A method to build a representation using a classifier and its use in a K Nearest Neighbors-based deployment

Vincent Lemaire, Marc Boullé, Fabrice Clérot, Pascal Gouzien

***Abstract*— The K Nearest Neighbors (KNN) is strongly dependent on the quality of the distance metric used. For supervised classification problems, the aim of metric learning is to learn a distance metric for the input data space from a given collection of pair of similar/dissimilar points. A crucial point is the distance metric used to measure the closeness of instances. In the industrial context of this paper the key point is that a very interesting source of knowledge is available : a classifier to be deployed. The knowledge incorporated in this classifier is used to guide the choice (or the construction) of a distance adapted to the situation Then a KNN-based deployment is elaborated to speed up the deployment of the classifier compared to a direct deployment.**

I. Introduction

A. Industrial problem

Data mining consists in methods and techniques which allow the extraction of information and knowledge from data. Its use allows establishing correlations between data and, for example within the framework of customer relationship management, to define types of customer's behavior.

Given a database, one common task in data analysis is to find the relationships or correlations between a set of input or explanatory variables and one target variable. This knowledge extraction often goes through the building of a model which represents these relationships. Faced with a classification problem, a probabilist model estimates the probabilities of occurrence of each target class for all instances of the database given the values of the explanatory variables. These probabilities, or scores, can be used to evaluate existing policies. The scores are used for example in customer relationship to evaluate the probability that a customer will buy a new product (appetency) or resign a contract (churn). The scores are then exploited by the marketing services to personalize the customer relationship.

To produce scores, a predictive model (M) has to be applied for all instances using explanatory variables. To speed up this process, [1, 2] have proposed to build a table of paragons containing representative individuals. This table contains representative examples (customers) given the explanatory variables used by the predictive model. The paragons are connected by an 'Index Table' to all the population.

The Figure 1 describes the deployment process with and without the Index Table. In a classic deployment, without the Index Table, the deployment process includes two main steps : (i) the data table T1 ($K1$ instances represented by J explanatory variables) is extracted and (ii) the classifier is applied on all instances of this table. In the industrial platform [1], the input data from information system are structured and stored in a simple relational database (table on the left in Figure 1). The extraction consists in (i) the construction of explanatory variables from joints between different tables (in the relation database), (ii) the elaboration of a flat instance x variables representation. The explanatory variables are built and selected automatically for each specific marketing project. The deployment cost is the addition of the extraction cost (C_{e1}) and the classification cost (C_{c1}). If the classification problem is supposed to be stationary the classifier does not need to be trained again and can be deployed other times by repeating these two steps.

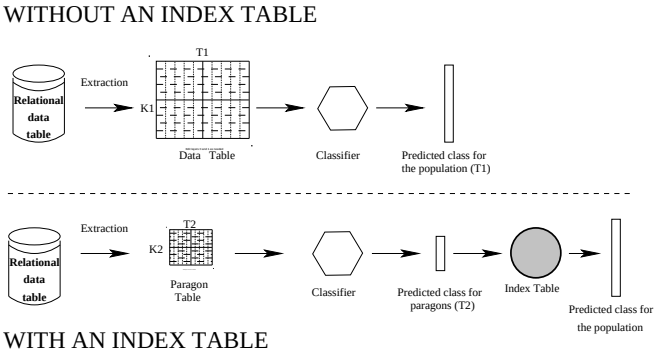

Fig. 1. Classifier deployment - With or without an Index Table

To decrease the cost of subsequent deployments, two tables are elaborated at the end of the first deployment : (i) the paragon table, T2 ($K2$ instances represented by J explanatory variables) using random sampling on T1 and (ii) the Index Table which contains for each instance of T1 its K nearest neighbors in T2 ; these two elaborations have respectively the costs C_{rs} and C_{id}.

