# Prediction of Methane Outbreak in Coal Mines from Historical Sensor Data under Distribution Drift

Marc Boullé

Orange Labs,
2 avenue Pierre Marzin,
22300 Lannion, France
marc.boulle@orange.com
http://www.marc-boulle.fr

**Abstract.** We describe our submission to the IJCRS'15 Data Mining Competition, where the objective is to predict methane outbreaks from multiple sensor readings. Our solution exploits a selective naive Bayes classifier, with optimal preprocessing, variable selection and model averaging, together with an automatic variable construction method that builds many variables from time series records. One challenging part of the challenge is that the input variables are not independent and identically distributed (i.i.d.) between the train and test datasets, since the train data and test data rely on different time periods. We suggest a methodology to alleviate this problem, that enabled to get a final score of 0.9439 (team marcb), second among the 50 challenge competitors .

**Keywords:** Multi-relational data mining, supervised classification, feature selection, drift detection

## 1 Introduction

The IJCRS'15 Data Mining Competition[1] is related to a problem of prediction of methane outbreaks in a coal mine. The coal mine is equipped with 28 sensors of different types (barometer, anemometer, temperature meter, humidity meter, methane meter...). Sensor readings are available as time series for time periods of 10 minutes long with measurements taken every second. The train data consists of 51,700 samples (time periods of 10 minutes long), whereas the test data contains 5,076 samples with time periods that do not overlap with those in the train data. The objective is to predict whether a warning threshold has been reached in a delay between three and six minutes after the end of the time period, for three methane meters. The evaluation criterion is the mean AUC of the three target classes. In this paper, we present our submission to the challenge. It exploits a Selective Naive Bayes classifier together with an automatic variable construction method (Section 2). We motivate the choice of this classification

[1] https://knowledgepit.fedcsis.org/contest/view.php?id=109

framework and describe its application to the challenge in Section 3. A good classifier trained on the train data obtained a poor leaderboard score. This is not caused by over-fitting, but by a severe distribution drift between the train and test data. We suggest in Section 4 a methodology to alleviate this problem. Finally, Section 5 summarizes the paper.

## 2 Supervised Classification Framework

We summarize the Selective Naive Bayes (SNB) classifier introduced in [4]. It extends the Naive Bayes classifier [16] using an optimal estimation of the class conditional probabilities, a Bayesian variable selection and a Compression-based Model Averaging. We also describe the automatic variable construction framework presented in [5], used to get a tabular representation from the times series.

### 2.1 Optimal Discretization

The Naive Bayes (NB) classifier has proved to be very effective in many real data applications [16, 10]. It is based on the assumption that the variables are independent within each class, and solely relies on the estimation of univariate conditional probabilities. The evaluation of these probabilities for numerical variables has already been discussed in the literature [8, 18]. Experiments demonstrate that even a simple equal width discretization brings superior performance compared to the assumption using a Gaussian distribution per class. Using a discretization method, each numerical variable is recoded as a categorical variable, with a distinct value per interval. Class conditional probabilities are assumed to be piecewise constant per interval, and obtained by counting the number of instances per class in each interval. These class conditional probabilities are used as inputs for the naive Bayes classifier.

In the MODL approach [3], the discretization is turned into a model selection problem and solved in a Bayesian way. First, a space of discretization models is defined. The parameters of a specific discretization model $M$ are the number of intervals, the bounds of the intervals and the class frequencies in each interval. Then, a prior distribution is proposed on this model space. This prior exploits the hierarchy of the parameters: the number of intervals is first chosen, then the bounds of the intervals and finally the class frequencies. The choice is uniform at each stage of the hierarchy. Finally, the multinomial distributions of the class values in each interval are assumed to be independent from each other. A Bayesian approach is applied to select the best discretization model, which is found by maximizing the maximum a posteriori (MAP) model. Owing to the definition of the model space and its prior distribution, the Bayes formula is applicable to derive an exact analytical criterion to evaluate the posterior probability of a discretization model. The optimized criterion is $p(M)p(D|M)$, where $p(M)$ is the prior probability of a preprocessing model and $p(D|M)$ the conditional likelihood of the data given the model.

