

**AUTOMATIC DETECTION
OF
RWIS SENSOR MALFUNCTIONS
(PHASE I)**

FINAL REPORT

**Northland Advanced Transportation Systems
Research Laboratories
Project A: Fiscal Year 2005**

**Carolyn J. Crouch
Donald B. Crouch
Richard M. Maclin
Aditya Polumetla**

**Department of Computer Science
University of Minnesota Duluth
1114 Kirby Drive
Duluth, Minnesota 55812**

cs@d.umn.edu

AUTOMATIC DETECTION OF RWIS SENSOR MALFUNCTIONS

(PHASE I)

Objective

The overall goal of this project (Phases I & II) is to develop computerized procedures that detect Road Weather Information System (RWIS) sensor malfunctions. In this first phase of the research we apply two classes of machine learning techniques to data generated by RWIS sensors in order to predict sensor malfunctions and thereby improve accuracy in forecasting temperature, precipitation, and other weather-related data.

Project Overview

The Minnesota Department of Transportation (MN/DOT) uses environmental sensor station data, that is, Road Weather Information System data, to ascertain optimal deployment and use of winter maintenance crews, equipment, and deicing materials. [1, 2] These sensors collect data on weather conditions every 10 to 20 minutes at or near the road surface. The data is broadcast to a central monitoring station and used to forecast winter road conditions. Since forecasts are only as good as the sensor data on which they are based, it is critical that RWIS sensors be properly calibrated and functioning correctly. At present, routine maintenance and calibration are performed annually at each site. Yet missing from the RWIS system are automated procedures to monitor sensor data continuously for availability and accuracy and to detect sensor malfunctions. This needed RWIS component is the essence of this project.

Our research focuses on developing procedures to identify malfunctioning RWIS sensors by analyzing the sensor data. Repairing and/or recalibrating these sensors as malfunctions are detected or at least identifying those sensors that are misreporting weather conditions can assure confidence in the overall accuracy of the data. In this section, we provide an overview of this pilot project and introduce the environment in which we are conducting this research.

Minnesota has 91 RWIS meteorological measurement stations positioned alongside major highways to collect local pavement and atmospheric data. The State is laid out in a fixed grid format and RWIS stations are located within each grid. The stations themselves are positioned some 60 km apart. Each RWIS station utilizes various sensing devices, which are placed both below the highway surface and on towers above the roadway. Road sensors are used to determine if the road surface is wet, dry, frosted, snow covered, or iced. Tower sensors record such weather conditions as air temperature, relative humidity, wind speed, wind direction, and precipitation. Some of the stations use video cameras and visibility sensors to gather data about related conditions such as rain, fog and snow.

The sensor data, that is, the variables that will be used in determining sensor malfunctions, are dependent on the particular type of sensor being evaluated. Each sensor data item consists of a site identifier, sensor identification, date, time (Greenwich Mean Time), and the raw sensor values. The primary variables encountered when working with RWIS data include the following:

- **Air Temperature** - recorded in units of 0.01 degrees Celsius. Values range from -999 to 9999.
- **Visibility** - horizontal visibility. Values range from 0 to 99999 and are recorded in one tenth of a meter. All sites can report visibility up to 1.1 miles. Some detect and record visibility up to 10 miles.

- **Surface Temperature** - road surface temperature in units of 0.01 degrees Celsius. Values range from -999 to 9999.
- **Precipitation Type** - the form of current precipitation represented by one of the following codes:

Code	Type
0	No precipitation detected
1	Precipitation detected, not identified
2	Rain
3	Snow
41	Moderate Snow
42	Heavy Snow

- **Precipitation Intensity** - the intensity of current precipitation represented by one of the following codes:

Code	Intensity
0	None
1	Light
2	Slight
3	Moderate
4	Heavy
5	Other
6	Unknown

- **Precipitation Rate** - rate of rain/snow fall measured in mm/hour with values ranging from 000 to 999.
- **Precipitation Accumulation** - the total amount of rain/snow reported in millimeters. Values are available for 1, 3, 6, 12 and 24 hour periods.
- **Wind Speed** - measured in units of 0.1 meters/second. Values range from 0 to 9999.
- **Wind Direction** - measured as an angle, where 0° indicates north. Values range from 0° to 360°.
- **Air Pressure** - measured in units of 0.1 millibar. Values range from 0 to 9999.
- **Dew Point** - recorded in units of 0.1° Celsius. Values range from -999 to 9999.

Not all sensors (nor all RWIS sites) are being studied during this project. Our research focuses on a subset of sensors that includes precipitation, air temperature, and visibility. These sensors were identified by Mn/DOT as being of particular interest. However, data from other sensors will be used to determine if these particular sensors are operating properly.

A set of 13 RWIS stations were chosen for this phase of the project. The selected sites are not subject to microclimatic conditions and are contained in the same or adjacent grids located in the Northern part of the State. Each RWIS station is located near one or more regional airports so that the data generated by their Automated Weather/Surface Observing System (AWOS/ASOS) sensors can be compared with the data generated by the RWIS site. Airport AWOS/ASOS sites are located no more than 30 miles from each of the selected RWIS sites (a distance ensuring that each RWIS site is linked to at least one regional airport yet keeping to a minimum the number of airports lying within the 30-mile radius of any given RWIS site). The airports associated with each RWIS site are noted in Table 1. Figure 1 illustrates the location of each site and its adjacent airports. As may be noted, the 13

RWIS Site Id	Airport Code	Distance (miles)
14	KFFM	15.6
19	KELO	9.71
20	KAIT	16.3
	KBRD	21.8
25	KROX	12.2
27	KINL	2.82
35	KBRD	17.6
35	KLXL	17.8
49	KPKD	16.5
56	KROX	29.2
60	KCKN	13.9
62	KDTL	11.2
67	KINL	27.3
	KORB	19.9
68	KCKN	29.2
	KFSE	16.2
	KTVF	22.1
78	KBDE	20.4
	KROX	29.7

Table 1. RWIS sites and associated AWOS/ASOS sites.

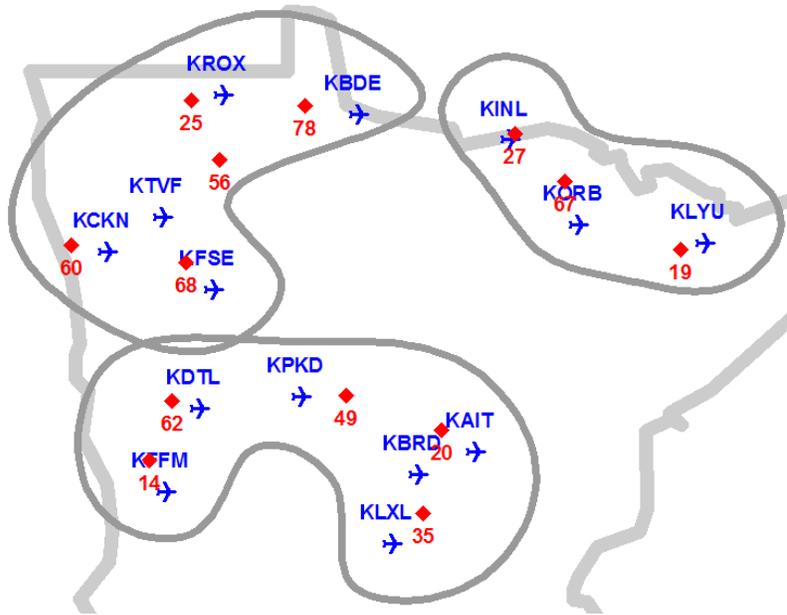


Figure 1. Location of RWIS/AWOS sites.

sites cluster naturally into three groups; this clustering was used to advantage as we refined our analysis of the sensor data.

The data generated by the airport sensors represent information from either an AWOS or ASOS site. [3] Data values are generally reported every hour and in units of measurement that often differ from those of the RWIS sites. Following are the variables of interest for the AWOS/ASOS data.

- **Air temperature** - measured in whole (integer) degrees in Fahrenheit. Range is from -140° F to 140° F.
- **Visibility** - prevailing horizontal visibility, coded as one of 19 values ranging from 3/16 mile to 10+ miles.
- **Weather Code** - prevailing weather condition occurring at the time of the observation. There are over 80 different possible codes available. [We note that 12 specific sites seem to report only from a subset of 15 codes.]
- **Dew point temperature** - the temperature at which dew forms. This variable is reported in whole (integer) degrees Fahrenheit. It ranges from -140F to 140F.
- **Air pressure** - the atmospheric pressure, when reduced to sea level, expressed in units of 0.1 millibars.
- **Wind speed** and **Wind direction** - encoded in the same variable **Wind**. Speed is in knots and direction is in compass degrees of the source of prevailing wind. When variable winds are detected (when no single direction is present for more than half the time), a value of 0 is recorded. The range of wind speed is between 0 (calm) and 999 knots.

The RWIS and AWOS/ASOS data was downloaded to our local server. For the initial 13 RWIS and corresponding AWOS/ASOS sites, we have historical sensor data from March 2001 to October 2005. We have converted the data to common units of measurement to support direct comparison of various system data.

We have also acquired and downloaded approximately 4 years of maintenance data for the RWIS sensors within the State – a total of 624 records. Since there is little standardization to the recording of maintenance data, we cast this data into a form that can be tied to the data of those sensors that have been recalibrated. Each maintenance record identifies a sensor as being in one of four states:

- Fully Operational – 338 entries
- Not Commissioned (offline on purpose) – 4 entries
- Problem Reported – 55 entries
- Problem Verified – 227 entries

An *explanation* field is also included in each record that may be used to better classify the data entry. The five most frequently occurring states included in this field for the historical data are listed below:

- Status to Fully Operational – 145 entries
- Log Appended – 144 entries
- No Data Received – 137 entries
- Incomplete Data – 33 entries
- Incorrect/Suspicious Data – 21 entries

One of the methodologies we explored to identify defective sensors and sensor calibration errors may be referred to as *neighborhood associations*. This technique uses AWOS/ASOS sensor data to predict the values of the corresponding RWIS sensors for sites that are nearest neighbors (as defined by Table 1). Disparity among results may imply RWIS sensor malfunction. This method can only be employed if the data values produced by the RWIS and AWOS/ASOS sensors are actually comparable; earlier research raised the question whether they are. [4] For example, unlike AWOS/ASOS sensors, RWIS temperature and dew point sensors are not aspirated.

We conducted a series of calculations to determine if data from the RWIS and nearby AWOS/ASOS sites are actually compatible. In particular, we computed the correlation between each pair of RWIS and AWOS/ASOS sites, a total of 19 data sets. The variables examined were air temperature, visibility, and precipitation. For illustration, we limit our discussion here to air temperature. Similar results were obtained for the other two variables.

Table 2 contains the correlation values for air temperature. The average correlation was 0.923, a significant degree of correlation. This suggests that although some calibration may be necessary to account for differences in RWIS and AWOS/ASOS sensors, the different systems allow for direct comparisons. Thus, we concluded that we should be able to build reliable prediction models using neighborhood associations for RWIS sites based on AWOS/ASOS data and data from nearby RWIS sites.

RWIS Site Id	Airport Code	Correlation
14	KFFM	0.968
19	KELO	0.908
20	KAIT	0.948
20	KBRD	0.917
25	KROX	0.963
27	KINL	0.912
35	KBRD	0.914
35	KLXL	0.915
49	KPKD	0.943
56	KROX	0.971
60	KCKN	0.948
62	KDTL	0.919
67	KINL	0.945
67	KORB	0.865
68	KCKN	0.892
68	KFSE	0.888
68	KTVF	0.900
78	KBDE	0.904
78	KROX	0.918

Table 2. Correlation of air temperature for RWIS and AWOS/ASIS pairs.

In Phase I of this project, we developed methods for recognizing malfunctions in RWIS sensors by building models using machine learning methods that employ data from nearby sensors in order to predict likely values of the sensor we are interested in. For example, consider the situation shown in Figure 2. In this case we are interested in predicting values such as the temperature of RWIS sensor 67. Our approach involves taking data from surrounding sensors (in this case RWIS sensors 19 and 27) and AWOS sensors INL, ORB and LYU) and using the values of these sensors to make predictions about sensor 67. A sensor that begins to deviate noticeably from values inferred from nearby sensors serves as an indication that the sensor has begun to fail. Figure 3 exemplifies the result of such a prediction model. In this figure the actual sensor values are shown along with the model's predicted values. The fault is easy to detect due to the abrupt differences in the predicted values.

We developed models to predict values of the temperature, precipitation and visibility sensors using a variety of different methods, including machine learning algorithms and basic statistical methods such as linear regression, naïve Bayes, decision trees, neural networks, and Bayes networks. To make the data more uniform and to remove some of the variance due to time of year and day, we normalized the data to obtain for each data point a normalized measure of how the value of each sensor related to the expected value of that sensor. For example, to report the temperature value from other sensors we first normalized with respect to the historic average values for that time frame. A temperature reading for sensor 19 of 25 degrees Fahrenheit would be considered out of the norm for August but perhaps reasonable for March at noon. After normalizing the data, we tested our methods by dividing the data repeatedly into training and testing groups using 10-fold cross validation, repeated 10 times. In 10-fold cross validation the data is randomly permuted and then divided into 10 equal sized groups. Each group is then in turn used as test data while the other 9 groups are used to create a model (in this way every data point is used as a test point exactly once). The results for the ten groups are summed, and the overall result for the test data is reported.

