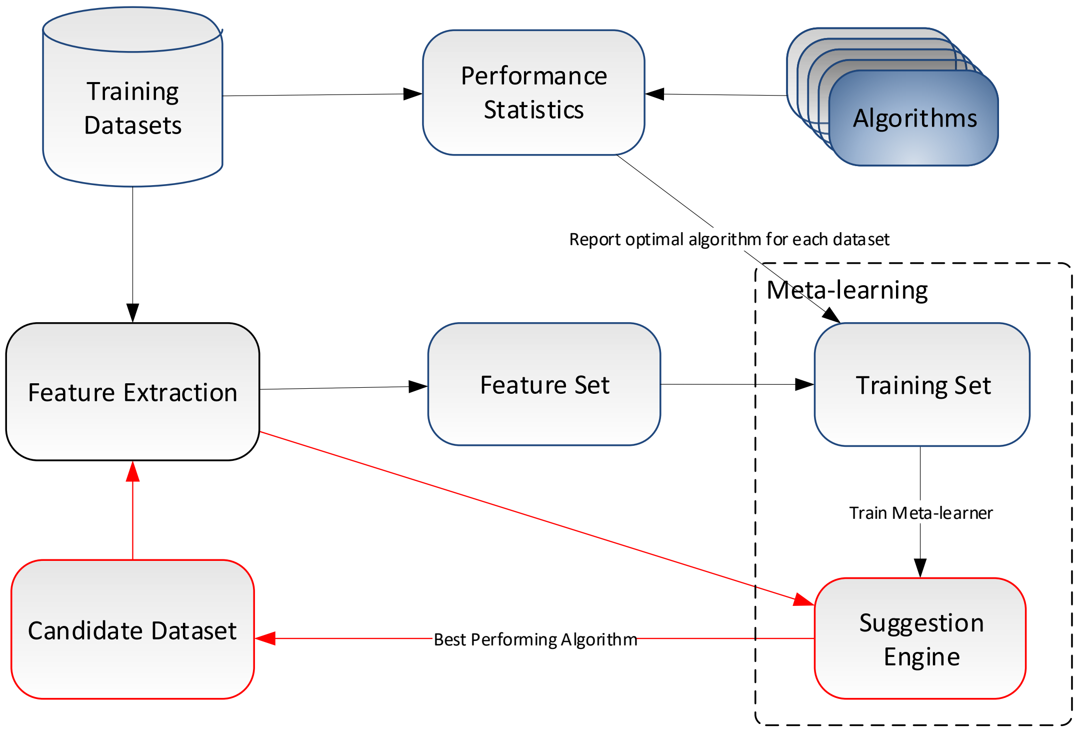


Figure 2.3: Setup for Meta-learning Based Model Suggestion

should be represented in the ontology. This is done by creating an abstract representation of the candidate dataset in the ontology which involves automatically extracting representative features from the dataset itself and from metadata if available.

A logical reasoner would then be able to infer suitable model types for the candidate dataset from the ontology using domain knowledge encoded in formal rules and ontological axioms.

Details and implementation of this approach have been presented in Chapter 3.

Meta-learning based Approach

Meta-learning can be briefly described as "learning to learn" or "learning about learning". There are large number of machine learning algorithms (*i.e.,* learners) one can choose among for a given problem. According to the no free lunch theorems for optimization [Wolpert and Macready, 1997], no single algorithm outperforms other algorithms for all different classes of problems. Therefore, identifying the most predictive learning algorithm for a given problem itself can be treated as a learning problem. Hence, the term first appeared in the field of machine learning in 90's to describe efforts for "learning" a mapping between datasets and machine learning techniques to achieve the best performance. Since then, it has been studied extensively, focusing mostly on classification problems (*i.e.*, selecting the top-performing classifier).

A typical meta-learning system is developed similarly to the workflow displayed in Figure 2.3. First, one needs to collect datasets representative of the problem domain. Next, a fixed set of meta-features should be extracted from each dataset. The number and type of the meta-features should relate to the nature of the task. For example, one might choose to extract information-theoretic meta-features such as class entropy for the response variable for a classification problem whereas statistical meta-features such as skewness and kurtosis of the response might be more appropriate for a regression problem. Then, depending on the chosen metric, performance statistics should be collected in order to rank algorithms based on performance. A meta-learner is then trained with the meta-features and the performance statistics. After a meta-learner is trained with the training data, it is then possible to use it in production for suggesting top modeling algorithm(s) for future candidate datasets.

A meta-learner can be built in many different ways. For example, it is possible to build a regression model per each target technique using meta-features of the dataset collection as predictors and the performance of the target technique for the datasets as the response. The regression models would then be used to predict performance of a future dataset from its meta-features. The predicted performance of each target technique can then be used to select the top-k performing techniques or to generate a full ranking of all target techniques. Another common approach is to build a binary classifier for every pair of technique and predict which technique would have better performance. The pair-wise comparisons from each meta-learner can then be aggregated to identify the winner. These approaches work best when the number of target techniques is relatively small as the number of meta-learners to train can become unmanageable when the number of candidate techniques becomes larger.

Alternatively, it is also possible to train a meta-learner capable of multi-class classification. In this case, the classifier output would be the most predictive technique from the set of target modeling techniques. We have preferred this approach as it provides more flexibility to update the meta-learning model as the modeling technique space grows and used a random-forest classifier and a k-nearest neighbors (k-NN) classifier as the meta-learner. Our work on using meta-learning for selecting top performing modeling technique (*i.e.*, model type) is presented in Chapter 4.

Other Approaches

In addition to aforementioned systems, there are fully automated commercial solutions encapsulating most if not all the details of the model building and diagnostics process from the end user and produce a ready to use prediction model from the user provided dataset. Most prominent examples include the Watson Analytics platform from IBM[2] and Google Prediction API[3]. Although these systems allow a low entry point into data analysis, more seasoned data analysts including domain experts may find this approach limiting.

[2]`https://www.ibm.com/watson-analytics`

[3]`https://cloud.google.com/prediction/docs`

Chapter 3

Automated Predictive Big Data Analytics Using Ontology-based Semantics[1]

[1]Mustafa V. Nural, Michael E. Cotterell, Hao Peng, Rui Xie, Ping Ma, John A. Miller. 2015. International Journal of Big Data. (2)2:43-56.
Reprinted here with permission of the publisher.

Abstract

Predictive analytics in the big data era is taking on an ever increasingly important role. Issues related to choice on modeling technique, estimation procedure (or algorithm) and efficient execution can present significant challenges. For example, selection of appropriate and most predictive models for big data analytics often requires careful investigation and considerable expertise which might not always be readily available. In this paper, we propose to use semantic technology to assist data analysts and data scientists in selecting appropriate modeling techniques and building specific models as well as the rationale for the techniques and models selected. To formally describe the modeling techniques, models and results, we developed the Analytics Ontology that supports inferencing for semi-automated model selection. The ScalaTion framework, which currently supports over thirty modeling techniques for predictive big data analytics is used as a testbed for evaluating the use of semantic technology.

3.1 Introduction

Predictive big data analytics relies on decades worth of progress made in Statistics and Machine Learning. Several frameworks are under development to support data analytics on large data sets. Included in the group are Drill[2] , Hadoop[3] , Mahout[4] , Storm[5] , Spark[6] , and ScalaTion [Miller et al., 2010]. These frameworks target large data sets by using databases and distributed file systems as well as parallel and distributed processing to speed up computation and support a greater volume of data. As large amounts of data become readily available, one would expect greater use of such frameworks, even by scientists, engi-

[2]Drill: http://drill.apache.org/
[3]Hadoop: http://hadoop.apache.org/
[4]Mahout: http://mahout.apache.org/
[5]Storm: https://storm.apache.org
[6]Spark: http://spark.apache.org

neers and business analysts that may not be familiar with the state-of-the-art in Statistics and Machine Learning.

The rapidly growing need for more people to analyze, or more importantly, make sense of, ever increasing amounts of data is an important challenge that needs to be addressed. One way to address this challenge is more education. Many universities are adding academic and professional programs on data analytics and data science. In addition to this, technology can also help address the problem. As there are over one hundred popular modeling techniques in the fields of Statistics and Machine Learning (ScalaTion already supports over thirty), how can one decide? Furthermore, given a modeling technique (type of model), there is still much work left to build a model, including use of data transformation functions, choices of predictor variables, etc.

With smaller data sets and high expertise on the part of the analyst, one practice is to try all possible models for a set of preferred techniques. For big data and less experienced analysts, this practice cannot be relied upon.

We propose to use Semantic Web technology to assist analysts in selecting, building and explaining models. Statistical and Machine Learning models are formally described using the Analytics Ontology. It is defined using the Web Ontology Language (OWL)[7] and built using the Protégé[8] Ontology Editor and Framework . Its taxonomy (class hierarchy) of model types, equivalence axioms between the model types in the taxonomy and property restrictions can be used to help choose appropriate modeling techniques using a Description Logic (DL) reasoner. This thesis focuses on the use of the ScalaTion framework and semantic technology to assist in the development and execution of large-scale models.

The rest of this paper is organized as follows: Section 3.2 discusses the workflow that we use to oversee the entire analytics process. Related work, on model selection and the use of

[7]OWL: http://www.w3.org/TR/owl2-overview/

[8]Protégé: http://protege.stanford.edu/

semantics in analytics, is presented in Section 3.3. Section 3.4 provides an overview of the ScalaTion Framework. Extraction of metadata is presented in Section 3.5. The structure and design of the Analytics Ontology as well as how this ontology is used in our analytics process is presented in Section 3.6. Finally, Section 3.7 concludes the paper.

3.2 Predictive Analytics Workflow

Abstractly, a univariate predictive model can generally be formulated using a prediction function f as follows:

$$y \ = \ f(\mathbf{x},\ t;\ \mathbf{b}) + \epsilon$$

where y is the response variable, $\mathbf{x}$ is a vector of predictor variables, t is a time variable, $\mathbf{b}$ is a vector of parameters and ϵ represents the residuals (what the model does not account for). The objective is to pick functional forms and then fit the parameters, $\mathbf{b}$ to the data to in some sense minimize the residuals. Estimation procedures for doing this include the following [Godambe, 1991]: Ordinary Least Squares (OLS), Weighted Least Squares (WLS), Maximum Likelihood Estimation (MLE), Quasi-Maximum Likelihood Estimation (QMLE), Method of Moments (MoM) and Expectation Maximization (EM). Closely related to the prediction problem is the classification problem. Although some view classification as a special case of prediction, classification takes center stage when the response variable y takes on values from a set small enough for its elements to be named (*e.g.,* reject, weak reject, neutral, weak accept, accept).

Predictive big data analytics involves many complex steps, many of which require a high-level of expertise. To help manage the complexity, we developed a hierarchical workflow for the predictive analytics process. The top level of our hierarchical workflow is illustrated in Figure 3.1.

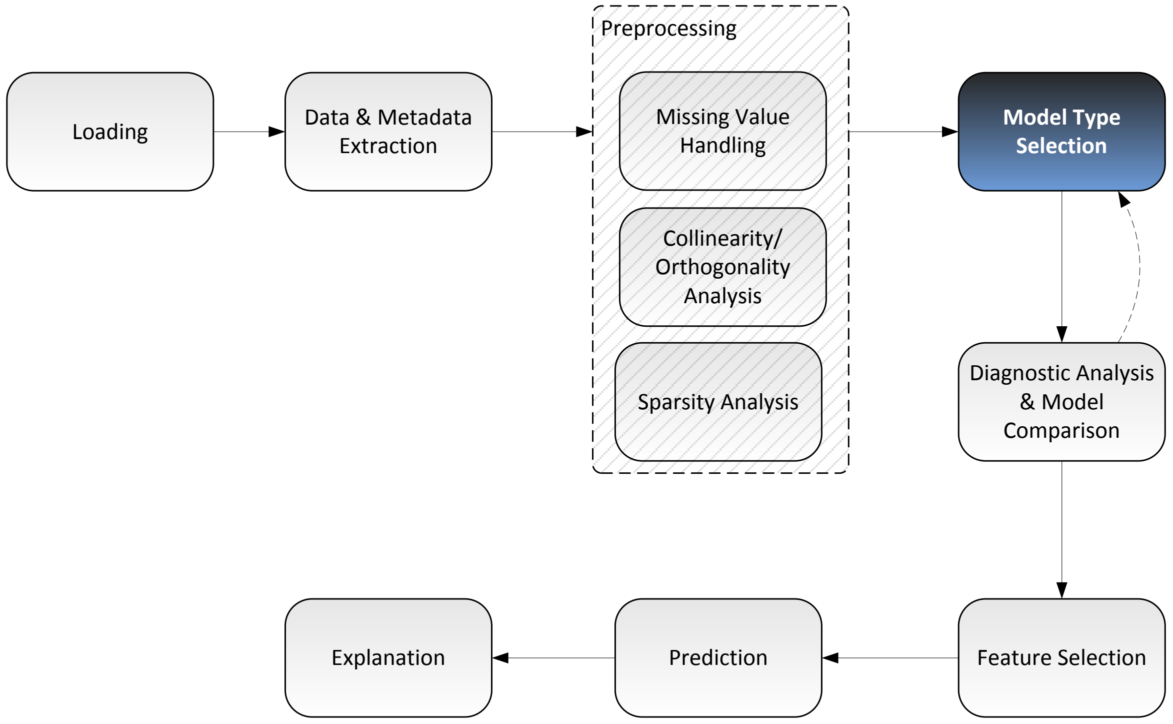


Figure 3.1: Predictive Analytics Workflow

The first step is to load the dataset in a flexible data structure that will make common manipulation operations easy. We use a table structure that support relational algebra operations [Codd, 1970].

The second step involves extracting key information regarding the dataset from data and metadata. This information plays an essential role during model type selection phase in addition to the explanation phase described later. Data & metadata extraction is discussed in more detail in Section 5.

The third step in the workflow involves further refinement of the selection problem as well as preparation of the data for analysis. This step involves the handling of missing values, multicollinearity and sparsity of data. It also involves the preprocessing of the various columns in a dataset so that they are encoded numerically. Certain other considerations may be taken at this point as well. For example, if the number of predictor variables and the multicollinearity are both high, some form of data/dimensionality reduction should be considered (*e.g.,* Regression using Singular Value Decomposition (SVD) or Principal Component Analysis (PCA)).

For example, instead of performing PCA, using Singular Value Decomposition (SVD) when estimating the coefficients of final models may be sufficient.

The fourth step in the workflow, and the main focus of this paper, is the selection of practical modeling techniques or model types based on the characteristics of a dataset. This is performed by first identifying the domains of discourse for each column in the dataset, then applying semantic inferencing using an ontology to suggest potential model types. For example, if the response variable is binary or dichotomous (*e.g.,* grant loan or deny loan), then Logistic Regression becomes a candidate since the residuals are distributed according to a Binomial distribution. If the response variable represents counts, Poisson Regression should be considered. The characteristics of domains of the predictor variables can be used in selecting between Analysis of Variance (ANOVA), Analysis of Covariance (ANCOVA) and Multiple Linear Regression (MLR).

The fifth step is a diagnostic analysis of the full models (all predictors included) in order to validate the chosen model(s) and go back to first step if a model(s) does not conform to the assumptions and requirements of the model type and therefore needs to be refined. One of the most important diagnostics is residual analysis. For example, if the first step suggested a General Linear Model (GLM), but the residuals are found not to be normally distributed with zero mean and constant variance, then various stabilizing transformations may be considered. Additionally, the distribution of the residuals (*e.g.,* Multinomial) can suggest a different type such as Multinomial Logistics Regression. In addition, outlier analysis can be performed in this step. Finally, in the previous step, more than one model type may be suggested. Diagnostic analysis also involves comparison of models in such cases and choosing the most suitable one according to the metrics provided.

The sixth step in the workflow performs various model building and model reduction techniques to suggest the subset of predictor variables that are used for each model type. For example, in regression the following procedures may be used: Forward Selection, Backward Elimination, Least absolute shrinkage and selection operator (Lasso) [Tibshirani, 1996], Forward Stagewise, Least Angle Regression (LARS) [Efron et al., 2004].

Finally, once a model is validated and finalized, it can be used for prediction. Additionally, the results may be explained based on the information available to the system. Information sources include both the metadata extracted from the dataset and the information gathered from external sources such as domain ontologies. In-depth discussion on this topic is provided in later sections.

3.3 Related Work

Semi-automated/automated model selection has been studied extensively in the literature. Similar to our approach, [Bernstein et al., 2002] describes an ontology-based Intelligent Discovery Assistant (IDA). After analyzing an input dataset, the system generates all possible workflows from the ontology that are valid within the characteristics of the input. The recommendations are then ranked based on user specified criteria (*e.g.,* simplicity, performance, etc.).

A more widely adopted technique used in automated model selection for classification is called Meta-learning. Meta-learning treats the selection problem itself as a learning problem. A meta-learning model uses characteristic properties of a dataset and performance metrics of different classification algorithms on that dataset as a training sample and learns to pick the most suitable model for a given dataset from this training exercise. The dataset properties used in meta-learning (*i.e.,* meta-features) include simple properties such as number of attributes, sample size, number of classes, and, in most cases, more complex properties such as noise-signal ratio, normalized class entropy, skewness, etc. A comprehensive list of meta-features used in meta-learning is provided in [Reif et al., 2012]. Despite the fact that the field of meta-learning is well established, there are only a few studies that focus on applying meta-learning for model selection for disciplines other than classification [Smith-Miles, 2008]. Our approach differs from meta-learning in two ways. First, we capture expertise in ontologies and rule bases. Second, we make use of metadata in addition to properties computed from the data.

Another approach was taken by [Calcagno and de Mazancourt, 2010]. In their paper, they describe an R package `glmulti` for the model selection problem with Generalized Linear Models (GZLM). `glmulti` focuses mainly on feature selection which is a sub-problem of automated model selection. `glmulti` takes the list of all predictor variables and generates all possible unique models from the combination of these variables. After generating and fitting all models using a specified metric iteratively, `glmulti` returns top n models for the input. Since the number of possible unique models grows exponentially with respect to the number of variables in the input, their approach becomes prohibitive as fitting all models requires significant resources. The authors try to overcome this limitation by using a genetic algorithm to assist model generation and try not to fit all possible models. Finally, `glmulti` does not take into consideration metadata when reducing the sample space as opposed to our approach.

Finally, traditional techniques for model selection include using decision trees hand crafted by domain experts[9]. These trees classify techniques based on the goal of the analysis and basic dataset properties such as number and types of predictors, number and types of

[9]See [Tabachnick and Fidell, 2013] for an example. There are also websites that serve similar purpose. Examples include http://www.ats.ucla.edu/stat/mult_pkg/whatstat/ and http://www.socialresearchmethods.net/selstat/ssstart.htm

responses. Based on the properties of the dataset to be analyzed, one can find a suitable technique (*i.e.,* model type) for analysis. However, these approaches usually provide general outlines and leave the majority of the work including the implementation to the analyst. Even though it is possible to extend this approach, we suggest using an ontology-based approach. This approach has several advantages over a decision tree based approach. First, the expertise is captured in description logic. Capturing the expertise in a formal language such as description logic makes it easier to point out inconsistencies, validate suggestions and create executable models automatically. This may not carry much importance when the domain knowledge rarely changes, however, there is no universal truth in statistics. Additionally, each scientific discipline has well-established preferences when it comes to statistical analysis. Therefore it is important to have an adaptable platform that is designed with changes in mind. Additionally, since model suggestion is done by a description logic reasoner, it is possible to generate formal justifications for why a particular model is suggested.

3.4 ScalaTion Framework

We use the ScalaTion [Miller et al., 2010] framework which supports multi-paradigm modeling for running our analyses. ScalaTion provides many analytics techniques for optimization, clustering, predicting, etc. which could be easily integrated in a large scale data analysis pipeline. Additionally, ScalaTion supports discrete event and continuous simulation.

ScalaTion is organized in three major package groups. The `analytics` package, analytics, includes implementations of major analytics algorithms which can be categorized under four types: predictors, classifiers, clusterers and reducers. Additionally, the `graphalytics` package provides implementations for graph-based analytics.

The optimization packages, `minima` and `maxima`, provide algorithms for optimization and implement major optimization paradigms such as Linear, Integer, Quadratic and Nonlinear

Programming and the Simplex method. The `minima` package is for minimization, while the `maxima` package is for maximization.

Finally, the simulation packages provide simulation engines for a variety of different modeling paradigms. Currently, ScalaTion has implementations for tableau, event, process, activity and state oriented models in addition to system dynamics. ScalaTion also has 2D (and prototype 3D) visualization support for such models.

