Investigation of Machine Learning Algorithms for the Classification of Atmospheric Measurements

David Hamblin¹, Dali Wang*¹, Gao Chen² and Ying Bai³
¹Christopher Newport University, Newport News, VA, USA (*Contact Author)
²Science Directorate, NASA Langley Research Center, Hampton, VA, USA
³Johnson C. Smith University, Charlotte, NC, USA

Abstract - The goal of this research work is to identify and develop computer algorithms capable of classifying atmospheric measurements, derived from the raw data files gathered by NASA through its airborne data collection campaigns. Four machine learning algorithms are investigated – Naive Bayes, Classification and Regression Tree (CART), Boosted CART, and Bagged CART – using the data gathered during a period of 10 years. Enhancement and modification have been made to these algorithms in order to address needs of our application. Accordingly, software is developed for each of the four algorithms. We have tested this development using the real data measurements from NASA’s data repository, and the results demonstrate the effectiveness and robustness of the algorithms.

Keywords: machine learning, classification, decision tree

1 Background

Beginning in the 1980s, NASA introduced tropospheric chemistry studies to gather information about the contents of the air. The data collection compiled, called missions or campaigns, are supported by the federal government with the goal of collecting data for scientific studies on climate and air quality issues. For the last decade, data has been compiled into a single format, ICARTT (International Consortium for Atmospheric Research on Transport and Transformation, thus the ICARTT format), which is designed to allow for sharing across the airborne community – with other agencies, universities, or any other group working with the data. The ICARTT file format is broken into two primary sections: the metadata or header section and the measurement data section. A typical ICARTT file contains, at the minimum, about 30 lines of metadata that describe the data gathered, the who/where/what of the data gathering process, along with additional comments supplied by the principal investigator that is pertinent to the data [1].

Over the past few years, NASA has made it a priority to open up its data archive to the general public in order to promote their usage in scientific studies. As part of a data management effort, NASA aims at establishing a centralized airborne data archive for airborne science data, and developing a set of web-based tools for data access and processing. The overall development is under the umbrella project entitled Toolsets for Airborne Data (TAD). An essential component of the system is to ingest the past measurement data files into a centralized database. The aircraft data ingestion system takes information from ICARTT files and translates it into entries in the data archive database. We have been involved in the three major components of this data ingestion process: 1. Cleaning and reformatting raw data in order to remove all inconsistencies. This part of work has been completed and reported in [2]. 2. Identifying measurement instrument in the metadata section, and classifying each instrument into one of the 300+ standardized names. This part of work has been completed and reported in [3]. 3. Identifying measurement variables and classifying them into one of many common names. This component is reported in this paper.

2 Problem Statement

The decentralized structure of the data collection campaigns – each campaign is funded separately and managed by its own group of scientists – introduces variations in the implementation of the ICARTT standard among missions. One such variation is the naming convention of measurement variables. Although they can be found in different missions, conducted by separate investigators, much of the data associated with a certain variable may be equivalent to others, save for the distinguishing metadata. In order to ingest the raw ICARTT data into NASA’s central data repository for public access, each variable name has to be converted to a standardized name (called common name), which is defined systematically. There are over 300 such common names; each uniquely defines a chemical or physical property along with further specifying information. The standardization of the names (called common naming) is essential for making the central data repository truly usable. Without that, users would not be able to search and compare measurements through a common interface across different files and data collections.
At an early stage of the project, the matching process between a ICARTT variable and its common name was done manually. However, there are only a handful of domain experts at NASA who can perform such a task. This, as one could imagine, takes a considerable amount of time. Given the great wealth of data accumulated since the 1980s, it becomes clear that automating the common naming process is the only path going forward. The goal of this research is to automate the process.

The common name collections that have been created manually in the early data ingestion process have become a valuable asset for this research work. We can use the small collections to serve as our training data set; therefore, their existence makes it possible for us to investigate machine learning algorithms as the solution to classifying variables.

### 3 Overview of Candidate Solutions

In its most common form, the common naming process is to choose a specific label (i.e., a common name) for a measurement variable from a list of predefined labels, given the metadata associated with this variable. It belongs to the category of label/text classification problems.

Machine learning algorithms have been proven to be very effective in many classification and identification problems [4][5][6]. The most commonly used techniques in problems similar to common name matching include: Artificial Neural Networks (ANN) – especially the perceptron-based network, Support Vector Machines (SVM), instance-based learning – especially the k-Nearest Neighbor (k-NN) algorithm, logic based algorithms – especially Decision Trees (DT) and the rule-based classifiers associated with it, and finally statistical learning algorithms such as Naive Bayes (NB) and the Bayesian Network associated with it.

Generally, SVMs and ANN tend to perform much better when dealing with multi-dimensional and continuous-valued features [4]. However, this problem only involves single dimensional and discrete features. K-NN is very sensitive to irrelevant features and intolerant to noise. These make it unattractive for this problem, as not all features are relevant to every common name, and the information in general is very noisy due to errors and inconsistencies in ICARTT files. Also, not all features can be found with each common name, but only those that are pertinent.