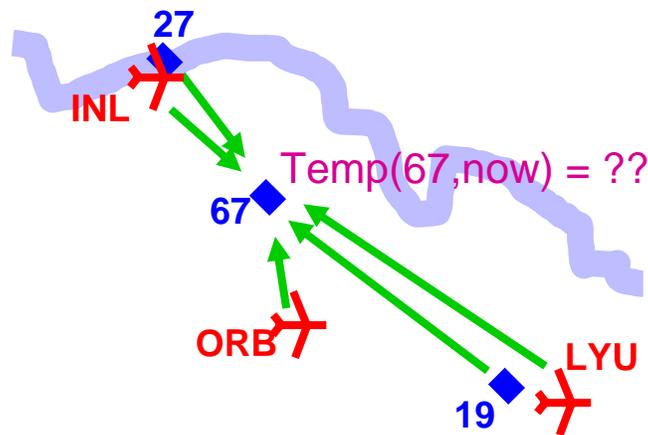


Figure 2: Predictions for a sensor are made using neighboring sensor data.

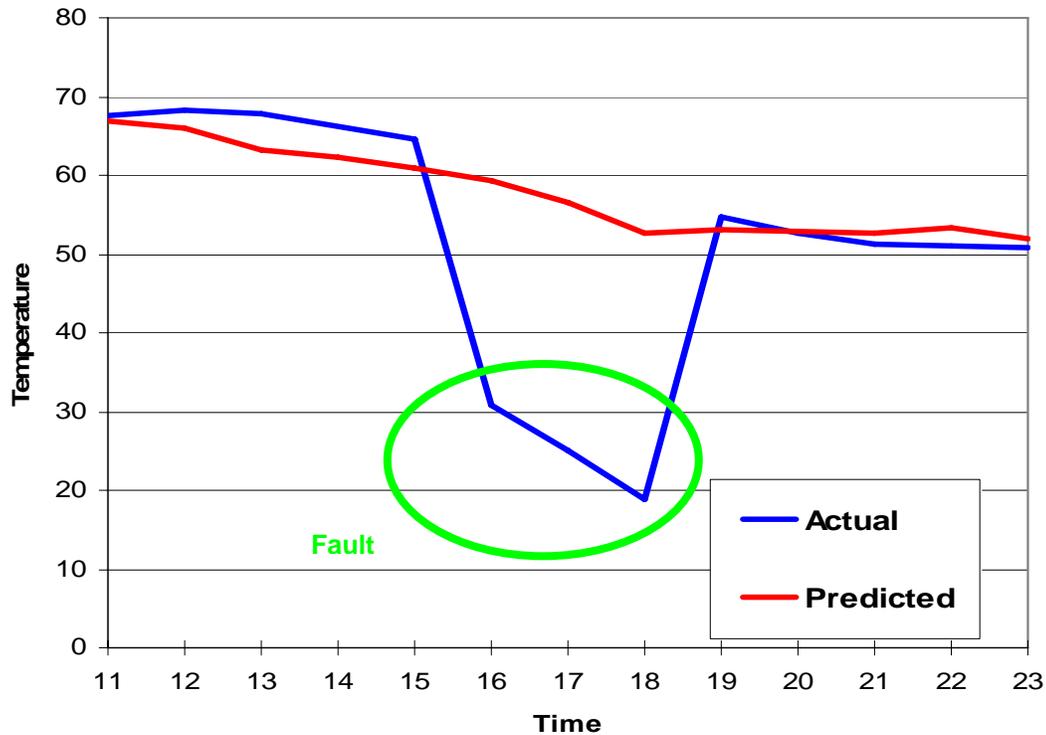


Figure 3: Sample of prediction of sensor temperature values over a 12-hour period

Testing was conducted using several years of RWIS and AWOS data. For each set of tests each RWIS site was in turn used as the site to be predicted. All data from the other sites (relating to temperature, precipitation, etc.) that had been collected immediately preceding the time to be predicted were input to the model. Results indicate that temperature is relatively easy to predict. While there is much greater variability in the precipitation and visibility sensors, we still found significant prediction capability even for these sensors.

Machine Learning Models

In this section we will describe the machine learning models used in the first phase of this project. *Machine learning* (ML) is a subfield of artificial intelligence whose concern is the development, understanding and evaluation of algorithms and techniques to allow a computer to learn. Since many ML algorithms use analysis of data for building models, statistics plays a major role in this field.

A process or task that a computer is assigned to perform can be termed the knowledge or task domain (or just the domain). The information that is generated by or obtained from the domain constitutes the knowledge base. The knowledge base can be represented in various ways using Boolean, numerical, and discrete values, relational literals, and their combinations. The knowledge base is generally represented in the form of input-output pairs, where the information represented by the input is given by the domain and the result generated by the domain is the output. The information from the knowledge base can be used to depict the data generation process (i.e., output classification for a given input) of the domain. Knowledge of the data generation process does not define the internals of the working of the domain, but can be used to classify new inputs accordingly.

As the knowledge base grows in size or becomes more complex, manually inferring new relations about the data generation process (the domain) increases in difficulty for humans. ML algorithms try to learn from the domain and the knowledge base to build computational models that represent the domain in an accurate and efficient way. The model captures the data generation process of the domain; algorithms use the model to match previously unobserved examples from the domain.

Models assume different forms based on the ML algorithm itself, including, for example, decision lists, inference networks, concept hierarchies, state transition networks, and search-control rules. Although the concepts and working of various ML algorithms may differ from one another, their common goal is to learn from the domain they represent.

In order to build a model of the domain, the ML algorithms must have a dataset, which constitutes the knowledge base. The dataset is a collection of instances from the domain. Each instance consists of a set of attributes which describe the properties of that example from the domain. An attribute takes in a range of values based on its attribute type, which can be discrete or continuous. Discrete (or nominal) attributes take on distinct values (e.g., *weather = sunny*) whereas continuous (or numeric) attributes take on numeric values (e.g., *temperature = 20°F*). Each instance consists of a set of input attributes and an output attribute. The input attributes are the information given to the learning algorithm, and the output attribute contains the feedback of the activity on that information. The value of the output attribute is assumed to depend on the values of the input attributes. The attribute along with the value assigned to it define a feature, which makes an instance a feature vector. The model built by an algorithm can be seen as a function that maps the input attributes in the instance to values of the output attribute.

ML algorithms are used to learn a model that describes the data generation process based on the dataset given to it. The data given to the algorithm for building the model is called the training data, as the computer is being trained to learn from this data, and the model built is the result of the learning process. The resulting model can be used to predict or classify previously unseen examples. New examples used to evaluate the model are called a test set. The accuracy of a model can be estimated from the difference between the predicted and actual value of the target attribute in the test set.

ML algorithms can be used to predict weather conditions. Using the weather data collected from a location for a certain period of time, we can build a model to predict variables such as temperature at a given time based on the input to the model. As weather conditions tend to follow patterns and are not totally random, we can use current meteorological readings along with those taken a few hours earlier at a location and also readings taken from nearby locations to predict a condition such as the temperature at that location. Thus, the data instances that will be used to build the model may contain current and previous readings from a set of nearby locations as input attributes. The sensor value that is to be predicted at a given RWIS site is the target attribute. The type and number of conditions that are included in an instance depend on the sensor value that we are trying to predict and on the properties of the ML algorithm used.

Consider the following example. Suppose we want to predict the current temperature at a site *C* (see Figure 4). To do this, we use eight input attributes: temperature values for the previous two hours at *C* and two nearby locations *A* and *B* together with the current temperature values at *A* and *B*. The output attribute is the predicted current temperature at *C*. Let $temp_{<site><hour>}$ denote temperature taken at time $<hour>$ at location $<site>$, then the data instance for this case will take the form:

$$temp_{A_{t-2}}, temp_{A_{t-1}}, temp_{A_t}, temp_{B_{t-2}}, temp_{B_{t-1}}, temp_{B_t}, temp_{C_{t-2}}, temp_{C_{t-1}}, temp_{C_t}$$

with the last attribute, $temp_{C_t}$, being the output attribute.

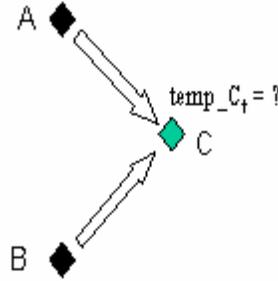


Figure 4: Using data from nearby sites to predict temperature for location C

ML algorithms can be broadly classified into two groups: classification and regression algorithms. These algorithms take input in any form, discrete or continuous, although some require all the input to be discrete. Classification algorithms use a training set to identify a set of rules that classify a given input into one of a set of discrete values; output is always in the form of a discrete value. Regression algorithms, on the other hand, develop a model based on equations or mathematical operations on the input attribute values and produce as output a continuous value. We used both classification and regression algorithms in this study; in particular, we used three classification algorithms: J48 decision trees, naïve Bayes, and Bayesian networks, and six regression algorithms: linear regression, least median squares, M5P, multilayer perceptron, RBF network, and the conjunctive rule algorithm. We will briefly describe these nine algorithms.

J48 Decision Tree Algorithm

J48 is a decision tree learner based on algorithm C4.5 [6]; C4.5 is an update of the ID3 algorithm [7]. A decision tree classifies a given instance by passing it through the tree starting at the root node and moving downward through the tree until a leaf node is reached. The value at that leaf node gives the predicted output for the instance. At each non-terminal node an attribute is tested and the appropriate branch taken based on the value of the attribute of the instance. A possible decision tree that could be used to predict the current temperature for site C (Figure 4) is illustrated in Figure 5.

The ID3 algorithm builds a decision tree based on a given set of training instances. It employs a greedy top-down approach to construct the tree, starting with the creation of the root node. For each node the attribute that best classifies all the training instances that have reached that node is selected as that node's test attribute; only those attributes that were not used for classification at other nodes above it in the tree are considered. To select the best attribute at a node, the *information gain* for each attribute is calculated; the attribute with the highest information gain is selected. Information gain is defined as the reduction in *entropy* caused by splitting the instances based on values taken by the attribute. The information gain for an attribute A at a node is calculated as follows:

$$InformationGain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \left(\frac{|S_v|}{|S|} Entropy(S_v) \right),$$

where S is the set of instances at that node, $|S|$ is its cardinality, and S_v is the subset of S for which attribute A has value v .

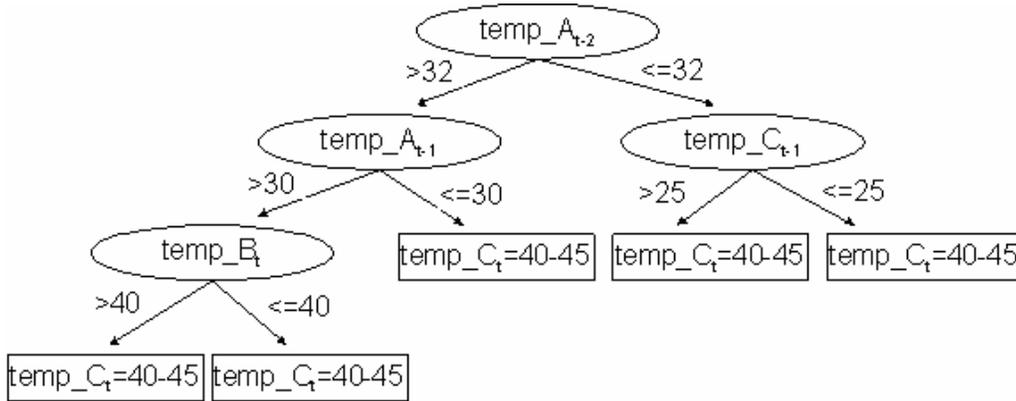


Figure 5: A decision tree to predict temperature at site C using input from sites A and B

Entropy of the set S is calculated as

$$Entropy(S) = \sum_{i=1}^{numclasses} -p_i \log_2 p_i,$$

where p_i is the proportion of instances in S that have the i^{th} class value as output attribute.

A new branch is added below the node for each value taken by the test attribute. The training instances that have the test attribute value associated with the branch taken are passed down the branch and subsequently used for the creation of further nodes. If this subset of training instances has the same output class value, then a terminal node is generated, and the output attribute is assigned that class value. In the case where no instances are passed down a branch, then a terminal node is created that assigns the most common class value in the training instances as its output attribute. This process of generating nodes is continued until all the instances are correctly classified, all the attributes have been used, or it's not possible to divide the examples.

In order to use the ID3 algorithm for predicting RWIS sensor values, the algorithm must be modified (1) to deal with continuous valued attributes, (2) to allow for instances that have missing (unknown) attribute values, and (3) to prevent over fitting the data. When a discrete valued attribute is selected at a node, the number of branches formed is equal to the number of possible values taken by the attribute. In the case of a continuous valued attribute, two branches are formed based on a threshold value that best dichotomizes the instances. For example, suppose that in building the tree in Figure 5, the attribute at the root node, $temp_{A_{t-2}}$, has a value of 32. This threshold is selected as the value of the attribute that maximizes the information gain of the given training instances. It should be noted that Fayyad [9] further extended the ID3 algorithm to allow a continuous-valued attribute to be split into more than two intervals.

If an instance has no value for an attribute, the missing value can be replaced by the most common value for that attribute among the training instances that reach the node where this attribute is tested. The decision tree algorithm C4.5 incorporates this modification of ID3. In particular, the probability for each possible value taken by the attribute with a missing value is calculated, based on the number

of times it is seen in the training instances at a node. The probability values are then used for to calculate the information gain at the node.

In the ID3 algorithm, over fitting may occur if the training set is relatively small. In this case, the resulting tree correctly classifies the training instances but fails when applied to the entire distribution of data; this over fitting occurs because the algorithm focuses on the spurious correlation in the data. To avoid over fitting, algorithm C4.5 uses the so-called rule post-pruning technique. In rule post-pruning, the tree is first built and then converted into a set of rules. For example, in Figure 5, the rule generated for the leftmost path of the tree is

IF ($temp_A_{t-2} > 32$ AND $temp_A_t > 30$ AND $temp_B_t > 40$) THEN $temp_C_t = 40-45$.

Naïve Bayes Algorithm

Naïve Bayes [9, 10] is a simple probabilistic classifier based on Bayes rule. The naive Bayes algorithm builds a probabilistic model by learning the conditional probabilities of each input attribute given a possible value taken by the output attribute. This model is then used to predict an output value for a given set of inputs by applying Bayes rule¹ on the conditional probability of seeing a possible output value when the attribute values in the given instance are seen together.