ScalaTion , coded in the Java Virtual Machine (JVM) based the Scala language, makes use of Scala's native parallelism support via `.par` functions in addition to processing using Akka[10]. In ScalaTion , the `linalgebra.par` package, which currently contains 3,326 lines of code, contains parallel versions Cholesky and QR factorizations as well as parallel versions of many operations for both dense and sparse matrices. For some operations like matrix multiplication, the `.par` function available in Scala makes it easy to convert sequential to parallel code. Below is the definition of the matrix multiplication function in ScalaTion 's `MatrixD` class. The parallel ranges in the `for` loop split the iterations of the loop into manageable units of execution that can be performed by parallel threads.

```
def * (b: MatrixD): MatrixD =
{
    val c  = new MatrixD (dim1, b.dim2)
    val bt = b.t // transpose the b matrix
    for (i <- range1.par; j <- c.range2.par) {
        val va = v(i); val vb = bt.v(j)
        var sum = 0.0
        for (k <- range2) sum += va(k) * vb(k)
        c.v(i)(j) = sum
    } // for
```

[10]Akka Framework: http://akka.io/

```
    c
} // end * function
```

For other operations such as Cholesky and QR factorizations, matrix inversion and SVD a substantial speedup via parallelism is not so easy to achieve. For example, in [Lahabar and Narayanan, 2009] speedup of SVD was only achieved by utilizing Graphics Processing Units (GPUs) via the CUDA[11] platform. We plan on exploring the use of co-processors (*e.g.,* Intel Xeon Phi), custom thread pooling, and frameworks for distributed computation (*e.g.,* Akka) to facilitate speedup.

There are also other modeling environments such as Weka [Hall et al., 2009] sharing similar functionality with ScalaTion . Weka is a popular data analytics and machine learning platform written in Java. It provides a very intuitive user interface that allows pre-processing of data in addition to visualization. Weka also provides a Java API for programmatic access to the algorithms. In contrast to ScalaTion , Weka is focused on machine learning. By providing an integrated approach, ScalaTion reduces the cost of development time for a multi-paradigm modeling task.

In contrast to statistical software like R and SAS, ScalaTion has a cleaner syntax that allows for mathematical formulas and expressions to more closely resemble their standard textbook notations. For example, Figures 3.2, 3.3 & 3.4 show how to estimate the coefficients in a Principal Component Regression (PCR) model of the form $\mathbf{y} = X\mathbf{b} + \epsilon$ using SVD in R, SAS and ScalaTion , respectively. Let U, Σ and V be the factors obtained from applying SVD to X. Then, an estimate for $\mathbf{b}$ is

$$\widehat{\mathbf{b}} \;=\; V\Sigma^{-1}U^t y \;=\; V\frac{1}{\Sigma}U^t y$$

where Σ^{-1} and $\frac{1}{\Sigma}$ both represent the inverse of the diagonal matrix containing the singular

[11]CUDA: http://www.nvidia.com/object/cuda_home_new.html

values of X [Mandel, 1982]. Of the three code examples provided, one could argue that the ScalaTion example is not only more readable, but looks closer to the mathematical notation provided above.

```
svd <- svd(x)
u <- svd$u
d <- diag(svd$d)
v <- svd$v
b <- v %*% ginv(d) %*% t(u) %*% y
```

Figure 3.2: Estimating coefficients of a PCR model using SVD in R

```
call svd(u, d, v, x);
b = v * inv(diag(D)) * u` * y;
```

Figure 3.3: Estimating coefficients of a PCR model using SVD in SAS

```
val (u, d, v) = x.svd.factors()
val b = v ** d.reciprocal * u.t * y
```

Figure 3.4: Estimating coefficients of a PCR model using SVD in ScalaTion

Additionally, since ScalaTion is written in Scala for the JVM, users can easily integrate existing APIs in Scala, Java, and other JVM languages. For integration with native code libraries, the Java Native Interface (JNI) can be used. While Spark is also available on the JVM, it utilizes Stochastic Gradient Descent SGD) to produce least squares estimates in regression whereas ScalaTion utilizes factorization techniques such as Cholesky Factorization, QR Decomposition and SVD Decomposition which are commonly considered to work better in practice.

3.5 Extraction of Metadata

In order to reduce model type space and find suitable modeling techniques for a given dataset, some information needs to be gathered. Some of this information can be obtained directly by performing a quick analysis of the dataset. We list a number of properties in Table 3.1 that can be obtained in such a way.

Table 3.1: Data Extraction Tasks

TASK	**DESCRIPTION**
Domain of Variables	Continuous, Discrete (Binary, Integer, etc.)
Variable Diagnostics	Mean, variance, probability distribution and etc.
Dimension Analysis	Dimension reduction based on multicollinearity / rank (PCA etc.)
Feature Space Analysis	Analyzing relationship between sample size and number of predictors to prevent overfitting
Sparsity	Yes, No
# of Samples	Integer

The domains of variables are essential when choosing an appropriate model type. For example, if the domain of the response variable is binary, then a logistic regression model can be selected. Similarly, for a dataset which contains both discrete and continuous predictor variables, an ANCOVA model can be more appropriate. Although capturing the domain of a variable can be difficult, it can be useful when this information is not available in the metadata. For example, a technique based on repeated differences can be employed to help identify whether the values are discrete.

Diagnostic analysis of variables includes computing basic statistics such as mean and variance as well as more advanced information such as the probability distribution of the variable. This information may be useful in several ways. First, it can help the model suggestion process. As an example, Poisson regression may be used for modeling count data (*i.e.,* non-negative integer response). However, an important assumption for Poisson regression is that the mean is equal to the variance. If the diagnostics reveal that the variance is significantly larger than mean (*i.e.,* overdispersion) negative-binomial regression may be suggested instead. It may also help determine the validity of a selected model after fitting.

A very important factor in predictive analytics is reliability. [Harrell, 2015] defines reliability as the ability of the fitted model to predict future instances as well as existing instances upon which the model is trained. If a model is overfitted, its reliability of predicting future instances will be low. In addition to choosing an inappropriate model, a major cause for overfitting is having too many predictors for the given sample size. According to [Harrell, 2015], a fitted relation model is likely to be reliable in most cases if the number of predictors p is less than $m/15$ where m is the limiting sample size. m is equal to the number of samples in the dataset for a continuous response. For a binary response, m is equal to $min(n_1, n_2)$ where n_1 and n_2 are marginal frequencies of two response levels. In cases where $p > m/15$, a dimension reduction approach such as Principal Component Analysis (PCA) should be taken. PCA can also be performed to reduce the dimensionality of the matrix when the data matrix for the predictor variables has high multicollinearity. Additionally, this step can help make some of the underlying matrix operations performed by various algorithms easier (*e.g.,* matrix factorization). Another consideration that should be taken with respect to the data matrix is whether or not it is sparse. Certain algorithms can take advantage of specialized data structures for storing sparse matrices in order to reduce memory consumption and avoid unnecessary operations [Smailbegovic et al., 2005].

Additionally, any available metadata can also be used. As mentioned in the related work, existing systems operate solely on the input data ignoring problem definition, field descriptions and other metadata. However, predictive analytics often requires knowledge of the experiment design which was used in order to generate the data. Additionally, selection of a modeling technique is often dependent on the goal of the analysis.

Table 3.2: Metadata Extraction Tasks

TASK	**DESCRIPTION**
Associated Task (Goal)	Prediction, Classification, etc.
Multivariate Response?	Yes, No
Multivariate Predictor?	Yes, No
Variable Type	Continuous, Discrete (Binary, Integer, etc.)
Response Column	Column number(s) of the response variable
Missing Values	Yes, No

Table 3.2 lists a number of useful properties that can potentially be obtained from metadata. An associated task can be as simple as whether this problem is a prediction or a classification problem. In some cases, the Associated Task could be as specific as ANOVA or ANCOVA (*e.g.,* if you are interested in the relationship between the predictors themselves with respect to the response). Often times, data is stored in a matrix format where a row indicates a sample and columns indicate the features of the sample. The Response Column indicates the column index of the data matrix in which the response variable is stored. Variable Type indicates the domain of a variable (*e.g.,* continuous, binary, ordinal, etc.) and can be extracted for each variable in the dataset depending of the availability of the metadata. Whether the dataset contains missing values or not and which variables have missing values can affect the selection of appropriate model types. Depending on the situation, our system may choose a model that can handle missing values or otherwise samples containing missing values would be discarded.

Whether a dataset has multivariate response or predictors can be implicitly available given the above information, however, the provided metadata may contain only partial information.

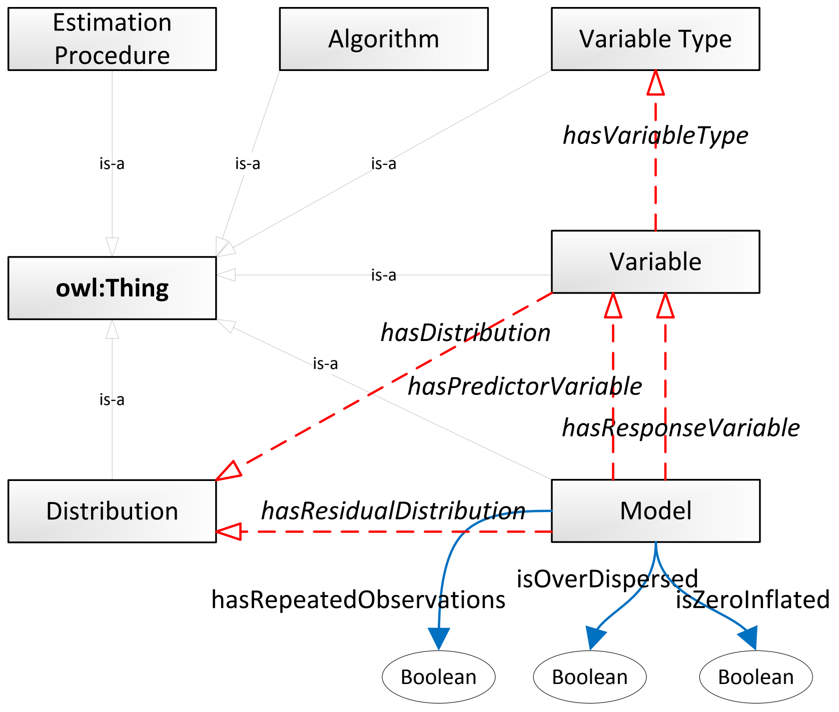


Figure 3.5: Main Object and DataType properties in the analytics ontology

3.6 Analytics Ontology

The Analytics Ontology[12] is an ontology for supporting automated model selection. Currently, it is more geared towards predictive analytics. However, the ontology is designed to be extensible in order to support a wide range of analytics paradigms including classification and clustering.

The most important class in the ontology is Model. The Model class along with its subclasses is both indicative of a model type as well as a partial realization of a statistical model. This is due to the fact that all other classes such as Variable, Function, and Distribution

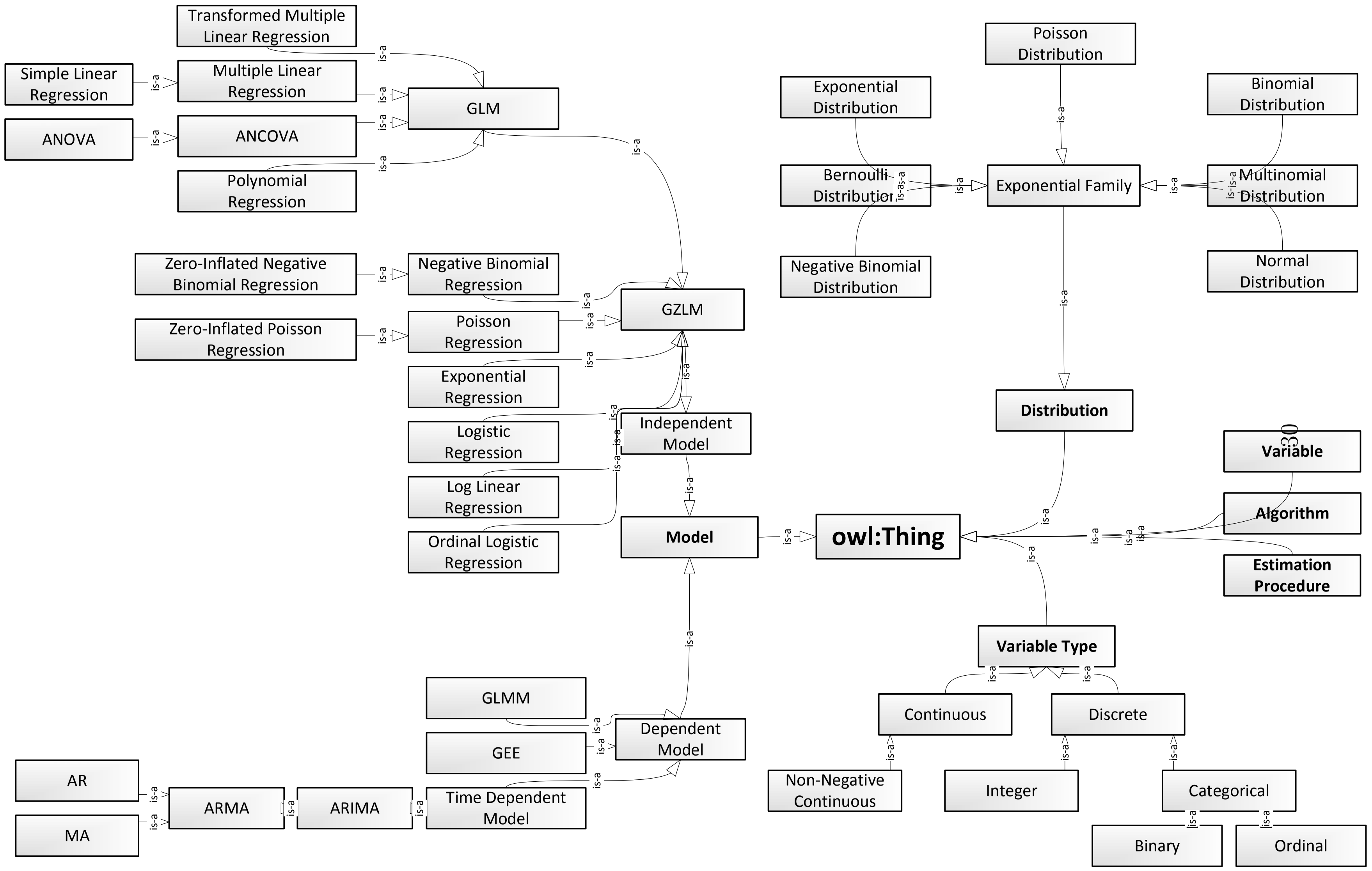
[12]Analytics Ontology can be accessed from https://github.com/scalation/analytics.

describe a part of the model.

Figure 3.6 displays the major classes and their hierarchy. We now give working definitions of the major classes of the ontology in the analytics context.

Model defines different model types that can be used for analyzing data. There are many ways of specifying the class hierarchy for a collection of model types, however, we have given priority to correspondence with implementations of these types (*e.g.,* ScalaTion). This becomes important when running models generated from the abstract models represented in the ontology. There are two top level Model classes, namely *DependentModel* and *IndependentModel*. The main distinction between the two models is the dependency (*i.e.,* correlation) among responses of the observations belonging to the same individual in the dataset. The dependency usually occurs when there are repeated observations (*i.e.,* measurements) of the same individual in time or space dimension. Time-series models are a typical example of a *DependentModel*. Other major members of *DependentModel* are Generalized Estimating Equation (GEE) models and Generalized Linear Mixed Models (GLMM).

IndependentModel includes Generalized Linear Models (GZLM) [Nelder and Baker, 2004]. The most basic independent model is simple linear regression, which quantifies the linear relationship between the response and a single explanatory variable. An equivalent model of regression is ANOVA model, which target the problem from a different angle that focuses on analyzing the differences among group means and their associated procedures. The extension of simple linear regression and ANOVA are multiple linear regression and MANOVA, respectively, which are used when there are two or more predictors. Those models, together with other models, such as before-mentioned ANCOVA, and polynomial regression that describe the linear relation between predictors and responses, belong to the general linear model class. The generalized linear model is a flexible generalization of general linear model that allows for response variables that have error distributions other than normal distributions. The common generalized linear models include logistic regression, Poisson regression, Log-Linear


Transformed Multiple Linear Regression
Simple Linear Regression
Multiple Linear Regression
GLM
ANOVA
ANCOVA
Polynomial Regression
Zero-Inflated Negative Binomial Regression
Negative Binomial Regression
GZLM
Zero-Inflated Poisson Regression
Poisson Regression
Exponential Regression
Logistic Regression
Independent Model
Log Linear Regression
Ordinal Logistic Regression
Model
owl:Thing
Poisson Distribution
Exponential Distribution
Binomial Distribution
Bernoulli Distribution
Exponential Family
Multinomial Distribution
Negative Binomial Distribution
Normal Distribution
Distribution
Variable
Algorithm
Estimation Procedure
Variable Type
Continuous
Discrete
Non-Negative Continuous
Integer
Categorical
Binary
Ordinal
GLMM
GEE
Dependent Model
AR
MA
ARMA
ARIMA
Time Dependent Model
is-a

Models, etc. Changing the relationship between the parameters and the linear predictor, i.e. changing the link function, usually will lead to different generalized linear models.

Variable represents a feature of a Model. A variable can have a role of a predictor or a response which is defined by the relationship with Model. This relationship is defined by the object properties *hasPredictorVariable* and *hasResponseVariable*. Additionally, a predictor variable can also have a role of representing the time component in a time series model. In that case, *hasTemporalVariable* property may be used.

Variable Type restricts a *Variable*'s corresponding domain of discourse. The relationship between a *Variable* and *Variable Type* is defined by the object property *hasVariableType*. As seen in Figure 3.6, a *Variable* can either have a *Continuous* or *Discrete* type. Even though *Discrete* class may be considered a subclass of *Continuous* class algebraically, their statistical interpretation mandate being represented as two distinct classes in the ontology. Similarly, subclasses of *Categorical* class, *Binary* and *Ordinal* are represented as distinct classes albeit *Binary* being a special case of *Ordinal* variable type.

In some families of models such as GZLM (Generalized Linear Models) a link *Function* is used to relate the predictor variables to the mean response. The relationship between a *Model* and a *Function* is defined by the object property *hasLinkFunction*.

Distribution class refers to the (probability) distribution of a random variable which specifies how to assign the probability of occurrence for each element in the variable's range. Considering the fact that the residual (error term) of a model is considered to be a random variable, this class is used to specify the residual distribution of a model. This relationship can be defined using the *hasResidualDistribution* object property.

3.6.1 Equivalence Class Axioms

In addition to the class axioms (*i.e.,* class definition, hierarchy of classes) and object properties, the Analytics Ontology defines equivalence class axioms to capture key characteristics of different model types. According to the OWL language specification, an equivalent class axiom states the equivalence of two named classes which is defined with Description Logic syntax. Using these axioms, it becomes possible to deduce implicit information hidden in the ontology with a reasoner.

```
IndependentModel ≡
        Model and (hasRepeatedObservations value false)
GZLM ≡
        Model and (hasRepeatedObservations value false)
Poisson Regression ≡
        GZLM
        and (
                (hasResidualDistribution only 'Negative Binomial Distribution') or
                (hasResidualDistribution only 'Poisson Distribution') or
                (hasResidualDistribution exactly 0 Thing)
        )
        and (hasPredictorVariable some (hasVariableType only Continuous))
        and (hasResponseVariable only (
                (hasDistribution only 'Poisson Distribution')
                and (hasVariableType only 'Non-Negative Integer')
        ))
Exponential Regression ≡
        GZLM
        and ((hasResidualDistribution only 'Exponential Distribution') or
        (hasResidualDistribution exactly 0 Thing)
        )
        and (hasPredictorVariable only (hasVariableType only Continuous))
        and (hasResponseVariable only (hasVariableType only 'Non-Negative Continuous'))
GLM ≡
        GZLM
        and (
                (hasResidualDistribution only 'Normal Distribution')
                 or (hasResidualDistribution exactly 0 Thing)
        )
        and (hasResponseVariable only (hasVariableType only Continuous))
Multiple Linear Regression ≡
        GLM
        and (hasPredictorVariable some (hasVariableType only Continuous))
        and (hasResponseVariable only (hasDistribution only 'Normal Distribution'))
ANCOVA ≡
        GLM
        and (hasPredictorVariable some (hasVariableType only Categorical))
        and (hasResponseVariable only (hasDistribution only 'Normal Distribution'))
```

Figure 3.7: Example of equivalence axioms based on the variable type and residual distribution

Figure 3.7 lists a few of the equivalence class axioms from the ontology. Based on these axioms it is possible to infer that a dataset can be modeled using *ANOVA* if it has only categorical predictor variables and a continuous response variable. We can easily see that

Multiple Linear Regression is left out according to the equivalence axiom restricting the domain of at least one predictor variable to be *Continuous*. We should note that the restrictions defined by the equivalence axioms are used for elimination of unsuitable models. Therefore, a dataset can be modeled by using any other *Model* which is not eliminated by the reasoner during inference.