Efficient search heuristics allow to find the most probable discretization given the data sample. Extensive comparative experiments report high performance.

*Univariate informativeness evaluation* A 0-1 normalized version of the optimized criterion provides a univariate informativeness evaluation of each input variable. Taking the negative log of the MAP criterion, $c(M) = -(\log p(M) + \log p(D|M))$, the approach receives a Minimim Description Length (MDL) [21] interpretation, where the objective is to minimize the coding length of the model plus that of the data given the model. The null model $M_\emptyset$ is the preprocessing model with one single interval, which represents the case with no correlation between the input and output variables. We then introduce the *I(V)* criterion in Equation 1 to evaluate the informativeness of a variable $V$.

$$I(V) = 1 - \frac{c(M)}{c(M_\emptyset)}. \tag{1}$$

The value of $I(V)$ grows with the informativeness of an input variable. It is a between 0 and 1, 0 for irrelevant variables uncorrelated with the target variable and 1 for variables that perfectly separate the target values.

### 2.2 Bayesian Approach for Variable Selection

The naive independence assumption can harm the performance when violated. In order to better deal with highly correlated variables, the Selective Naive Bayes approach [17] exploits a wrapper approach [13] to select the subset of variables which optimizes the classification accuracy. Although the Selective Naive Bayes approach performs quite well on datasets with a reasonable number of variables, it does not scale on very large datasets with hundreds of thousands of instances and thousands of variables, such as in marketing applications or text mining. The problem comes both from the search algorithm, whose complexity is quadratic in the number of variables, and from the selection process which is prone to overfitting. In [4], the overfitting problem is tackled by relying on a Bayesian approach, where the best model is found by maximizing the probability of the model given the data. The parameters of a variable selection model are the number of selected variables and the subset of variables. A hierarchic prior is considered, by first choosing the number of selected variables and second choosing the subset of selected variables. The conditional likelihood of the models exploits the Naive Bayes assumption, which directly provides the conditional probability of each class. This allows an exact calculation of the posterior probability of the models. Efficient search heuristic with super-linear computation time are proposed, on the basis of greedy forward addition and backward elimination of variables.

### 2.3 Compression-Based Model Averaging

Model averaging has been successfully exploited in bagging [6] using multiple classifiers trained from re-sampled datasets. In this approach, the averaged classifier uses a voting rule to classify new instances. Unlike this approach, where each classifier has the same weight, the Bayesian Model Averaging (BMA) approach [11] weights the classifiers according to their posterior probability. In the

case of the Selective Naive Bayes classifier, an inspection of the optimized models reveals that their posterior distribution is so sharply peaked that averaging them according to the BMA approach almost reduces to the MAP model. In this situation, averaging is useless. In order to find a trade-off between equal weights as in bagging and extremely unbalanced weights as in the BMA approach, a logarithmic smoothing of the posterior distribution, called Compression-based Model Averaging (CMA), is introduced in [4]. The weighting scheme on the models reduces to a weighting scheme on the variables, and finally results in a single Naive Bayes classifier with weights per variable. Extensive experiments demonstrate that the resulting Compression-based Model Averaging scheme clearly outperforms the Bayesian Model Averaging scheme. In the rest of the paper, the classifier resulting from model averaging is called Selective Naive Bayes (SNB).

### 2.4 Automatic Variable Construction for Multi-Table

In a data mining project, the data preparation phase aims at constructing a data table for the modeling phase [20, 7]. The data preparation is both time consuming and critical for the quality of the mining results. It mainly consists in the search of an effective data representation, based on variable construction and selection. Variable construction [19] has been less studied than variable selection [9] in the literature. However, learning from relational data has recently received an increasing attention. The term Multi-Relational Data Mining (MRDM) was initially introduced in [12] to address novel knowledge discovery techniques from multiple relational tables. The common point between these techniques is that they need to transform the relational representation. Methods named by propositionalisation [14, 15, 1] try to flatten the relational data by constructing new variables that aggregate the information contained in non target tables in order to obtain a classical tabular format.