The naive Bayes algorithm uses a set of training examples to classify a new instance using the Bayesian approach. For an instance, the Bayes rule is applied to find the probability of observing each output class given the input attributes; the class that has the highest probability is assigned to the instance. The probability values used are obtained from the counts of attribute values in the training set.

Consider the example given in Figure 4. For a given instance with two input attributes $temp_A_t$ and $temp_B_t$, with values a and b respectively, the value v_{MAP} assigned by the naive Bayes algorithm to the output attribute $temp_C_t$ is the one that has the highest probability across all possible values taken by the output attribute. The probability of the output attribute having value v_j when the given input attribute values are seen together is given by $P(v_j/a,b)$. Applying Bayes theorem, $P(v_j/a,b)$ can be calculated using the following expression:

$$P(v_j | a,b) = \frac{P(a,b | v_j)P(v_j)}{P(a,b)} = P(a,b | v_j)P(v_j),$$

where $P(v_j)$ is the probability of observing v_j as the output value,

If the number of input attributes (a, b, c, d, \dots) is large, then there may not be enough data to estimate the probability $P(a, b, c, d, \dots | v_j)$. The naïve Bayes algorithm solves this problem by assuming conditional independence for all the input attributes yielding v_j for the output. The probability value $P(a, b | v_j)$ can then be simplified as

$$P(a,b | v_j) = P(a | v_j)P(b | v_j),$$

where $P(a | v_j)$ is the probability of observing the value a for the attribute $temp_A_t$ when output value is v_j . Thus the probability of an output value v_j being assigned for the given input attributes is

$$P(v_j | a,b) = P(v_j)P(a | v_j)P(b | v_j).$$

¹ Bayes rule states that $P(A/B) = [P(B/A) P(A)] / P(B)$, where $P(A/B)$ is defined as the probability of observing A given that B occurs. Bayes' rule allows one to find $P(A/B)$ when the individual probabilities of A and B are known, and the probability of observing B given A is also known.

Learning in the Naive Bayes algorithm involves finding the probabilities of $P(v_j)$ and $P(a_i/v_j)$ for all possible values taken by the input and output attributes based on the training set provided. $P(v_j)$ is obtained from the ratio of the number of times the value v_j occurs for the output attribute to the total number of instances in the training set. For an attribute at position i with value a_i , the probability $P(a_i/v_j)$ is obtained from the number of times a_i occurs in the training set when the output value is v_j .

The naive Bayes algorithm requires all attributes in the instance to be discrete. Continuous valued attributes have to be discretized before they can be used. Missing values for an attribute are not allowed, as they can lead to difficulties when calculating the probability values for that attribute. A typical solution is to replace missing values by a default value for that attribute.

Bayesian Belief Networks (Bayes Nets)

The naive Bayes algorithm assumes that the values of the input attributes are conditionally independent for a given value of the output attribute. However, there may be cases where this assumption results in inappropriate predictions. Bayesian belief networks or Bayes nets applies conditional independence to a subset of inputs rather than all of them, therein avoiding a global assumption of conditional independence among the inputs while still maintaining some amount of conditional independence,

A Bayesian belief network [11, 12] is a directed acyclic graphical network model that yields the joint probability distribution for a set of attributes. Each attribute in the instance is represented as a node in the network. A directed link from node X to node Y indicates that X is a parent of Y and reflects that Y is dependent on X . Y may have multiple parent nodes. In the Bayes network, an attribute is conditionally independent of any attributes that are not its children. Only the parents are considered for calculating the joint probability, as only the direct parents of a node influence the conditional probabilities at this node. Using conditional independence between nodes, the joint probability for a set of attribute values y_1, y_2, \dots, y_n represented by the nodes Y_1, Y_2, \dots, Y_n is given by

$$P(y_1, \dots, y_n) = \prod_{i=1}^n P(y_i | Parents(Y_i)),$$

where $Parents(Y_i)$ are the immediate parents of node Y_i . The probability values for a node are represented in a tabular form called a Conditional Probability Table (CPT). In the case of nodes with no parents, the CPT gives the distribution of the attribute at that node. It should be noted that a Bayesian network requires that both input and output attributes be discrete.

Consider again the example illustrated in Figure 4. A simple Bayesian network for predicting temperature at site C, using only a few of the input instances of sites A and B, is shown in Figure 6. Each node in the tree is associated with a CPT; for example, the CPT for the node $temp_{A_{t-2}}$ will contain the probability of each value taken by it when all possible values for $temp_{A_{t-1}}$ and $temp_{C_t}$ (i.e., its parents) are seen together. For a given instance, the Bayesian network can be used to determine the probability distribution of the target class by multiplying all the individual probabilities of values of the individual nodes. The class value with the highest probability is selected. The probability of a class value taken by the output attribute $temp_{C_t}$ for the given input attributes, using parental information of nodes from the Bayesian network in Figure 6, is

$$\begin{aligned} P(temp_{C_t} | temp_{A_{t-1}}, temp_{A_{t-2}}, temp_{B_t}, temp_{B_{t-2}}, temp_{C_{t-2}}) = \\ P(temp_{C_t}) * P(temp_{A_{t-1}} | temp_{C_t}) * P(temp_{A_{t-2}} | temp_{A_{t-1}}, temp_{C_t}) * P(temp_{B_t} | temp_{C_t}) * \\ P(temp_{B_{t-2}} | temp_{A_{t-1}}, temp_{C_t}) * P(temp_{C_{t-2}} | temp_{A_{t-2}}, temp_{C_t}). \end{aligned}$$

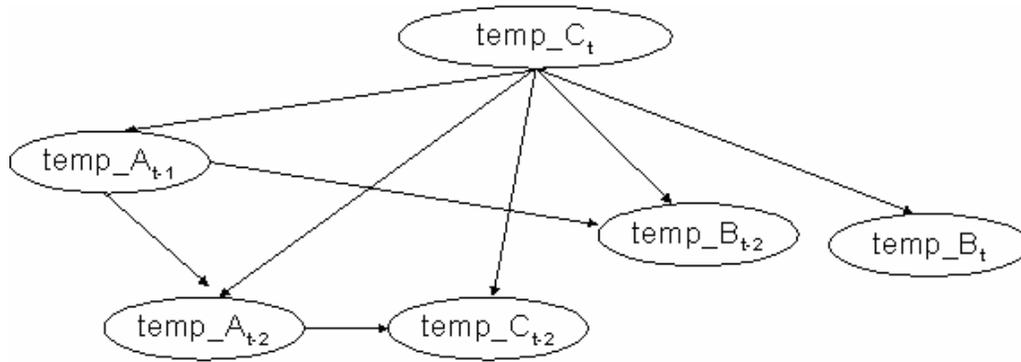


Figure 6: A Bayesian network to predict temperature at site C using input from sites A and B

Learning in Bayes' Nets involves finding the best performing network structure for the training set and calculating CPTs. To build the network structure, we start by assigning each attribute a node. Learning the network connections involves moving through the set of possible connections and finding the accuracy of the network for the given training set. The accuracy of the network can be determined by using a scoring criterion such as the Akaike's information criterion [13], the minimum description criterion [14], or the cross-validation criterion. Allen and Greiner [15] present a brief description of these scoring criteria along with their empirical comparisons.

The K2 algorithm [16] can be used to learn the Bayesian network structure. K2 puts the given nodes in an order and then processes one node at a time. It adds an edge to this node from previously added nodes only when the network accuracy is increased after this addition. When no further connections can be added to the current node that increase the accuracy, the algorithm moves to another node. This process continues until all nodes have been processed. When all variables present in the network are seen in the training data, the probability values in the CPTs can be generated by counting the required terms. In the case of training data with missing variables, the gradient ascent training method [17] can be used to learn values for the CPTs.

Linear Regression

The Linear Regression algorithm of WEKA [5] performs standard least squares regression to identify linear relations in the training data. This algorithm gives the best results when there is some linear dependency among the data. It requires the input attributes and target class to be numeric and it does not allow missing attributes values. The algorithm calculates a regression equation to predict the output (x) for a set of input attributes $a_1, a_2 \dots, a_k$. The equation to calculate the output is expressed in the form of a linear combination of input attributes with each attribute associated with its respective weight w_0, w_1, \dots, w_k , where w_i is the weight of a_i and a_0 is always taken as the constant 1. An equation takes the form

$$x = w_0 + w_1 a_1 + \dots + w_k a_k .$$

For the temperature example in Figure 4, the regression equation would take the form

$$temp_C_t = w_0 + w_{A_{t-2}} temp_A_{t-2} + w_{A_{t-1}} temp_A_{t-1} + w_{A_t} temp_A_t + w_{B_{t-2}} temp_B_{t-2} + w_{B_{t-1}} temp_B_{t-1} + w_{B_t} temp_B_t + w_{C_{t-2}} temp_C_{t-2} + w_{C_{t-1}} temp_C_{t-1} ,$$

where $temp_C_t$ is the value assigned to the output attribute.

The accuracy of predicting the output by this algorithm (that is, the error) can be measured as the absolute difference between the actual output observed and the predicted output as obtained from the regression equation. Weights must be chosen in such a way as to minimize the error. To obtain better accuracy, higher weights must be assigned to those attributes that influence the result the most.

A set of training instances is used to update the weights. Initially, weights can be assigned random values or a constant (such as 0). For the first instance in the training data the predicted output is obtained as

$$w_0 + w_1 a_1^{(1)} + \dots + w_k a_k^{(1)} = \sum_{j=0}^k w_j a_j^{(1)},$$

where the superscript for attributes gives the instance position in the training data. After the predicted outputs for all instances are obtained, the weights are reassigned so as to minimize the sum of squared differences between the actual and predicted outcome. Thus the aim of the weight update process is to minimize

$$\sum_{i=1}^n \left(x^{(i)} - \sum_{j=0}^k w_j a_j^{(i)} \right)^2,$$

the sum of the squared differences between the observed output for the i^{th} training instance ($x^{(i)}$) and the predicted outcome for that training instance obtained from the linear regression equation.

Least Median Squares

The WEKA Least Median Squares of Regression algorithm [18] is a linear regression method that minimizes the median of the squares of the differences from the regression line. The algorithm requires input and output attributes to be continuous, and it does not allow missing attribute values. Standard linear regression is applied to the input attributes to get the predict the output. The predicted output x is obtained as

$$w_0 + w_1 a_1^{(1)} + \dots + w_k a_k^{(1)} = \sum_{j=0}^k w_j a_j^{(1)},$$

where a_j are input attributes and w_j , the weights associated with them.

Using the training data, weights are updated in such a way that they minimize the median of the squares of the difference between the actual output and the predicted outcome using the regression equation. Weights can be initially set to random values or assigned a scalar value. The aim of the weight update process is to determine new weights to minimize

$$\text{median}_i \left(x^{(i)} - \sum_{j=0}^k w_j a_j^{(i)} \right)^2,$$

where i ranges from 1 to the number of instances in the training data that is being used and $x^{(i)}$ is the actual output for the training instance i .

M5 Prime Algorithm

The M5 Prime algorithm (M5P) [19] is a regression-based decision tree algorithm, based on the M5 algorithm by Quinlan [20]. For a given instance the tree is traversed from top to bottom until a leaf node is reached. At each node in the tree a decision is made to follow a particular branch based on a test condition on the attribute associated with that node. Each leaf node has a linear regression model associated with it of the form

$$w_0 + w_1 a_1 + \dots + w_k a_k,$$

where the input attributes are represented by a_1, a_2, \dots, a_k and the weights w_0, w_1, \dots, w_k are calculated using standard regression.

An M5 tree is built using a divide-and-conquer method. At each interior node, a test condition is chosen that splits the instances into subsets based on the test outcome. A test is based on an attribute's values. In M5 the test that maximizes the error reduction is used:

$$\Delta error = stdev(S) - \sum_i \left(\frac{|S_i|}{|S|} stdev(S_i) \right),$$

where S is the set of instance passed to the node, $stdev(S)$ is its standard deviation, and S_i is the subset of S resulting from splitting at the node with the i^{th} outcome for the test. This process of creating new nodes is repeated until there are too few instances to proceed further, or the variation in the output values in the instances that reach the node is small.

Once the tree has been built, a linear model is constructed at each node. The linear model is a regression equation. The attributes used in the equation are those that are tested or are used in linear models in the subtrees owned by the node. The attributes that are ancestors of the node are not used in the equation, since their effect on predicting the output has already been captured. The linear model is further simplified by eliminating attributes whose removal leads to a reduction in the error. The error is defined as the absolute difference between the output value predicted by the model and the actual output value seen for a given instance.

An M5 tree can be complex. To simplify the tree without losing its basic functionality, it may be pruned. Starting from the bottom of the tree, the error is calculated for the linear model at each node. If the error is less than the model subtree owned by the node, then the subtree for this node is pruned.

In the case of missing values in training instances, M5P changes the expected error reduction equation to

$$\Delta error = \frac{m}{|S|} * \beta(i) * \left[stdev(S) - \sum_i \left(\frac{|S_i|}{|S|} stdev(S_i) \right) \right],$$

where m is the number of instances without missing values for that attribute, S is the set of instances at the node, $\beta(i)$ is the factor multiplied in case of discrete attributes, and j assumes values L and R with S_L and S_R being the sets obtained from splitting at that attribute.

MultiLayer Perceptron

A MultiLayer Perceptron (MLP) [21] is a neural network that is trained using back propagation. MLPs consist of multiple layers of computational units with connections from lower units to units in a subsequent layer. The basic structure of MLP consists of an input layer, one or more hidden layers, and one output layer. Output from units in the hidden layer is used only in the network and is not seen outside the network. An MLP that can be used to predict temperature for the example in Figure 4 is shown below in Figure 7. The output from a unit is used as input to units in the subsequent layer. The connection between units in subsequent layers has an associated weight.