The statistical-based Bayesian approach is also investigated in this research. The major advantage of the Naive Bayes classifier is its short computational time, small storage space requirement, and robustness to noise and missing values. In addition, the algorithm is very transparent, as the output is easily grasped by users through a probabilistic explanation [4][7]. A detriment to Naive Bayes, however, is that it may become less accurate when compared with some sophisticated algorithms, and less tolerant to redundant features. Given its advantages, despite the problems, Naive Bayes is chosen as the algorithm for the common naming process in order to provide a baseline performance measurement for decision-tree based algorithms.

Random forest is an ensemble algorithm that trains many decision trees, based on random decisions in the selection of feature splits to classify variables [8][9]. Once the forest has been trained, classification is chosen based upon the mode class for the entire forest, given the features. However, unlike typical bagging approaches where splits are based upon an impurity function, random forests do not work with the atmospheric dataset due to the large number of features and their possible values [10].

Typical decision trees, such as C4.5, are unreliable for classifying the airborne data. Preliminary testing with one of these algorithms led to low accuracy due to their preference for binary and discrete features. Instead, a separate type of decision tree is considered. Classification and Regression Trees (CART) have been used in the statistics community for over thirty years [11], and are well-proven methods for creating predictive models in the form of a “tree”, where branches are features and the end of a branch - the leaf - is the chosen class. Classification trees in particular have provided good results for datasets with categorical classes and features [12], while regression trees deal with continuous values unlike the atmospheric dataset. Classification trees are constructed through the partitioning of features based upon providing good splits, utilizing some metric to determine the “best” split. As they have the capability to work with datasets filled with categorical information and seeking a categorical outcome, classification trees align with the common naming process. Favorable preliminary results for implementation of a classification tree model lead to further investigation into utilizing this algorithm. In this work, the Classification and Regression Tree method along with several strategies for improving it - bagging and boosting - are tuned and tested against the atmospheric dataset.

### 4 Algorithms

#### 4.1 Construction of Feature Sets

The primary motivation behind the creation of common names is due to the lack of consistency in variable naming information between similar data; therefore, using the variable name alone to classify variables would not work. Before classification, a feature set is built for each variable, based upon certain attributes within the metadata. Feature sets are important for describing a variable through many different attributes that it possesses. There are two steps involved in building a feature set for each variable:

1. Using the fingerprint algorithm program OpenRefine (developed for and originally known as Google Refine) on variable names so that
variables with inconsistencies/typos are renamed to be the same.

2. Parsing the metadata using a separate configuration file with regular expression matching and more specific features matching for variables, in order to identify the feature values needed by the algorithms. The features obtained through this process include: variable name, unit, description, principal investigator’s name, instrument, segments of the variable’s name, and numbers appearing in the variable’s name.

4.2 CART

CART is split up into two models, as its name suggests: classification trees and regression trees [12]. For categorical data and outcome, like with common names labeled to various multi-level features, classification trees are selected. The classification tree from CART utilizes a generalization of the binomial variance called Gini impurity, defined below, to determine the “best” split for the construction of a tree at each branch.

\[
I_G(f) = 1 - \sum_{i=1}^{n} f_i^2 
\]  

where \( n \) is the number of classes (common names) in a set, and \( f_i \) is the proportion of items in that set labeled with the class \( i \).

The construction of a CART model begins by growing an overly large tree and beginning to prune it - reduce the size - to minimize the estimate of classification. The growing and pruning process is performed via 10-fold cross validation and taking the average set of splits as the final tree. In order to identify the appropriate classification for a feature set in the CART approach, the class (common name) with the lowest misclassification cost is chosen.

Classification trees are constructed through an exhaustive search approach, which are further refined by specific implementations. The following list details the process of building each tree, by recursively partitioning each feature into the best possible split, according to Gini impurity.

1. Start at the root node
2. For each feature \( X \), find the sample set \( S \) that minimizes the sum of the node impurities in the two child nodes and choose the split \( \{X* \in S*, \text{ where } S* \subseteq S\} \) that gives the minimum overall \( X \) and \( S \).
3. If a stopping criterion is reached, exit. Otherwise, apply step 2 to each child node in turn.

A stopping criterion would be the value of the impurity function crossing a given threshold, a leaf is created for a class instead of a new feature split.

4.3 Boosting

The boosting approach to machine learning works by building many sequential classifiers for the data, primarily used to reduce bias. Its purpose is to take a set of classifiers that may be considered weak, with many misclassifications, and combine their results into a single classifier that has the lowest misclassification (error) rate. For each sequential tree, the weighting of misclassified features is modified.

Originally, the boosted CART algorithm was developed for the two-class classification problem. However, problems including the common naming one possess many different choices for class. In order to classify these types of problems, multiple two-class algorithms could be combined together. For our purpose, we investigated the multi-class expansion of boosted CART known as AdaBoost [13] and more recent SAMME (stagewise additive modeling using a multi-class exponential loss function) [14]. The two algorithms are very similar, although we choose SAMME for the common naming problem. The algorithm is outlined below, where \( K \) is the number of common names.