In [5], an automatic variable construction method is proposed for supervised learning, in the multi-relational setting using a propositionalisation-based approach. Domain knowledge is specified by describing the multi-table structure of the data and choosing construction rules. The formal description of the data structure relies on a root table that contains the main statistical units and secondary tables in 0 to 1 or 0 to n relationship with the root table. For example, Figure 1 describes the structure of the data for the challenge. The construction rules available for automatic construction of variables are detailed below:

- *Selection(Table, Num)→Table*: selection of records from a secondary table according to a conjunction of selection terms (membership in a numerical interval of a variable *Num* in the secondary table),
- *Count(Table)→Num*: count of records in a table,
- *Mean(Table, Num)→Num*: mean value of variable *Num*,
- *Median(Table, Num)→Num*: median value,
- *Min(Table, Num)→Num*: min value,
- *Max(Table, Num)→Num*: max value,
- *StdDev(Table, Num)→Num*: standard deviation,

– *Sum(Table, Num)→Num*: sum of values.

The space of variables that can be constructed is virtually infinite, which raises both combinatorial and over-fitting problems. When the number of constructed variables increases, the chance for a variable to be wrongly considered as informative becomes critical. A prior distribution over all the constructed variables is introduced. This provides a Bayesian regularization of the constructed variables, which allows to penalize the most *complex* variables. An effective algorithm is introduced as well to draw samples of constructed variables from this prior distribution. Experiments show that the approach is robust and efficient.

## 3 Applying the Framework for the Challenge

We motivate our choice of the classification framework[2], then describe how we apply it on the challenge dataset.

### 3.1 Choice of the Classification Framework

In all our challenge submissions, we exploit the framework described in Section 2 to train a selective naive Bayes classifier, with optimal discretization, variable selection and model averaging. The classifier is trained on a flat data representation, obtained using the automatic variable construction method (Section 2.4) that builds many variables from the time series records data. Once the data schema is specified, the only parameter is the number of variables to construct. The method is fully automatic, scalable and highly robust, with test performance mainly equivalent to train performance.

The SNB classifier is resilient to noise and to redundancies between the input variables, but it is blind to non-trivial interactions between the variables. This can be leveraged by feature engineering, relying on domain expertise rather than on statistical expertise. More accurate classification methods are available, such as random forests, gradient boosting methods, support vector machines or neural networks. However, these methods require intensive feature engineering to get a flat input data table representation, are prone to over-fitting, are mainly black-box, not suitable for an easy interpretation of the models and finally require fine parameter tuning, both time consuming and expertise intensive. In an industrial context like the Orange telecommunication operator, the major issue is to quickly provide an accurate, robust and interpretable solution to many data mining problems, rather than a very accurate solution to few problems. In this context, the generic framework described in Section 2 and used in this challenge offers a good solution.

### 3.2 Application to the Challenge Dataset

For the IJCRS'15 Data Mining Competition, coal mines are described using a root table that contains the three class variables and a secondary table for the

[2] Available as a shareware at http://www.khiops.com

sensor readings. An identifier variable *Id* is added in each record of both tables, to enable the join between the root and secondary tables.

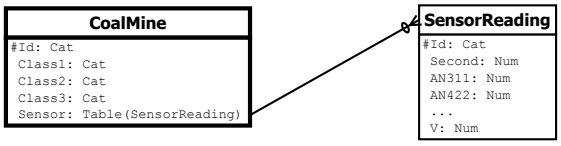



**Fig. 1.** Multi-table representation for the data of the IJCRS'15 challenge

The multi-table representation of the challenge data is presented in Figure 1. The root table (*CoalMine*) contains 51,700 train instances, with 4 variables: *Id* and the three class variables. The secondary table (*SensorReading*) contains 51,700 $\times$ 600 records, with 30 variables: *Id* as a join key, the time variable *Second* and the 28 time series variables for the sensor readings (*AN311*, *AN422*, ... , *V*). Using the data structure presented in Figure 1 and the construction rules introduced in Section 2.4, one can for example construct the following variables ("name" = *formula*: comment) to enrich the description of a *CoalMine*:

- "Mean(Sensor.TP1721)" =
  *Mean*(*Sensor*, *TP1721*):
  mean of the temperature sensor TP1721 readings,
- "Count(Sensors) where AN422 $\in ]1.75, 1.85]$" =
  *Count*(*Selection*(*Sensor*, *AN422* $\in ]1.75, 1.85]$)):
  number of sensor readings where the anemometer AN422 value is between 1.75 and 1.85,
- "Max(Sensors.MM263) where Second $> 300$" =
  *Max*(*Selection*(*Sensor*, *Second* $> 300$), *MM263*):
  max of the methane meter sensor MM263 readings in the last five minutes.