The table of paragons is drawn from the data table (T1) to be representative of the variables relevant for the model. To produce and maintain online a sample of size n, Reservoir Sampling algorithm [3]) is used. The indexing task has to be executed for all the instances of the database. The search of nearest neighbors is an expensive operation. In order to accelerate the research of nearest neighbors Locality Sensitive Hashing [4] is used. Due to place considerations for more details on data extraction, variable construction and variable selection the reader can find in [2] a list of the workflow which briefly points out, for each step, which technique is applied.

Authors are in the group 'Profiling and Datamining', Orange Labs, 2 avenue Pierre Marzin, 22300 Lannion, France (phone : +33 296 053 107 ; email : firstname.name@orange-ftgroup.com).

For the second deployment, the classifier is deployed on T2 to obtain the scores of the paragons (with a cost C_{c2}). The scores of all instances (T1) are obtained by a simple join between the paragon table and the Index Table (with a cost C_{jo}) : each instance of T1 gets the score of its nearest neighbors in T2.

This method of deployment is particularly effective when the model is deployed several times. For example for monthly marketing campaigns, only the reduced table of paragons is built each month to produce the scores of all instances. This approach makes it possible to increase dramatically the number of scores that can be produced on the same technical architecture.

The gain is twofold : (1) in the extraction step (K2 $\ll$ K1), (2) in the deployment step since the joint between the paragon table and the Index Table is faster ($(C_{c2} + C_{jo}) \ll C_{c1}$) than the application of the classifier on all instances.

In this particular industrial framework, the key point consists in building the Index Table (grey disk in the Figure 1) which contains for each instance its nearest neighbor(s) in the paragon table. The problem is therefore the building of a KNN-based deployment (between $T1$ and $T2$) with a minimum loss of performance by comparison with the "direct" deployment (classifier applied to the total population, $T1$). To understand the paper it is important to note that the aim is not to elaborate a new classifier based on a KNN. The purpose is to elaborate a metric so that the deployment based on the Index Table realizes at best the deployment of the initial classifier (M).

B. KNN

The algorithm based on KNN comes from the family of lazy learners : contrary to many learning methods, there is no training step to determine parameters (except the value of K). Given a training set of instances correctly labeled and an integer K, the KNN classifier determines the label of a new instance (in a test set) by attributing it to the majority class of the K instances (in the training set) which are the most similar to it. A crucial point is the distance metric used to measure the closeness of instances. There is no universal distance metric and a good knowledge of the classification problem generally guides the choice of this distance.

But, in our industrial context, the key point is that we have a very interesting source of knowledge : the classifier to be deployed. In this article we show how the knowledge of the classification problem to be solved and the existence of the classifier to be deployed can guide the construction of a distance adapted to the situation. The proposed approach is to project the instances in the importance space of the explanatory variables of the classifier to be deployed, then to build a KNN on this new representation. We will show that using a sensitivity analysis, not dedicated to the classifier to be deployed, a representation based on the classifier is obtained, and that deployment is very efficient using this new representation for the KNN.

C. Outline

The paper is organized as follow :

- The Section II positions our approach. Learning metric and projections are briefly presented and the conclusion is that they are not adapted to our industrial problem. Therefore, a new approach is proposed : to project the instances in the importance space of the explanatory variables of the classifier to be deployed, then to build a KNN on this new representation
- The Section III details the three main methods, of the state of art, to realize this projection whatever is the classifier.
- Since in our industrial context our classifier is a naïve Bayes classifier, the Section IV applies these three methods for this particular type of classifier.
- The method is then tested in Section V on three real classification problems which correspond to our industrial framework. One shows that the representation based on variable importance using a sensitivity analysis gives similar performance than a representation based on variable importance using a method dedicated to the classifier. The results are also close to those obtained using the direct deployment.