The hidden units and output units are based on sigmoid units (Figure 8). A sigmoid unit calculates a linear combination of its input and then applies the sigmoid function to the result. The sigmoid function for net input x is

$$sigmoid(x) = \frac{1}{(1 + e^{-x})}$$

$Sigmoid(x)$ is a continuous function of its input (x) and ranges from 0 to 1.

An MLP establishes values for its weights using the back propagation algorithm applied to a set of training instances. [22] The back propagation algorithm takes a set of training instances for the learning process. For the given feed forward network, the weights are initialized to small random numbers. Each training instance is passed through the network and the output from each unit is computed. The target output is compared with the output computed by the network to calculate the error; this error value is fed back through the network. The error value is used to adjust the weights of its connections. To adjust the weights, back propagation uses gradient descent to minimize the squared error between the target output and the computed output:

$$w_{ji} = w_{ji} + \eta \delta_j x_{ji}$$

where w_{ji} is the weight from unit i to unit j , x_{ij} is the input from unit i to unit j , η is the learning rate and δ_j is the error obtained at unit j . This process of adjusting the weights using training instances is iterated for a fixed number of times or continued until the error is small or cannot be reduced.

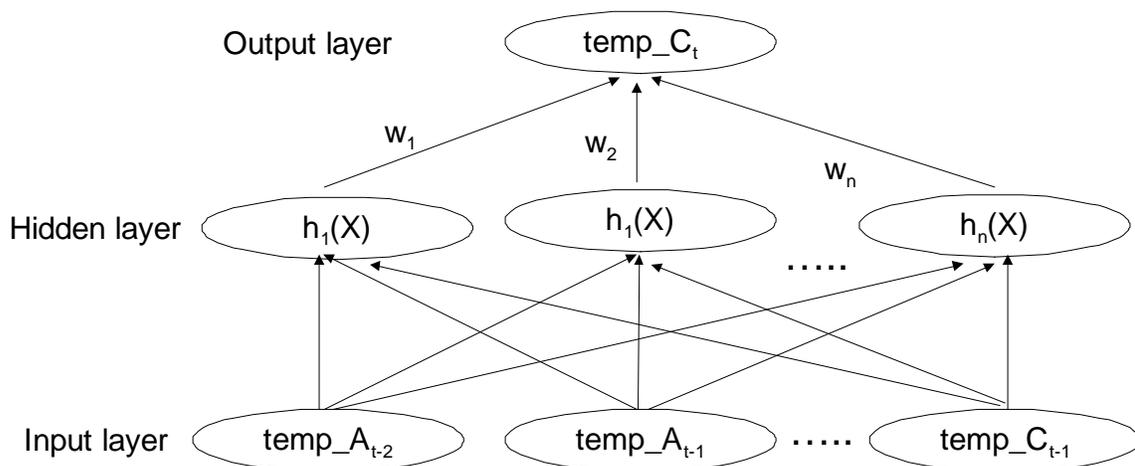


Figure 7: A multilayer perceptron with two hidden layers to predict temperature at a site.

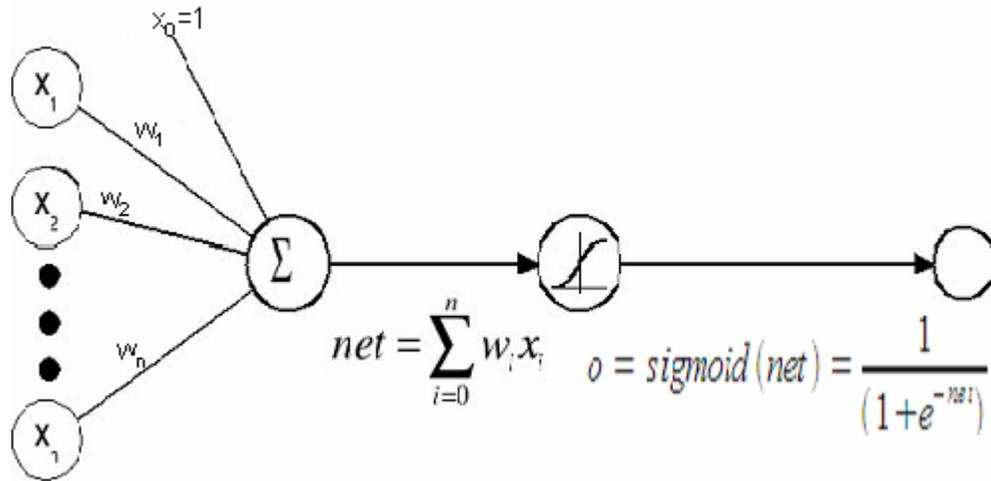


Figure 8: A sigmoid unit that takes inputs x_i with weights w_i

To improve the performance of the algorithm, the weight update made at the n^{th} iteration of the back propagation is made partially dependent on the amount of weight changed in the $n-1^{\text{st}}$ iteration. The contribution to the weight change by the $n-1^{\text{st}}$ iteration is determined by a constant term called momentum (α). The new rule used for weight-update at the n^{th} iteration is

$$\Delta w_{ji}(n) = \eta \delta_j x_{ji} + \alpha \Delta w_{ji}(n-1).$$

This momentum term is added to achieve faster convergence to a minimum.

Radial Basis Function Network

A Radial Basis Function (RBF) network [23, 24] is a feed forward neural network with three layers: input, hidden and output. It differs from an MLP in the way the hidden layer units perform calculations. An RBF Network can build both regression and classification models. We utilize the regression model.

In an RBF network, inputs from the input layer are mapped to each of the hidden units. The hidden units use radial functions such as the bell-shaped Gaussian function for activation. The activation $h(x)$ of the Gaussian function for a given input x decreases monotonically as the distance between x and the center c of the Gaussian function increases. The Gaussian function is of the form

$$h(x) = \exp\left(\frac{-(x-c)^2}{r^2}\right).$$

An RBF network treats the inputs and the hidden units as points in space. The activation of a hidden unit depends on the distance between the input values and the hidden unit. The distance is converted into a similarity measure by the Gaussian function. The point in space for the hidden unit is obtained from the center of the Gaussian for that unit; the width of the Gaussian is a learned parameter as well. An RBF network is trained to learn the centers and widths of the Gaussian function for hidden units, and then to adjust weights in the linear regression model that is used at the output unit.

Conjunctive Rule Algorithm

The Conjunctive Rule algorithm learns a single rule that can predict an output value. An example of a rule that could be developed for our temperature example is

IF (temp_{A_{t-2}} > 30 AND temp_{C_{t-1}} < 90 AND temp_{B_{t-1}} > 40) THEN temp_{C_t} = 30.

A conjunctive rule consists of a set of relations between relevant attributes whose consequent yields an output value. The learning process in the conjunctive rule algorithm attempts to define a rule for all relevant attributes based on the training data. The algorithm learns by calculating the variance reduction for all possible antecedents and selects the one that reduces the variance the most.

Summary

These nine machine learning algorithms will be used to build models to detect abnormal behavior of various types of RWIS sensors. A model provides us with a predicted output value for a given sensor at the current point in time. The actual value reported by the sensor can be compared with the predicted value; the difference between the two is an indication of the probability of a sensor malfunction.

To test this theory, we developed an MSP decision tree to predict temperature values of an RWIS site. We utilized two data elements in performing these predictions: the temperatures of neighboring RWIS and AWOS sites and the distance between the sites. We included distance to aid in the generation of a realistic decision model. The MSP decision tree automatically discards variables that do not contribute to prediction. We chose neighbors using the three clusters shown on the map in Figure 1 to prevent comparisons across macroclimates.

We used three months of data for the training set and one month for the test set. Each set included data from different years. We performed this experiment for various RWIS sites. Figure 9 contains the predicted results and actual results for temperature predictions for January 2003 for a given RWIS site. As may be noted, prediction results were very good.

We conducted a similar experiment for another site that was known to have failed during May 2003. Table 3 contains a partial list of the actual and predicted temperature values during the period of time when a sensor failure occurred. We note the large differences between the corresponding values near the point of failure.

These exploratory experiments suggested that we can successfully build models to predict sensor values for RWIS sites and in some cases actually use these results to identify sensor failures. In the following section we will describe the general performance of the various models in predicting sensor malfunctions.

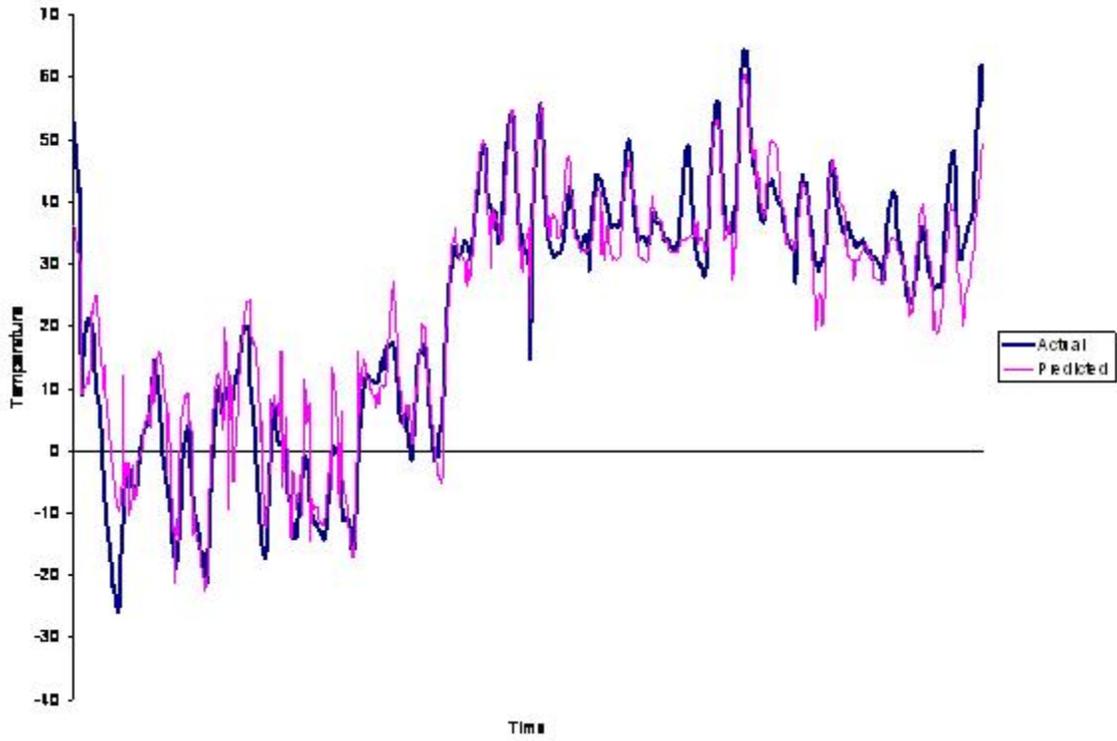


Figure 9. Predicted temperature values versus actual sensor values for January 2003.

Actual	Predicted	Error
67.58	66.86	-0.73
68.27	65.90	-2.37
67.88	63.20	-4.68
66.26	62.30	-3.96
64.58	60.84	-3.74
30.73	59.37	28.64
	56.58	
18.95	52.54	33.59
54.71	53.20	-1.51
52.64	52.77	0.13
51.20	52.68	1.48
50.93	53.43	2.50
50.72	51.90	1.18

Table 3. Actual and predicted temperature values around a sensor failure.

Evaluating the ML Models

To detect RWIS sensor malfunctions, we seek significant variations and/or deviations between the values reported by a sensor and our models' predictions of the values of the sensor for a given time period. Of course, it is imperative that the predictions made by the ML models be highly accurate. To ensure reliability in our assessments of sensor performance, the performance of an algorithm is evaluated using cross-validation. In cross-validation, a subset of the data provided is reserved as a test set and the remaining data, the training set, is used by the algorithm to build a model. The test set is used to evaluate the performance of the model by measuring the accuracy with which the model classifies the test set instances.

In n -fold cross-validation, the dataset (historical RWIS/AWOS sensor data) is divided into n subsets of equal size. One subset is used as a test set and the remaining $n-1$ subsets are used for training. Cross-validation is performed n times with each of the subsets being used as a test set exactly once. The average performance of the all the test sets reflects the overall performance of the algorithm. The advantage of using n -fold cross-validation is that each instance of the dataset is a member of some test set and is used to evaluate the performance of the model within a single dataset. Kohavi [25] suggests using 10-fold or 20-fold cross-validations for optimal estimates.

In this study, we used both classification and regression algorithms to model sensor data. Classification algorithms are used to classify a given instance into a set of discrete categories; precipitation type and temperature are predicted using this approach. The three classification algorithms previously described, namely J48 decision trees, naive Bayes, and Bayesian Networks are used to predict values for these types of sensors. Since classification algorithms allow continuous input, the data used to build the models included both current and historical temperature readings for nearby RWIS/AWOS sites.

Each dataset was constructed according to an input format we derived based on the weather data available for the RWIS and the AWOS sites. The dataset was then split into a training set and a test set using the cross-validation method. Multiple 10-fold cross-validation was used to obtain estimates of the model's performance.

Classification algorithms predict the class value taken by the output attribute, in our case precipitation type and temperature class value, for a given instance in the test set. The accuracy of the prediction results is represented in the form of a confusion matrix, with rows representing sensor values and columns representing predicted values for the output attributes. Each cell in the confusion matrix contains the number of times the actual class given by the column is predicted. The numbers along the main diagonal indicate the number of times the predicted class values are equal to the actual class values. Thus, the sum of entries along the diagonals divided by the total number of instances present in the test set yields the percentage of the number of correctly classified instances. In the case of multiple n -fold cross-validations, the confusion matrices obtained for each test set are averaged to obtain a confusion matrix for the model as a whole. An example of confusion matrices employing only two test sets is shown in Figure 10.

