Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 389–396

DOI: 10.15439/2015F423
ACSIS, Vol. 5

Tagging Fireworkers Activities from Body Sensors under Distribution Drift

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

***Abstract*—We describe our submission to the AAIA'15 Data Mining Competition, where the objective is to tag the activity of firefighters based on vital functions and movement 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. The most 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. We suggest a methodology to alleviate this problem, that enabled to get a final score of 0.76 (team marcb).**

I. Introduction

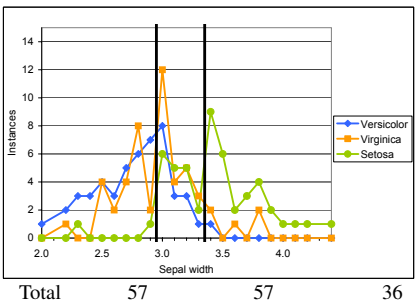
The AAIA'15 Data Mining Competition [1] is related to a problem of activity tagging. Firefighters are equipped with body sensors that register vital functions and movements. Vital function records are summarized by statistics (minimum, maximum, median...) using a fixed number of input variables, whereas movement records are available as 42 times series of length 1.8 s with measures every 4.5 ms. Train data consists of 20,000 samples for activities recorded from four firefighters, whereas the test data contains 20,000 samples coming from a different group of four firefighters. The objective is to tag the activity of firemen, among 24 activities, and the evaluation criterion is the balanced accuracy (BAC). 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 II). We motivate the choice of this classification framework and describe its application to the challenge in Section III. A good classifier trained on the train data obtains a disastrous 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 IV a methodology to alleviate this problem. In Section V, we present related work and discuss our approach. Finally, Section VI summarizes the paper.

II. Supervised Classification Framework

We summarize the Selective Naive Bayes (SNB) classifier introduced in [2]. It extends the Naive Bayes classifier [3] owing to 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 [4], used to get a tabular representation from times series.

A. Optimal discretization

The Naive Bayes (NB) classifier has proved to be very effective in many real data applications [3], [5]. 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 [6], [7]. 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. Figure 1 shows an example of discretization into three intervals, for the *Sepal width* input variable of the Iris dataset [8].



Total	57	57	36
Versicolor	34	15	1
Virginica	21	24	5
Setosa	2	18	30
	]-∞, 2.95[	[2.95, 3.35[	[3.35, +∞[

Fig. 1. Number of instances per class in the Iris dataset, for a discretization of the *Sepal width* input variable into three intervals

978-83-60810-66-8/$25.00

In the MODL approach [9], 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) [10] 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.

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

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.

B. 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 [11] exploits a wrapper approach [12] 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 [2], 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.

C. Compression-Based Model Averaging

Model averaging has been successfully exploited in bagging [13] 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 [14] 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 [2]. 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).

D. 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 [15], [16]. 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 [17] has been less studied than variable selection [18] 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 [19] 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 [20], [21], [22] 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 [4], 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 several secondary tables in 0 to 1 or 0 to n relationship with the root table. For example, Figure 2 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 original or 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.

III. Applying the framework for the challenge

We motivate our choice of the classification framework[1], then describe how we apply it on the challenge dataset.

A. Choice of the classification framework

In all our challenge submissions, we exploit the framework described in Section II 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 II-D) 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 II and used in this challenge offers a good solution.

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

B. Application to the challenge dataset

For the AAIA'15 Data Mining Competition, firefighters are described using a root table that contains the target activity as well as the summary variables for the vital function sensors and a secondary table for the movement sensors readings. An identifier variable *Id* is added in each record of both tables, to enable the join between the root and secondary tables.