3.6.2 Model Selection Using Semantic Reasoning

We use description logic reasoning for reducing the model type space. Given background knowledge and observations, an explanation is computed. In our context, background knowledge is realized with the axioms in the ontology. These include the class axioms which define the hierarchy between model types and the equivalence class axioms. Similarly, observations are the acquired characteristics of the dataset of focus. These are captured by the facts which are defined by linking the instances describing the dataset to the ontology axioms using the object property relationships. Finally, explanation constitutes the set of inferred possible model types that could be used for analyzing the specified dataset.

Our reasoning framework is fully captured in the Analytics Ontology which is expressed in OWL 2 DL profile. The DL profile imposes certain restrictions over the full OWL 2 language so that certain computational guarantees can be made by the reasoners who support reasoning with OWL 2 DL profile. Since the Analytics Ontology fully adheres to OWL 2 DL profile, any existing OWL 2 reasoner (*e.g.,* HermiT [Shearer et al., 2008], Pellet [Sirin et al., 2007], etc.) can be used for deducing the model types for a given dataset. Also, this ensures that our approach is decidable as all possible inferences are guaranteed to be made. A logical reasoner serves several different purposes for an ontology. First, it helps extend the knowledge base by discovering hidden (*i.e.,* implicit) information related to the concepts and individuals in the ontology. For a dataset, the information of the most suitable model type(s) is hidden in the characteristics and properties of it. In our case, this hidden (implicit) information is discovered by the logical reasoner from the explicit information added to the ontology about the dataset. Additionally, it performs consistency checking for all axioms in the ontology including the axioms that were inferred during the discovery phase. This step help ensure that the domain expertise is properly captured in the ontology and may be used for suggesting suitable models for datasets.

3.6.3 Scala-Dash

In order to demonstrate the approach described in this paper, we have developed *scala-dash*[13]. Developed in scala language for leveraging ScalaTion framework to full extent, *scala-dash* is an application with a graphical user interface that is designed to handle all steps of predictive analytics workflow as described in Section 3.2.

We use the Auto-MPG dataset [Quinlan, 1993] from UCI Machine Learning Repository [Lichman, 2013] as a small example for illustration. The dataset is for prediction of fuel consumption in miles per gallon and contains 3 discrete variables; cylinders, model year, origin and 5 continuous variables; mpg, displacement, horsepower, weight and acceleration. The dataset contains 398 samples including 6 samples with missing values for horsepower. According to this specification, the dataset can be represented in the ontology as shown in Figure 3.8. By default, all individuals are an instance of *owl:Thing* in OWL. Note that the *AutoMPGModel* is not an instance of any specific *Model* class in the hierarchy as this is not explicitly expressed (see Figure 3.8). It is not known at this point which models are suitable for this dataset.

As opposed to other logic frameworks OWL takes an *open-world assumption*(OWA) when dealing with missing/incomplete knowledge. In a *closed-world assumption* (CWA) system, any information that does not exist in the knowledge is considered false[14]. As an example,

[13]scala-dash is available for download at https://github.com/scalation/analytics
[14]For a more in-depth discussion on OWA and CWA, see [Russell et al., 2005]

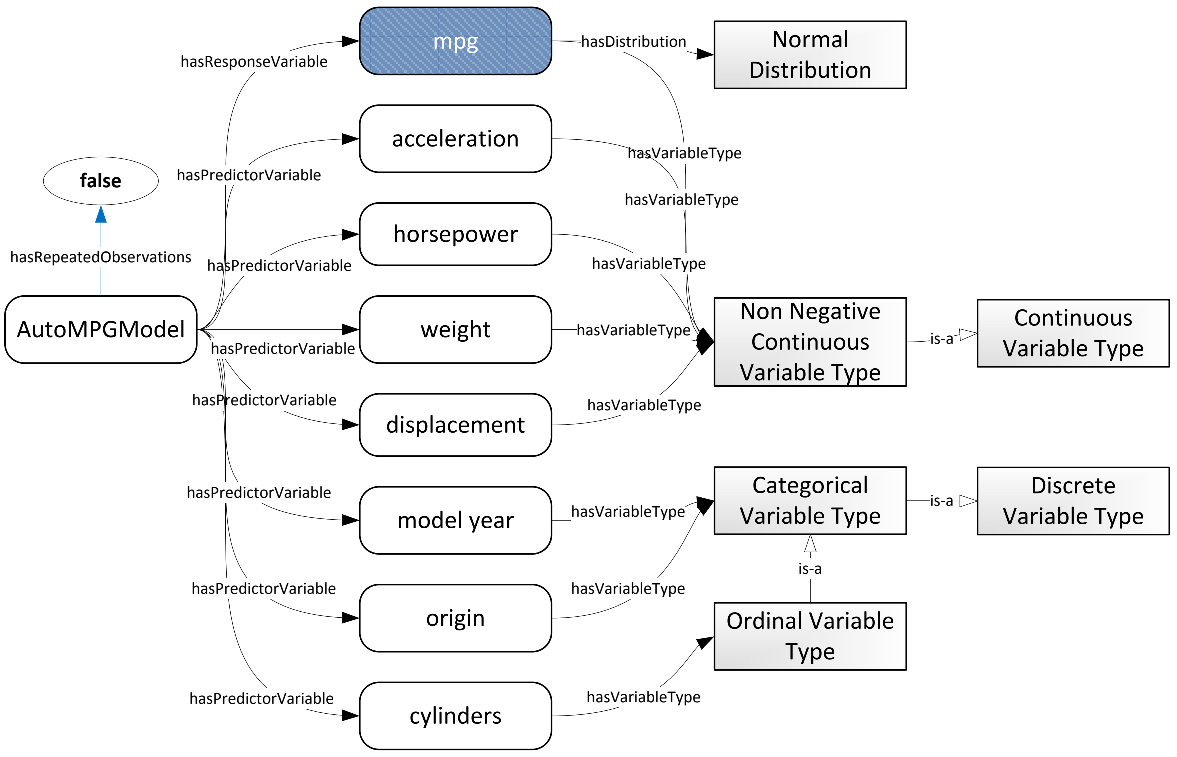


Figure 3.8: Representation of Auto MPG Model in the ontology

in a closed-world, a logical reasoner can deduce that *AutoMPGModel* only has 8 variables as shown in Figure 3.8. However, an open-world reasoner cannot deduce the same fact as there might be other variables which may still exist in another knowledge base. For this reason, besides asserting the facts as in Figure 3.8, we also need to assert closure axioms that are implicitly asserted in a closed-world assumption (CWA). Following the same example, the following assertions must be added to the ontology to "close" the axioms for *AutoMPGModel*.

- *AutoMPGModel* hasVariable only ({*acceleration, cylinders, displacement, horsepower, model_year, mpg , origin, weight*})
- *mpg* hasDistribution only ({*Normal_Distribution_Instance*})

- *mpg* hasVariableType only ({*Non_Negative_Continuous_Variable_Type*})[15]

Another related characteristic of OWL is the lack of unique name assumption (UNA). As a result, OWL makes no assumption that individuals with different unique names are in fact different individuals. To let the reasoner be aware of the fact that the variables are different, we add the following axiom to the ontology.

- DifferentIndividuals ({*acceleration, cylinders, displacement, horsepower, model_year, mpg, origin, weight*})

After the *AutoMPGModel* instance is properly closed, the following facts about the dataset are inferred when a reasoner is run on the ontology:

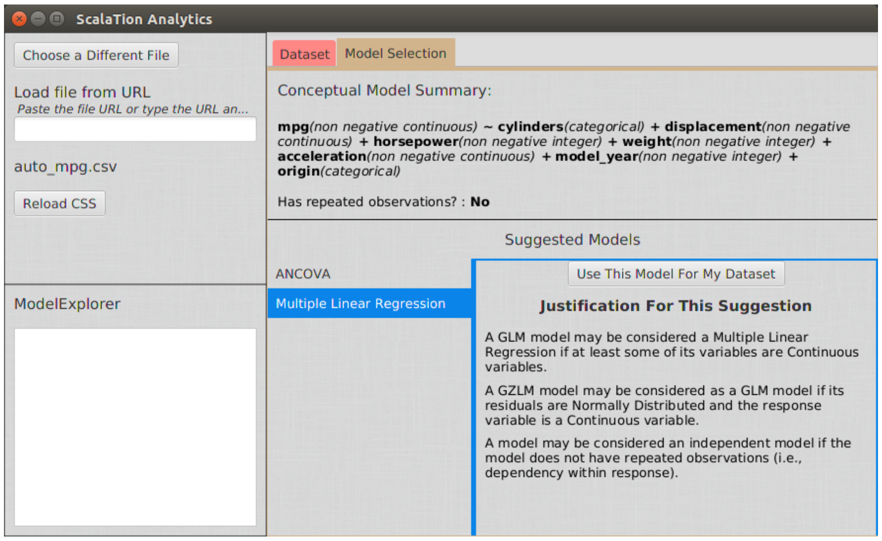
- *AutoMPGModel* is-a *Model.*
- *AutoMPGModel* is-a *IndependentModel.*
- *AutoMPGModel* is-a *GZLM.*
- *AutoMPGModel* is-a *GLM.*
- *AutoMPGModel* is-a *ANCOVA.*
- *AutoMPGModel* is-a *Multiple Linear Regression.*

As one can quickly notice, some of these inferences such as *AutoMPGModel* is-a *Model* do not carry much importance in terms of information content. Therefore, a filtering based on the model hierarchy is performed. An inference is not added to the list of suggestions if one of its subclasses already exist in the list. The pseudocode for the filtering algorithm is given in Figure 3.10. After the filtering function is run, only the following inferences are returned to the user as suggestions as shown in Figure 3.9:

[15]Similar assertions are made for other variables as well. They are excluded for brevity.

- *AutoMPGModel* is-a *ANCOVA*.

- *AutoMPGModel* is-a *Multiple Linear Regression*.



Conceptual Model Summary:

mpg(non negative continuous) ~ **cylinders**(categorical) + **displacement**(non negative continuous) + **horsepower**(non negative integer) + **weight**(non negative integer) + **acceleration**(non negative continuous) + **model_year**(non negative integer) + **origin**(categorical)

Has repeated observations? : **No**

Justification For This Suggestion

A GLM model may be considered a Multiple Linear Regression if at least some of its variables are Continuous variables.

A GZLM model may be considered as a GLM model if its residuals are Normally Distributed and the response variable is a Continuous variable.

A model may be considered an independent model if the model does not have repeated observations (i.e., dependency within response).

Figure 3.9: A Screenshot from scala-dash Displaying Suggestions for AutoMPGModel

```
function FILTER ($datasetId, ontology$)
 $suggestions = retrieveTypes(datasetId,\ ontology)$
 for $s$ in $suggestions$
  $subs = getSubclasses(s, ontology)$
  for $sub$ in $subs$
   if $sub$ in $suggestions$
     $removeFromSuggestions$
   end if
  end for
 end for
 return $suggestions$
```

Figure 3.10: Algorithm for Filtering Suggestions

Based on the current knowledge, all the inferred models can be suitable for analyzing this dataset. However, additional information can be asserted as new information becomes available. For example, during the full model residual analysis phase after the candidate models were run, if the residual distribution of the model is determined to be an exponential distribution we add this information by asserting the following statement "*AutoMPGModel* hasResidualDistribution *Exponential Distribution*" to the ontology. When the reasoner is re-run including this new information, *Exponential Regression* is added to the list. Also note that *ANCOVA* and *Multiple Linear Regression* are no longer possible models since both models are a sub-class of *GLM* model which expects the residuals to be Normally Distributed.

Based on the suitable models inferred by the reasoner, running a model with ScalaTion can be performed in the following fashion. Given a data matrix X and response vector $\mathbf{y}$, a Multiple Linear Regression model can be run as shown in the code snippet below. The `fit` function returns coefficient estimates and as well as various model diagnostics (*e.g.,* R^2 and $F - Statistic$, etc.).

```
val mpgMLRModel = new Regression (x, y)
mpgMLRModel.train ()
println ("fit = " + mpgMLRModel.fit)
```

In a different example, we look into the resin defects dataset[16]. The dataset captures three predictor variables: *hours* since last hose cleaning, *temperature* of the resin and the *size* of the screw that moves the resin pellets through the hoses, to predict the number of *discoloration* defects per hour during the manufacturing process. Since the response variable represents a count (Non-Negative Integer), *Poisson Regression* is inferred as the suitable model by our system. In contrast, glmulti tool does not have a mechanism to capture information of this nature and therefore suggests a general linear model (glm) formula. A quick analysis reveals that *Poisson Regression* model provides a better fit for this dataset.

Finally, we demonstrate scala-dash by using the airline dataset[17] . The airline dataset has become popular in the big data domain after the 2009 Data Expo competition organized by

[16]http://support.minitab.com/en-us/datasets/
[17]http://stat-computing.org/dataexpo/2009/

American Statistical Association Sections on Statistical Computing. The dataset is produced by RITA[18] and covers flight delay information for all commercial flights since 1987. Dataset may be analyzed in many ways as it is quite comprehensive but we focus on modeling delay between estimated and actual arrival of a flight as a function of following predictors: Month, Day of Month, Day of Week, Departure Time, Arrival Time, Departure Delay, Flight Time and Distance. After the dataset is loaded into the system, scala-dash suggested a Multiple Linear Regression model for the dataset. The model diagnostics revealed a R^2 value of .775.

3.6.4 Incorporation of Knowledge Through Domain Ontologies

Since the Analytics Ontology is written in the OWL language, it can be easily supplemented with a domain ontology related to the input domain. Besides top-level ontologies such as Dublin Core (DC) that define common metadata terms, many domain-specific ontologies have been created and been in use extensively. It is becoming more common that data is published with metadata defined in these domain-specific ontologies especially in the scientific and biomedical domains. As a prominent example, the Gene Ontology (GO) contains over 40,000 biological concepts and has been used for annotating more than 100,000 peer-reviewed scientific papers with information on genes and gene properties[19]. The well-defined knowledge captured in these domain ontologies would assist reducing the possible model space even further. For example in a traffic dataset, the response variable might capture a *count* such as number of cars passing by an intersection in a certain period of time. Even though the variable has an *Integer* domain, it can be modeled using *Poisson Regression* since the response variable captures a *count*. Many insights such as in the example are available in domain ontologies. Additionally, it can also be beneficial for making inferences based on the model since an alignment provides more information.

[18]http://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp
[19]Gene Ontology: http://geneontology.org/page/about

3.7 Conclusion

We have described a framework for supporting semi-automated model selection and model execution for conducting predictive analytics. Particularly, we show that instance classification can be used for reducing the set of applicable model types. After the model types are chosen, feature selection can be performed in order to find suitable subsets of the full model for each type. This is important because as the number of predictor variables and/or the sample size increase, exploration of all possible model types becomes less feasible. The ability to provide an explanation of why a particular model is selected based upon the inference provided by the reasoner is an important advantage over existing model selection tools. For future work, we plan to extend the ontology with concepts (*e.g.,* skewness, kurtosis, etc.) to assist automating workflow steps after model selection. Additionally, we plan to facilitate alignments with top-level and domain-specific ontologies to aid in output analysis. Finally, we plan to conduct extensive evaluations of the usability of this approach and provide performance results.

Chapter 4

Using Meta-learning for Model Type Selection in Predictive Big Data Analytics[1]

[1]Mustafa V. Nural, Hao Peng, John A. Miller. To be submitted to International Journal of Data Mining Science.

Abstract

One of the biggest challenges for today's data scientists is to be able to make an informed decision among an exhaustive number of different modeling techniques. As no single algorithm can perform optimally in all cases, the context for the modeling task including the dataset characteristics plays an unsurprisingly important role in deciding which modeling algorithm to choose. In our previous work, we have presented an ontology-based automated model-selection system extending the Scala-based ScalaTion data framework. In this study, we present a meta-learning based model-selection system for regression problems using a Random Forest classifier as the meta-learner and provide an extensive evaluation of the system. Additionally, we compare the meta-learning approach with the ontology-based approach and evaluate performance of both methods.

4.1 Introduction

With the rapid increase in the quantity of very large datasets, the need for improvements in the technological infrastructure to handle these datasets has become obvious. Recent technological advancements, including large-scale storage, huge main memories, multi-core servers and high- performance clusters, provide the foundation for handling big data. Software to facilitate the processing of big data, *e.g,* Hadoop [2] and Spark [3], has made tremendous strides. Many of these frameworks now provide some support for advanced predictive analytics. Some attempt to work with existing software like R with its hundreds (including 232 in the *caret* package) of analytics techniques [Kuhn, 2008]. Others, like Spark's MLlib [4] and ScalaTion [5] [Miller et al., 2013], are providing better performance and scalability by

[2] `http://hadoop.apache.org/`
[3] `https://spark.apache.org/`
[4] `https://spark.apache.org/mllib/`
[5] `https://github.com/scalation`

re-developing the analytics libraries.

Data scientists are now faced with vastly more data than ever before, a huge number of analytics techniques to chose from, many optimization techniques for structure or parameter learning (*e.g.,* parameter estimation), and configuration issues for running software efficiently on a cluster. As a result, the pace of data generation and the complexity of the field are surpassing the ability to process and analyze such data in the conventional way. Data scientists will need to depend more than ever on tools that provide support for automating parts of their data analytics workflow.

The primary contribution of this paper is an extensive study on meta-learning for regression algorithms. Past studies have mostly focused on classification problems, and in a few cases, a limited number of regression-based techniques. Our extensive study also introduces new meta-features that can be effectively applied to algorithm selection.

This paper discusses advances in the automation of big data analytics, focusing on assisting data scientists in key steps in the development of data analytics workflows. Related work discussing various approaches to automating the development of analytics workflows is given in Section 4.2. Section 4.3 presents the key steps within predictive analytics workflows. It highlights the issues that need to be addressed in automated modeling. Section 4.4 discusses the use of meta-learning to support automated modeling. An evaluation of the approaches is given in section 4.5. Finally, conclusions and future work are presented in Section 4.6.

4.2 Related Work

Automation of analytics workflows, or parts of them, has been studied and attempted over the years. As a result, there are various approaches to support the data scientist.

4.2.1 Exhaustive Approach

As there are literally hundreds of different techniques for predictive modeling, choosing an optimal (or one of the top few similarly performing) technique(s) for a given problem proves to be hard. One way to tackle this issue is to take an exhaustive approach and try as many different techniques as possible. However, even though this seems straightforward, often times each modeling technique requires a slightly different setup and therefore the approach quickly becomes prohibitive.

The `caret` package [Kuhn, 2008] available for the open-source R language is designed to address this problem by providing a common interface for running, evaluating, and tuning over 200 available techniques scattered across different packages in the R environment. Another R package that provides an infrastructure to streamline searching of the solution space is `performanceEstimation` [Torgo, 2014]. Additionally, it also has workflow support to include pre- and post- processing steps.

Similarly, the open-source WEKA [Hall et al., 2009] data-mining tool has an "Experimenter" module that performs an automatic experimentation for a dataset(s) against the user-selected algorithms based on the chosen performance metric.

Many commercial tools provide similar functionality to the end-users to perform this kind of automated modeling. IBM SPSS Modeler[6] has three Automated Modeling Nodes, namely, `Auto Classifier, Auto Numeric, and Auto Cluster` that allow its users to automatically search the algorithm space for classification, prediction, and clustering problems, respectively.

DataRobot[7]– a startup company founded by veterans from insurance companies – leverages cloud infrastructure to allow data scientists to evaluate all possible algorithms simultaneously by running them all in parallel in the cloud. The company claims to reduce the

[6] `https://www.ibm.com/support/knowledgecenter/en/SS3RA7_15.0.0/com.ibm.spss.modeler.help/ensemblemodeling_overview.htm`

[7] `https://www.datarobot.com`

model development time spent by a data scientist significantly, from weeks and months to less than a day.

Finally, Auto-WEKA [Thornton et al., 2013][8]performs model selection and hyperparameter (*i.e.,* model-specific parameters) optimization simultaneously for identifying best classification algorithm and its respective parameters. Auto-WEKA leverages existing classification algorithms in the WEKA algorithm space and employs a Bayesian optimization method, namely SMAC (*Sequential Model-based Algorithm Configuration*) [Hutter et al., 2011], to explore the model and parameter space instead of a full exhaustive search. Authors report promising results while completing significantly faster compared to a baseline random grid search. Auto-WEKA 2.0 [Kotthoff et al., 2017] further extends the algorithm space to include regression algorithms.

These solutions relieve a significant burden from an analyst's shoulders; however, it may quickly become prohibitive to depend on exhaustively evaluating a large number of techniques, especially as the dataset size becomes larger. Therefore, systems that help reduce the modeling algorithm/technique search space become essential. [9]

4.2.2 Ontology-based Approach

Intelligent Data Assistant (IDA) [Kietz et al., 2014][Kietz et al., 2012] takes an ontology-based approach to automated modeling focusing mainly on classification problems. The IDA is backed by a data mining ontology [Panov et al., 2014] and generates analytics workflows using hieararchical task network planning. As there literally may be hundreds of valid workflows satisfying input-output requirements of individual steps, IDA also performs ranking to highlight the best workflows to the users. Two different ranking approaches are used.