The number of variables to construct is the only user parameter. An input flat data table representation is then obtained from the set of all automatically constructed variables. All these variables are then preprocesses using the optimal discretization method (cf. Section2.1) to assess their informativeness and evaluate their class conditional probabilities, before training the SNB classifier.

## 4 Challenge Submissions

In this section, we describe our submissions to the challenge and suggest a methodology to alleviate the problem of the drift between the train and test distributions of the challenge dataset.

### 4.1 Preliminary Trials

To get familiar with the challenge evaluation protocol, we made preliminary trials, using all the sensor variables and constructing 100, 1000 and 10000 variables to summarize the sensors times series. We collect in Table 1 the train and test AUC (averaged over the three target classes) using a 70%-30% split of the train dataset as well as the leaderboard score obtained using the corresponding submissions.

**Table 1.** Performance per number of constructed variables

| Variables | Train AUC | Test AUC | Leaderboard score |
|---|---|---|---|
| 100 | 0.9760 | 0.9684 | 0.8067 |
| 1000 | 0.9835 | 0.9766 | 0.7419 |
| 10000 | 0.9885 | 0.9802 | 0.7876 |

We obtained surprisingly robust and high train AUC. With only 100 variables constructed from the 28 sensors, the train AUC is about 0.97, with less than 1% difference between the train and test splits. However, these promising performance dropped down on the challenge leaderboard, with a non monotonous behavior w.r.t. the number of constructed variables. This large drop of performance was not cause by over-fitting, but by a drift between the train and test (based on two distinct time periods).

### 4.2 A Methodology to Reduce the Drift Problem

Let us consider two tasks: classification of the methane outbreaks and detection of the drift. The drift detection task can be turned into a classification task as in [2], by merging the train and test datasets and using the dataset label ('train' or 'test') as the target variable. Using the initial input representation (cf. Section 4.1) with 10000 constructed variables, the drift detection task achieved an almost perfect performance with an AUC of 0.9999. This means that the train and test distributions can be well separated using the sensors data. As the data is not i.i.d, obtaining good classification performance on the train data does not guarantee good performance on the test data. Intuitively, if we are able to select an input representation with good classification performance on the train data but poor drift detection, we expect that our classifier will be less sensitive to drift and its performance drop on the test dataset will be reduced.

The objective is then to explore varying input representations and select the one with the best classification performance together with the poorest drift detection. To do so, we represent in Figure 2 the informativeness (cf. Formula 1) of the 10,000 constructed variables for the classification and drift detection tasks. The results show that there are variables with large drift informativeness and small classification informativeness (top-left of the figure), or on the contrary variables with small drift informativeness and large classification informativeness

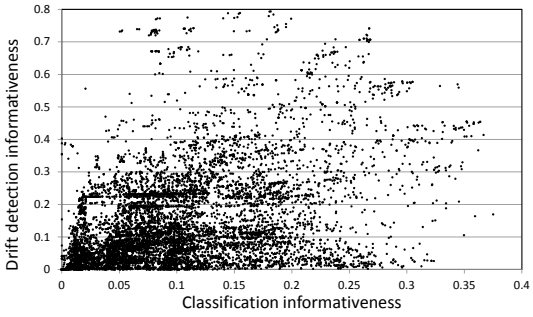



**Fig. 2.** Informativeness of 10,000 variables

(bottom-right). The interesting variables are those on the right and close to the X axis, with small drift informativeness.

To gain further insights, we collect the mean informativeness (cf. formula 1) per input sensor (gathering all constructed variables involving each sensor) and per target class.