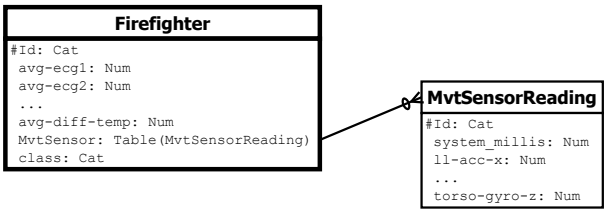


Fig. 2. Multi-table representation for the data of the AAIA'15 challenge

The multi-table representation of the challenge data is presented in Figure 2. The root table (*Firefighter*) contains 20,000 instances, with 44 variables: *Id*, the 42 vital function input variables (*avg-ecg1*, *avg-ecg2*, ... *avg-diff-temp*) and the class variable. The secondary table (*MvtSensorReading*) contains 20,000 $\times$ 400 records, with 44 variables: *Id* as a join key and the 43 time series variables for the movement sensor readings (*system_millis*, *ll-acc-x*, ... *torso-gyro-z*). Using the data structure presented in Figure 2 and the construction rules introduced in Section II-D, one can for example construct the following variables ("name" = *formula*: comment) to enrich the description of a *Firefighter*:

- "StdDev(MvtSensor.rl-acc-x)" = *StdDev(MvtSensor, rl-acc-x)*: standard deviation of rl-acc-x in the sensor readings,
- "Count(MvtSensor) where ll-gyro-y $>$ 60" = *Count(Selection(MvtSensor, ll-gyro-y $>$ 60))*: number of sensor readings where ll-gyro-y $>$ 0,
- "Max(MvtSensor.torso-acc-z) where torso-acc-x $>$ 5" = *Max(Selection(MvtSensor, torso-acc-x $>$ 5), torso-acc-z)*: max of torso-acc-z for sensor readings where torso-acc-x $>$ 5.

The number of variables to construct is the only user parameter. An input flat data table representation is then obtained from the initial input variables coming from the root table and the set of all automatically constructed variables. All these variables are then preprocesses using the optimal discretization method (cf. SectionII-A) to assess their informativeness and evaluate their class conditional probabilities, before training the SNB classifier. For advanced use, it is possible to impose a constraint on the granularity of the discretizations (cf. Section II-A): instead of obtaining the optimal number of intervals, the preprocessing method output discretizations with at most I_{Max} intervals, where I_{Max} is a user parameter.

IV. Challenge submission

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.

A. First trials

a) Submission 1: To get familiar with the challenge evaluation protocol, we made a first quick trial, using only the 43 vital function variables. We obtained a surprisingly high train accuracy, with few over-fitting: 0.9840 and 0.9680 on a 70%-30% split of the train dataset. However, our first submission obtained only a 0.1859 score on the challenge leaderboard. This dramatic drop of performance was not caused by over-fitting, but by a drift between the train and test distributions (based on two different groups of four firefighters).

b) Submission 2: We then generated 100 additional variables to summarize the movement sensors times series, using the framework described in Section II-D. As we observed that the optimal discretizations (see Section II-A) were very fine grained, with up to hundred of intervals, we decided to constrain the discretization method to build at most 10 intervals. We obtained a 0.9499 train accuracy (on the test split of the train dataset), and a 0.4566 leaderboard score. Constraining the discretisations thus reduced the drift effect.

c) Submission 3: Using discretizations with at most two intervals, we obtained a 0.8603 train accuracy and a 0.6372 leaderboard score. This confirmed the benefit of the constrained discretisations to reduce the drift effect.

d) Submission 4.: Still using discretizations with at most two intervals, we generated 1,000 variables from the movement sensors times series. We obtained a 0.9254 train accuracy and a 0.6951 leaderboard score.

Table I summarizes the performance obtained for each preliminary submission as well as the related user parameters: number of constructed variables (cf. Section II-D) and constrain on the maximum number of intervals in the discretizations (cf. variable preprocessing in SectionII-A).

These preliminary trials took only one hour and gave interesting insights.

TABLE I
Method parameters and performance per submission

Submission	Constructed variables	Interval max nb	Train accuracy	Leaderboard score
Submission 1	0		0.9680	0.1859
Submission 2	100	10	0.9499	0.4566
Submission 3	100	2	0.8603	0.6372
Submission 4	1000	2	0.9254	0.6951

B. Analysis, trials and errors

Let us consider two tasks: classification of the activity and detection of the drift. The drift detection task can be turned into a classification task as in [23], by merging the train and test datasets and using the dataset label ('train' or 'test') as the target variable. Intuitively, if we are able to select an input representation with good classification accuracy on the train dataset but poor drift detection, we expect that our classifier will be less sensitive to drift and that the 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 accuracy together with the poorest drift detection. To do so, we mainly considered the following dimensions: max number of intervals for discretizations, representation of times series, selection of variables, both for the root and secondary tables, choice of construction rules, number of constructed variables. Exploiting the informativeness of variables both from the classification and drift detection tasks, we made many trials and errors and finally obtained a solution with a leaderboard score of 0.7892. This solution mainly consists of:

1) discretizations with at most two intervals,
2) no use of any vital function variables, selection of part of the movement variables (mainly, the *acc-x* and *acc-z* for the *torso*, the *acc-x, acc-z, gyro-x, gyro-y* for the *legs* and *acc-x, gyro-x, gyro-z* for the *hands*),
3) construction of 10,000 variables using only the *Count* and *Selection* construction rules.

C. A methodology to reduce the drift problem

While the approach described in the preceding section was insightful and allowed to improve significantly the leaderboard score, it heavily relies on human expertise and is time consuming. We now suggest a methodology that aims at automatizing the approach. First, we use all the initial input representation and all the available construction rules to build 10,000 variables. Using discretizations with at most two intervals, we evaluate the informativeness of each input variable (cf. Formula 1), both for the drift detection and the classification tasks. The results, displayed in Figure 3, 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 (bottom-right). The interesting variables are those close to the X axis, with small drift informativeness.

We then sort the variables by increasing drift informativeness and select subsets of variables of increasing sizes, for a list of thresholds of drift informativeness (0, 0.0001, 0.00025...). Figure 4 displays the number of informative

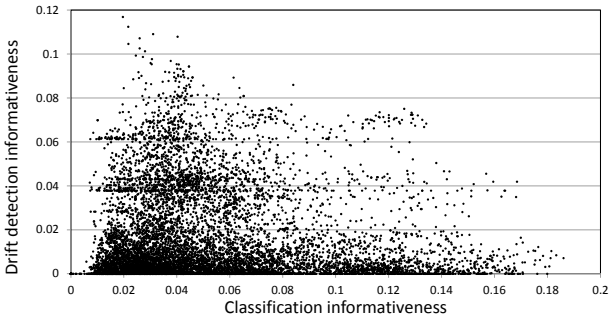


Fig. 3. Informativeness of 10,000 variables

variables for both the classification and detection tasks, for each drift threshold. For example, using a threshold of 0 that excludes any variable with drift informativeness, the smallest selected subset contains around 2,000 variables that are informative for the classification task.

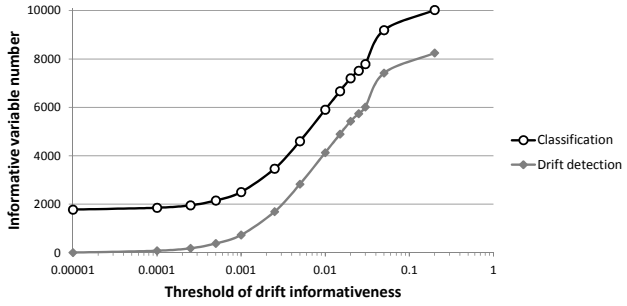


Fig. 4. Number of informative variables per drift threshold for the classification and drift detection tasks

For each of these subsets, we train the selective naive Bayes classifier both for the classification and drift detection tasks, collect the resulting accuracies as well as the leaderboard score obtained with the related submissions. For both tasks, we use a 70%-30% split of the data to evaluate the robustness of the classifier, and always observe a small difference ($\approx 1\%$) between the train and test performance.

Training times are more than ten times longer for the classification task (24 target values) than for the drift detection task (two labels). The largest classification task consists of 14,000 train instances (70%*20,000) and 10,000 constructed variables, each summarizing 400 movement sensor readings. On a PC Windows with Intel Xeon 2.3 Ghz processor, it took about half an hour for the data preparation and four hours to train the SNB classifier.