II. Representation built using a classifier - Description and positioning

A. Learning metric

Many machine learning algorithms, such as K Nearest Neighbors (KNN), heavily rely on the distance metric for the input data patterns. The aim of distance Metric learning is to optimize a distance to separate a given collection of pairs of similar/dissimilar points. This metric has to preserve (resp. increase) the distance among the training data similar (resp. different). Many studies have demonstrated [5], both empirically and theoretically, that a learned metric can significantly improve the performance in classification and this particularly when the classifier is a KNN.

B. Projection

Many projection techniques (global or local, linear or nonlinear) exist [6] such as Principal Component Analysis (PCA) that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components, Factorial Analysis (FA) which describes variability among observed variables in terms of fewer unobserved variables called factors ; Isometric Feature Mapping (ISOMAP) ; Multidimensional Scaling (MDS), etc... These projections can be considered as unsupervised learning of metrics.

Within the framework of a supervised classification these various projections aim at projecting the data in a space where the problem will be easier to solve. It is also the case of Support Vector Machine (SVM) [7] and kernel methods [8].

C. A Projection using a classifier

In this paper we will show how to exploit the knowledge of the classification problem to be solved and the existence of a classifier to be deployed. Learning metric is here not adapted since the model already exists and the deployment process does not have to change of model or to train a new model. Projection is not appropriate since this kind a method does not use the knowledge included in the classifier to be deployed.

We propose to project the instances in the importance space of the explanatory variables of the classifier to be deployed, then to build a KNN on this new representation. The obtained representation should incorporate information related to the topology of the classification problem [9] and allow the KNN to reach the performance of the original classifier.

III. Projection in the Importance space - General case

In this section, notations used below in the paper and the tools, which allow the projection in the importance space, are presented. Different methods to compute the variable importance are detailed in the last part of the section.

A. Description

Let :

- T be a training data table (K instances and J explanatory variables : $V1, ..., V_J$) ;
- C be the number of classes of the classification problem ;
- M be a probabilistic classifier trained using T ;
- G be a method, knowing M, which computes the importance of the explanatory variables of the classifier ; this importance is computed instance by instance ;
- an instance x_k represented by a vector of J components : $x_k = (V_1 = x_{1k}, ..., V_J = x_{Jk})$.

After the training step, M classifies the instances in T (or in another deployment table) so that any instance (k) belongs to a class c ($x_k \in C_c$).

From then armed with M, T and G, all instances in T can be projected in the importance space of the classifier. This projection is illustrated in Figure 2.

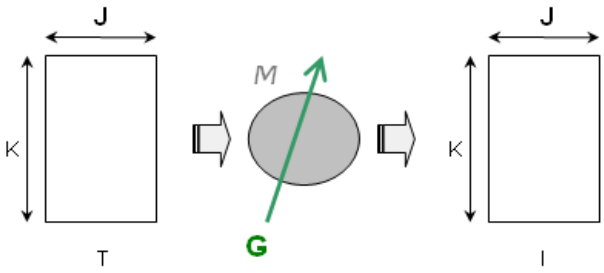

Fig. 2. Projection in the importance space

B. Which measure of importance ?

Specific methods : For many classifiers, the literature proposes one or several methods [10] for the computation of the importance of explanatory variables for the classification by M of each instance in T. We name I the importance table such that $I_j^z(x_k)$ is the importance value of the input variable V_j for the instance x_k given a class z. Such methods are methods where the computation of the importance is dedicated to a specific classifier (decision tree, SVM . . .).

Generic methods : Generic method often consider $I_j^z(x_k)$ as an indicator which "measure" the difference between the output of the model with an explanatory variable and without this explanatory variable [11]. This indicator is noted in this paper $I_j^z(x_k)$ = diff $[M(x_k)\ ,\ M(x_k \backslash V_j)]$ where $M(x_k)$ is the output value of the model and $M(x_k \backslash V_j)$ the output value of the model in the absence of the input variable V_j. The use of these generic methods requires the computation of $M(x_k \backslash V_j)$ which is not always easy depending on the classifier used. In practice, this often requires to have access to the internal parameters of the model.