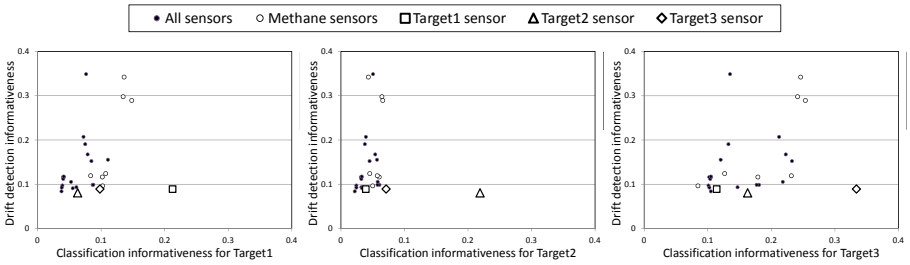



**Fig. 3.** Mean informativeness per sensor for each target

The results, shown in Figure 3, suggest to consider the following subsets of variables, by decreasing number of variables with high drift informativeness.

1. all the 28 input sensors,
2. only the 11 sensors related to methane,
3. only the 3 sensors related to the three target classes,
4. only one sensor per target class.

We then build classifiers using only 100 constructed variables (which seems enough from Section 4.1), and obtain the classification and drift detection train AUC as well as the leaderboard scores, reported in Table 2.

Table 2 shows that the drift detection AUC rapidly decreases with smaller number of sensors while the train classification AUC remains almost the same. Meanwhile, the leaderboard score increases, from 0.8067 using all the sensors to 0.9304 when only the target sensor is used for the prediction.

**Table 2.** Performance per number of input sensors

| Input sensors | Drift AUC | Train AUC | Leaderboard AUC |
|---|---|---|---|
| 28 | 0.9998 | 0.9684 | 0.8067 |
| 11 | 0.9996 | 0.9723 | 0.8978 |
| 3 | 0.7625 | 0.9742 | 0.9225 |
| 1 | 0.6210 | 0.9675 | 0.9304 |

### 4.3 Simplification of the Solution

As only 100 variables and one single sensor per target variable where enough to get a good leaderboard score, we made several trials and errors to simplify the solution, improve its interpretability and performance. We finally kept four time periods (full 10 minutes period, last 5 minutes, last 2 minutes 30 seconds, and last minute) and two construction rules (*Mean* and *Max*), representing the input data by only 8 variables per target class.

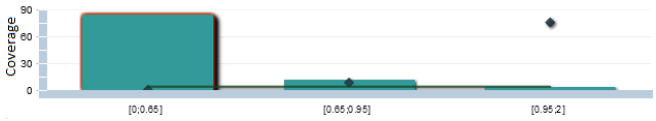
### 4.4 Further Improvement

According to the challenge organizers, the time periods in the training data are overlapping and are given in a chronological order. This raises an additional issue, where instances in short time periods are non i.i.d. and over-sampled: this might cause additional over-fitting. Inspecting the data, we estimated that the over-sampling factor was about 10, and decided to train again the solution described in Section 4.3, using 10% of the train data. To improve the robustness, we divided the train data into 10 folds, trained the solution on each fold and averaged the predictions. Using this method, our final chosen submission obtained a leaderboard score of 0.9461, very close to our final score of 0.9439.

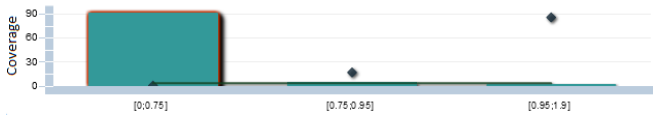
Interestingly, many time series problems suffer from the two same kind of problems: different time periods that cause drift between the distributions in case of non stationary data, and over-sampled data when the sampling rate is large compared to the typical size of the time windows that govern the behavior of the time series. For example, for climate time series (temperature, humidity, pressure...), different data collection periods (e.g. winter and summer) involve distinct distributions of the data. And for a given period, a dataset with one record every second is clearly over-sampled, compared to the typical change rate in climate times series. This results in dramatic over-fitting, since predicting future value to be the same as the last past value is likely to be very accurate for a prediction windows of one second, but valueless for longer windows.