Figure 5 shows that the accuracy of drift detection rapidly decreases with smaller number of variables while the decrease is slower for the train classification accuracy. Meanwhile, the leaderboard score increases, from 0.7083 using all the variables to a plateau with a leaderboard score of about 0.78 between 2,500 and 6,000 variables . Using this methodology, our final chosen submission obtained a leaderboard score of

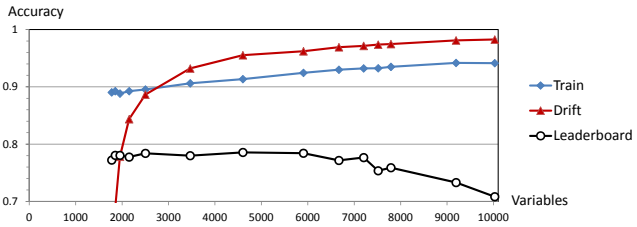


Fig. 5. Train classification, drift detection and leaderboard accuracies w.r.t. number of input variables

0.7856 (using around 4,500 variables) and a final score of 0.76.

D. Insights on relevant variables

Looking at Figure 4, we can see that the selected subset that contains around 4,500 variables is related to a drift threshold of 0.005, which is very small. Figure 3 shows that this corresponds to variables very close from the X axis, with potentially large class informativeness but very small drift detection informativeness.

For interpretability purposes, it is interesting to further investigate on which kind of variables are kept in the best classifier. In Figure 6, we reuse the same kind of plot as in Figure 3 and focus on the initial data representation, per family of variables. The 42 vital function variables are summarized in five families: EGC, heart rate, breath rate, respiration and temperature. The figure shows that these variables (especially the heart rate ones) have large drift informativeness and small classification informativeness. They are excluded from the best classifier. As for the 42 movement sensor time series, we divided the analysis using separate plots for the 7 body parts (torso, left and right hand, arm and legs) with 6 families per body part: x, y and z readings for the accelerometer and gyroscope. From the 10,000 automatically constructed variables, we collected the subset of variables related to each body part per family. The results are summarized in the 7 body part plots in Figure 6. This brings interesting insights w.r.t. to the relevance of each movement sensor for the classification task. For example, the arm sensors are the least interesting, since they are related to many variables with large drift informativeness. On the opposite, the other body part sensors are related to many constructed variable close to the X axis, with small drift informativeness and large classification informativeness. Overall, the torso and leg movement sensor bring the most useful information. The dissimilarity between the left and right body part sensors appears clearly. For example, for the left hand, the z accelerometer is too sensible to the drift, and for the left leg, this the case for the y accelerometer. All these insights may be useful to optimize data collection and pre-processing in order to improve the performance of the classifier.

E. Limits of the approach

Using the leaderboard to chose the best solution is likely to overfit the test dataset, and the solution might not be reliable