[8]Ports to different environments including python based scikit-learn may be found at the AutoML website. See `http://www.ml4aad.org/automl/`

[9]Interested readers may refer to the survey at `https://thomaswdinsmore.com/2017/01/16/the-year-in-machine-learning-part-four/`

One is frequency based in which workflows containing most frequently used operators (*based on usage data*) are ranked higher. In the second approach, IDA utilizes a meta-learning based system to rank workflow suggestions [Nguyen et al., 2011][Nguyen et al., 2014].

4.2.3 Meta-learning-based Approach

[Bilalli and Abell, 2016][Bilalli et al., 2016] study meta-learning approach for performing automated pre-processing of data. The authors use 17 features extracted from the datasets to train a meta-learner to rank preprocessing steps by prediction accuracy for 5 different classification algorithms (Naive Bayes, Logistic, IBk, JRip, J48). Meta-learner is implemented using a regression tree and training data are created by testing each dataset with each preprocessing transformations using 10-fold cross-validation. For each dataset-algorithm pair, possible transformations are classified as either good, bad, or neutral by the meta-learner which corresponds to whether the transformation increases the prediction accuracy, decreases it or doesn't have a significant contribution. The authors argue that the number of datasets for which the meta-learner suggested transformations were successful (*i.e.,* increased the prediction accuracy) outnumber the number of datasets for which the suggested transformations were unsuccessful (*i.e.,* decreased the prediction accuracy) .

[Loterman and Mues, 2015] compares a subsampling-based approach with meta-learning based approach for model type selection of 5 regression techniques, namely, linear regression, linear spline, regression tree, spline tree, and a decision tree. They conclude that subsample-performance based approach is significantly better than the dataset features based or average performance based approach. However, it is important to note that the authors consider only a small number of dataset features compared to other studies in the literature. The authors use 90 datasets from KEEL[10] dataset repository and expand the number of datasets available for training by switching the response with each of the numeric predictors and

[10] `http://www.keel.es/`

randomly subsampling the resulting dataset (*i.e.,* datasetoid). However, the final training dataset collection is not available publicly.

[Smith et al., 2001] uses 57 classification datasets from the UCI machine learning repository, 6 target classification algorithms (3 rule-based and 3 statistical) and 21 dataset features. A supervised neural network is used as the meta-learner although the size of the training set (67 datasets) is noted as being rather limited for this kind of process, with a danger of over-fitting. Evaluation of meta-learning is performed using 10-fold cross-validation. When predicting the best performing algorithm, 77% accuracy is obtained using the loose accuracy metric ($LA@1$) [Kalousis, 2002]. $LA@2$ and $LA@3$ measures would yield 95% and 98% accuracy, respectively.

[Sun and Pfahringer, 2013] compares different meta-learning algorithms for top-k ranking of classification algorithms using evaluation measures that include Normalized Cumulative Discounted Gain (NCDG) and Loose Accuracy(LA).

Finally, [Serban et al., 2013] and [Smith-Miles, 2008] provide a comprehensive survey of automated model development and data analysis using meta-learning and other approaches.

Although meta-learning for classification problems have been studied extensively, there has been only a limited number studies on meta-learning for regression. Our study focuses only on the regression algorithms for predictive analytics, introduces several new meta-features, and includes detailed evaluation comparing it to a number of different approaches including the baseline exhaustive search and ontology-based model selection approach.

4.2.4 Other Approaches

On the other side of the spectrum, some commercial platforms offer fully-automated prediction services. That is, the end-users have little or no control on the prediction workflow execution. Often times, most details of the prediction workflow are hidden from the end-users as well.

The Google Prediction API[11] provides an cloud endpoint for its users to upload training data in several ways. Once the upload is complete, training will be performed automatically. The users will then be able to submit prediction queries (*e.g.,* one or more instances) and will receive a response with the predicted answer(s) (either numeric or categorical based on the training data). The details of the implementation are proprietary but speculations is that they select from multiple machine learning algorithms.

The Watson Analytics platform[12] provides a web-based user interface for less technically-inclined users to upload their data into the platform. Once the data are available in the platform, the users can then follow the graphical interface to automatically develop a model and obtain visualizations and summaries of key insights from their datasets. An appropriate model type is automatically chosen by the system based on the dataset, however, end-users can not make any modifications.

These systems provide a very low-barrier entry point to data analytics for an audience with little or no data science background such as domain experts, software engineers and business decision makers. However, this approach can easily be limiting for more advanced scenarios when analysts would like to have more control over or knowledge of their analytics workflow.

4.3 Predictive Analytics Workflow

A prediction analytics problem may generally be modeled as

$$y \ = \ f(\mathbf{x}; \boldsymbol{\beta}) + \epsilon$$

where the response variable y is modeled as a function of the vector of predictor variables $\mathbf{x}$

[11]`https://cloud.google.com/prediction/docs/`
[12]`https://www.ibm.com/watson-analytics`

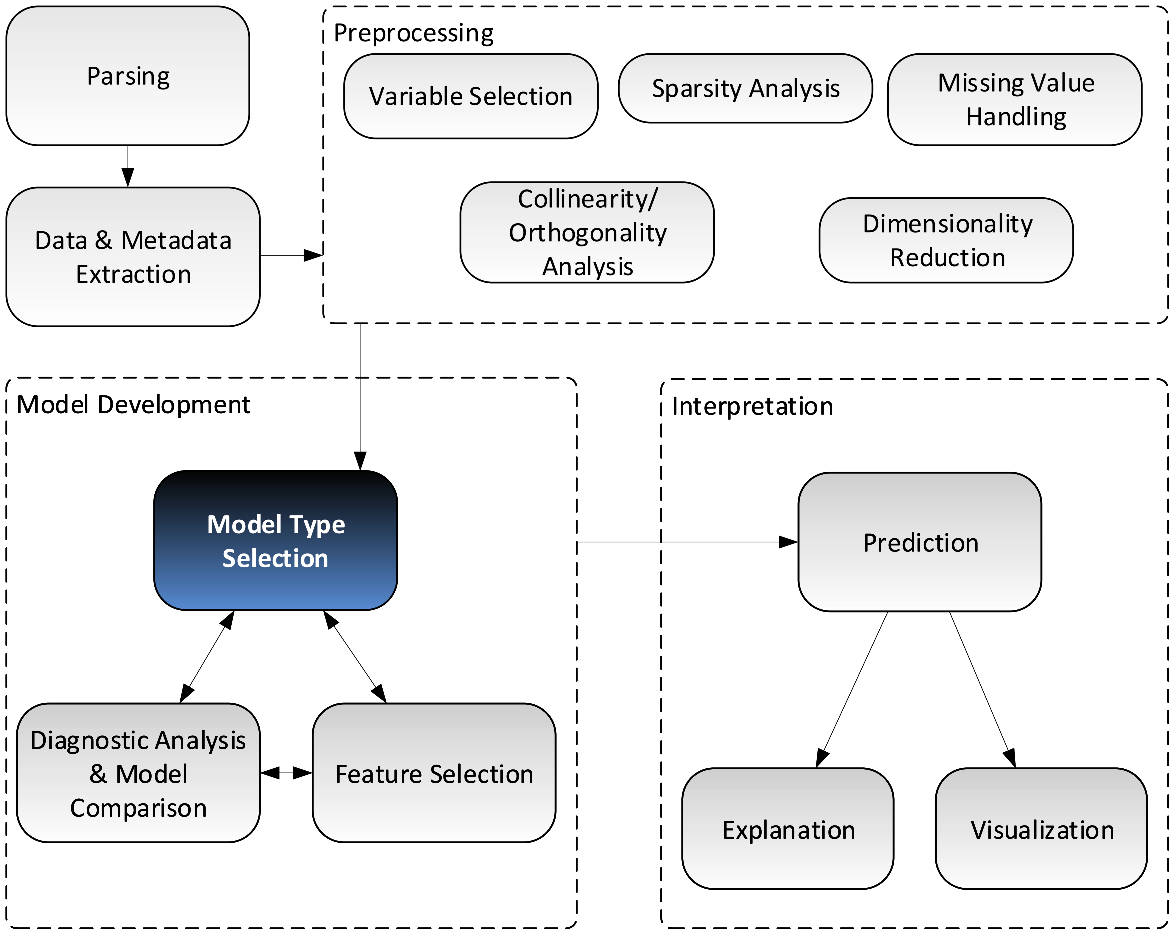


Figure 4.1: Typical Predictive Analytics Workflow

(also commonly referred as *features* or *attributes*) and the $\boldsymbol{\beta}$ parameter/coefficient vector plus the ϵ error/residual term representing what the model does not account for. For example, if the model is linear in the parameters, then

$$y = \boldsymbol{\beta} \cdot \mathbf{x} + \epsilon \ .$$

The parameter/coefficient vector $\boldsymbol{\beta}$ can be estimated by collecting training data (*e.g.,* a response vector $\mathbf{y}$ and a data matrix X) and minimizing some norm of the error/residual vector $\|\boldsymbol{\epsilon}\|$. For a linear model and L_2 norms, this involves using matrix factorization to solve the normal equations, $X^t X \boldsymbol{\beta} = X^t \mathbf{y}$. More general optimization techniques are required in other cases.

Other models for predictive analytics include time and/or prior values of y, *e.g.,* the auto-regressive order two time series model, $AR(2)$, predicts a future value for y based on prior values,

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \epsilon \ .$$

In almost all real life scenarios, predictive analytics involve many steps and are iterative exercises. Figure 4.1 shows a typical workflow for performing predictive big data analytics that covers most scenarios. We describe three major components of the workflow in the following.

4.3.1 Preprocessing

Preprocessing plays a significant role for building successful predictive models. At the very least, it involves preparing the training dataset for the input requirements of the target algorithms/techniques.

Below are the preprocessing operations we apply to each individual dataset automatically as needed to satisfy input requirements of individual modeling algorithms.

Nominal/Categorical variables are often encoded as strings. However, as most algorithms require numerical input for training and prediction, we create an injection from the string variables to integers using a bidirectional map. Given the string, it will give you the corresponding integer and vice versa. Additionally, if the data includes an ID string, that is, the variable uniquely identifies each instance of the dataset, it is removed from the dataset.

Missing values are rather common in many datasets and can be caused by issues such as sensor failures, incomplete data collection and etc. Missing values in a dataset must be handled as most algorithms expect complete instances in the training data. Ideally, missing value imputation should be performed with careful consideration based the inherent reason for why the missing values exist [Gelman and Hill, 2006]. However, this is not always possible in the context of automated modeling. We use the following imputation function to handle missing values in the input datasets[13].

Let X_j be a random variable with mean μ_j and variance σ_j^2,

$$X_j \sim \mathcal{N}(\mu_j, \sigma_j^2)\,.$$

then, each missing value in the jth attribute of the input matrix will be replaced with X_j.

In addition to meeting the input requirements of a target modeling algorithm, preprocessing can also be performed to **condition** the data to increase model performance. A few of the most commonly used approaches include feature selection, dimensionality reduction, outlier analysis and removal, sparsity analysis and *etc.* Except in cases where a chosen modeling algorithm implicitly performs any of these operations (*e.g.,* L_1 regularization via Lasso Regression), we do not count for these additional preprocessing operations in the meta-learning system.

[13]for more details see *Routine multivariate imputation* in [Gelman and Hill, 2006]

4.3.2 Model Development

Model development is a highly iterative process involving selecting a suitable model type, building and performing model diagnostics, and feature selection. In most use cases, many models are built in parallel and later on compared.

Model Type Selection (*i.e.,* Algorithm Selection)

Different modeling techniques have different assumptions about the dataset that must be satisfied (*e.g.,* OLS requires a tall input matrix $m \geq k$) in order to make effective predictions. For a given dataset, certain modeling techniques may be automatically eliminated due to the violations of their assumptions.

Diagnostic Analysis & Model Comparison

Among the remaining applicable modeling techniques, the best performing one for a given dataset is of great interest. Metrics such as the R^2 values and cross-validated root mean squared errors may be used to select the best or top modeling techniques.

Feature Selection

Eliminating features that are not needed for making good predictions may increase the robustness of the model and reduce the risk of over-fitting. This procedure is often carried out during the process of model comparisons but may also be done during preprocessing or model-type selection via selecting a modeling technique with implicit feature selection.

4.4 Meta-learning

Using notions from Rice's formal theory for algorithm selection problem [Rice, 1976], we may represent the model type selection problem as in Figure 4.2. Following the diagram, the problem can be defined as the following: Given a dataset $d \in D$, features extracted from the dataset $f(d) \in F$, the available metadata $m \in M$, and the modeling algorithms $\alpha \in A$, find a mapping $\alpha \in S(f(d), m)$ that maximizes performance of $p(\alpha(d))$.

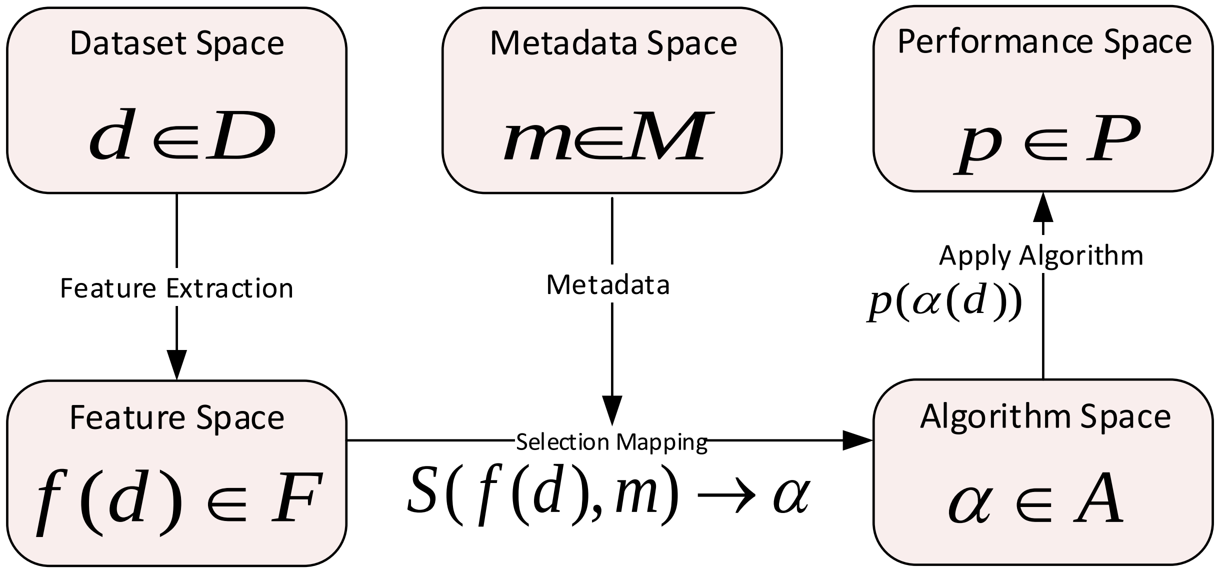


Figure 4.2: Rice Algorithm Selection Theory Diagram adapted from [Smith-Miles, 2008]

The abstraction of a dataset by its features allows us to treat this as a learning problem that creates a mapping S that for every $d \in D$, $S(f(d), m))$ would return the top performing algorithm. Since predictive analytics techniques can be considered as learners, learning the optimal selection mapping S becomes **meta-level**, hence the name for meta-learning. Without abstraction, algorithm selection can only be done as an exhaustive search in which one needs to explore all $\alpha \in A$ to find the optimal α that maximizes performance $p(\alpha(d))$.

We argue that feature extraction introduces minimal overhead and therefore provides significant advantage over exploring the full algorithm space provided that the meta-learning algorithm can provide a satisfactory selection mapping for $\alpha \in S(f(d), m)$.

For the implementation and evaluation of the meta-learning system, we use the open source ScalaTion framework which supports multi-paradigm modeling for running our analyses. Written in Scala language with an extensive support for simulation, optimization, data analytics including clustering, prediction and classification, *etc.* which could be easily integrated in a large scale data analysis pipeline, ScalaTion is an ideal platform for predictive data analytics.

In the following subsections, we describe individual components of the meta-learning system.

4.4.1 Feature Space

In order to train our meta-learner, we first need to represent datasets using meta-features extracted from the dataset d. Table 4.1 lists all the meta-features that we currently automatically extract from the datasets using ScalaTion . Most of the listed 21 meta-features are commonly used statistical features from the literature, [Brazdil et al., 1994], [Bilalli et al., 2016]. Since the literature mainly focuses on meta-learning for classification problems, we have omitted the features such as *majority/minority class ratio*, *class entropy*, *number of classes* and etc. Additionally, we have added a few extra features we believe that might be useful for regression problems (*i.e.*, prediction).

Dimensionality represents the log ratio of instances and features, and serves as another scale for the degrees of freedom. This value becomes negative when there are fewer instances than features, which implies negative degrees of freedom.

Matrix Condition can be used to check for multi-collinearity. If a matrix is ill-conditioned, then regularization may be needed.

Response Skewness and *Response Kurtosis* are measures of skewness and long-tailedness, respectively, of the distribution of the response. Both serve as measures of deviations from the normal distribution, which is an important factor for developing Generalized Linear Models (GLM).

Response Coefficient of Variation measures the relative variability of the response. If it is close to 1, then the response most likely follows an exponential distribution. A value less than 1 typically means the variability is relatively low. If higher than 1, then the variability is relatively high.

Distinct Ratio Response measures the the level of uniqueness in the values of the response. If this ratio approaches 0, then the set of unique responses is very small.

Table 4.1: Dataset Meta-features($f(d)$)

META-FEATURE	**DOMAIN**	**IMP**	**DESCRIPTION**
# of Instances	Integer	N/A	Total number of instances (m)
# of Attributes	Integer	N/A	Total number of predictor variables (k)
Degrees of Freedom	Integer	0.686	Base degrees of freedom computed as $df = m - k - 1$
Ratio of Degrees of Freedom	Decimal	0.671	$rdf = \frac{m-1}{df}$, $rdf = 1$ is ideal.
Dimensionality	Decimal	0.645	$log_{10}(m/k)$ Ratio of #instances to#features in logarithmic scale.
Matrix Condition	Decimal	0.627	$log_{10}(ConditionNumber)$ of the input matrix. $ConditionNumber$ of matrix is the ratio of largest singular value to smallest.
Nonnegative Response	Binary	0.571	Whether the response is non-negative or not (Yes, No)
Domain Response	Binary	0.699	Domain of the response variable (Integer or Decimal)
Distinct Ratio Response	Decimal	0.639	Ratio of number of distinct values in the response to the total number of instances (m)

Table 4.1: Dataset Meta-features($f(d)$)

META-FEATURE	**DOMAIN**	**IMP**	**DESCRIPTION**
Response Coefficient of Variation	Decimal	0.638	Ratio of the response standard deviation to the response mean ($s/\overline{y}$) where $$\overline{y} = \frac{\sum_{i=1}^{m} y_i}{m}\ ,\ s = \sqrt{\frac{\sum_{i=1}^{m}(y_i - \overline{y})^2}{m-1}}$$
Response Skewness	Decimal	0.646	Skewness parameter of the response variable computed as $$\frac{\sum_{i=1}^{m}(y_i - \bar{y})^3/m}{s^3}$$
Response Kurtosis	Decimal	0.633	Kurtosis of the response variable computed as $$\frac{\sum_{i=1}^{m}(y_i - \bar{y})^4/m}{s^4}$$
% Numeric Attributes	Decimal	0.537	Percentage of numeric attributes among all attributes $\#numericAttrs/\#allAttrs$
% Nominal Attributes	Decimal	0.493	Percentage of nominal/categorical attributes among all attributes ($2 < \#distinctVals < 20$) $\#nominalAttrs/\#allAttrs$
% Binary Attributes	Decimal	0.529	Percentage of binary attributes among all attributes $\#binaryAttrs/\#allAttrs$

Table 4.1: Dataset Meta-features($f(d)$)

META-FEATURE	DOMAIN	IMP	DESCRIPTION
Mean Means of Numeric Attributes	Decimal	0.560	Let N be the number $numericAttrs$ $$meanMeans = \left(\sum_{i=1}^{N} mean(numericAttrs_i)\right)/N$$
Mean Stddev of Numeric Attributes	Decimal	0.545	$meanStddev = \left(\sum_{i=1}^{N} stddev(numericAttrs_i)\right)/N$
Mean Kurtosis of Numeric Attributes	Decimal	0.521	$meanKurtosis = \left(\sum_{i=1}^{N} kurtosis(numericAttrs_i)\right)/N$
Mean Skewness of Numeric Attributes	Decimal	0.529	$meanSkewness = \left(\sum_{i=1}^{N} skewness(numericAttrs_i)\right)/N$
Max # Nominal Distinct Values	Integer	0.516	Number of distinct values in the nominal attribute with the maximum number of distinct values among all nominal attributes
Min # Nominal Distinct Values	Integer	0.531	Number of distinct values in the nominal attribute with the minimum number of distinct values among all nominal attributes
Mean # Nominal Distinct Values	Decimal	0.532	Let N_2 be the number of $nominalAttrs$: $$meanDistinctVals = \frac{\sum_{i=1}^{N_2} \#distinctVals(i)}{N_2}$$

Table 4.1: Dataset Meta-features($f(d)$)

META-FEATURE	**DOMAIN**	**IMP**	**DESCRIPTION**
Mean Stddev # Nominal Distinct Values	Decimal	0.509	$\sqrt{\frac{\sum_{i=1}^{N_2} \#distinctVals(i) - meanDistinctVals}{N_2}}$

4.4.2 Metadata Space

In addition to the meta-features extracted from the data, existing metadata may be leveraged to represent the dataset as well. Even though compiling metadata might seem cumbersome, many communities in life sciences have well-established minimum information standards (*e.g.,* Minimum information about a microarray experiment (MIAME) [Brazma et al., 2001]) to accompany published datasets to ensure the data could be analyzed, verified and reproduced by the broader scientific community. We limit this paper's scope to only utilizing $f(d)$ for selection mapping (S) and consider inclusion of metadata as future work.