### 4.5 Insights on Relevant Variables

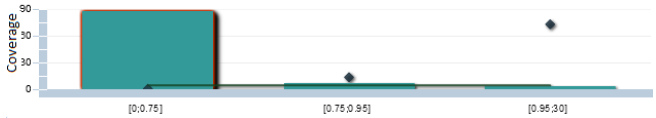
The IJCRS'2015 challenge comes with abundant data: 28 sensors with records every seconds during 10 minutes, which amounts to 28*600=16,800 values per train instance. Remarkably, our final solution exploits only 8 variables per class: mean and max of the target methane meter readings during the last 10mins, 5mins, 2mins 30secs and last min.



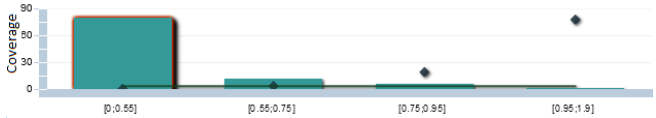

Max of methane meter MM264 in last 5mins, $I(V) = 0.4561$



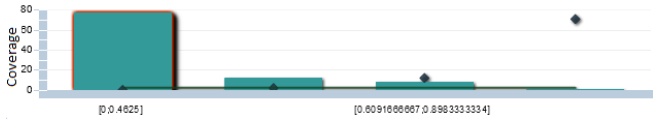

Max of methane meter MM264 in last 2mins 30secs, $I(V) = 0.4367$



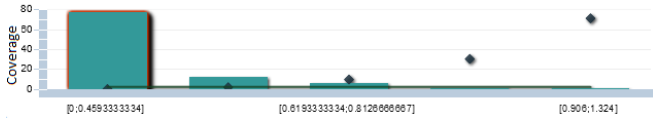

Max of methane meter MM264 in last 10min, $I(V) = 0.4252$



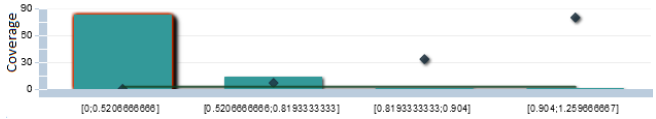

Max of methane meter MM264 in last min, $I(V) = 0.4115$



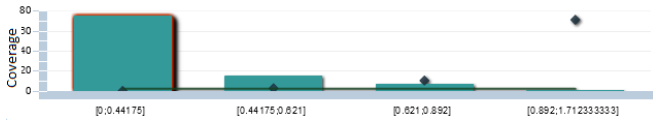

Mean of methane meter MM664 in last min, $I(V) = 0.4105$




Mean of methane meter MM664 in last 2mins 30secs, $I(V) = 0.4074$




Mean of methane meter MM664 in last 5mins, $I(V) = 0.4038$




Mean of methane meter MM664 in last 10mins, $I(V) = 0.3772$

**Fig. 4.** Optimal discretizations of the input variables for the second class

To get further insights on these variables, we inspect their discretizations, obtained from the first fold in our ten fold process described in Section 4.4. The overall behavior is of the same kind for the three classes, with slight differences. The variables are all discretized into 3 intervals for the first class, into 3 to 5 intervals for the second class, and into 3 or 4 intervals for the third class. Figure 4 displays the optimal discretizations for the 8 variables related to the second class (methane meter *MM264*, with two class values: *normal* and *warning*). For example, the first histogram is related to the variable "Max of methane meter MM264 in last 5mins", which have the highest informativeness ($I(V) = 0.4561$) among the 8 variables. The three intervals cover respectively 85%, 12% and 3% of the instances. The overall mean coverage of the minority class value *warning* is represented by the horizontal line (2.6%). The coverage of *warning* per interval is represented by the diamonds: 0.16% in the first interval, 6% in the second one and 54% in the last one.

The optimal discretizations consist of 3 to 5 intervals, but they all represent the same kind of information. Most of the instances belong to the first interval, with very small value of the input variable and very small proportion of the *warning* target value. The second interval comes beyond a first threshold (typically between 0.5 and 0.75) and the last one is generally beyond a threshold of about 0.9. The proportion of *warning* quickly grows with the value of input variable, with between 40% and 80% of *warning* in the last interval.

Overall, the probability of a methane outbreak in the next few minutes quickly increases with the value of the related methane meter in the last minutes. This behavior, represented for the second class, is the same for the other classes (not displayed in this paper). The methodology presented in this paper allowed to retrieve this interpretable behavior and to quantify it precisely.