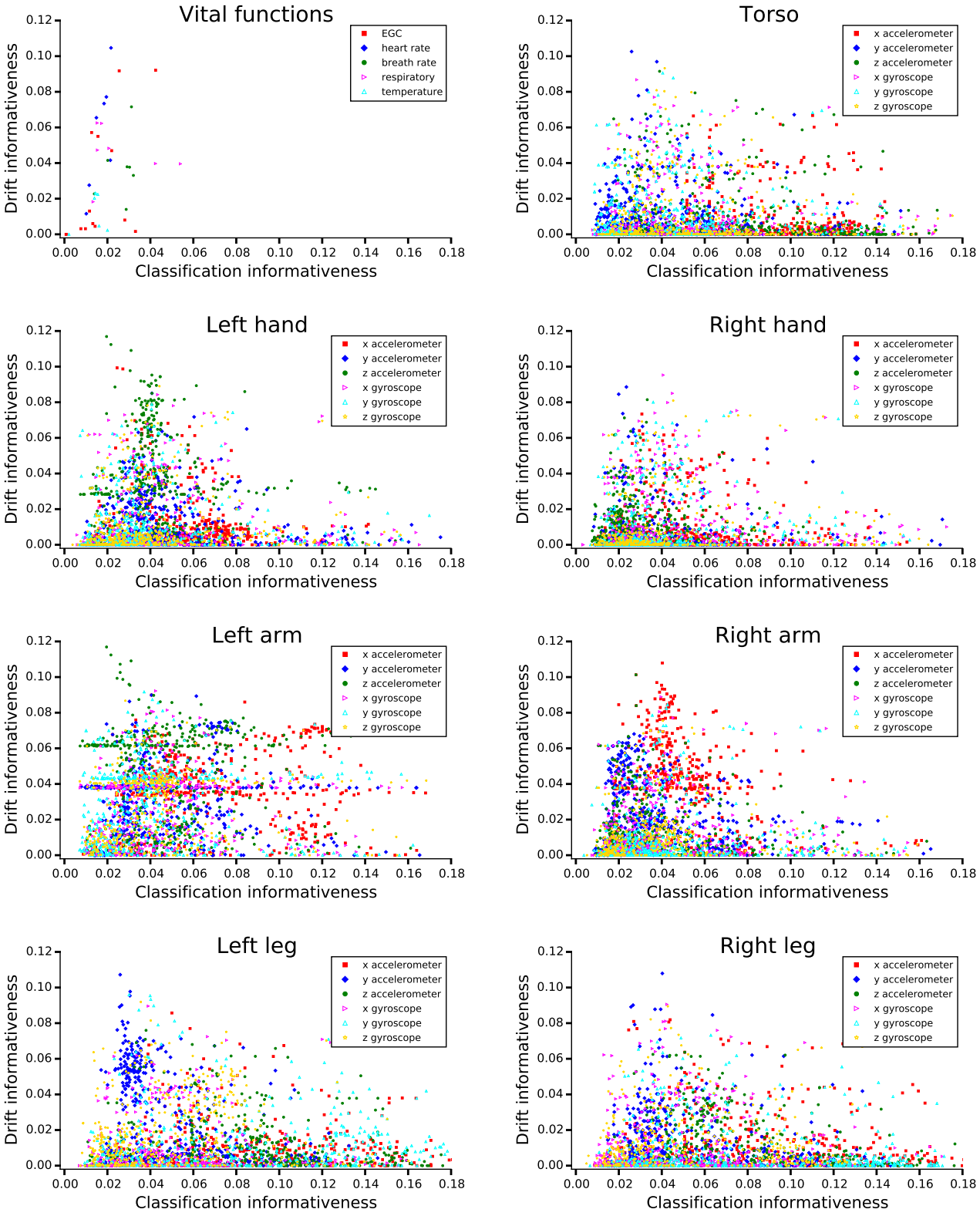


Fig. 6. Informativeness per family of variables

when applied to new firefighters not in the train nor test datasets. Ideally, data should be collected from more distinct firefighters to improve the reliability and performance of the solution. Also, the firefighter identifiers should be available in the datasets. This would enable a cross-validation process with splits based on distinct firefighters, to select the best solution using the train dataset only.

V. Related work and discussion

The challenge settings rely on a train and test datasets with the same task. As in supervised learning, labels are available in the train dataset and not available in the test dataset. However, the train and test data do not come from the same distribution (different firefighters in train and test).

In semi-supervised learning, [24] the objective is to exploit both labeled and unlabeled data, as in our approach, but the distributions of the labeled and unlabeled data are assumed to be the same. The case of transfer learning (see [25] for a comprehensive survey) is close to the challenge settings. Transfer learning aims at exploiting two source and target domains and tasks, with or without available labels for each task. It has been studied under different names (learning to learn, knowledge transfer, inductive transfer, multitask learning...) and covers a variety of settings. The closest one is called *Transductive Transfer Learning* (also named Domain Adaptation, Sample Selection Bias, Co-variate shift...), where the source and target domains are different but related, the tasks are the same, and the labels are available only in the source domain. In this setting, our approach is related to *Feature-Representation Transfer* approaches, where the objective is to find a good feature representation that reduces the difference between the source and target domains and improves the accuracy for the target task. According to Pan and Yang [25], most feature-representation transfer approaches to the transductive transfer learning settings are under unsupervised learning frameworks. In structural correspondence learning (SCL) [26], a set of domain specific *pivot* features is defined and treated as a new label vector. The corresponding classification problems are assumed to be solved by linear classifiers. SCL then learns a matrix of parameters and applies a singular value decomposition on this matrix. This allows to create new features that encodes a correspondence between the source and target domains. The tricky part is how to well design the pivot features. This can be done heuristically or using mutual information [27]. Many approaches focus on the natural language processing (NLP) domain. In [28], a kernel-mapping function is proposed to map the data from both the source and target domains. However, the kernel is domain knowledge driven and is not easy to generalize to other applications. In [29], a co-clustering based approach is proposed to propagate the labels across the domains. In [30], an algorithm named *bridged refinement* is proposed to correct the labels predicted by a shift-unaware classifier toward a target distribution and take the mixture distribution of the training and test data as a bridge to better transfer from the training data to the test data. Other approaches summarized in [25] extend traditional approaches in the NLP domain, such as spectral analysis, probabilistic latent semantic analysis (PLSA) or dimensionality reduction.