For precipitation, we can use the percentage of instances that were classified correctly (when no precipitation was present and when precipitation was present) to determine the accuracy of the model built. This method is particularly important for determining the accuracy of predicting precipitation because in most cases no precipitation is reported; thus, in cases when precipitation is reported, the algorithm may try to classify it as no precipitation by assuming the data is noise.

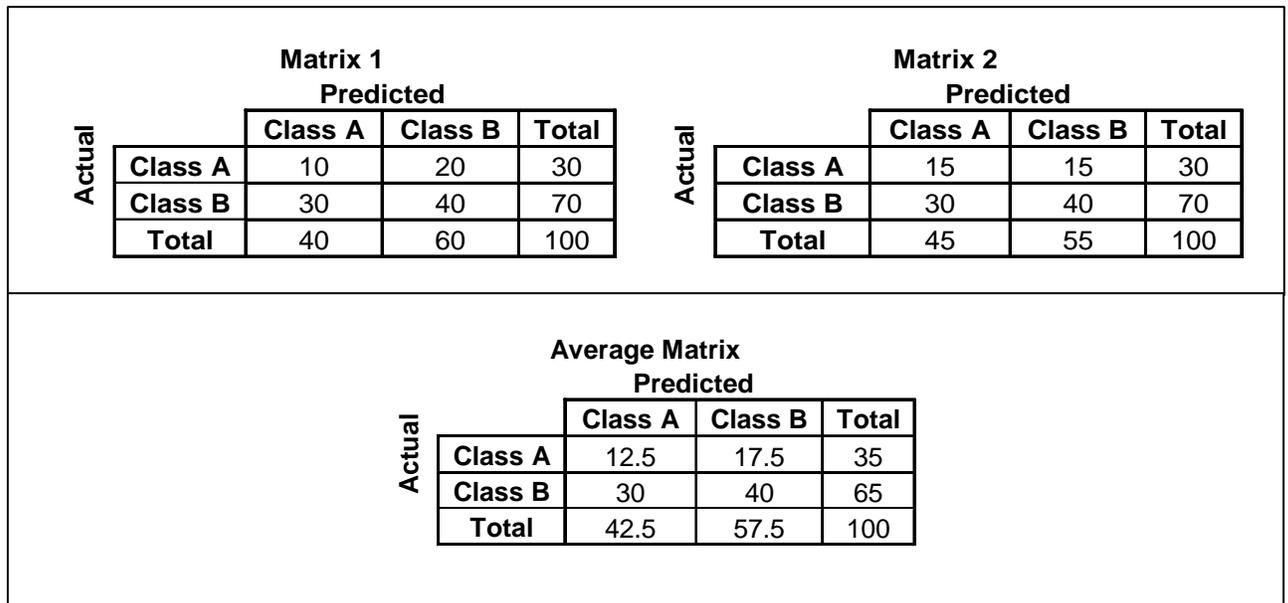


Figure 10: Example of Confusion Matrices

For temperature, the error in the results is obtained using the absolute distance between actual and predicted class values. A distance between two adjacent classes is taken as 1; for identical class values, the distance is 0. The greater the distance the poorer the prediction is.

Regression algorithms may be used to predict temperature and visibility. The performance of regression algorithms can be determined by the difference between the actual value and predicted value, which gives the amount of prediction error (as previously illustrated in Table 3). The mean of the absolute errors across all instances in the test set represents the performance of the algorithm. In the case of multiple n -fold cross-validations, the error value of course is averaged across all the test sets. For a model whose prediction accuracy is high, even a slight difference between the reported sensor value and the predicted value may indicate sensor malfunction. We evaluated the performance of each regression algorithm using the 10-fold cross-validation technique.

Empirical Analysis

We conducted a set of experiments to predict temperature, precipitation type, and visibility at various RWIS sites using the machine learning methods discussed previously. We used regression algorithms to predict visibility, classification algorithms to predict precipitation, and both machine learning methods to predict temperature. For each experiment we present the methodology, summarize the results obtained, and classify the performance of each method in predicting sensor values. The raw results of each experiment are detailed in the Appendix.

Predicting Temperature using Regression Algorithms

In this set of experiments, we used linear regression (LR), least median squares (LMS), M5 Prime (M5P), multilayer perceptron (MLP), RBF Network (RBF), and Conjunctive Rule (CR) to predict the current temperature at an RWIS site. We used the current temperature and three hours of historical temperature data from neighboring RWIS/AWOS sites to create a model for each algorithm. As noted

in Figure 1, we are working with three clusters containing a total of 13 RWIS sites and 14 AWOS sites.

We calculate temperature as the difference between the temperature values reported at an RWIS site and the projected hourly temperature at its corresponding AWOS site. To calculate the projected hourly temperature for an AWOS site, we used temperature data from 1997 to 2004 for this site. The projected hourly temperature for a given hour of the day is defined as the sum of the average temperature reported for that day and the monthly average difference in temperature of that hour in the day for the respective month. For example, if the temperature at the RWIS site 67 is 32°F at time t and the projected hourly temperature at time t for its corresponding AWOS site, KLYU, is 30°F, then the temperature of site 19 for hour t is 2°F. Using historical data to extract temperature values as deviations from the average provides additional information apart from what we already have (i.e., the RWIS and the AWOS data).

We used the regression algorithms mentioned above and the dataset generated to try to predict the current hour temperature value at the selected 13 RWIS sites. Our testing involved ten 10-fold cross-validation runs; each instance in the dataset is predicted 10 times (i.e., one in each cross-validation). The average of these 10 values gives the final predicted value. We used absolute error between the actual value reported by the sensor and the value predicted by the algorithm to evaluate the performance of the algorithm.

Figure 11 shows the mean absolute error and the standard deviation obtained across all RWIS sites for each regression algorithm. The actual performance values are contained, in a more detailed format in the Appendix.

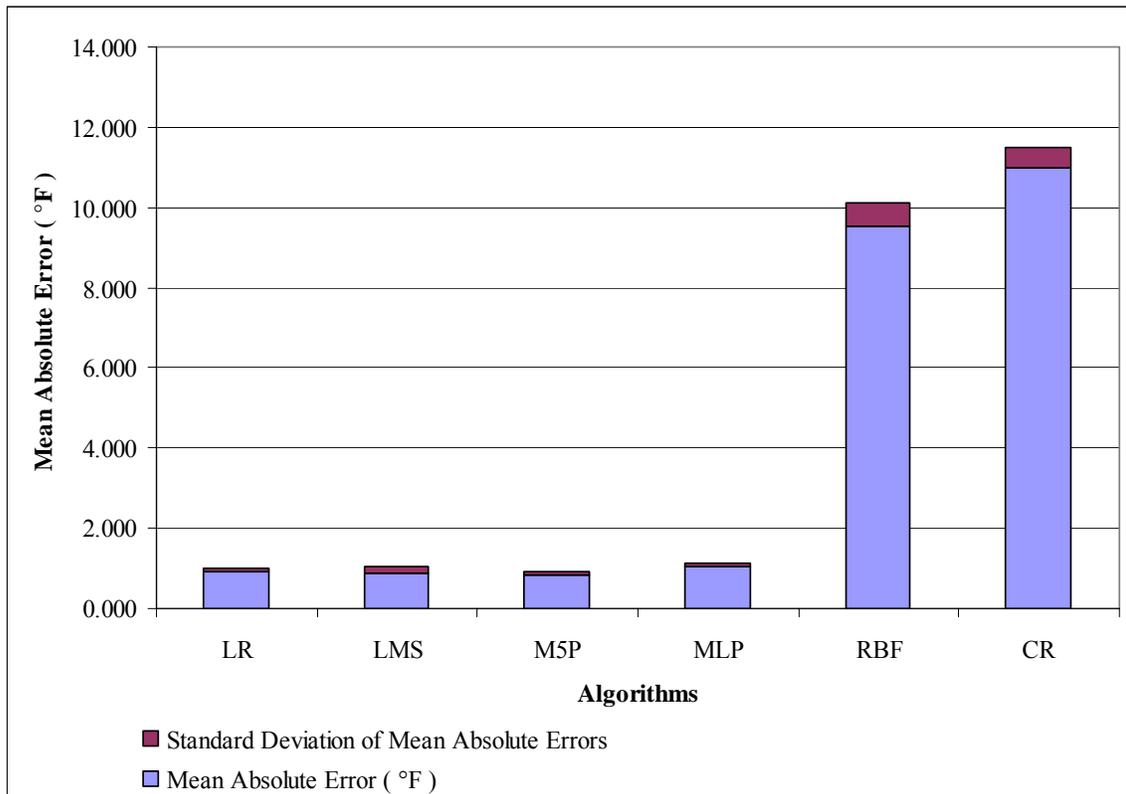


Figure 11: Performance of Regressions Algorithms for Predicting Temperature

The radial basis function network and the conjunctive rule algorithm failed to predict temperature values for all sites. These two algorithms are very sensitive to parametric values and would have performed better if we had significantly tuned the parameters. The other four models performed well and were accurate to $\pm 1^\circ\text{F}$.

The standard deviation value for the mean absolute errors across different sites for an algorithm measures the variation in the error values across the sites. A small standard deviation suggests the model is fairly independent of the site location and can predict temperature with similar accuracy for any site. This is desirable since such a model can be used across all sites. The RBF and CR algorithms have higher standard deviations in comparison to the other algorithms, but this is likely a function of their high error in predicting temperature values. The best standard deviation results are obtained for the M5 prime algorithm with 0.058 as the standard deviation value across sites.

Combining the results and giving priority to the algorithms that have a low absolute error and exhibit similar performance across all sites, we find that M5 prime and linear regression performed best, closely followed by least median squares. It can be clearly seen that CR and RBF did not do well concerning prediction of temperature; they are not used in the experiments that follow.

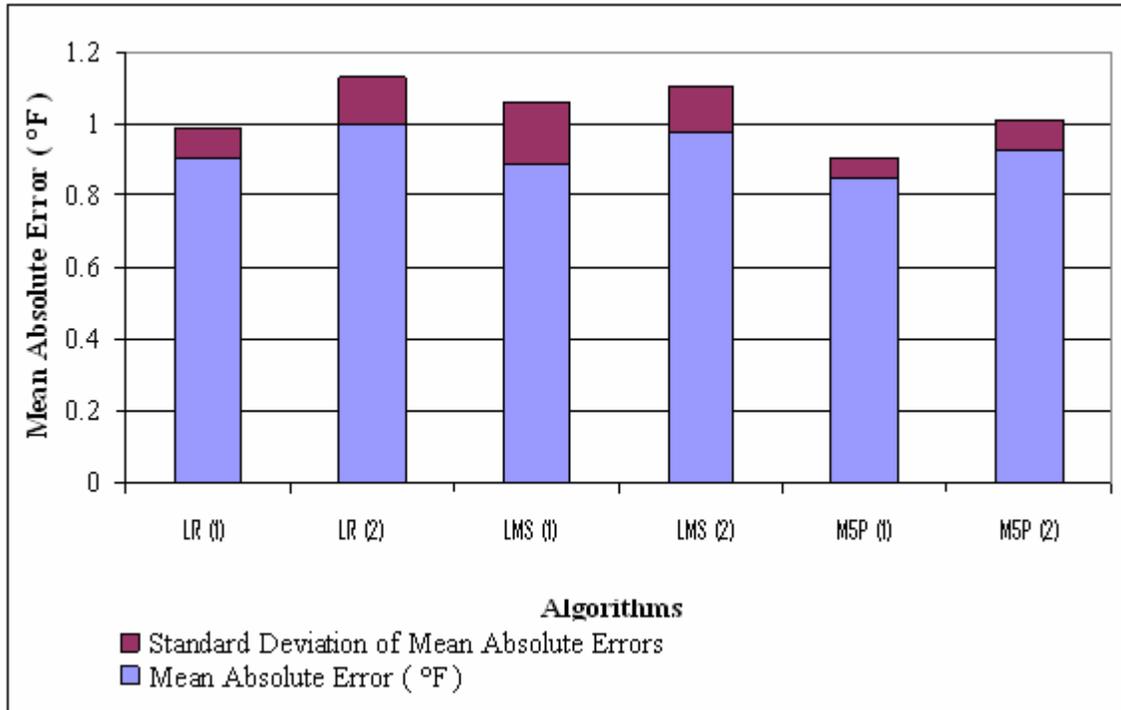
To detect RWIS temperature sensor malfunctions, we recommend building models from M5P, LR, or LMS. When the difference between the error reported for an hour and the mean absolute error obtained from testing the model is greater than 1.96 standard deviations, with standard deviation of error calculated from the test results, we can say with 95% accuracy that the sensor has failed.

Due to the fact that the presence of precipitation affects the temperature at a location, we decided to investigate the use of models for predicting temperature that also include precipitation as part of their input data (in addition to temperature values). We used the regression algorithms LMS, LR, and M5P to predict the current temperature class value at an RWIS site. Our input dataset consists of the current and the previous three hour temperature values for clustered RWIS and AWOS sites and the precipitation type observed at the current hour at the RWIS sites.

We again used the mean absolute error obtained from the ten 10-fold cross-validation runs to evaluate the performance of these algorithms. Figure 12 shows the mean absolute error and the standard deviation averaged for all RWIS sites. For comparison, the results of the first experiment for each method are included for each algorithm.

Of the three algorithms (LR, LMS and M5P) used, we see that M5P again yields better results in predicting temperature with lesser mean absolute error and consistency in predictions across the sites (with respect to the standard deviation of errors across various sites). Significant variation of absolute errors reported by these three algorithms was not seen, with all predicting temperature with accuracy close to 0.95°F .

Including precipitation type in the dataset as an additional source of information does not improve performance of the models. It can be observed from Figure 12 that the mean absolute error for the algorithms LR, LMS and M5P is slightly higher for this experiment when compared with the first experiment; the mean absolute error increased by approximately 0.09°F when precipitation type was included in the dataset.