4.4.3 Algorithm Space

We consider the following prediction algorithms that make up the algorithm space.

- Ordinary Least Squares Regression (OLS) (ScalaTion)
- Weighted Least Squares Regression (WLS) (ScalaTion)
- Back-elim Regression ($BackElim$) (ScalaTion)
- Response Surface Analysis *Quadratic Expansion* ($ResponseSurface_{quad}$) (ScalaTion)
- Response Surface Analysis *Cubic Expansion* ($ResponseSurface_{cubic}$) (ScalaTion)
- Log Transformed Regression ($LogTrans$) (ScalaTion)
- Root Transformed Regression ($RootTrans$) (ScalaTion)
- Exponential Regression (Exp) (R)
- Poisson Regression ($Poisson$) (R)
- Inverse Gaussian Regression ($InvGauss$) (R)

- Gamma Regression (*Gamma*) (R)
- Ridge Regression (*Ridge*) (R, ScalaTion)
- Lasso Regression (*Lasso*) (R, ScalaTion)
- Partial Least Squares Regression (*PLS*) (R)
- Principal Components Regression (*PCR*) (R)

Algorithms marked as ScalaTion have a native ScalaTion implementation, whereas algorithms marked as R have been integrated to the ScalaTion framework using the JRI (Java/R Interface) library[14].

One may notice that above is not an exhaustive list of algorithms for performing predictive analytics and can be extended to include popular algorithms such as neural-nets, support vector regression, regression trees and many others. The selection of algorithms are mainly based on the availability in ScalaTion , ease of interpretation of created models, and reasonable performance with default options and therefore not requiring manual tuning of parameters.

4.4.4 Performance Space

We collect a number of performance metrics for evaluation. Root Mean Squared Error ($RMSE$) is defined as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{n}}$$

where y_i stands for the observed response for the ith instance and $\hat{y}_i$ stands for the predicted response for the ith instance. The $RMSE$ metric is useful for comparing models with the same input dataset as it is within the same scale of the response variable. However, it is

[14]See `https://www.rforge.net/JRI/` for the official library page.

Procedure 1 Creating Performance Space

```
 1: procedure COLLECTMETRICS ($D, A$)
 2:     $k \leftarrow 10$                                        ▷ # folds for cross-validation
 3:     $n \leftarrow 10$                                        ▷ # runs
 4:     $P \leftarrow Map((d, \alpha) \rightarrow Tuple_3[n])$
 5:     for all $d \leftarrow D$ do
 6:         for all $\alpha \leftarrow A$ do
 7:             $perf \leftarrow Tuple_3[n]$
 8:             for $i \leftarrow 0\, until\, n$ do
 9:                 $model \leftarrow$ BUILDMODEL ($d, \alpha$)
10:                 $(RRSE, RMSE, time) \leftarrow$ TRAINANDCROSSVALIDATE ($model, k$)
11:                 $perf[i] \leftarrow (RRSE, RMSE, time)$
12:             end for
13:             $P(d, \alpha) \leftarrow perf$
14:         end for
15:     end for
16:     return $P$
17: end procedure
```

of little use if one wants to compare performance of a regression technique across different datasets since response variable for each dataset will be in a different scale.

To overcome this limitation, Root Relative Squared Error ($RRSE$) may be used. The $RRSE$ metric is defined as

$$RRSE = \sqrt{\frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{m}(y_i - \overline{y})^2}}$$

which provides a ratio of the prediction error to the error of predicting the mean response ($\overline{y}$) for every instance. Therefore, a perfect model would yield $RRSE = 0$ whereas any $RRSE > 1$ means the model performs worse than just predicting the mean response for every instance.

The performance space may be extended to include other important metrics such as Akaike information criterion (AIC), Mean absolute error (MAE), and Coefficient of determination ($R^2 = 1 - RRSE^2$), however, they were not considered in the context of this initial study.

Procedure 1 describes how the performance space is built using the performance metrics described above. As a summary, a regression model is built for each dataset and algorithm combination and $RMSE, RRSE$, and *time* are recorded after performing a 10-fold cross-validation. In order to account for randomness from the cross-validation, this process is repeated 10 times for each run and results are averaged.

4.4.5 Meta-learning for Selection Mapping

Random forest is a very popular classification algorithm due to its relative simplicity and resistance to overfitting [Hastie et al., 2009]. Additionally, random forests can also be used for ranking using the individual class probabilities of a multi-class classifier. Hence, we employ a random forest classifier to learn a mapping $S(f(d)) \rightarrow \alpha$ from the features of a dataset $f(d)$ to an algorithm $\alpha \in A$ that yields maximum performance among the algorithms in the algorithm space A.

Procedure 2 may be referred for a visual representation of how the training set for meta-learning is created. First, meta-features shown in Table 4.1 are extracted from each dataset. Next, the best performing algorithm for each dataset is selected by picking the algorithm that produces the lowest cross-validated $RMSE$ for the given dataset. Procedure 1 summarizes the process of generating the performance metrics including $RMSE$ in Section 4.4.4. All experiments are run on a workstation with an 8 core Intel Core i7-4900MQ CPU operating at 2.80GHz with 32GB RAM.

The training set is then used to train a multi-class random forest classifier with dataset features as input and the best performing algorithm label as the output. In addition to the top prediction, 2 more predictions for each dataset are made by the picking models that yield top-2 and top-3 class probabilities.

Procedure 2 Generating The Training Set for $S(f(d))$

```
1: procedure GenerateTrainingSet ($D, P$)
2:     $S \leftarrow Map\ (f \rightarrow \alpha)$
3:     for all $d \leftarrow D$ do
4:         $f \leftarrow$ extractFeatures ($d$)                    ▷ $f(d)$
5:         $\alpha_{best} \leftarrow argmin_{\alpha \in A}(P(d, \alpha)_{RMSE})$
6:         $S$ += $(f \rightarrow \alpha_{best})$
7:     end for
8:     return $S$
9: end procedure
```

4.5 Evaluation

4.5.1 Training Datasets

In order to evaluate our system, we have created a dataset collection from a number of different sources and domains. The criterion for selection was to have a numeric response in order to run regression algorithms. We have performed preprocessing on the datasets such as numeric encoding of nominal or binary string variables, removing instances with missing values etc. only when needed to satisfy input requirements of the regression algorithms. We have removed any dataset from the collection if majority of the algorithms failed to complete due any reason such as being ill-conditioned, not having enough instances to converge, etc. As a result, we have ended up with a total of 114 regression datasets in our collection obtained from various sources listed below. As a summary:

- 43 datasets from UCI Machine Learning Repository [Lichman, 2013]
- 17 datasets from OpenML [Kietz et al., 2014]
- 16 datasets from publicly available packages in R[15]

[15]See `https://vincentarelbundock.github.io/Rdatasets/datasets.html` for an unofficial compilation.

- 12 datasets from Luis Torgo Regression datasets collection[16]
- 9 datasets from Bilkent University Function Approximation Library[17]
- 9 datasets from NCI-60 Cell Line panel: Similar to [Lee et al., 2011], we have used gene expression data obtained from Affymetrix HG-U133A and B chips normalized using the GCRMA method as predictors of proteins with top 9 most variance obtained from Reverse-phase protein lysate arrays (RPLA).
- 8 datasets from other sources[18]

The collection includes a very diverse set of datasets from various domains including but not limited to life sciences, finance, engineering, physical sciences and so on. Additionally, datasets have quite different characteristics in terms of size, feature types (*e.g.,* binary, numeric, categorical), response variable properties and etc. Table 4.2 shows some key statistics for the collection.

Table 4.2: Dataset Collection by Numbers

Number of Datasets	114
Min # Instances	50
Max # Instances	999249
Min # Predictors	1
Max # Predictors	3489
Min Dimensionality	200 (50 x 4)
Max Dimensionality	44910250 (583250 x 77)
Min Input Size	1.4KB
Max Input Size	283MB

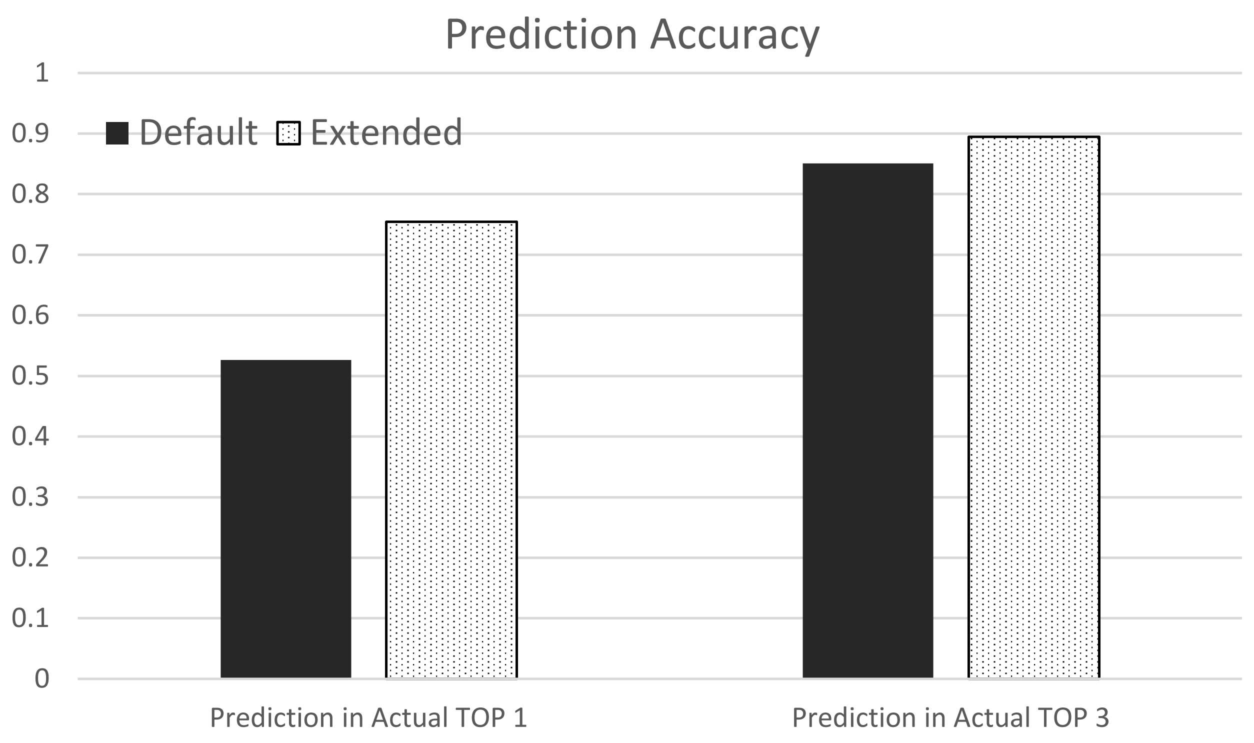


Figure 4.3: Accuracy of Meta-learning system

4.5.2 Meta-learning Performance Results

As described in Section 4.4.5, we have trained a random forest classifier using the extracted features and performance metrics for the 114 datasets in the dataset space. The resulting classifier had 100 random trees with an average tree size of 75. The largest tree had a size of 91 while the smallest tree in the forest had a size of 59.

Random forests are resilient to overfitting provided that the fraction of relevant (*i.e.,* non-noise) attributes in the input dataset is not small and the number of trees is large [Hastie et al., 2009].

The classifier is evaluated with 10-fold cross-validation and top 3 predictions from each fold are recorded.

In the *default* evaluation strategy, the actual top performing algorithm is chosen based on the lowest average $RMSE$ over 10 runs. The *default* evaluation revealed that the algorithm meta-learning system predicted was in fact the actual top performing algorithm in only 52% of the cases (60 out of 114). When checked whether the predicted algorithm was in the actual top 3 performing algorithms, the accuracy was 85% (97 out of 114).

However, since each model, $\alpha(d)$ (*i.e.,* dataset-algorithm pair), is run 10 times and cross validated error metrics (*cv-folds are created randomly in each run*) are recorded, it was not always straightforward to identify the top performing algorithm due to overlapping errors between different algorithms for the individual runs. In those cases, an algorithm is also marked as top if a paired t-test against the algorithm having the lowest average $RMSE$ fails to reject the null hypothesis that the mean error of two algorithms for the dataset are not significantly different. Algorithm 1 presents how the top algorithm(s) are identified for each dataset in the testing collection. A similar approach is taken for identifying actual top-3 performing algorithms.

[16]`http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html`

[17]`http://funapp.cs.bilkent.edu.tr/DataSets/`

[18]For more details on the datasets see `github.com/scalation/analytics`

Algorithm 1 Identifying Top Performing Models

```
1: procedure TopAlgorithms ($D, P, A$)
2:     $top_1 Algs \leftarrow Map(d \rightarrow Set_\alpha)$
3:     for all $d \leftarrow D$ do
4:         $algs \leftarrow Set_\alpha$
5:         $\alpha_{best} \leftarrow argmin_{\alpha \in A}(P(d, \alpha)_{RMSE})$
6:         $algs \mathrel{+}= \alpha_{best}$
7:         for all $\alpha \leftarrow A$ do
8:             $p \leftarrow$ t-test$(P(d, \alpha)_{RMSE}, P(d, \alpha_{best})_{RMSE})$
9:             if $p > .05$ then                    ▷ failed to reject the null hypothesis
10:                $algs \mathrel{+}= \alpha$
11:            end if
12:        end for
13:        $top_1 Algs(d) \leftarrow algs$
14:    end for
15:    return $top_1 Algs$
16: end procedure
```

Following the *extended* strategy described above, the evaluation revealed that the predicted algorithm by the meta-learner was in fact the actual top performing model in 75% of the cases (86 out of 114). Similarly, the predicted algorithm by the meta-learner was in the actual top-3 performing models in 89% of the cases (102 out of 114). Figure 4.3 displays how the two evaluation strategies compare.

We have also trained a meta-learner using kNN to serve as a comparison with the random forest classifier. Ranking using a k-nearest neighbors (kNN) classifier is a commonly used method for obtaining top-k rankings in meta-learning. After identifying the closest neighbors of a dataset using a distance metric such as "Euclidean Distance" , a weighted average (neighbors are weighted by the inverse of their distance) of each individual neighbor's actual ranking is used for computing the candidate dataset's predicted ranking of modeling algorithms.

To compare the two meta-learners, we use the following metrics commonly used for evaluating top-k ranking.

Mean Average Precision ($MAP@k$) is calculated as the following. For each instance, if the predicted algorithm at $rank_i$ ($1 \leq i \leq k$) matches the actual algorithm at $rank_i$, then it is scored as 1, otherwise 0. The precision scores for each rank are averaged and the MAP@k is calculated by taking the mean of individual average rankings.

Loose Accuracy ($LA@k$) [Kalousis, 2002] is a metric used as an alternative for the traditional accuracy measure and it is particularly useful for cases where the top predictions may not differ significantly. $LA@k$ uses a binary scoring method in which an instance is scored 1.0 if any of the top-k predictions match the actual top-1 prediction, 0 otherwise.

Normalized Discounted Cumulative Gain ($NDCG@k$) [Wang et al., 2013] is another popular metric for evaluating top-k rankings. In contrast to the binary relevance score used in MAP, $NDCG$ uses a graded penalty logarithmically proportional to position of the suggestion if a top modeling algorithm appears in the top-k results but out of order. For example $MAP@k$ would yield a score of 0 for an instance if all top-k performing algorithms are in the results but in a slightly different order (e.g, actual top-1 appears in second rank, actual top-2 appears first, etc.). On the other hand, $NDCG$ may yield a much higher score in this instance as the rankings are only penalized log proportional to their actual position in the top-k results. In a hypothetical case where A_1, A_2 and A_3 are the top-3 ordered performing algorithms for a dataset and (A_3, A_1, A_2) is the ranked list of suggestions, $MAP@3$ would score this instance as 0 and $NDCG@3$ would score it as 0.82.

Figure 4.4 presents how meta-learners implemented using a random forest and kNN compare using different evaluation measures listed above. The random forest based meta-learner clearly performs better than the kNN classifier based meta-learner according to all of the metrics discussed.

Finally, the importance of the features used in the meta-learning is evaluated. The random forest classifier employs an entropy-based impurity measure during the training phase.

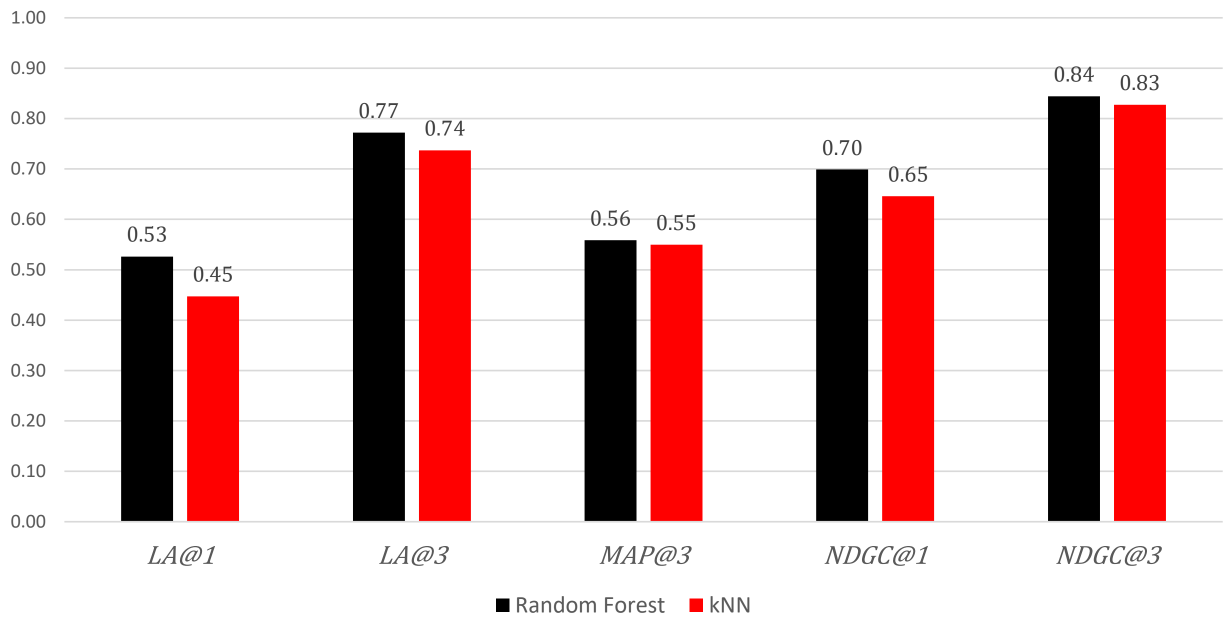


Figure 4.4: Performance Comparison of Random Forest vs. kNN

The average impurity decrease measure for individual features is obtained from the classifier and the features are ranked by the measure averaged over 10 cross-validation folds to identify most important features. The results revealed that the meta-features related to the dimensionality (dim, df, rdf) and the meta-features based on the response variable $(skewness, kurtosis, distinct\_ratio, sttdev\_mean\_ratio,\ domain)$ are more important for selecting optimally performing algorithm (*i.e.,* model-type). Additionally, condition number of the input matrix is also ranked higher in importance. Refer to the Table 4.1 for the importance scores of all features.