Our approach is clearly related to the settings of Transductive Transfer Learning based on Feature-Representation Transfer. One main difference is that our method relies on a multi-table data representation of the data and exploits the automatic variable construction framework summarized in Section II-D to explore many representations. Still, in the flat table case, our approach could be applied in the NLP domain where many input variables are available using the bag of words representation of texts.

Compared to related transfer learning approaches, another difference is that our approach focuses on the methodology rather than on new specific modeling techniques. The unlabeled train and test data are first exploited jointly to sort the input variables by decreasing drift informativeness. Then, a list of embedded classifiers is build on subsets of variables of increasing size, with expected increasing classification performance but also increasing sensibility to drift. The final classifier can be found either using a tolerance threshold compared to the best classification performance in the source domain, or using labels in the target domain if they are partly available.

Let us now focus on some settings that could benefit from our approach or more generally from transfer learning:

- Like in the AAIA'15 Data Mining Competition, there are many problems where the train and test data are not governed by the same distributions, and where the task is to train a classifier from many train instances coming from few sources (like the many train samples coming from only four firefighters). This is the case in domains where data is hard to collect and relies on few volunteers, each contributing to several train instances. For example, the MNIST database of handwritten digits [31] contains 60,000 train instances and 10,000 test instances, with 250 train writers and 250 other test writers. The UCI repository [8] contains many other such datasets, including for example the *Australian Sign Language signs Data Set* with 95 signs were collected from five signers for a total of 6650 sign samples, or the *Spoken Arabic Digit Data Set* with 10 digits collected from 88 speakers for a total of 8800 samples (represented using timeseries of mel-frequency cepstrum coefficients).
- When the train and test data come from different time periods, the assumption of i.i.d. distribution is violated as soon as the data are not stationary. This applies to many times series prediction problems, where samples are collected from one single source in a train period and where the trained model is applied to a future test period. For example, the IJCRS'15 Data Mining Competition[2] is related to a problem of prediction of methane outbreaks in a coal mine equipped with 28 sensors of different types (barometer, anemometer, temperature meter, humidity

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

meter, methane meter...), with 51,700 train samples and 5,076 test samples with time periods that do not overlap with those in the train data.

- A variant of the preceding settings usually occurs in the marketing field, where there is abundant data from millions of customers, which allows extracting train datasets with one sample per customer. However, the marketing tasks are not classification, but prediction on a future period in a non-stationary market environment (for tasks such as churn, fraud or up-selling for example). In the Orange telecommunication operator, we have many such problems and plan to evaluate our transfer learning approach.

VI. Conclusion

Whereas most data mining methods rely on i.i.d. data, this is not the case in AAIA'15 Data Mining Competition, where the train and test data where collected from two different groups of four firefighters. In this case, a robust classifier was able to achieve 100% accuracy in a 70%-30% split of the train data, with a dramatic drop of the test performance down to 18% leaderboard score. This is not an overfitting problem, but a problem of distribution drift between the train and test datasets. 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. Applying this methodology, we were able to build a classifier with 0.76 final score, which is a tremendous improvement compared to our initial solution. In future work, we plan to refine the methodology and to evaluate it on new problems, in particular for Orange marketing prediction tasks, where the data are abundant, complex and governed by non stationary distributions.

References

[1] M. Meina, A. Janusz, K. Rykaczewski, D. Ślęzak, B. Celmer, and A. Krasuski, "Tagging firefighter activities at the emergency scene: Summary of AAIA'15 data mining competition at Knowledge Pit," in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems, M. Ganzha, L. Maciaszek, and M. Paprzycki, Eds., 2015,* in print September 2015.

[2] M. Boullé, "Compression-based averaging of selective naive Bayes classifiers," *Journal of Machine Learning Research,* vol. 8, pp. 1659–1685, 2007.