Fig

Figure 12: Performance of Regressions Algorithms for Predicting Temperature without Precipitation (1) and with Precipitation (2)

Predicting Temperature using Classification Algorithms

In this experiment we built models to predict temperature at an RWIS site using classification algorithms. As in the previous work, we used the current temperature and three hours of historical temperature data from neighboring RWIS/AWOS sites to create a model for each algorithm. The temperature values had to be discretized for use with these algorithms.

Discretization of temperature involves finding a set of values that split the continuous sequence into intervals, each interval having a single discrete value. We use the projected hourly temperature for an AWOS site along with the average current temperature at the closest RWIS neighboring site to determine the class value for the current RWIS temperature value. Specifically, the reported temperature at an RWIS site is subtracted from the projected hourly temperature for the AWOS site closest to it for that specific hour. This difference is then divided by the standard deviation of the projected hourly temperature for that AWOS site. The result indicates how much the actual value deviates from the projected value, that is, the number of standard deviations from the projected value:

$$num_stdev = (actual_temp - proj_temp) / std_dev$$

Classes may be defined based on the number of standard deviations from the mean.

Class Value		Class Value	
1	$\text{num_stdev} < -2$	6	$0.25 < \text{num_stddev} \leq 0.5$
2	$-2 \leq \text{num_stdev} \leq -1$	7	$0.5 < \text{num_stddev} \leq 1$
3	$-1 < \text{num_stdev} \leq -0.5$	8	$1 < \text{num_stddev} \leq 2$
4	$-0.5 < \text{num_stdev} \leq -0.25$	9	$\text{num_stdev} > 2$
5	$-0.25 < \text{num_stdev} \leq 0.25$		

Table 4: Classes used in Discretization of Temperature

Consider the class values defined in Table 4. To discretize temperature, the number of standard deviations from the mean obtained for a given temperature value is mapped to one of the nine ranges in the table and the temperature is assigned the respective class value. For example, to convert the actual temperature 32°F at an RWIS site into a class value, we calculate the projected temperature for that hour at the associated AWOS site KORB, say 30°F, and the standard deviation of projected temperatures at KORB for the year, say 5.06. Our new representation for 32°F is calculated as 0.396. We thus arrive at the class value of 6 for the temperature of 32°F. Such a representation has an advantage as the effect of season and time of day are at least partially removed from the input data.

We used two classification algorithms, J48 decision trees and naïve Bayes, to predict temperature. We ran a series of experiments to predict the class for each temperature sensor at the 13 RWIS sites used previously. We used the absolute distance between the class value of the temperature reported by the RWIS sensor and the predicted temperature class to evaluate the performance of the classification algorithms; a distance of 0 indicates that the predicted class value is the same as the class value reported by the sensor. The percentage of instances that were reported with a distance ranging from 1 to 6 for both J48 and NB are shown in the Figure 13.

The J48 decision tree clearly outperforms the Naive Bayes algorithm by classifying 93.6% of the instances in the dataset; only 32.3% of the instances were correctly classified by the NB algorithm. No instances were reported having a distance of more than 3 between actual and predicted class value when prediction was done using J48; in fact, 99.4% of the instances in the dataset were predicted with a distance of either 0 or 1. Using the J48 algorithm, we can readily predict RWIS sensor malfunctions when the distance between the actual and predicted temperature class value is greater than 1. For all the known malfunction cases in our dataset, the difference in the class value is actually 2 or greater. Thus, malfunctions can be detected with high accuracy with the J48 algorithm.

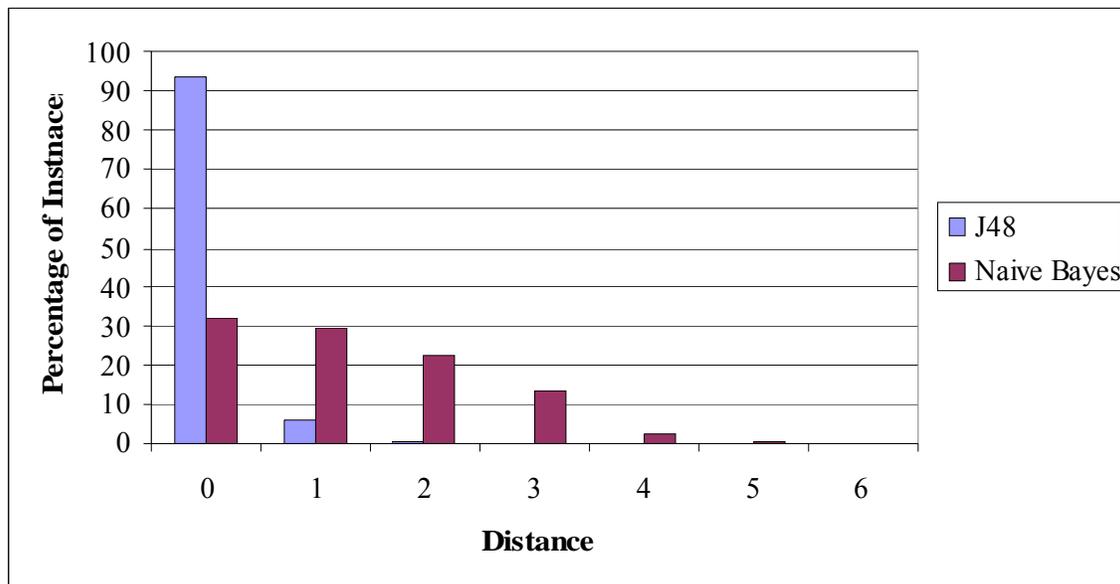


Figure 13: Performance of Classification Algorithms in Predicting Temperature

Predicting Precipitation Type using Classification Algorithms

We conducted experiments to determine the performance of J48 decision trees, naive Bayes, and Bayesian networks to predict precipitation type. We focused on classification algorithms because precipitation is reported by RWIS sensors in the form of class values. Since the output of classification models are also class values, we can readily compare these values with the sensor’s reported values. Temperature data at the neighboring RWIS/AWOS sites along with the precipitation type reported at the RWIS sites are used to form the input dataset. We included the temperature information to help in the prediction process as there is a correlation between temperature and precipitation observed at a location. Precipitation information from AWOS sites was not used because the effect of precipitation is localized and does not affect the occurrence of precipitation at nearby and other locations.

The output of our models is presented as a confusion matrix; in addition, statistical results such as the classification error, root mean squared errors, and the percentage of correctly classified instances are calculated. Figure 14 shows, for each of the three algorithms, the classification error and standard deviation obtained from predicting precipitation type across all 13 RWIS sites. These values were obtained after averaging the reported values for each cross-validation run. The classification error and standard deviation values obtained for each individual site as well as a detailed analysis of the confusion matrices are given in the Appendix.

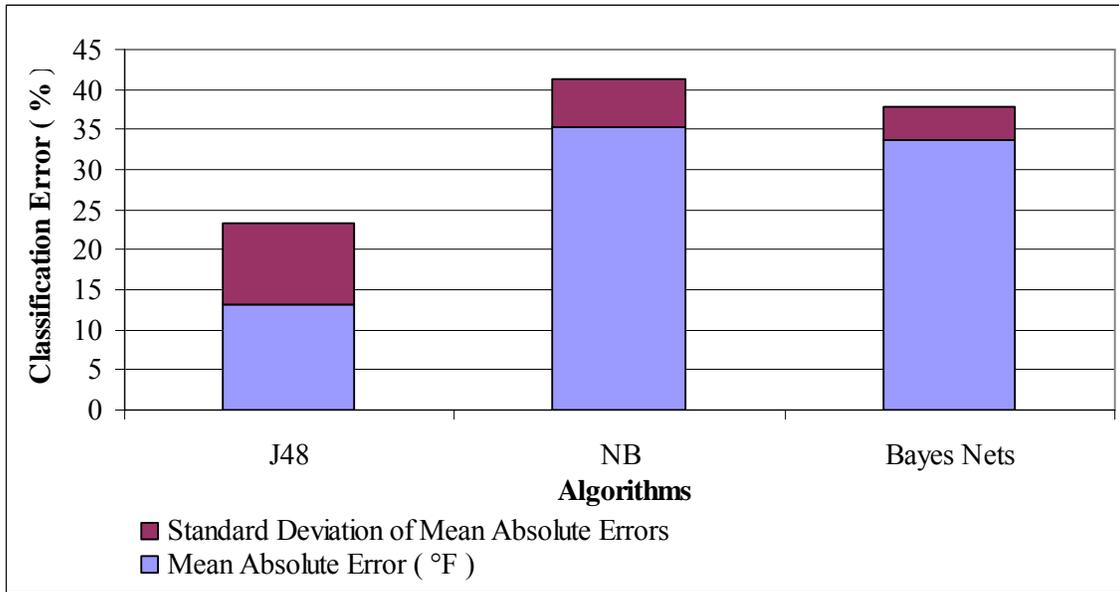


Figure 14: Performance of Classification Algorithms in Predicting Precipitation

The overall results in Figure 14 reveal that J48 performs best in predicting precipitation type, but its high standard deviation reveals that this method is inconsistent in its predictions across the sites. An analysis of the raw data (see the Appendix) indicates a 0.077 classification error for site 35 and a 0.356 classification error for site 67. Other than site 67, none of the sites have a classification error above 0.26, a value lower than the mean absolute errors reported by NB and Bayes net.

Depending on the type of sensor, some RWIS sites report precipitation as present or not present, while other sites actually report the type of precipitation observed. In order to compare sites, we combined the different types of precipitation together and just reported any type as precipitation present. This allows comparison of the percentage of instances when precipitation was correctly classified between sites. Figure 15 shows the percentage of instances that were correctly classified by the classification models when precipitation was present and when no precipitation was present. Instances with precipitation present were very few, because precipitation does not exist for a long period and, in fact, may only be present for an hour or two (leading to very few positive observations being reported over any given period of time).

As seen in Figure 15, naive Bayes and Bayesian networks predict only about 19% of the instances as no precipitation present, when in actuality 81% of the instances were reported as no precipitation. These algorithms perform poorly for classifying no precipitation. However, J48 classifies instances with no precipitation present correctly, identifying 75% of the instances as no precipitation present. But it fails to report the presence of precipitation with the accuracy it predicts precipitation.

In general, none of the algorithms perform well in correctly classifying precipitation. The combination of J48 and Bayes Net can be used to detect malfunctions, with J48 being used to predict the absence of precipitation and Bayesian networks, the presence of precipitation. We choose Bayesian networks over naive Bayes because of its smaller mean absolute error and standard deviation.

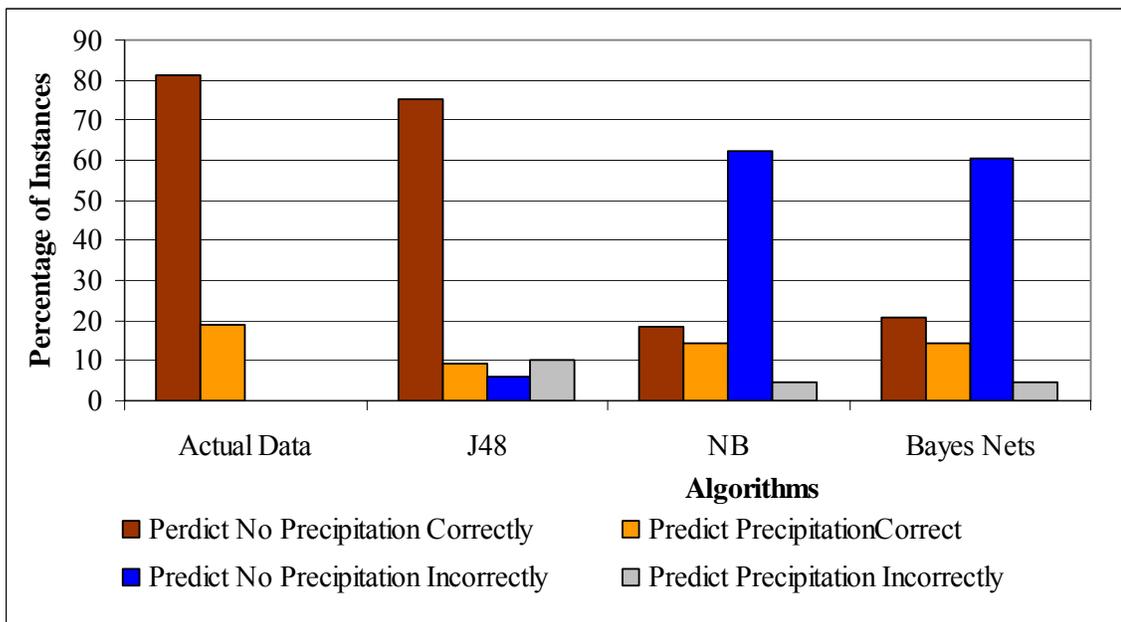


Figure 15: Accuracy of the Classification Algorithms in Predicting Precipitation

Due to varying accuracies in prediction among different sites (see Appendix), each individual site requires its own model and specific percentages with which they correctly classify precipitation.. For example, for site 62, J48 misclassifies 3.84% of the instances when predicting no precipitation and Bayes nets misclassifies 0.77% of instances when reporting presence of precipitation. For this site, when J48 predicts incorrectly that no precipitation is present, we can say with 95% accuracy that the sensor has failed. A similar rate of accuracy may be concluded when Bayes nets wrongly reports presence of precipitation. The combination of J48 and Bayes nets produces high accuracy in detecting sensor malfunctions when each model is individually responsible for classifying absence of precipitation and presence of precipitation, respectively.

Predicting Visibility using Regression Algorithms

In this set of experiments we predict visibility for a given RWIS site using four regression algorithms: least median square, linear regression, M5 prime, and multilayer perceptron. The temperature data from the RWIS site and the nearby RWIS/AWOS sites along with the precipitation type and visibility reported at the RWIS sites are used to form the input data set. Precipitation is included as an attribute since visibility is affected by its presence. The mean absolute error obtained from site 67 was excluded from calculating performance results, since it reports visibility up to ten miles whereas all other sites report only up to about one mile.