Methods based on sensitivity analysis : Another approach, called sensitivity analysis, consists in analyzing the model as a black box by varying its input variables. In such "what if" simulations, the structure and the parameters of the model are important only as far as they allow accurate computations of dependent variables using explanatory variables. Such an approach works irrespective of the model.

The measure of importance is then based on sensitivity analysis of the output of the model. Such a method is described in [12].

IV. Case of a naive Bayesian classifier

This section first presents the naive Bayes classifier and the version that comes from an averaging of selective naive Bayes classifiers. The second part of the section presents the different methods used to measure the variable importance which will be tested below in the experiments section of this paper.

A. Introduction

The naive Bayes classifier [13] assumes that all the explanatory variables are independent knowing the target class. This assumption drastically reduces the necessary computations. Using the Bayes theorem, the expression of the obtained estimator for the conditional probability of a class C_z is :

$$P(C_z|x_k) = \frac{P(C_z)\prod_{j=1}^{J} P(V_j = x_{jk}|C_z)}{\sum_{t=1}^{C}\left[P(C_t)\prod_{j=1}^{J} P(V_j = x_{jk}|C_t)\right]} \quad (1)$$

The predicted class is the one which maximizes the conditional probabilities $P(C_z|x_k)$. The probabilities $P(V_j = x_{jk}|C_z)(\forall j, k, z)$ are estimated using counts after discretization for numerical variables or grouping for categorical variables. The denominator of the equation 1 normalizes the result so that $\sum_z P(C_z|x_k) = 1$.

When the naive Bayes classifier comes from an averaging of selective naive Bayes classifiers [14] each explanatory variable j is weighted by a weight W_j ($W_j \in [0-1]$). The formulation of the conditional probabilities becomes :

$$P(C_z|x_k) = \frac{P(C_z)\prod_{j=1}^{J} P(V_j = x_{jk}|C_z)^{W_j}}{\sum_{t=1}^{C}\left[P(C_t)\prod_{j=1}^{J} P(V_j = x_{jk}|C_t)^{W_j}\right]} \quad (2)$$

Note 1 : Each instance, x_k, is a vector of values (numerical or categorical) such as : $x_k = (x_{1k}, x_{2k}, ..., x_{Jk})$. After a discretization / grouping respectively for numerical / categorical variables, each explanatory variable j is coded on H_j values. Every instance is then coded in the form of a vector of discrete values. In this case the equations 1 and 2 should incorporated $P(V_j = (x_{jk} \in [.,.])|C_z)$ in place of $P(V_j = x_{jk}|C_z)$. Below in this paper this coding is implied without being formally noted.

Note 2 : Below in the paper, the class conditional probabilities ($P(V_j = x_{jk}|C_z)$) are estimated using the MODL discretization method [15] for the numeric variables and the MODL grouping method [16] for the categorical variables.

B. The measures of importance studied

1) Representation dedicated to the classifier: For the naive Bayes Classifier an exact representation can be written knowing the parameters of the model. The representation is defined as :

$$I_j^z = log(P(V_j = x_{jk}|C_z)) \quad (3)$$

Indeed, starting with the naive Bayes predictor which takes recoded explanatory variables (supervised discretization or grouping) and using the log function, one has :

$$log(P(C_z|x_k)) = \sum_{j=1}^{J} log(P(V_j = x_{jk}|C_z)) + log(P(C_z)) - log(P(x_k)) \quad (4)$$

We then introduce the following distance :

$$\text{Dist}(x_k, x_k^{'}) = \sum_{j=1}^{J}\sum_{z=1}^{C}\left|log(P(V_j = x_{jk}|C_z)) - log(P(V_j = x_{jk}^{'}|C_z))\right| \quad (5)$$

This relates to a representation of every instance on a vector of $J * C$ components (for example for $C = 2$) :

$$(log(P(V_1 = x_{1k}|C_1)), ..., log(P(V_J = x_{Jk}|C_1)), log(P(V_1 = x_{1k}|C_2)), ..., log(P(V_J = x_{Jk}|C_2))) \quad (6)$$

The proposed distance corresponds to the L1 norm using this coding.