## 5 Conclusion

Whereas most data mining methods rely on i.i.d. data, this is not the case in IJCRS'15 Data Mining Competition, where the train and test data where collected from two different time periods. In this case, a robust classifier was able to achieve 0.98 AUC in a 70%-30% split of the train data, with a severe drop of the test performance down to 0.80. This is not an overfitting problem, but a problem of distribution drift between the train and test data. In this paper, we have suggested a methodology to alleviate this problem by evaluating the informativeness of each variable for the classification and drift detection tasks. We follow the intuition that the classifiers that exploit input variables with high class informativeness and low drift informativeness are more likely to be resilient to drift. We explored several axis for choosing representations that are robust to drift: selection of sensors, selection of construction rules that summarize sensor readings and number of constructed variables. In the end, we kept only one sensor per target class, summarized by 8 input variables. We were then able to build a classifier with 0.9439 final score, which is a large improvement compared to our initial solution.

## References


1. Blockeel, H., De Raedt, L., Ramon, J.: Top-Down Induction of Clustering Trees. In: Proceedings of the Fifteenth International Conference on Machine Learning. pp. 55–63. Morgan Kaufmann (1998)
2. Bondu, A., Boullé, M.: A supervised approach for change detection in data streams. In: Proceedings of International Joint Conference on Neural Networks. pp. 519–526 (2011)
3. Boullé, M.: MODL: a Bayes optimal discretization method for continuous attributes. Machine Learning 65(1), 131–165 (2006)
4. Boullé, M.: Compression-based averaging of selective naive Bayes classifiers. Journal of Machine Learning Research 8, 1659–1685 (2007)
5. Boullé, M.: Towards automatic feature construction for supervised classification. In: ECML/PKDD 2014. pp. 181–196. Springer-Verlag (2014)
6. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
7. Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R.: CRISP-DM 1.0 : step-by-step data mining guide. Tech. rep., The CRISP-DM consortium (2000)
8. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: Proceedings of the 12th International Conference on Machine Learning. pp. 194–202. Morgan Kaufmann, San Francisco, CA (1995)
9. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L. (eds.): Feature Extraction: Foundations And Applications. Springer (2006)
10. Hand, D., Yu, K.: Idiot's bayes ? not so stupid after all? International Statistical Review 69(3), 385–399 (2001)
11. Hoeting, J., Madigan, D., Raftery, A., Volinsky, C.: Bayesian model averaging: A tutorial. Statistical Science 14(4), 382–417 (1999)
12. Knobbe, A.J., Blockeel, H., Siebes, A., Van Der Wallen, D.: Multi-Relational Data Mining. In: Proceedings of Benelearn '99 (1999)
13. Kohavi, R., John, G.: Wrappers for feature selection. Artificial Intelligence 97(1-2), 273–324 (1997)
14. Kramer, S., Flach, P.A., Lavrač, N.: Propositionalization approaches to relational data mining. In: Džeroski, S., Lavrač, N. (eds.) Relational data mining, chap. 11, pp. 262–286. Springer-Verlag (2001)
15. Krogel, M.A., Wrobel, S.: Transformation-based learning using multirelational aggregation. In: ILP. pp. 142–155. Springer (2001)
16. Langley, P., Iba, W., Thompson, K.: An analysis of Bayesian classifiers. In: 10th National Conference on Artificial Intelligence. pp. 223–228. AAAI Press (1992)
17. Langley, P., Sage, S.: Induction of selective Bayesian classifiers. In: Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence. pp. 399–406. Morgan Kaufmann (1994)
18. Liu, H., Hussain, F., Tan, C., Dash, M.: Discretization: An enabling technique. Data Mining and Knowledge Discovery 4(6), 393–423 (2002)
19. Liu, H., Motoda, H.: Feature Extraction, Construction and Selection: A Data Mining Perspective. Kluwer Academic Publishers (1998)
20. Pyle, D.: Data preparation for data mining. Morgan Kaufmann Publishers, Inc. San Francisco, USA (1999)
21. Rissanen, J.: Modeling by shortest data description. Automatica 14, 465–471 (1978)