4.5.3 Meta-learning vs. ScalaTion

Additionally, we provide a comparison of meta-learning with two different baselines. The first baseline is ScalaTion in which we exhaustively run all algorithms listed in the section 4.4.3

and record the total training time and the lowest $RRSE$ score from the top performing algorithm. As a second baseline, we have used the popular AutoWEKA [Thornton et al., 2013] tool. We have run AutoWEKA with default parameters and used a time-limit of 30 minutes per dataset. As AutoWEKA suggests trying at least thousands of configurations to obtain reliable results, we have only included datasets that are less than 1MB to make sure each dataset is run sufficient number of times. This criterion matched 47 out of 114 datasets in our collection.

Table 4.3 displays both the lowest error and the total time taken to complete in seconds for all 47 datasets included in the comparison. The $RRSE$ values highlighted in bold denote the lowest error obtained for each dataset. After inspecting the results in detail, we have identified that for the majority of the datasets where AutoWEKA outperformed meta-learning, it was consistently better than both meta-learning and the ScalaTion baseline at the same time. We attribute this difference due to the additional algorithms available to AutoWEKA such as neural nets, k-nearest neighbors, MultilayerPerceptron, Random Forest, KStar, Additive regression, REPTree, M5Rules, M5P and ensemble techniques such as RandomCommittee[19]

To present a better comparison, future work should include the missing techniques in the ScalaTion framework and consider additional metrics such as Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC).

[19]See `https://www.cs.waikato.ac.nz/ml/weka/documentation.html` for a detailed documentation of available techniques in the WEKA environment.

Table 4.3: Performance Comparison Against Baseline

	BASELINE				META-LEARNING	
Dataset	Auto-WEKA		ScalaTion		Top Prediction	
	$RRSE$	$time(s)$	$RRSE$	$time(s)$	$RRSE$	$time(s)$
wisconsin_breast	0.9454	$1.80E+03$	**0.9244**	1.229	**0.9244**	0.32
auto_price	**0.3302**	$1.80E+03$	0.4287	0.603	0.4463	0.038
forest_fire	**0.9059**	$1.80E+03$	0.9330	0.749	0.9344	0.052
housing	**0.3279**	$1.80E+03$	0.4591	1.037	0.5303	0.035
servo	**0.3479**	$1.80E+03$	0.6590	0.936	0.6590	0.01
student_1	**0.8953**	$1.80E+03$	0.9355	1.806	0.9355	0.051
yacht	**0.0553**	$1.80E+03$	0.0761	1.037	0.0761	0.012
fb_metric_1	0.7249	$1.80E+03$	**0.2319**	1.169	**0.2319**	0.028
fb_metric_2	0.1822	$1.80E+03$	**0.1754**	1.215	**0.1754**	0.025
fb_metric_3	0.8813	$1.80E+03$	**0.6734**	1.240	0.7846	0.022
fb_metric_4	0.6705	$1.80E+03$	**0.6050**	1.221	0.6294	0.021
chick_weight	**0.4841**	$1.80E+03$	0.4992	0.923	0.5147	0.005
life_cycle_savings	0.9119	$1.80E+03$	**0.8886**	0.930	0.9042	0.004
body_fat	0.5831	$1.80E+03$	**0.5594**	2.739	**0.5594**	0.02
plastic	**0.2314**	$1.80E+03$	0.2316	1.895	0.2316	0.081
quake	**0.9962**	$1.80E+03$	0.9987	1.940	0.9987	0.019
solar	**0.9283**	$1.80E+03$	0.9425	1.208	0.9481	0.015
olympic2000	**0.1635**	$1.80E+03$	0.4428	1.111	0.4493	0.007
prostate	0.7407	$1.80E+03$	**0.6400**	0.343	**0.6400**	0.005
nist_gauss_1	**0.0631**	$1.80E+03$	0.6925	0.272	0.6937	0.003
concrete_slump_3	**0.1001**	$1.80E+03$	0.3471	0.535	0.3477	0.005

Table 4.3: Performance Comparison Against Baseline

Dataset	BASELINE				META-LEARNING	
	Auto-WEKA		SCALATION		Top Prediction	
	RRSE	*time(s)*	*RRSE*	*time(s)*	*RRSE*	*time(s)*
concrete_slump_2	**0.6883**	$1.80E+03$	0.7385	0.483	0.7590	0.005
concrete_slump_1	**0.8026**	$1.80E+03$	0.8821	0.459	0.8839	0.006
auto-mpg	**0.3372**	$1.80E+03$	0.3787	0.824	0.4338	0.011
computer_activity_2	**0.1634**	$1.80E+03$	0.1997	8.458	0.1997	0.157
istanbul_stock	0.6581	$1.80E+03$	**0.6561**	0.479	**0.6561**	0.01
tecator_moisture	0.2278	$1.80E+03$	**0.2068**	17.299	0.2073	0.072
tecator_fat	**0.1734**	$1.80E+03$	0.1763	17.203	0.1763	0.076
tecator_protein	0.2577	$1.80E+03$	**0.2190**	18.660	**0.2190**	0.068
bike_sharing_total_day	**0.3315**	$1.80E+03$	0.4553	0.754	0.4553	0.018
visualizing_soil	**0.0042**	$1.80E+03$	0.2169	2.363	0.4082	0.052
abalone	**0.6691**	$1.80E+03$	0.6892	2.338	0.6894	0.052
student_2	0.892	$1.80E+03$	**0.8566**	2.166	0.8588	0.026
cars	**0.331**	$1.80E+03$	0.5038	2.124	0.5262	0.075
weather_1	**0.0834**	$1.80E+03$	0.2492	2.271	0.2492	0.033
weather_2	**0.0836**	$1.80E+03$	0.2440	2.194	0.2471	0.029
treasury	**0.0568**	$1.80E+03$	0.0719	3.149	0.2700	0.023
qsar_47555	**0.7925**	$1.80E+03$	0.8192	2.793	0.8192	0.095
qsar_31274	–	–	**0.5414**	81.918	0.5521	0.463
kin8nm	**0.43**	$1.80E+03$	0.7664	6.320	0.7666	0.517
computer_activity_1	**0.133**	$1.80E+03$	0.1745	43.128	0.1745	0.261
bank8fm	**0.2045**	$1.80E+03$	0.2553	6.687	0.2553	0.095

Table 4.3: Performance Comparison Against Baseline

Dataset	BASELINE				META-LEARNING	
	Auto-WEKA		SCALATION		Top Prediction	
	$RRSE$	$time(s)$	$RRSE$	$time(s)$	$RRSE$	$time(s)$
crime_norm	0.5841	$1.80E+03$	**0.5793**	62.058	0.5824	0.611
parkinson_1	**0.4382**	$1.80E+03$	0.9252	7.014	0.9252	0.157
parkinson_2	**0.4364**	$1.80E+03$	0.9140	6.601	0.9140	0.153
puma8nh	**0.6573**	$1.80E+03$	0.7936	2.704	0.7938	0.103
pol	**0.1293**	$1.80E+03$	0.5400	138.935	0.7308	0.6

4.5.4 Meta-learning vs. Ontology-based Suggestion

In this section, we present a comparison of the meta-learning approach described in this paper with an ontology-based suggestion approach.

In [Nural et al., 2015], we have introduced an ontology-based suggestion approach for the model selection problem. Using the Analytics Ontology[20] describing various prediction modeling techniques, parameters, variables, and model features such as dimensionality, condition of the input matrix etc., candidate datasets are modeled in the ontology as an abstraction via its extracted features. Modeling techniques are annotated by domain experts with logical rules (*i.e.,* equivalence axioms) to describe when it is appropriate to use that technique to analyze a candidate dataset.

Using an ontological reasoner such as HermiT [Shearer et al., 2008], it becomes possible to retrieve suggestions from the ontology via matching the features of a candidate dataset with appropriate modeling techniques for prediction. Interested readers may refer

[20]`https://github.com/scalation/analytics/`

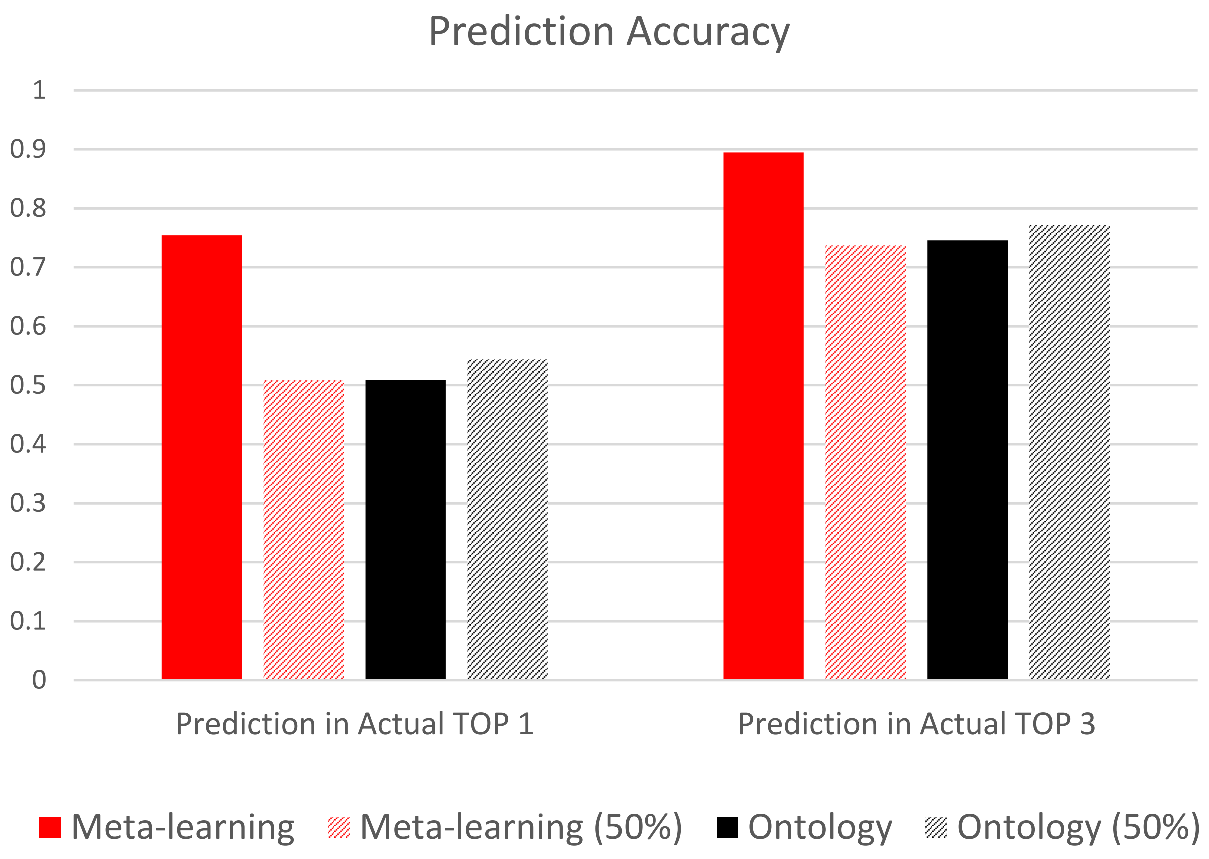


Figure 4.5: Comparison of Meta-learning vs. Ontology-based approaches

to [Nural et al., 2015] for detailed description of the approach.

For comparison, the same collection of 114 datasets (see 4.5.1) is used. Using ScalaTion , each dataset is provided as an input to the ontology-based suggestion engine and the suggested modeling technique is recorded. The suggestions are then evaluated using the extended evaluation strategy described in section 4.5. The figure 4.5 shows how the ontology-based approach compares to meta-learning based approach. When predicting the top-1 performing technique, the ontology-based approach achieves 51% accuracy compared to 75% with meta-learning. When evaluating whether the suggested technique is within the actual top-3 performing models, the ontology-based approach achieves 75% accuracy compared to 89% by the meta-learning approach.

Based on the results reported above, the ontology-based approach is not a strong alternative for the model-selection problem alone. However, the ontology-based approach may still be relevant in the following scenarios.

- Performance of meta-learning depends on the quality and the size of the training set. Therefore, extending the algorithm space requires significant investment for collection of additional datasets to increase the size of the training set. To demonstrate this, we have performed a small-scale experiment and re-evaluated meta-learning and ontology-based suggestion performance on randomly selected 50% of the original dataset collection. As seen on Figure 4.5, the performance of ontology-based system stays relatively stable. However, meta-learning performance drops significantly. So, one can argue that ontology-based suggestion approach can be relevant with limited training set and large number of target algorithms.

- As discussed in Section 4.3, data analytics requires an iterative workflow with many other steps in addition to selecting the most predictive modeling algorithm. Even though the ontology-based approach may be insufficient to support most predictive modeling algorithm selection, it can still be utilized successfully for determining most predictive workflows. [Nguyen et al., 2014] presents a successful application of a well curated data-mining ontology to generate valid workflows for a data-analysis task using hierarchical task planning.

- In some data analysis settings, there may be other constraints in addition to the top performance for selecting a modeling technique. A few examples include model interpretability, parsimony, ability to inject theory and etc.

- Where there are multiple possible goals (prediction, forecast) and breadth, ontology may be very helpful to reduce the search space and maybe used in tandem with meta-learning in a hybrid setting.

4.6 Conclusion

This paper discusses the progress made in automated modeling, with particular emphasis on modeling via meta-learning for predictive analytics with regression-based algorithms. The major contributions and findings of this paper include that

- Metalearning may be utilized to efficiently reduce the algorithm space for predictive big data analytics.

- Dimensionality and characteristics of the response variable are the most important indicators of filtering algorithms. This is not surprising as Generalized Linear Models have specific assumptions on the response variable. Similarly, low dimensionality and negative degrees of freedom is an important indicator for using a regularization algorithm such as Lasso or Ridge Regression.

- Highly efficient and robust implementations of prediction algorithms, including *OLS*, *ridge*, and *lasso* regression exist in the ScalaTion framework.

- ScalaTion framework provides a viable alternative for big data analytics with its support for multi-paradigm modeling and efficient learning algorithms.

- Provide a comparison of meta-learning and ontology-based approaches for the model selection problem.

In addition to developing a meta-learning system, we have also introduced a number of new meta-features for better capturing characteristics related to the regression algorithms.

An emphasis of future work will be to extend the meta-learning system with additional learning algorithms in the algorithm space, provide support for alternative performance metrics for identifying optimally performing algorithms, and extend the meta-learning system to consider of available contextual metadata in addition to features automatically extracted from the datasets. Additionally, the future work will include comparison of the presented automated modeling approach to the approaches currently employed for automated modeling.

Chapter 5

Summary

This thesis presents our contributions to the field of predictive big data analytics with special focus on automated modeling. We have explored the landscape of automated modeling and investigated how ontology-based semantics and meta-learning can play a role assisting data analysts by automating one or more of the steps of a typical predictive data analytics workflow (refer to the Figure 4.1). Existing literature on automated modeling mainly focuses on classification problems. Additionally, the small number of studies focusing on regression (*i.e.,* prediction) consider only a limited number of modeling techniques (< 5). Our work extends the algorithm space with popular statistical techniques such as Generalized Linear Models family, Lasso Regression, Partial Least Squares Regression and considers 15 different algorithms in total.

Major contributions of this thesis may be summarized as follows:

- Investigation of machine learning based meta-learning approach for assisting automated /semi-automated construction of a predictive analytics workflow.

  We have successfully demonstrated a meta-learning system for predicting top performing model(s) among 15 different candidate algorithms based on dataset characteristics. Our system achieved 75% accuracy for predicting top-1 performing model correctly.

Additionally, the predicted model was in the actual top-3 in 89% of the cases. Moreover, we were the first to use a random-forest classifier as the meta-learner to predict the performance of regression algorithms. We were not able to directly compare the performance of our system with the literature on meta-learning for regression due to several reasons such as the differences in target algorithms, inability to reproduce the training datasets used in the studies and so on. The only quantitative study was performed by [Loterman and Mues, 2015] and as mentioned in related work, we were unable to acquire the datasets and there is minimal overlap with our modeling techniques.

- Investigation of ontology-based semantics for assisting automated/semi-automated construction of a predictive analytics workflow.

  We have demonstrated that ontology-based semantics can be useful for providing model-type suggestions for a user-provided dataset automatically. Leveraging the logical reasoning infrastructure the system is built upon, justifications of the suggestions can also be provided for more educated model building. In our evaluations, the meta-learning approach had better performance for correctly predicting top-1 performing algorithm, however, we have shown that the ontology-based approach can still be relevant when resources for collecting training datasets are limited and size of the algorithm space is large.

- Highly efficient implementations of popular algorithms including Lasso and Ridge regression in Scala base ScalaTion framework.

- A recent survey of the state-of-the-art automation efforts for assisting data analysts for performing predictive big data analytics.

- Identifying the key areas of improvement for building systems to support automated predictive data analytics efforts.

- The `scala-dash` tool as a demonstration of ontology supported interactive data analytics.

- Extensive evaluation of ScalaTion framework as a platform for predictive big data analytics.

After laying the foundation with the work presented in this thesis, we believe the following might be appropriate directions to explore in future studies.

- Incorporating metadata along with the features automatically extracted from the data for identifying most appropriate models. For example, the probability distribution of a response variable is one of the most important factors for choosing an appropriate model type for generalized linear models. If a response variable of a dataset follows an exponential distribution, Exponential Regression should be chosen, Similarly, Gamma Regression should be chosen when the response variable follows a Gamma distribution. Even though it is possible to automatically extract information such as skewness and kurtosis relating to the shape of the response distribution, it is challenging to correctly identify the probability distribution a variable follows. Some tools such as Expert-Fit [Law and McComas, 2003] provide identification however it is proprietary. If this information can be made available as metadata (*e.g.,* from problem definition or by visualization) it can be used as an additional feature during the model-type suggestion process.

- Investigation of a hybrid approach combining ontology-based semantics and meta-learning together to support a wider range of scenarios and/or steps in predictive big data analytics.

- Investigation of sub-sampling for very large datasets and extending meta-features to better characterize a dataset.

- Expanding the candidate modeling technique space in ScalaTion to provide a more competitive platform and to be able to provide a fair evaluation with existing alternatives.
- Expanding the `scala-dash` graphical data analysis tool with the contributions presented in this thesis and evaluate its usability.

We believe the automated modeling will continue to keep its relevance and become more essential to any data analytics effort amidst the current trend of explosive increase in the data production pace. Data analysts will be facing new challenges to timely analyze and interpret incoming big data streams from many sources and therefore will be seeking automated solutions to deal with the overwhelming amounts of data.

Bibliography

[Bernstein et al., 2002] Bernstein, A., Hill, S., and Provost, F. (2002). Intelligent Assistance for the Data Mining Process : An Ontology-based Approach.

[Bilalli and Abell, 2016] Bilalli, B. and Abell, A. (2016). Towards Intelligent Data Analysis : The Metadata Challenge. In *Proceedings of the International Conference on Internet of Things and Big Data - Volume 1: IoTBD*, number IoTBD, pages 331–338.

[Bilalli et al., 2016] Bilalli, B., Abelló, A., Aluja-Banet, T., and Wrembel, R. (2016). Automated Data Pre-processing via Meta-learning. In Bellatreche, L., Pastor, Ó., Almendros Jiménez, J. M., and Aït-Ameur, Y., editors, *Model and Data Engineering: 6th International Conference, MEDI 2016, Almer{í}a, Spain, September 21-23, 2016, Proceedings*, pages 194–208. Springer International Publishing, Cham.

[Boyd et al., 2011] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122.

[Brazdil et al., 1994] Brazdil, P., Gama, J., and Henery, B. (1994). Characterizing the Applicability of Classification Algorithms Using Meta-level Learning. In *Proceedings of the European Conference on Machine Learning on Machine Learning*, ECML-94, pages 83–102, Secaucus, NJ, USA. Springer-Verlag New York, Inc.

[Brazma et al., 2001] Brazma, A., Hingamp, P., Quackenbush, J., Sherlock, G., Spellman, P., Stoeckert, C., Aach, J., Ansorge, W., Ball, C. A., Causton, H. C., Gaasterland, T., Glenisson, P., Holstege, F. C. P., Kim, I. F., Markowitz, V., Matese, J. C., Parkinson, H., Robinson, A., Sarkans, U., Schulze-Kremer, S., Stewart, J., Taylor, R., Vilo, J., and Vingron, M. (2001). Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. *Nat Genet*, 29(4):365–371.

[Calcagno and de Mazancourt, 2010] Calcagno, V. and de Mazancourt, C. (2010). glmulti: An R Package for Easy Automated Model Selection with (Generalized) Linear Models. *Journal of Statistical Software*, 34(12):1–29.

[Codd, 1970] Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*, 13(6):377–387.

[Efron et al., 2004] Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *Ann. Statist.*, 32(2):407–499.

[Friedman et al., 2010] Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1):1–22.

[Gelman and Hill, 2006] Gelman, A. and Hill, J. (2006). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Analytical Methods for Social Research. Cambridge University Press.