Knowing that the classifier comes from the **averaging of classifiers :**

$$(\ log(P(V_1 = x_{1k}|C_1))W_1, ..., log(P(V_J = x_{Jk}|C_1))W_J, log(P(V_1 = x_{1k}|C_2))W_1, ...log(P(V_J = x_{Jk}|C_2))W_J\ ) \quad (7)$$

In **this case** the representation is therefore defined as :

$$MODL_j^z = log(P(V_j = x_{jk}|C_z))W_j \quad (8)$$

The equation 8 is the indicator of the representation named "MODL" below in this paper when the classifier is a selective naive Bayes classifier averaged [14]. This indicator represents an optimal result knowing the internal parameters of the model. This is this indicator which is tested the experiments part of this paper.

2) Generic Representation: The use of these generic methods requires the computation of $P(C_z|x_k \backslash V_j)$ which is not always easy depending on the classifier used. A way to have an estimation of this is to use the equation 5 (applied to a single instance) in [12] (also employed in equation 6 in [11]) :

$$P(C_z|x_k \backslash V_j) = \sum_{s=1}^{m_i} P(C_z|x_k \leftarrow V_j = a_s)P(V_j = a_s)$$

Here m_i represents the number of values of the variable V_j, the term $P(C_z|x_k \leftarrow V_j = a_s)$ represents the probability we get for C_z when in x_k we replace the value of the component V_j with the value a_s ; and $P(V_j = a_s)$ is prior on the value a_s.

In the case of the naive Bayes classifier, the appendix 1 in [11] shows that the use of this measure for sensitivity analysis produces an exact result. There is an exact matching between generic methods and methods based on sensitivity analysis. Therefore methods to compute indicator importance, described in [11], can be used. These indicators are the references in the state of art for the naive Bayes classifier :

– "Information Difference (IDI)" - This indicator measures the difference of information :

$$\text{IDI}_j^z = log\left(P(C_z|x_k)\right) - log\left(P(C_z|x_k \backslash V_j)\right) \quad (9)$$

– "Weight of Evidence (WOE)" - This indicator measures the log of the odd ratio :

$$\text{WoE}_j^z = log\left(odds(C_z|x_k)\right) - log\left(odds(C_z|x_k \backslash V_j)\right) \quad (10)$$

$$\text{where } odds(.) = p(.)/(1-p(.))$$

– "Difference of probabilities (DOP)" - This indicator simply measures the mathematical difference between the output of the classifier with and without the explanatory variable j :

$$\text{DOP}_j^z = P(C_z|x_k) - P(C_z|x_k \backslash V_j) \quad (11)$$

To this state of art, another indicator of importance is added :

– "Kullback-Leibler divergence (KLD)" - This indicator measures the Kullback-Leibler divergence in the case where the distribution of reference is the distribution of the output of the classifier with the explanatory variable

j. The distribution to compare is the distribution of the output of the classifier in absence of the explanatory variable j :

$$\text{KLD}_j^z = P(C_z|x_k) log\left(\frac{P(C_z|x_k)}{P(C_z|x_k \backslash V_j)}\right) \quad (12)$$

3) Supervised representation without classifier: The state of art [17] shows that supervised discretization is better than unsupervised discretization for classification problems. Therefore only the supervised representation is tested in this paper. For this supervised case, the problem is to define for every explanatory variable a representation which takes into account the conditional distribution of the target classes. Another problem is to be able to cumulate the contribution of each explanatory variable.