Figure 16 shows the mean absolute error and standard deviation obtained across all sites for predicting visibility, excluding site 67. The overall mean absolute error for each algorithm was obtained by averaging the values reported for each cross-validation run. Raw performance results are included in the Appendix.

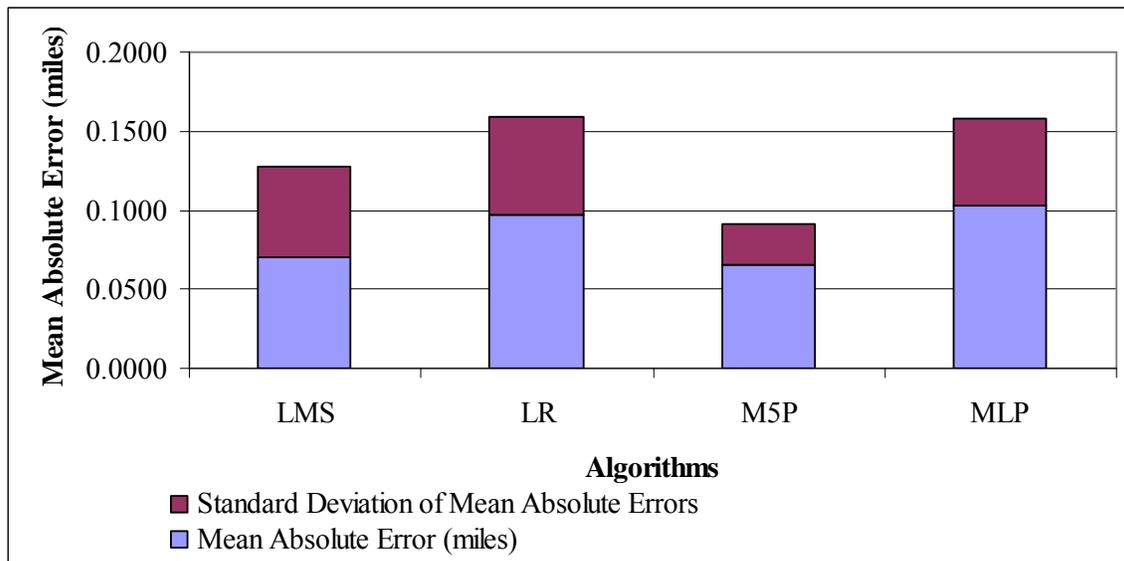


Figure 16: Performance of Regression Algorithms for Predicting Visibility

The LMS and M5P algorithms yield similar performance in predicting visibility and exceed the performance of LR and MLP. However, the standard deviation of errors across various sites was lower for M5P. Thus, M5P would serve as a good predictor of visibility across different sites; in fact, an M5P model created using one site can be used to predict values at another site. When the difference between prediction error and the mean absolute error for M5P is greater than 1.96 standard deviations (with standard deviation being calculated from the error values obtained during cross-validations), one can reasonably conclude that the sensor is malfunctioning.

Conclusion

In this project, we built various machine learning models that employ data from nearby sensors in order to predict likely values of the sensors we are interested in. We used both classification and regression algorithms; in particular, we used three classification algorithms: J48 decision trees, naïve Bayes, and Bayesian networks, and six regression algorithms: linear regression, least median squares, M5P, multilayer perceptron, RBF network, and the conjunctive rule algorithm. We performed a series of experiments to determine which of these models can be used to detect malfunctions in RWIS sensors. We compared the values predicted by the various ML methods to the actual values observed at an RWIS sensor to detect sensor malfunctions.

Accuracy of an algorithm in predicting values plays a major role in determining the accuracy with which malfunctions can be identified. From the experiments performed to predict temperature at an RWIS sensor, we concluded that the classification algorithms LMS and M5P gave results accurate to $\pm 1^\circ\text{F}$ and had low standard deviation across sites. Both models were identified to be able to detect sensor malfunctions accurately. RBF Networks and CR failed to predict temperature values. A threshold distance of 2°F between actual and predicted temperature values is sufficient to identify a sensor malfunction when J48 is used to predict temperature class values. The use of precipitation as an additional source of decision information produced no significant improvement on the accuracy of these algorithms in predicting temperature.

The J48, naive Bayes, and Bayes net exhibited mixed results when predicting the presence or absence of precipitation. However, a combination of J48 and Bayes nets can be used to detect precipitation sensor malfunctions, with J48 being used to predict the absence of precipitation and Bayesian networks, the presence of precipitation.

Visibility was best classified using M5P. When the difference between prediction error and the mean absolute error for M5P is greater than 1.96 standard deviation, one can reasonably conclude that the sensor is malfunctioning.

References

- [1] How does the Road Weather Information System Work? U.S. Department of Transportation Federal Highway Administration, <http://www.fhwa.dot.gov/winter/roadsvr/rwis.htm>
- [2] Monitoring System Gives Highway Crews the Edge in Winter Maintenance, U.S. Department of Transportation Federal Highway Administration, Pub. No. FHWA-SA-96-045.
- [3] Automated Surface Observing System, U.S. Department of Commerce and National Weather Service, Manual S100, July 1998.
- [4] Use of Road Weather Information Systems in the Improvement of Transportation Operations in the Complex Terrain of Western Nevada, Collaborative Research on Road Weather Observations and Predictions by Universities, State Departments of Transportation, and National Weather Service Forecast Offices, U. S. Department of Transportation, Publication No. FHWA-HRT-04-109, 2004.
- [5] Witten, I. and Frank, E., *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.
- [6] Quinlan, R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco, 1993.
- [7] Quinlan, R., *Induction of Decision Trees*, Machine Learning, vol. 1, pp. 81-106, 1986.
- [8] Fayyad, U. and Irani, K., *Multi-interval discretization of continuous-valued attributes for classification learning*, Proceedings of 13th International Joint Conference on Artificial Intelligence, pp 1022-1027, Morgan Kaufmann, 1993.
- [9] Good, I., *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*, M.I.T. Press, 1992.
- [10] Langley, P., Iba, W. and Thompson, K., *An Analysis of Bayesian Classifiers*, Proceedings of the Tenth National Conference on Artificial Intelligence, pp. 223-228, AAAI Press, 1992.
- [11] Friedman, N., Geiger, D. and Goldszmidt, M., *Bayesian network classifiers*, Machine Learning, vol. 29, pp. 131-163, 1997.
- [12] Pearl, J., *Probabilistic reasoning in intelligent systems*, Morgan Kaufman, 1988
- [13] Akaike, H., *A new look at the statistical model identification*. IEEE Transaction on Automatic Control, vol. AC-19, pp. 716-723, 1974.
- [14] Rissanen, J., *Modeling by shortest data description*. Automatica, vol. 14, pp. 465-471, 1978.
- [15] Allen, T. and Greiner, R., *A model selection criteria for learning belief nets: An empirical comparison*, Proceedings of the International Conference on Machine Learning, pp. 1047-1054, 2000.
- [16] Cooper, G. and Herskovits, E., *A Bayesian Method for the Induction of Probabilistic Networks from Data*, Machine Learning, vol. 9, pp. 309-347, 1992

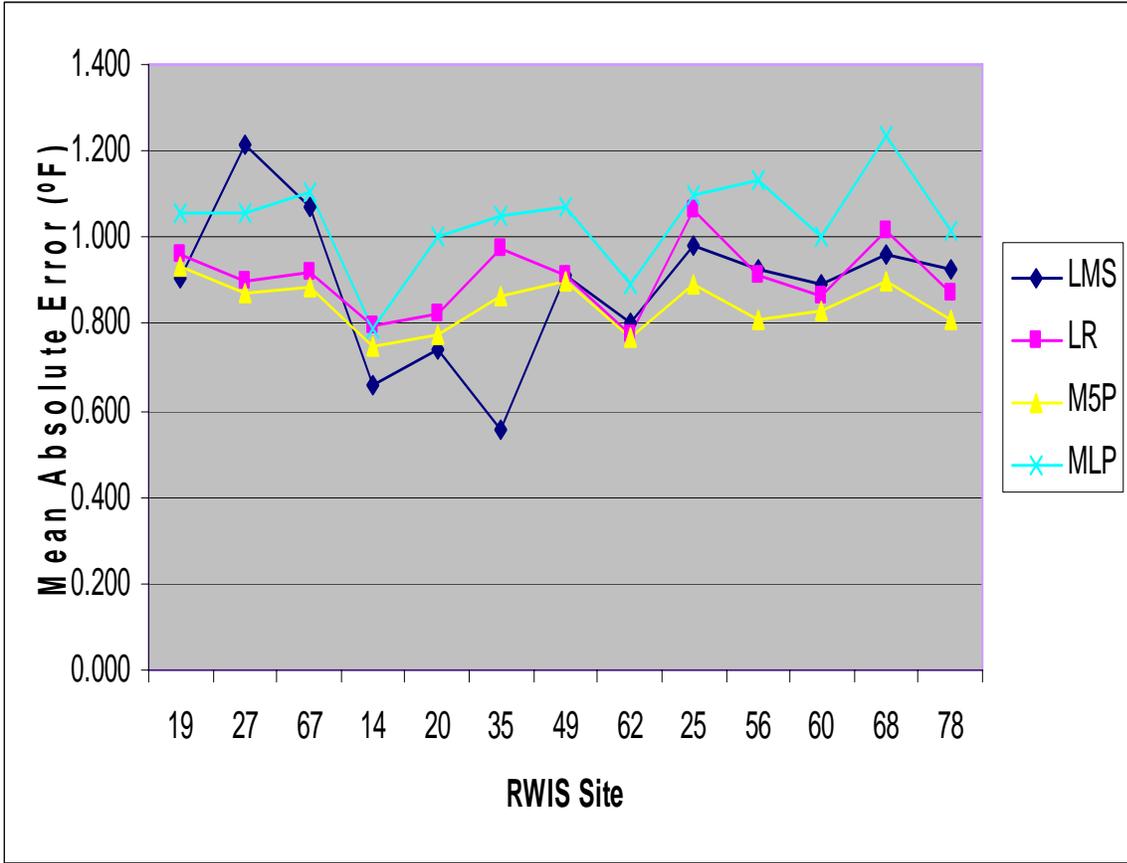
- [17] Russell, S., Binder, J., Koller, D. and Kanazawa, K., *Local Learning in Probabilistic Networks with Hidden Variables*, Proceedings of the 14th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1995.
- [18] Rousseeuw, P., *Least Median Squares of Regression*, Journal of American Statistical Association, vol. 49, pp. 871-880, December 1984
- [19] Wang, Y. and Witten, I., *Inducing Model Trees for Continuous Classes*, In Poster Papers of the Ninth European Conference on Machine Learning, pp. 128-137, Prague, Czech Republic, April, 1997.
- [20] Quinlan, R., *Learning with Continuous Classes*, Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, pp. 343-348. World Scientific, Singapore, 1992.
- [21] Bishop, C., *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.
- [22] Rumelhart, D., Hinton, G. and Williams, R., *Learning internal representations by error propagation*, Parallel Distributed Processing: Explorations in the Microstructures of Cognition, vol.I, pp. 318–362, MIT Press, 1986.
- [23] Buhmann, M., Albowitz, M., *Radial Basis Functions: Theory and Implementations*, Cambridge University Press, 2003.
- [24] Orr, M., *Introduction to radial basis function networks*. Technical report, Institute for Adaptive and Neural Computation of the Division of Informatics at Edinburgh University, Scotland, UK, 1996, <http://www.anc.ed.ac.uk/~mjo/papers/intro.ps.gz>.
- [25] Kohavi, R., *A study of cross-validation and bootstrap for accuracy estimation and model selection*, Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995.

APPENDIX

Experimental Results

		ML Algorithms					
RWIS Site		LMS	LR	M5P	RBF	CR	MLP
Set 1	19	0.908	0.960	0.936	9.521	11.150	1.059
	27	1.217	0.896	0.873	9.465	10.199	1.058
	67	1.069	0.918	0.885	10.062	11.596	1.108
Set 2	14	0.659	0.795	0.751	8.478	10.605	0.789
	20	0.743	0.820	0.776	9.417	10.821	1.001
	35	0.553	0.977	0.864	9.523	10.817	1.051
	49	0.916	0.913	0.898	9.579	11.017	1.074
	62	0.800	0.779	0.769	9.383	11.040	0.892
Set 3	25	0.984	1.062	0.889	10.386	11.957	1.097
	56	0.925	0.913	0.807	10.510	11.512	1.133
	60	0.889	0.867	0.833	9.675	11.078	1.002
	68	0.958	1.015	0.901	9.017	10.439	1.235
	78	0.929	0.875	0.809	8.945	10.449	1.012
Mean of Abs. Errors (°F)		0.888	0.907	0.845	9.535	10.975	1.039
StdDev of Abs. Errors		0.171	0.083	0.058	0.559	0.503	0.110