[Godambe, 1991] Godambe, V. P. (1991). *Estimating Functions*. Oxford Statistical Science Series: 7. New York : Oxford University Press.

[Goodman and Flaxman, 2016] Goodman, B. and Flaxman, S. (2016). European Union regulations on algorithmic decision-making and a “right to explanation”. *arXiv preprint 1606.08813*.

[Hall et al., 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

[Harrell, 2015] Harrell, F. E. (2015). *Springer Series in Statistics Regression Modeling Strategies*. Springer, 2nd. editi edition.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer-Verlag New York.

[Hutter et al., 2011] Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). *Sequential Model-Based Optimization for General Algorithm Configuration*, pages 507–523. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Kalousis, 2002] Kalousis, A. (2002). *Algorithm Selection via Meta-Learning*. PhD thesis, University of Geneva.

[Kiefer and Monro, 1953] Kiefer, J. and Monro, S. (1953). Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, 4(3):502–502.

[Kietz et al., 2012] Kietz, J.-U., Serban, F., Bernstein, A., and Fischer, S. (2012). Designing KDD-workflows via HTN-planning. In *Proceedings of the 20th European Conference on Artificial Intelligence*, ECAI'12, pages 1011–1012, Amsterdam, The Netherlands, The Netherlands. IOS Press.

[Kietz et al., 2014] Kietz, J. U., Serban, F., Fischer, S., and Bernstein, A. (2014). "Semantics inside!" but let's not tell the data miners: Intelligent support for data mining. In *Lecture Notes in Computer Science*, volume 8465 LNCS, pages 706–720.

[Kotthoff et al., 2017] Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., and Leyton-Brown, K. (2017). Auto-WEKA 2.0: Automatic Model Selection and Hyperparameter Optimization in WEKA. *J. Mach. Learn. Res.*, 18(1):826–830.

[Kuhn, 2008] Kuhn, M. (2008). Building predictive models in R using the caret package. *Journal of Statistical Software*, 28(5):1–26.

[Lahabar and Narayanan, 2009] Lahabar, S. and Narayanan, P. J. (2009). Singular value decomposition on GPU using CUDA. In *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–10.

[Law and McComas, 2003] Law, A. M. and McComas, M. G. (2003). ExpertFit Distribution-fitting Software: How the ExpertFit Distribution-fitting Software Can Make Your Simulation Models More Valid. In *Proceedings of the 35th Conference on Winter Simulation: Driving Innovation*, WSC '03, pages 169–174. Winter Simulation Conference.

[Lee et al., 2011] Lee, D., Lee, W., Lee, Y., and Pawitan, Y. (2011). Sparse partial least-squares regression and its applications to high-throughput data analysis. *Chemometrics and Intelligent Laboratory Systems*, 109(1):1–8.

[Lichman, 2013] Lichman, M. (2013). UCI Machine Learning Repository.

[Loterman and Mues, 2015] Loterman, G. and Mues, C. (2015). Learning algorithm selection for comprehensible regression analysis using datasetoids. *Intelligent Data Analysis*, 19:1019–1034.

[Mandel, 1982] Mandel, J. (1982). Use of the Singular Value Decomposition in Regression Analysis. *The American Statistician*, 36(1):15–24.

[Miller et al., 2013] Miller, J. A., Cotterell, M. E., and Buckley, S. J. (2013). Supporting a Modeling Continuum in Scalation: From Predictive Analytics to Simulation Modeling. In

Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World, pages 1191–1202. IEEE Press.

[Miller et al., 2010] Miller, J. A., Han, J., and Hybinette, M. (2010). Using domain specific language for modeling and simulation: scalation as a case study. In *Proceedings of the Winter Simulation Conference*, WSC'10, pages 741–752. Winter Simulation Conference.

[Nelder and Baker, 2004] Nelder, J. A. and Baker, R. J. (2004). Generalized Linear Models. In *Encyclopedia of Statistical Sciences*. John Wiley & Sons, Inc.

[Nguyen et al., 2011] Nguyen, P., Hilario, M., and Kalousis, A. (2011). A meta-mining infrastructure to support KD workflow optimization. In *European Conference on Machine Learning and Pronciples and Practice of Knowledge discovery in Databases*, pages 1–10.

[Nguyen et al., 2014] Nguyen, P., Hilario, M., and Kalousis, A. (2014). Using meta-mining to support data mining workflow planning and optimization. *Journal of Artificial Intelligence Research*, 51(1):605–644.

[Nural et al., 2015] Nural, M. V., Cotterell, M. E., Peng, H., Xie, R., Ma, P., and Miller, J. A. (2015). Automated Predictive Big Data Analytics Using Ontology Based Semantics. *International Journal of Big Data (IJBD)*, 2(2):43–56.

[Panov et al., 2014] Panov, P., Soldatova, L., and Džeroski, S. (2014). *Ontology of core data mining entities*, volume 28.

[Quinlan, 1993] Quinlan, J. R. (1993). Combining instance-based and model-based learning. In *Machine Learning. Proceedings of the Tenth International Conference*, pages 236–243. Morgan Kaufmann.

[Reif et al., 2012] Reif, M., Shafait, F., Goldstein, M., Breuel, T., and Dengel, A. (2012). Automatic classifier selection for non-experts. *Pattern Analysis and Applications*, 17(1):83–96.

[Ribeiro et al., 2016] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA. ACM.

[Rice, 1976] Rice, J. R. (1976). The Algorithm Selection Problem. *Advances in Computers*, 15:65–118.

[Russell et al., 2005] Russell, S. J., Norvig, P., Canny, J., and Bratko, I. (2005). *Artificial Intelligence: A Modern Approach*. Pearson Education, Limited.

[Serban et al., 2013] Serban, F., Vanschoren, J. ., Kietz, J.-U., and Bernstein, A. (2013). A Survey of Intelligent Assistants for Data Analysis. *ACM Computing Surveys*, 45(3):1–35.

[Shearer et al., 2008] Shearer, R., Motik, B., and Horrocks, I. (2008). HermiT: A Highly-Efficient OWL Reasoner. In Dolbear, C., Ruttenberg, A., and Sattler, U., editors, *Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions, collocated with the 7th International Semantic Web Conference (ISWC-2008), Karlsruhe, Germany, October 26-27, 2008*, volume 432 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Sirin et al., 2007] Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53.

[Smailbegovic et al., 2005] Smailbegovic, F. S., Gaydadjiev, G. N., and Vassiliadis, S. (2005). Sparse matrix storage format. In *Proceedings of the 16th Annual Workshop on Circuits, Systems and Signal Processing, ProRisc 2005*, pages 445–448. Citeseer.

[Smith et al., 2001] Smith, K. A., Woo, F., Ciesielski, V., and Ibrahim, R. (2001). Modelling the relationship between problem characteristics and data mining algorithm performance using neural networks, in: C. Dagli, et al. In *Eds.), Smart Engineering System Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Data Mining, and Complex Systems*, pages 357–362.

[Smith-Miles, 2008] Smith-Miles, K. A. (2008). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, 41(1):1–25.

[Sun and Pfahringer, 2013] Sun, Q. and Pfahringer, B. (2013). Pairwise meta-rules for better meta-learning-based algorithm ranking. *Machine Learning*, 93(1).

[Tabachnick and Fidell, 2013] Tabachnick, B. G. and Fidell, L. S. (2013). *Using Multivariate Statistics (6th Edition)*. Allyn & Bacon, Inc., Needham Heights, MA, USA.

[Thornton et al., 2013] Thornton, C., Hutter, F., Hoos, H., and Leyton-Brown, K. (2013). Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In *Proc.˜of KDD-2013*, pages 847–855.

[Tibshirani, 1996] Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

[Torgo, 2014] Torgo, L. (2014). An Infra-Structure for Performance Estimation and Experimental Comparison of Predictive Models in R. *CoRR*, abs/1412.0.

[Wang et al., 2013] Wang, Y., Wang, L., Li, Y., He, D., and Liu, T.-Y. (2013). A Theoretical Analysis of NDCG Type Ranking Measures. In Shalev-Shwartz, S. and Steinwart, I., editors, *Proceedings of the 26th Annual Conference on Learning Theory*, volume 30 of *Proceedings of Machine Learning Research*, pages 25–54, Princeton, NJ, USA. PMLR.

[Wolpert and Macready, 1997] Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Appendices

Appendix A

Performance of ScalaTion

We have also evaluated performance of algorithms we have implemented in ScalaTion . As a case study, *OLS*, *ridge* and *lasso* regression algorithms are compared with the *R* implementations. The *glm* package is used for creating *OLS* from *R* and *glmnet* [Friedman et al., 2010] package is used for *ridge* and *lasso* models.

The test setup is as follows. A subset of 59 datasets from the dataset collection (see Section 4.5.1) were selected if they had more rows than columns so that the resulting model would have a positive degrees of freedom. Each dataset is read and prepared for execution by creating an input matrix X and the response vector $\mathbf{y}$ using the ScalaTion framework.

Then, *OLS*, *ridge* and *lasso* models for each dataset is run with *R* and ScalaTion implementations 10 times each. For each run, total training *time* in seconds, cross-validated error measures *RMSE* and *RRSE* are recorded. The results from 10 runs are aggregated by taking mean *time*, *RMSE*, *RRSE* per dataset per implementation.

Ordinary Least Squares Regression (*OLS*)

Figure A.1 shows how *OLS* implementations in ScalaTion and *R* compare. Since the average training time for individual datasets differs significantly due to varying dataset charac-

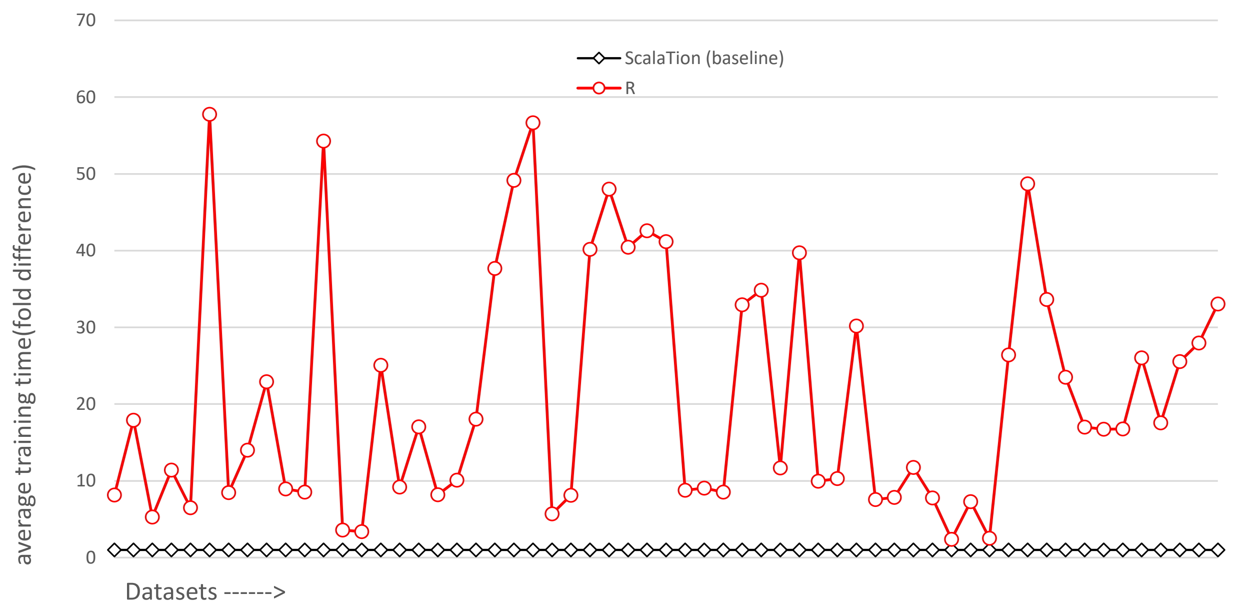


Figure A.1: Average Training Time *OLS* (*lower is better*)

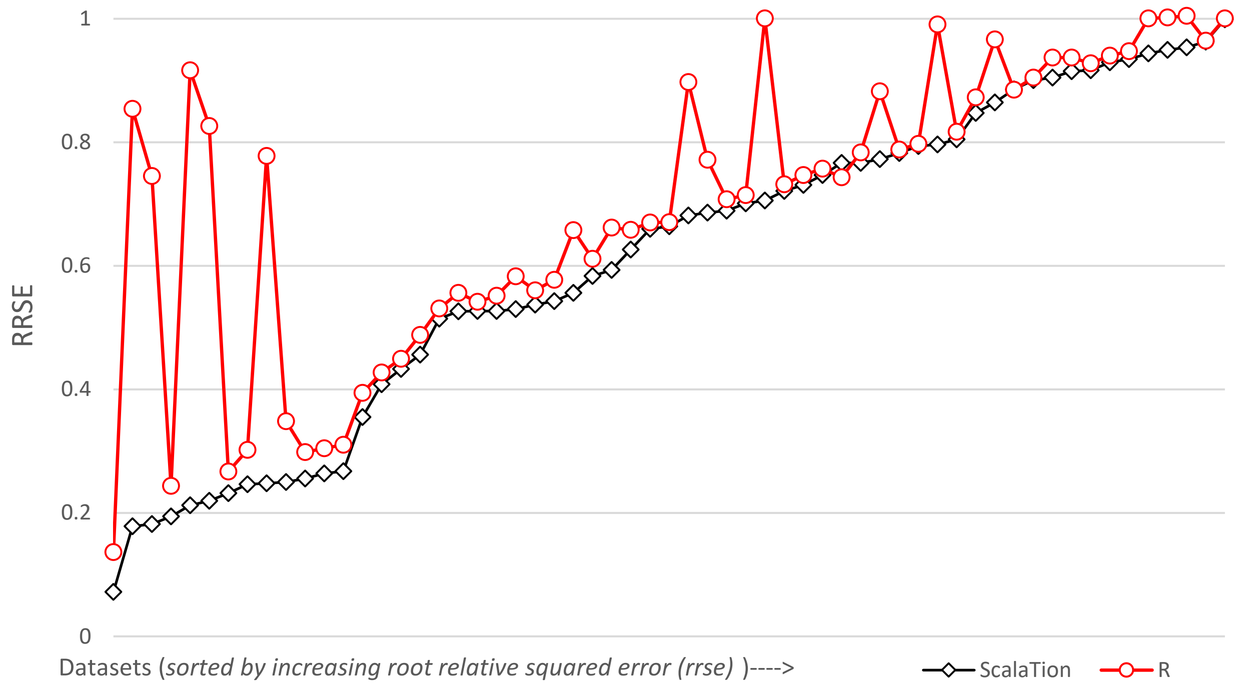


Figure A.2: Average RRSE *ridge* (*lower is better*)

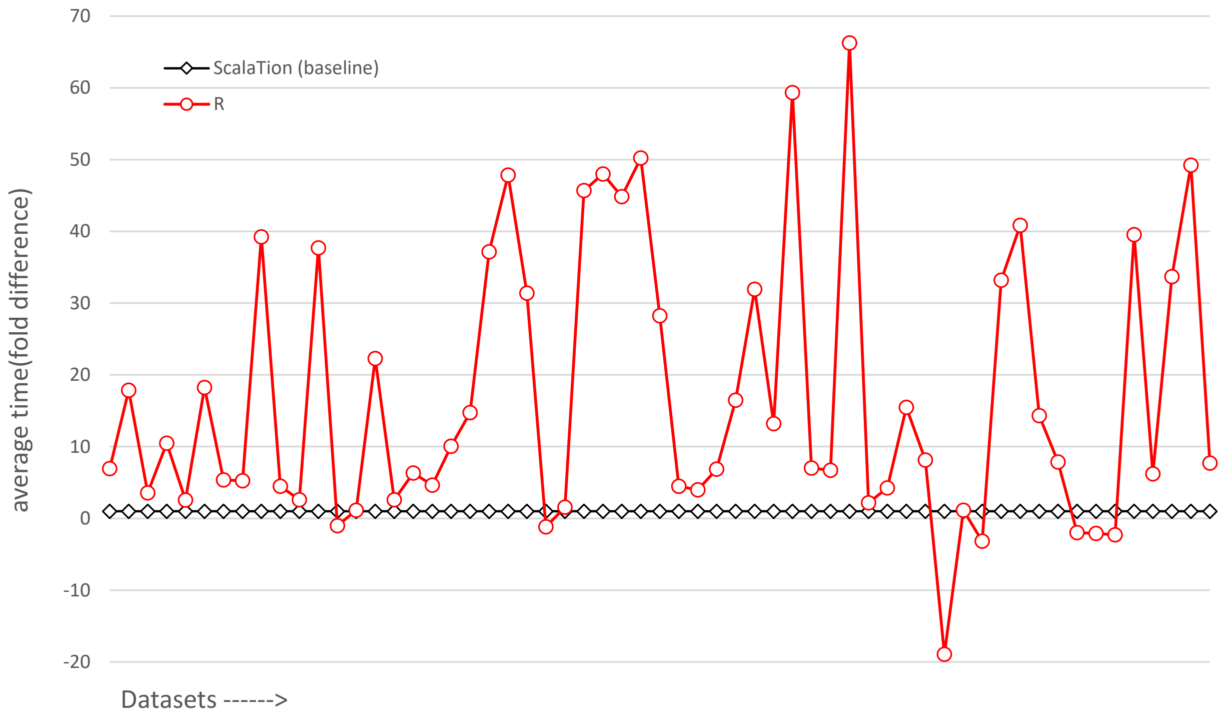


Figure A.3: Average Training Time *ridge* (*lower is better*)

teristics (*e.g.,* size, dimensionality, sparsity and etc.) in the collection, a relative comparison is presented. Each data point represents the relative average training time (over 10 runs) of a dataset in comparison to average training time taken by ScalaTion implementation. The y-axis represents fold difference in which a positive number n depicts R implementation completed training n times slower on average for that dataset. Similarly, a negative number $-n$ means, R was n times faster on average for the given dataset than ScalaTion .

Ridge Regression (*ridge***)**

The parameter vector $\hat{\boldsymbol{\beta}}$ learned from Ridge Regression (*ridge*) must satisfy the following constraint:

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left(\sum_{i=1}^{m} (y_i - \boldsymbol{x_i}\boldsymbol{\beta})^2 + \lambda ||\boldsymbol{\beta}||_2^2 \right)$$

where $\boldsymbol{x_i}$ is the i-th instance of the input matrix as a row vector. In other words, *ridge* attempts to minimize the sum of squared errors just like *OLS* but also places an additional penalty on the sum of squared parameters in order to stabilize them from extreme values which often causes over-fitting.

If the regularization parameter λ is set to 0, then *ridge* becomes *OLS*. However, the search for the optimal value of λ that results in the best performance for a dataset is usually of great interest. In the ScalaTion implementation of *ridge*, the Golden Section Line Search [Kiefer and Monro, 1953] is used to find the optimal value of λ that minimizes the cross-validated sum of squared errors: For each λ value that is searched, a 10-fold cross-validation is performed on the dataset using that particular value of λ. For each fold, the parameter vector $\hat{\boldsymbol{\beta}}$ is fitted from the training set using the Cholesky factorization technique, then predictions are made on the testing test and residuals/errors are computed. Finally, the sum of squared residuals/errors from all 10 folds is the criterion for determining the optimal

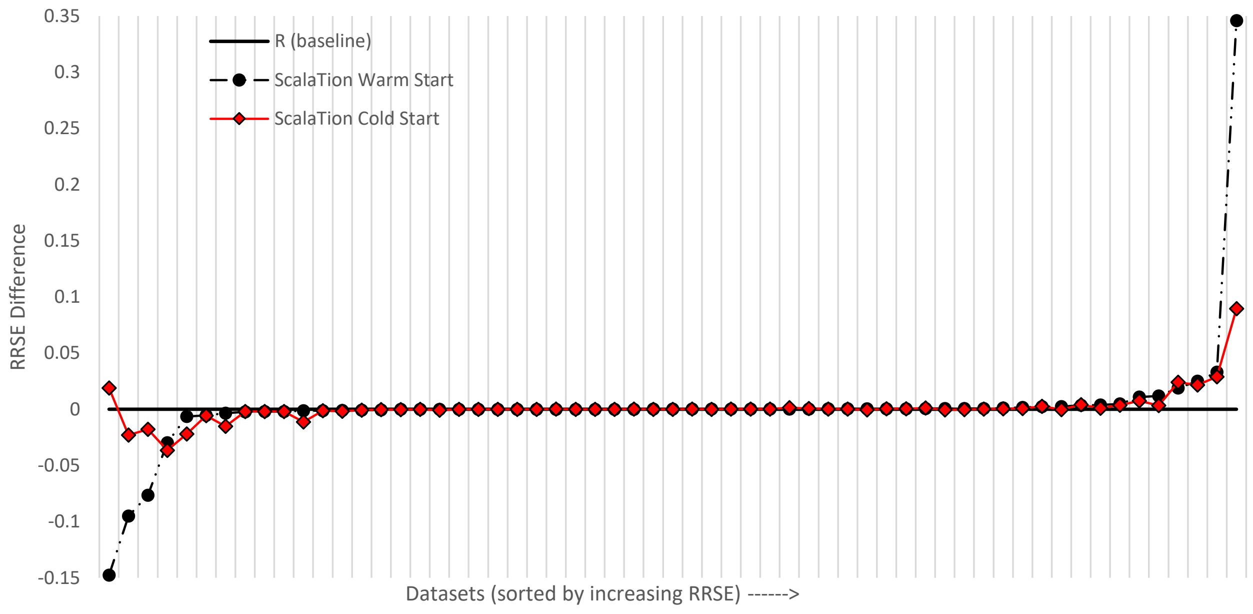


Figure A.4: RRSE Difference *lasso* (*lower is better*)

value of λ.