Three supervised indicators are introduced "MOP" (see equation 13), "LMOP" (see equation 14) and "VPD" (see equation 15). The MOP indicator is close to the indicator "Value Difference Metric" (VDM, ([18])). The inconvenient of VDM mentioned in ([18], section 2.5) is here relieved since explanatory variables are discretized/grouped with an efficient supervised method ([14]) (see classifier Section V-B.3). These three indicators are :

– "Modality probability (MOP)"- This indicator simply measures the probabilities of occurrence of the values of the explanatory variables such as :

$$\text{MOP}_j^z = P(V_j = x_{jk}|C_z) \quad (13)$$

This indicator is not really an indicator of importance but it allows to capture the distribution $P(x_k)$ knowing the discretization model (or grouping model).

– "Log Modality probability (LMOP)"- This indicator simply measures the information which contained in the probabilities of occurrence of the values of the explanatory variables such as :

$$\text{LMOP}_j^z = log\left(P(V_j = x_{jk}|C_z)\right) \quad (14)$$

Note that this representation is the representation associated to a not averaging naïve Bayes classifier.

– "Minimum of variable probabilities difference (VPD)" - This indicator measures the minimum difference between the probability of the variable j knowing a reference class and the probability of the same variable knowing another class such as :

$$\text{VPD}_j^z = P(V_j = x_{jk}|C_z) - \max_{q \neq z} P(V_j = x_{jk}|C_q) \quad (15)$$

The VPD values belongs to $[-1, 1]$ and measure the positive, neutral or negative contribution of the variable j in the probability $P(C_z|x_k)$. In the section V the reference class used is the predicted class by the classifier built.

4) Summary and discussion: Whatever is the chosen method, every instance x_k is represented by a vector of the importance indicator, this for all the classes (C) of the classification problem. The initial number of components of x_k is therefore multiplied by C. For example for the indicator 'WOE', the vector is :

$$x_k = \left(\text{WOE}_1^1, ..., \text{WOE}_J^1, \text{WOE}_1^2, ..., \text{WOE}_J^2, ..., \text{WOE}_1^C, ..., \text{WOE}_J^C\right) \quad (16)$$

For the indicators IDI, WOE and KLD, a Laplace estimator is used to estimate $P(.)$ such as $P(.)$ is always above 0 and below 1 ($P(.) \in]0, 1[$), this to avoid numerical problems as dividing by zero.

The distance between two instances corresponds to the L1 norm for all the indicators presented above such as :

$$\text{Dist}(x_k, x_k^{'}) = \sum_{j=1}^{J} \sum_{z=1}^{C} \left| I_j^z(x_k) - I_j^z(x_k^{'}) \right| \quad (17)$$

The Table I summarizes the eight indicators based or not on sensitivity analysis and dedicated or not to the naive Bayes classifier.

TABLE I
SUMMARY OF THE 8 INDICATORS OF IMPORTANCE (1) MODL, (2) IDI, (3) WOE, (4) DOP, (5) KLD, (6) MOP, (7) LMOP, (8)VPD

	1	2	3	4	5	6	7	8
Sensitivity Analysis (Yes/No)	N	Y	Y	Y	Y	N	N	N
Dedicated to the naive Bayes (Yes/No)	Y	N	N	N	N	N	Y	N
Takes into account the weights (W_j) (Yes/No)	Y	Y	Y	Y	Y	N	N	N

These eight indicators cover the 3 axis mentioned in the first column of the Table I. They allow the analysis of the generic behavior of the proposed method : the use of any classifier and a measure of the importance of the explanatory variables using sensitivity analysis.

V. IMPACT OF THE REPRESENTATION ON A KNN - EXPERIMENTATIONS

A. The data - The small KDD 2009 challenge

The purpose of the KDD Cup 2009 was to predict the propensity of customers to switch provider (churn), buy new products or services (appetency), or buy upgrades or add-ons proposed to them to make the sale more profitable (up-selling). In the 'small' version of the challenge, each classification problem was constituted of a database. The database consisted of 100 000 instances (customers), split randomly into equally sized train and test. An instance was constituted of 230 explanatory numerical and categorical variables. This dataset offers a variety of other difficulties : heterogeneous data, noisy data, unbalanced distributions of predictive variables, sparse target values (only 1 to 7 percent of the examples belong to the positive class) and many missing values. The entire description of the challenge and the analysis of the results can be found in [19].