[3] P. Langley, W. Iba, and K. Thompson, "An analysis of Bayesian classifiers," in *10th National Conference on Artificial Intelligence.* AAAI Press, 1992, pp. 223–228.

[4] M. Boullé, "Towards automatic feature construction for supervised classication," in *ECML/PKDD 2014,* 2014, pp. 181–196.

[5] D. Hand and K. Yu, "Idiot's bayes? not so stupid after all?" *International Statistical Review,* vol. 69, no. 3, pp. 385–399, 2001.

[6] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features," in *Proceedings of the 12th International Conference on Machine Learning.* Morgan Kaufmann, San Francisco, CA, 1995, pp. 194–202.

[7] H. Liu, F. Hussain, C. Tan, and M. Dash, "Discretization: An enabling technique," *Data Mining and Knowledge Discovery,* vol. 4, no. 6, pp. 393–423, 2002.

[8] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[9] M. Boull´ e, "MODL: a Bayes optimal discretization method for continuous attributes," *Machine Learning,* vol. 65, no. 1, pp. 131–165, 2006.

[10] J. Rissanen, "Modeling by shortest data description," *Automatica,* vol. 14, pp. 465–471, 1978.

[11] P. Langley and S. Sage, "Induction of selective Bayesian classifiers," in *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence.* Morgan Kaufmann, 1994, pp. 399–406.

[12] R. Kohavi and G. John, "Wrappers for feature selection," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[13] L. Breiman, "Bagging predictors," *Machine Learning,* vol. 24, no. 2, pp. 123–140, 1996.

[14] J. Hoeting, D. Madigan, A. Raftery, and C. Volinsky, "Bayesian model averaging: A tutorial," *Statistical Science,* vol. 14, no. 4, pp. 382–417, 1999.

[15] D. Pyle, *Data preparation for data mining.* Morgan Kaufmann Publishers, Inc. San Francisco, USA, 1999.

[16] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, "CRISP-DM 1.0: step-by-step data mining guide," *The CRISP-DM consortium,* Tech. Rep., 2000.

[17] H. Liu and H. Motoda, *Feature Extraction, Construction and Selection: A Data Mining Perspective.* Kluwer Academic Publishers, 1998.

[18] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, Eds., *Feature Extraction: Foundations And Applications.* Springer, 2006.

[19] A. J. Knobbe, H. Blockeel, A. Siebes, and D. Van Der Wallen, "Multi-Relational Data Mining," in *Proceedings of Benelearn '99,* 1999.

[20] S. Kramer, P. A. Flach, and N. Lavrač, "Propositionalization approaches to relational data mining," in Relational data mining, S. Džeroski and N. Lavrač, Eds. Springer-Verlag, 2001, ch. 11, pp. 262–286.

[21] M.-A. Krogel and S. Wrobel, "Transformation-based learning using multirelational aggregation," in ILP. Springer, 2001, pp. 142–155.

[22] H. Blockeel, L. De Raedt, and J. Ramon, "Top-Down Induction of Clustering Trees," in *Proceedings of the Fifteenth International Conference on Machine Learning,* 1998, pp. 55–63.

[23] A. Bondu and M. Boullé, "A supervised approach for change detection in data streams," in *Proceedings of International Joint Conference on Neural Networks,* 2011.

[24] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning.* MIT Press, Cambridge, MA, 2006.

[25] S. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering,* vol. 22, no. 10, pp. 1345–1359, 2010.

[26] J. Blitzer, R. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," in *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing,* ser. EMNLP '06, 2006, pp. 120–128.

[27] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics,* 2007, pp. 440–447.

[28] H. Daum´ e III, "Frustratingly easy domain adaptation," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics,* 2007, pp. 256–263.

[29] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu, "Co-clustering based classification for out-of-domain documents," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* ser. KDD '07. ACM, 2007, pp. 210–219.

[30] D. Xing, W. Dai, G.-R. Xue, and Y. Yu, "Bridged Refinement for Transfer Learning," in *Proc. 11th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD),* ser. PKDD 2007. Springer Berlin / Heidelberg, 2007, pp. 324–335.

[31] Y. LeCun and C. Cortes, "The MNIST database of handwritten digits," 1998, http://yann.lecun.com/exdb/mnist/.