In the *glmnet* package, Cyclical Coordinate Descent is used for fitting the parameter vector $\hat{\boldsymbol{\beta}}$ and a grid search is used to find the optimal λ value based on the cross-validated mean squared errors.

Figure A.2 shows how ScalaTion and *glmnet* implementations compare in terms of achieving lowest cross-validated error by tuning λ penalty. Each data point in the plot represents the mean $RRSE$ error of 10 runs for a dataset for the given implementation. The datasets in the horizontal axis are sorted by increasing $RRSE$ score achieved by ScalaTion . The plot clearly shows that ScalaTion implementation was consistently able to find a better λ to achieve slightly lower cross-validated error in comparison to *glmnet*. This is most likely due to the choice of different search algorithms for tuning λ in these two implementations. For a number of datasets, ScalaTion performs dramatically better than *glmnet* as it can be clearly seen from the plot.

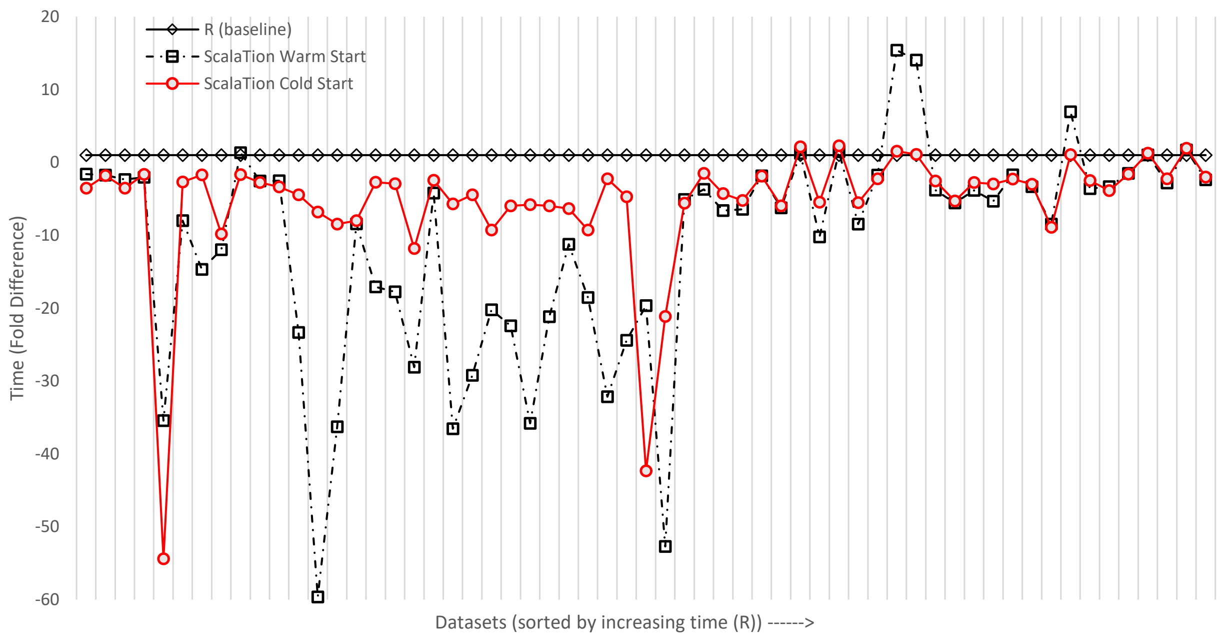


Figure A.5: Average Training Time *lasso* (*lower is better*)

Additionally, the ScalaTion implementation is able to train a better model without a compromise in time taken for training. According to the runtime profiling results during the testing phase, matrix multiplication and transpose are the most time-consuming operations in the ScalaTion implementation. Therefore some of the intermediate results have since been cached to improve efficiency. Figure A.3 provides a comparison for average training times for the same datasets. It is clear that ScalaTion implementation is able train a *ridge* model faster than the *glmnet* implementation with a few exceptions. In one such case, ScalaTion model completed training in 309 seconds on average, compared to 16.3 seconds for *glmnet* model for the $qsar_{191}$ dataset. This dataset is from a family Quantitative structure-activity relationship (*qsar*) datasets where the predictors are binary variables defining the physio-chemical properties of a chemical and the response denotes the biological activity. Having all binary features (*i.e.,* either 0 or 1) combined with high dimensionality and sparsity, *qsar* datasets are a typical example of datasets that are difficult to train for many algorithms. Comparing other metrics for the $qsar_{191}$ dataset revealed that even though *glmnet* requires significantly lower time to train, it is not able to find an optimal λ penalty to yield a reasonable model ($RRSE < 1$). The *ridge* model from *glmnet* has an average $RRSE = 1.0003$, whereas the model from the ScalaTion implementation has $RRSE = 0.7054$.

Lasso Regression (*lasso*)

The parameter vector $\hat{\boldsymbol{\beta}}$ learned from Lasso Regression (*lasso*)[Tibshirani, 1996] must satisfy the following constraint:

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left(\sum_{i=1}^{m} (y_i - \boldsymbol{x_i}\boldsymbol{\beta})^2 + \lambda ||\boldsymbol{\beta}||_1 \right)$$

where $\boldsymbol{x_i}$ is the i-th instance of the input matrix as a row vector. Similar to *ridge*, *lasso* also places a penalty on the size of the parameter vector with a small but an important difference. In contrast to *ridge*, the penalty parameter λ is applied to the L_1 norm instead of the L_2 (Euclidean) norm which allows some parameter coefficients to become exactly zero. Therefore, *lasso* effectively performs variable selection and becomes very suitable for problems where $p \gg n$.

glmnet uses the same method as *ridge* to solve *lasso* (*i.e.,* Cyclical Coordinate Descent). ScalaTion uses Alternating Direction Method of Multipliers (*admm*) [Boyd et al., 2011] method to solve *lasso*. *admm* is a very flexible algorithm that works particularly well for solving problems with two distinct parts. In the context of *lasso*, the minimization of the sum of squared errors and finding the optimal regularization penalty λ can be considered as two sub-problems.

When applied to the *lasso* problem as described in [Boyd et al., 2011], *admm* involves iteratively solving the following equations

$$\beta^{k+1} = (X^TX + \rho I)^{-1}(X^Ty + \rho(z^k - u^k))$$

$$z^{k+1} = S_{\lambda/\rho}(\beta^{k+1} + u^k)$$

$$u^{k+1} = u^k + \beta^{k+1} - z^{k+1}$$

until the convergence criteria is met where X is the input matrix, y is the response vector, β is the coefficient vector(*i.e.,* solution), S is the fast soft thresholding function, λ is the regularization penalty for lasso and ρ is the augmented Lagrangian parameter.

The most expensive computations $(X^TX + \rho I)^{-1}$ and X^Ty are precomputed and cached as the input and the ρ parameter stay constant. Selection of a larger value for the ρ parameter allows faster convergence, however, usually at the expense of algorithm stability. After careful tuning, we have determined to use $\rho = 0.1$ as the default. Using the same methodology used for *ridge* described in A, the optimal λ penalty is determined by using Golden Section Line Search minimizing the cross-validated sum of squared errors.

The original *admm* algorithm uses a "Cold Start" approach in which the z and u vectors are initialized to 0 ($z^0 = u^0 = 0$) at the beginning of an *admm* run for each λ value. For achieving faster convergence, the "Warm Start" approach uses initializing the z and u vectors from the *admm* run for the previous λ value. We have implemented and tested both approaches for comparison.

The figure A.4 shows how ScalaTion compares to *glmnet* in terms of cross-validated error. Each data point in the plot represents the mean *RRSE* error difference of the respective implementation and the R implementation(*baseline*) for a single dataset over 10 runs. It is clear that both "Warm Start" and "Cold Start" implementations are performing on identically (*i.e.,* the *RRSE* error difference is 0) with the baseline R implementation for most datasets. For a number of datasets, ScalaTion implementations perform better by achieving $RRSE$ errors than R. In a few other datasets, the baseline implementation performs better. When averaged across all datasets, all three algorithms where within $< 0.25\%$ error tolerance of each other.

The figure A.5 reveals that ScalaTion is able to train *lasso* much faster on average than *glmnet* albeit a few notable exceptions. Each data point in the plot refers to the fold difference of average time taken to train a dataset compared to the baseline (R). It can also be seen that while the "Warm Start" is dramatically faster than the "Cold Start" for majority of datasets, it fails to converge in a reasonable time period in a few exceptional cases. However, it is important to note that even though "Warm Start" ran 5 to 15 times slower for those exceptional cases (the three outliers can easily be spotted in figure A.5), "Warm Start" was able to achieve much better accuracy in terms of the $RRSE$ error (refer to lower left portion of the plot in A.4). When accounting total training time of all datasets with 10 runs each, the "Cold Start" and "Warm Start" implementations are 25% and 24% faster than the baseline R implementation, respectively.

Appendix B

Datasets

In order to evaluate our system, we have created a dataset collection from a number of different sources and domains. The criterion for selection was to have a numeric response in order to run regression algorithms. We have performed preprocessing on the datasets such as numeric encoding of nominal or binary string variables, missing value imputation etc. only when needed to satisfy input requirements of the regression algorithms. We have removed any dataset from the collection if majority of the algorithms failed to complete due any reason such as being ill-conditioned, not having enough instances to converge, etc. As a result, we have ended up with a total of 114 regression datasets in our collection obtained from various sources listed below. As a summary:

- 43 datasets from UCI Machine Learning Repository [Lichman, 2013]
- 17 datasets from OpenML [Kietz et al., 2014]
- 16 datasets from publicly available packages in R[1]
- 12 datasets from Luis Torgo Regression datasets collection[2]

[1]See `https://vincentarelbundock.github.io/Rdatasets/datasets.html` for an unofficial compilation.

[2]`http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html`

- 9 datasets from Bilkent University Function Approximation Library[3]
- 9 datasets from NCI-60 Cell Line panel: Similar to [Lee et al., 2011], we have used gene expression data obtained from Affymetrix HG-U133A and B chips normalized using the GCRMA method as predictors of proteins with top 9 most variance obtained from Reverse-phase protein lysate arrays (RPLA).
- 8 datasets from other sources[4]

[3]`http://funapp.cs.bilkent.edu.tr/DataSets/`

[4]For more details on the datasets see `github.com/scalation/data`

Table B.1: Dataset Collection

Dataset	$m^{\S}$	$k^{\dagger}$	**Description**
auto-mpg	392	9	Auto-MPG Dataset from UCI
airfoil	1503	6	Airfoil Self Noise Dataset from UCI(NASA)
concrete_compressive	1030	10	Concrete Compressive Strength Dataset from UCI
ccpp	9568	5	Combined Cycle Power Plant Dataset from UCI
concrete_slump_1	103	11	Concrete Slump Dataset from UCI (target: SLUMP)
concrete_slump_2	103	11	Concrete Slump Dataset from UCI (target: FLOW)
concrete_slump_3	103	11	Concrete Slump Dataset from UCI (target: Compressive Strength)
nist_gauss_1	250	2	NIST Gauss1 dataset. The data are two well-separated Gaussians on a decaying exponential baseline plus normally distributed zero-mean noise with variance = 6.25.
prostate	97	9	R Prostate Cancer dataset
kin8nm	8192	9	kin8nm dataset from OpenML (https://www.openml.org/d/189)
computer_activity_1	8192	22	Torgo Computer Activity Dataset
computer_activity_2	8192	13	Torgo Computer Activity Dataset - Small version
wisconsin_breast	194	33	Wisconsin Breast Cancer Dataset
auto_price	159	16	Torgo Auto Price Dataset
gym_crowdedness	62184	11	Kaggle Campus Gym Crowdedness Dataset

§: *number of instances* †: *number of features*

Table B.1: Dataset Collection

Dataset	$m^{\S}$	$k^{\dagger}$	**Description**
forest_fire	517	13	UCI Forest Fire Dataset
housing	506	14	Boston Housing Dataset
istanbul_stock	536	10	UCI Istanbul Stock Exchange Dataset
tecator_moisture	240	101	OPENML Tecator Dataset(target: Moisture)
tecator_fat	240	101	OPENML Tecator Dataset(target: Fat)
tecator_protein	240	101	OPENML Tecator Dataset(target: Protein)
bike_sharing_total_hour	17379	17	UCI Bike Sharing Dataset, Hourly Data Total Count
bike_sharing_total_day	731	16	UCI Bike Sharing Dataset, Daily Data Total Count
bng_breast	116640	10	OPENML BNG Breast Tumor Dataset
visualizing_soil	8641	5	OPENML Visualizing Soil Dataset
bank8fm	8192	9	OPENML Customer Bank Selection Dataset
abalone	4177	9	Torgo Abalone Dataset
electricity_prices	37682	17	OPENML ICON Electricity Challenge Dataset
casp	45730	10	UCI Protein Tertiary Structure DataSet
appliance_energy	19735	29	UCI Appliance Energy DataSet
crime_norm	1993	101	UCI Communities Crime(Normalized-ViolentPerPop) DataSet
parkinson_1	5875	19	UCI Parkinson Telemonitoring Dataset(target: total)
parkinson_2	5875	19	UCI Parkinson Telemonitoring Dataset(target: motor)

§: *number of instances* †: *number of features*

Table B.1: Dataset Collection

Dataset	$m^{\S}$	$k^{\dagger}$	**Description**
servo	167	5	UCI Servo Dataset
student_1	395	30	UCI Student Performance Dataset(target: mat)
student_2	649	30	UCI Student Performance Dataset(target: por)
yacht	308	7	UCI Yacht Hydodynamics Dataset
fb_metric_1	496	16	UCI Facebook Metric Dataset(target: total)
fb_metric_2	496	16	UCI Facebook Metric Dataset(target: like)
fb_metric_3	496	16	UCI Facebook Metric Dataset(target: comment)
fb_metric_4	496	16	UCI Facebook Metric Dataset(target: share)
cars	1447	14	Applied Predictive Modeling Cars Dataset(all)
chick_weight	578	3	R Caret Package Chick Weight Dataset
life_cycle_savings	50	5	R Caret Package Life Cycle Savings Dataset
hi	22272	12	R Health Insurance Housewives Dataset
body_fat	252	18	Bilkent Body Fat Dataset
fried	40768	11	Bilkent Fried Dataset
plastic	1650	3	Bilkent Plastic Dataset
quake	2178	4	Bilkent Quake Dataset
weather_1	1609	10	Bilkent Weather Ankara Dataset
weather_2	1461	10	Bilkent Weather Izmir Dataset
treasury	1049	16	Bilkent Treasury Dataset
pwlinear	177147	11	OPENML PWLinear Dataset
puma32h	8192	33	Torgo Puma32H Dataset
puma8nh	8192	9	Torgo Puma8NH Dataset
2dplanes	40768	11	Torgo 2dplanes Dataset

§: *number of instances* †: *number of features*

Table B.1: Dataset Collection

Dataset	m[§]	k[†]	**Description**
pol	15000	27	OPENML Pole Telecom Dataset
solar	1066	10	UCI Solar Flare Dataset
qsar_47555	1158	52	OPENML QSAR Dataset(47555)
qsar_31274	1189	132	OPENML QSAR Dataset(31274)
air	999249	19	RITA Airline on-time Performance Dataset (1987 only)
buzz_toms	28179	97	UCI Social Media Buzz Dataset - Toms Hardware)
buzz_twitter	583250	78	UCI Social Media Buzz Dataset - Twitter
qsar_47749	6003	611	OPENML QSAR Dataset(47749)
olympic2000	66	12	Olympic2000 Dataset from "Analyzing Categorical Data"
qsar_191	4442	1024	OPENML QSAR Dataset(191)
qsar_33511	6003	420	OPENML QSAR Dataset(33511)
corn_m5spec_moisture	80	701	NIR of Corn Samples for Standardization Benchmarking Dataset (Moisture)
corn_m5spec_oil	80	701	NIR of Corn Samples for Standardization Benchmarking Dataset (Oil)
corn_m5spec_protein	80	701	NIR of Corn Samples for Standardization Benchmarking Dataset (Protein)
corn_m5spec_starch	80	701	NIR of Corn Samples for Standardization Benchmarking Dataset (Starch)
qsar_12789	309	1025	OPENML QSAR Dataset(12789)

§: *number of instances* †: *number of features*

Table B.1: Dataset Collection

Dataset	$m^{\S}$	$k^{\dagger}$	**Description**
energy_efficiency_1	768	10	UCI Energy Efficiency Dataset(Heating Load)
energy_efficiency_2	768	10	UCI Energy Efficiency Dataset(Cooling Load)
cbm_1	11934	15	UCI CBM Dataset(Compressor)
cbm_2	11934	15	UCI CBM Dataset(Turbine)
triazines	186	59	Bilkent Triazines Dataset
cars_kbb	804	18	R Caret Package KBB Price Cars Dataset
chem	176	58	Applied Predictive Modeling Chemical Manufacturing Dataset
crime_unnorm_autoTheft	2211	103	UCI Communities Crime DataSet (target: autoTheft)
crime_unnorm_burgl	2211	103	UCI Communities Crime DataSet (target: burgl)
crime_unnorm_larc	2211	103	UCI Communities Crime DataSet (target: larc)
crime_unnorm_nonViol	2117	103	UCI Communities Crime DataSet (target: nonViol)
crime_unnorm_violent	1993	103	UCI Communities Crime DataSet (target: violent)
crime_unnorm_total	1901	103	UCI Communities Crime DataSet (target: total)
crime_unnorm_arsons	2123	103	UCI Communities Crime DataSet (target: arsons)
crime_unnorm_assault	2201	103	UCI Communities Crime DataSet (target: assault)
crime_unnorm_rapes	2006	103	UCI Communities Crime DataSet (target: rapes)
crime_unnorm_murd	2214	103	UCI Communities Crime DataSet (target: murd)

§: *number of instances* †: *number of features*

Table B.1: Dataset Collection

Dataset	$m^§$	$k^†$	**Description**
crime_unnorm_robbb	2213	103	UCI Communities Crime DataSet (target: robb)
ailerons	13750	41	Ailerons Dataset
elevators	16599	19	Elevators Dataset
transcoding	68784	20	UCI Video Transcoding Dataset
sol_1	1267	229	Applied Predictive Modeling Solubility Dataset
sol_2	632	229	Applied Predictive Modeling Solubility Dataset(trans)
blood_brain	208	128	Applied Predictive Modeling Blood Brain Barrier Dataset
aquatic_tox_1	322	24	R QSARData Package Aquatic Toxicity Dataset(lcalc)
aquatic_tox_3	322	66	R QSARData Package Aquatic Toxicity Dataset(moe3d)
aquatic_tox_4	319	49	R QSARData Package Aquatic Toxicity Dataset(qprop)
aquatic_tox_2	322	221	R QSARData Package Aquatic Toxicity Dataset(moe2d)
cox2	462	206	R Caret Package Cox2 Dataset
melting_point	4401	204	R QSARData Package Melting Point Dataset
aloi	108000	129	OPENML Aloi Dataset
nci_60_90th_1	59	3490	NCI-60 Dataset(target: KRT18)
nci_60_90th_2	59	3490	NCI-60 Dataset(target: KRT19)
nci_60_90th_3	59	3490	NCI-60 Dataset(target: KRT7)

§: *number of instances* †: *number of features*

Table B.1: Dataset Collection

Dataset	$m^{§}$	$k^{\dagger}$	**Description**
nci_60_90th_4	59	3490	NCI-60 Dataset(target: TP53_26_GBL00064)
nci_60_90th_5	59	3490	NCI-60 Dataset(target: VASP)
nci_60_90th_6	59	3490	NCI-60 Dataset(target: MSN_4)
nci_60_90th_7	59	3490	NCI-60 Dataset(target: CDKN2A)
nci_60_90th_8	59	3490	NCI-60 Dataset(target: KRT8)
nci_60_90th_9	59	3490	NCI-60 Dataset(target: TP53_10_24342)
qsar_36276	6003	40	OPENML QSAR Dataset(36726)
qsar_47652	1731	84	OPENML QSAR Dataset(47652)

§: *number of instances* †: *number of features*