In this section only the train set (50000 instances) is used since the labels of the test set remain hidden by the organizers

of the challenge. The percentages of positive instances for the three problems are : (1) Churn problem : 7.3% (3672/50000 on train) ; (2) Appetency problem : 1.8% (890/50000 on train) ; Up-selling problem : 7.4% (3682/50000 on train). Therefore all the data used in this section are publicly available and the results are reproducible.

B. Protocol

1) Data normalization: One weakness of many distance function is that if one of the input attributes has a relatively large range, then it can overpower the other attributes. In this article the calculation of indicators of importance project the data in a consistent space. All the projected variables contain the same type of information and therefore no normalization is required.

2) K-fold cross validation: A 5-fold cross validation process has been used. The performance of every model is computed on the fold which has not been used to train the initial classier or the KNN. The five 'test' results are then combined to give an estimation of the generalization error of the architecture tested. The folds used to do the training of the initial classifier or the KNN classifier do not cross the test set. For reproducibility reason the following indication is given : the initial train set of the KDD challenge has been divided in 5 folds of 10000 instances each in the order of the downloadable file (the order of the instances in the train set of the challenge does not has a particular structure). The five fold cross validation allows to have a mean result with its standard deviation for the initial classifier and the KNN classifier.

3) Classifier beforehand built: The Table II presents the obtained results by the initial classifier (M), the selective naive Bayes classifier (SNB), for the Train AUC [20] (4 folds) and the Test AUC (1 fold). This classifier is the classifier (M) to be deployed and described in the introduction of this paper.

TABLE II
PERFORMANCES OF "SELECTIVE NAÏVE BAYES" CLASSIFIER (AUC)

Problem	AUC Train	AUC Test	Nb variables
Appetency	0.834 ± 0.003	0.817 ± 0.009	17, 18, 17, 20, 18
Churn	0.737 ± 0.004	0.728 ± 0.018	34, 35, 32, 34, 33
Upselling	0.868 ± 0.002	0.863 ± 0.007	50, 50, 51, 59, 51

The last column of the Table II indicates the number of used variables by the selective classifiers (fold1, ... , fold5), number to compare to the 230 initial variables. These classifiers have been obtained using the Khiops[1] software.

4) KNN classifier: In the KNN procedure the following algorithm is used :

1) for each test instance (t) its k nearest neighbors ($\text{KNN}_1, ..., \text{KNN}_k$) are looked for in the train set data base using a distance based on the L1 norm and the considered representation (MOP, VPD, MODL, IDI, WOE, DOP, KLD), see equation 17 ;
2) each of the nearest neighbors of the instance t receives the conditional probabilities $P(C_j|\text{KNN}_k)$, $\forall j$; probabilities computed using the initial classifier (SNB) ;
3) the instance t receives the mean of the conditional probabilities of its nearest neighbors $P(C_j|t) = \frac{\sum_k P(C_j|\text{KNN}_k)}{k}$, $\forall j$.
4) the belonging class of t is the one which maximizes the conditional probability $P(C_j|t)$.
5) the train and test AUC are estimated.

[1] www.khiops.com

C. Results & Discussion

The Figures 3, 4, 5 present respectively the mean Test AUC for appetency, churn and upselling. On each Figure the horizontal axis represents the number of nearest neighbors and the vertical the mean AUC (on the 5 folds). The results of eight representations have been distributed on the left and right part of each figure to have a good readability. The left part of each figure gives the results of the initial classifier (SNB), the unsupervised representations and the dedicated representation, the right part the results of the representation based on sensitivity analysis. The scales are the same for the left and right part for left/right comparisons. The variances of the results are not presented for place or readability reasons. The eight variances have the same magnitude and belong to the following intervals : appetency [0.007-0.014], churn [0.014-0.020] and upselling [0.005-0.009] ; irrespective of the number of nearest neighbors.