1. Initialize the observation weights \( \omega_i = 1/n, \ i = 1, 2, ..., n \).
2. For \( m = 1 \) to \( M \):
   (1) Fit a classifier \( T^{(m)}(x) \) to the training data using the weights \( \omega_i \).
   (2) Compute \( \text{err}^{(m)} = \sum_{i=1}^{n} \omega_i \Pi \left( c_i \neq T^{(m)}(x_i) \right) / \sum_{i=1}^{n} \omega_i \)
   (3) Compute \( \alpha^{(m)} = \log \frac{1 - \text{err}^{(m)}}{\text{err}^{(m)}} + \log(K - 1) \)
   (4) Set \( w_i \leftarrow w_i \cdot \exp \left( \alpha^{(m)} \cdot \Pi \left( c_i \neq T^{(m)}(x_i) \right) \right) \)
   (5) Re-normalize \( \omega_i \)
3. Output \( C(x) = \arg \max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \Pi(T^{(m)}(x) = k) \)

4.4 Bagging

Bootstrap aggregating, or bagging, is a concept that was developed for the purpose of improving the accuracy of
machine learning algorithms [9], in a similar yet different way to boosting. It is particularly effective with decision trees in reducing variance - determined by Gini impurity - as opposed to bias, and avoiding overfitting by sampling a dataset and creating individual classifiers for each subset. While boosting relies on sequential classifiers building off of the previous, bagging takes an aggregation of multiple classifiers used on different datasets.

Given the total training set $D$ of feature sets extracted from extended maps, a number of new training subsets $D_i$ are generated through random sampling with replacement of a fixed size less than or equal to the total size of the training set. Each of these new training subsets are considered “bootstrap” samples. Each training subset is trained with a classification tree. Once trained, classification of feature sets to common names is decided by an average of the classification labels from the trees. By averaging the results of each tree instead of using a single tree, variance is reduced.

The bagging method is highlighted in figure 1. Data from the entire training set are first “sampled” into training subsets, and each is used to generate their own CART classifier. The aggregation of multiple classifiers determines the output model used for classification. New data for classification, through new atmospheric campaigns, is pumped into the combined system to determine the output common names [15].

5 Implementation and Results

All algorithms in this research are implemented either in Python or R. The data pre-processing, construction of feature sets, and post processing functions are implemented in Python. The Naive Bayes algorithm is implemented in Python; CART and the boosting and bagging algorithms are mostly implemented in R.

ICARTT data from the NASA online data repository were used in this research. The data are from 10 missions dated between 2004 and 2015. The 10 missions used in this study include: ARCPAC, ARCTAS, CalNex, DC3, DISCOVERAQ California, DISCOVERAQ Maryland, INTEX-A, INTEX-B, NEAQS-ITCT2004, and TexasAQS.

The results for each algorithm are acquired through the use of six-fold cross validation. For each category, six subsets are created and each subset is tested against the remaining five, producing an accuracy value for each. The average of these values is taken as the total algorithm accuracy, and these values are displayed in Table 1.

Neither bagging nor boosting is able to execute for the aerosol dataset because of limitations in the algorithms. Due to the implementation in R and nature of the algorithms, including the large number of features and samples, combined with hundreds of decision trees, the training models built exceed the memory limitations of 64-bit R. We expect this will be resolved over time when new R releases become available. However, an increase in category size with newer missions may also prevent other large datasets from training. While performing well for the other datasets with less samples and predictors, the inability to produce results for aerosol show the limitation of two enhanced algorithms.

The decision-tree based CART approach performs well consistently across all categories. When it underperforms Naive Bayes, the difference in performance is insignificant. However, when Naive Bayes’ performance becomes unacceptable - such as with J variables - CART outperforms it by a large margin. The enhanced CART algorithms – boosting and bagging – provide the highest accuracy in most cases in comparison with the CART model they are based upon; however, their failure in processing larger data sets (i.e. aerosol) limits their usability in the near term.
6 Conclusions

The goal of this research work is to find a reliable way to classify measurement variables in NASA airborne measurement data files. Given the history and reality of airborne data collection practice, the accuracy and robustness of a solution are considered as the top priorities. The research work investigates several machine learning algorithms – some of them were not elaborated in this paper due to their applicability. We have developed software that produces consistent results across all data sets. The four algorithms we chose – Naive Bayes, CART, and two ensemble algorithms (boosting and bagging) – have performed robustly throughout our test. Additionally, the transparent nature of the algorithms allows for better results through modification. The performance measured by evaluation results greatly exceeds the expectation set forth at the start of the project. The success of this project makes it possible to automate the common naming process, which greatly improves the efficiency of the centralized airborne data archiving process in the future.

7 Acknowledgments

This work is sponsored by NASA Langley Research Center, especially its Science Directorate and Atmospheric Science Data Center (ASDC). We would like to thank members of ASDC – Amanda Benson, Emily Northup and Aubrey Beach – for their guidance and support.

8 References


[15] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques 3rd Ed. (University of Illinois at Urbana-Champaign and Simon Fraser University, 2011).