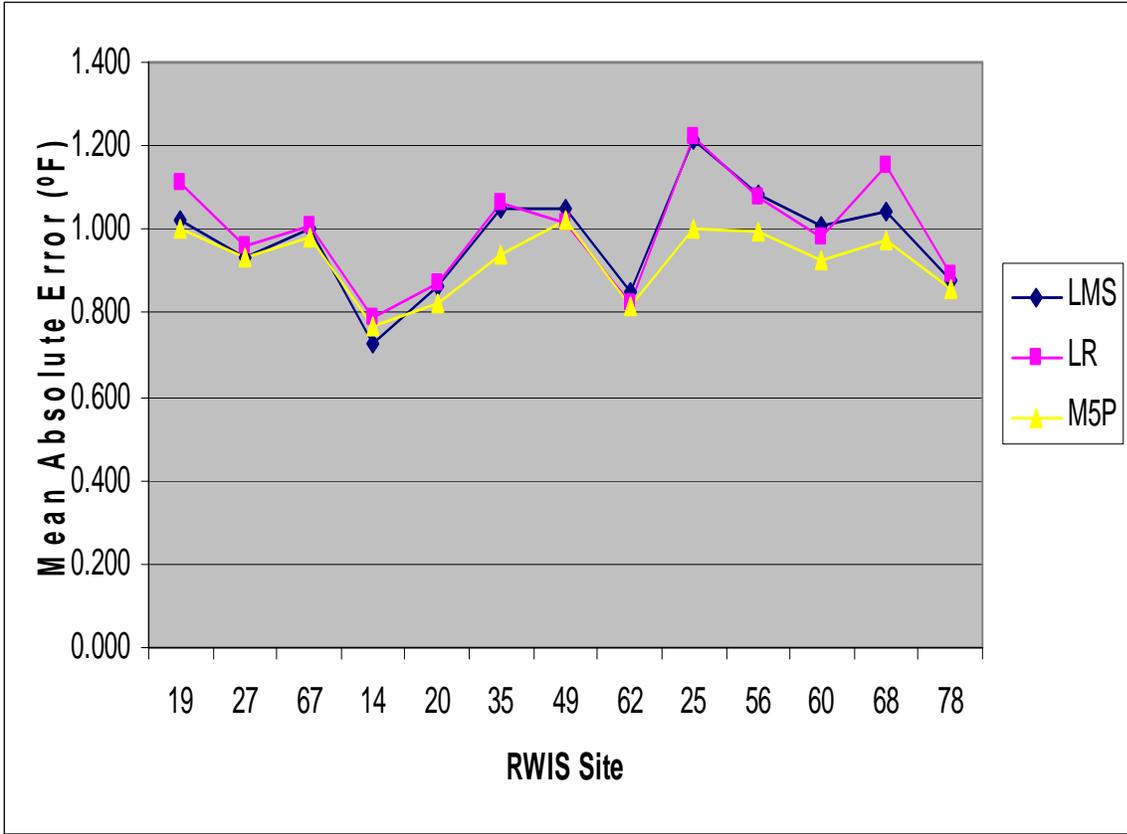
Results obtained from using regression algorithms to predict temperature at an RWIS site (see Figure 11). Input consists of temperature information from RWIS/AWOS sites along with temperature offset for the RWIS sites. The table has mean absolute error values averaged over ten 10-fold cross-validations.



Mean absolute errors for different RWIS sites obtained from predicting temperature using regression algorithms.

		ML Algorithms		
		LMS	LR	M5P
Set 1	19	1.023	1.115	1.001
	27	0.935	0.959	0.931
	67	1.001	1.006	0.984
Set 2	14	0.726	0.788	0.771
	20	0.862	0.872	0.827
	35	1.052	1.062	0.938
	49	1.051	1.014	1.022
Set 3	62	0.848	0.827	0.815
	25	1.217	1.222	1.004
	56	1.084	1.077	0.992
	60	1.007	0.981	0.924
	68	1.046	1.154	0.973
	78	0.876	0.891	0.856
Mean of Abs Errors (°F)		0.979	0.997	0.926
StdDev of Abs Errors		0.127	0.130	0.083

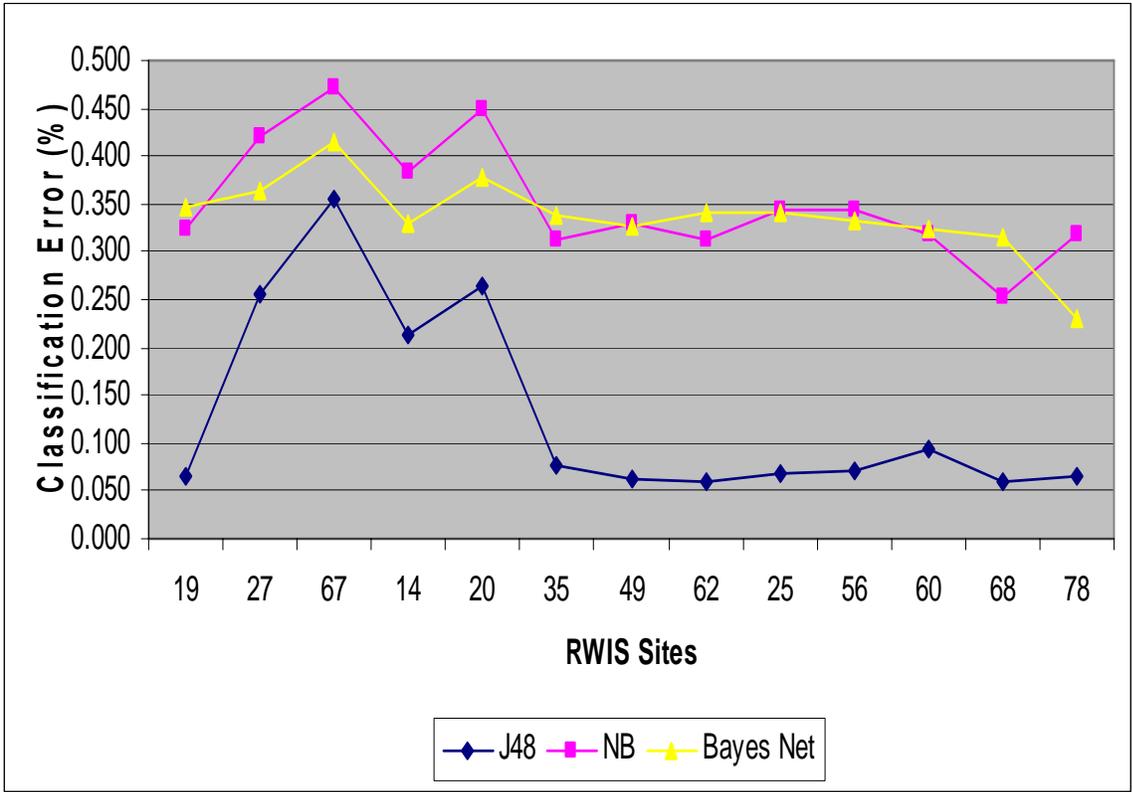
Results obtained from using regression algorithms to predict temperature. Input consists of temperature information from RWIS/AWOS sites along with precipitation type for the RWIS sites. (see Figure 12) The table has mean absolute error values averaged over ten 10-fold cross-validations.



Mean absolute errors for different RWIS sites obtained from predicting temperature using regression algorithms, with precipitation type information included in the input data set.

RWIS Site		ML Algorithms		
		J48	NB	Bayes Net
Set 1	19	0.064	0.325	0.346
	27	0.256	0.420	0.363
	67	0.356	0.472	0.414
Set 2	14	0.213	0.384	0.330
	20	0.265	0.450	0.379
	35	0.077	0.312	0.337
	49	0.062	0.328	0.328
	62	0.061	0.312	0.341
Set 3	25	0.068	0.342	0.342
	56	0.072	0.345	0.333
	60	0.095	0.317	0.323
	68	0.061	0.253	0.315
	78	0.065	0.318	0.231
Mean of Classification Errors		0.132	0.352	0.337
StdDev of Classification Errors		0.102	0.062	0.041

Results obtained from using classification algorithms to predict precipitation type at an RWIS site. (see Figure 14) Input data consists of temperature information from RWIS/AWOS sites along with precipitation type for the RWIS sites. The table has classification error values averaged over ten 10-fold cross-validations.



Classification errors for different RWIS sites obtained from predicting precipitation using classification algorithms.

RWIS Site	Actual Data		J48			
	NP	P	NP->NPP	P->PP	P->NPP	NP->PP
19	86.43	13.57	81.99	5.59	4.44	7.98
27	76.29	23.71	68.78	13.31	7.51	10.40
67	63.65	36.35	50.74	18.55	12.91	17.79
25	84.92	15.08	80.49	6.61	4.43	8.47
56	86.09	13.91	81.31	4.73	4.78	9.18
60	79.59	20.41	71.95	8.82	7.63	11.59
68	87.74	12.26	85.31	5.29	2.43	6.97
78	83.33	16.67	78.30	9.46	5.03	7.21
14	78.48	21.52	73.66	10.33	4.82	11.19
20	70.49	29.51	60.79	15.77	9.70	13.74
35	84.15	15.85	78.15	6.22	6.00	9.63
49	86.41	13.59	82.47	5.90	3.94	7.69
62	85.87	14.13	82.03	6.99	3.84	7.13
Average	81.02	18.98	75.06	9.05	5.96	9.92
Std Dev	7.23	7.23	9.97	4.36	2.84	3.12

NP: No precipitation reported
P: Precipitation reported
NP->NPP: No precipitation predicted when No precipitation was reported
P->PP: Precipitation predicted when precipitation was reported
P->NPP: No precipitation predicted when precipitation was reported
NP->P: Precipitation predicted when precipitation was reported

Percentage of instances predicted correctly using the J48 classification algorithm to predict precipitation type at an RWIS site. (see Figure 15)

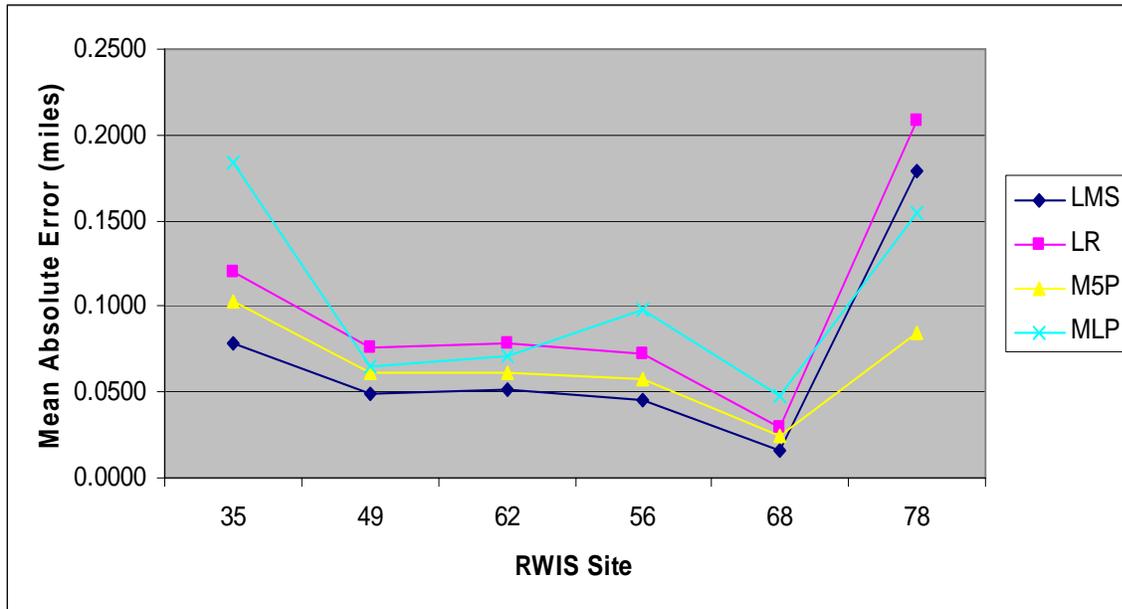
RWIS Site	Naive Bayes				Bayes Nets			
	NP->NPP	P->PP	P->NPP	NP->PP	NP->NPP	P->PP	P->NPP	NP->PP
19	7.88	13.11	78.55	0.46	1.81	13.48	84.62	0.09
27	43.72	14.14	32.57	9.57	52.02	11.92	24.27	11.79
67	27.66	12.55	35.99	23.80	40.62	18.89	23.03	17.46
25	2.53	14.92	82.38	0.16	2.29	15.02	82.63	0.07
56	2.99	13.78	83.10	0.13	4.63	13.75	81.46	0.16
60	3.33	19.95	76.26	0.46	1.17	20.32	78.41	0.09
68	28.22	10.78	59.52	1.48	13.82	11.43	73.90	0.86
78	13.18	15.68	70.15	0.99	35.23	13.41	48.10	3.26
14	49.44	12.06	29.04	9.46	57.24	9.69	21.24	11.83
20	35.03	19.88	35.45	9.63	45.98	16.11	24.50	13.40
35	9.00	14.79	75.15	1.06	2.44	15.65	81.71	0.20
49	7.44	13.20	78.97	0.39	6.81	13.41	79.60	0.18
62	10.29	13.36	75.58	0.76	2.28	13.97	83.60	0.16
Average	18.55	14.47	62.47	4.51	20.56	14.39	60.46	4.60
Std Dev	16.31	2.73	21.19	6.98	22.01	2.90	27.49	6.47

NP: No precipitation reported
P: Precipitation reported
NP->NPP: No precipitation predicted when No precipitation was reported
P->PP: Precipitation predicted when precipitation was reported
P->NPP: No precipitation predicted when precipitation was reported
NP->P: Precipitation predicted when precipitation was reported

Percentage of instances predicted correctly using the naïve Bayes and Bayes net classification algorithms to predict precipitation type at an RWIS site. (see Figure 15)

		ML Algorithms			
	RWIS Site	LMS	LR	M5P	MLP
Set 1	67	1.6519	1.7436	1.6584	1.8020
Set 2	35	0.0785	0.1204	0.1024	0.1838
	49	0.0491	0.0756	0.0613	0.0643
	62	0.0511	0.0785	0.0618	0.0716
Set 3	56	0.0456	0.0722	0.0573	0.0975
	68	0.0160	0.0296	0.0244	0.0478
	78	0.1791	0.2081	0.0845	0.1545

Results obtained from using regression algorithms to predict visibility at an RWIS site (see Figure 16). Input data set consists of temperature information from RWIS-AWIS sites in a set along with precipitation type and visibility for the RWIS sites. The table has mean absolute error values averaged over ten 10-fold cross-validations.



Mean absolute errors for different RWIS sites obtained from predicting visibility using regression algorithms.