The KNN-based deployment is based on two things : (i) the use of a representation (a geometry) to decide the proximity of the instances and (ii) the use of the model to be deployed (SNB) to set the conditional probability of the classes. The Figures 3, 4, 5 give an evaluation of the quality of the geometry. They indicate that a representation based on sensitivity analysis (IDI, WOE, DOP and KLD) allow obtaining the same results as the dedicated representation. This point is the main result of this paper and the goal is reach : the KNN-based deployment is equal to the direct deployment. The use of a representation not based on the classifier does not allow obtaining this result. In this case, the degradation in the deployment is important.

The generic representations and the dedicated representation reach the same performance than the initial classifier except for the Churn problem where a small degradation of the results can be seen : mean AUC $\approx 0.721 \pm 0.017$ with the KNN (for k from 1 to 10) against a mean AUC = 0.728 ± 0.018 for the SNB.

The analysis of the results versus the number of nearest neighbors shows a very good performance even for a small number of neighbors. This good performance is kept when the number of neighbors increases except for the MOP and LMOP representations where a relative degradation of the performances can be seen more (Churn) or less (Appetency, Upselling).

Another interesting point comes from the results of the representations MOP, LMOP and VPD which allow having a better result than a direct deployment realized using the

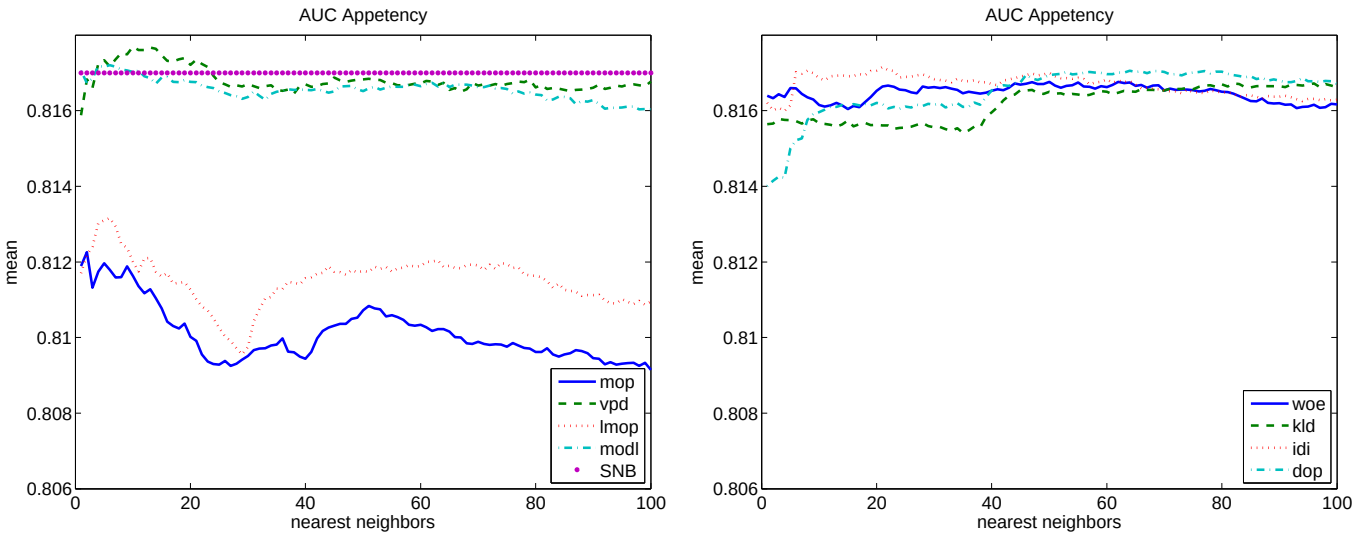

Fig. 3. AUC Test Appetency

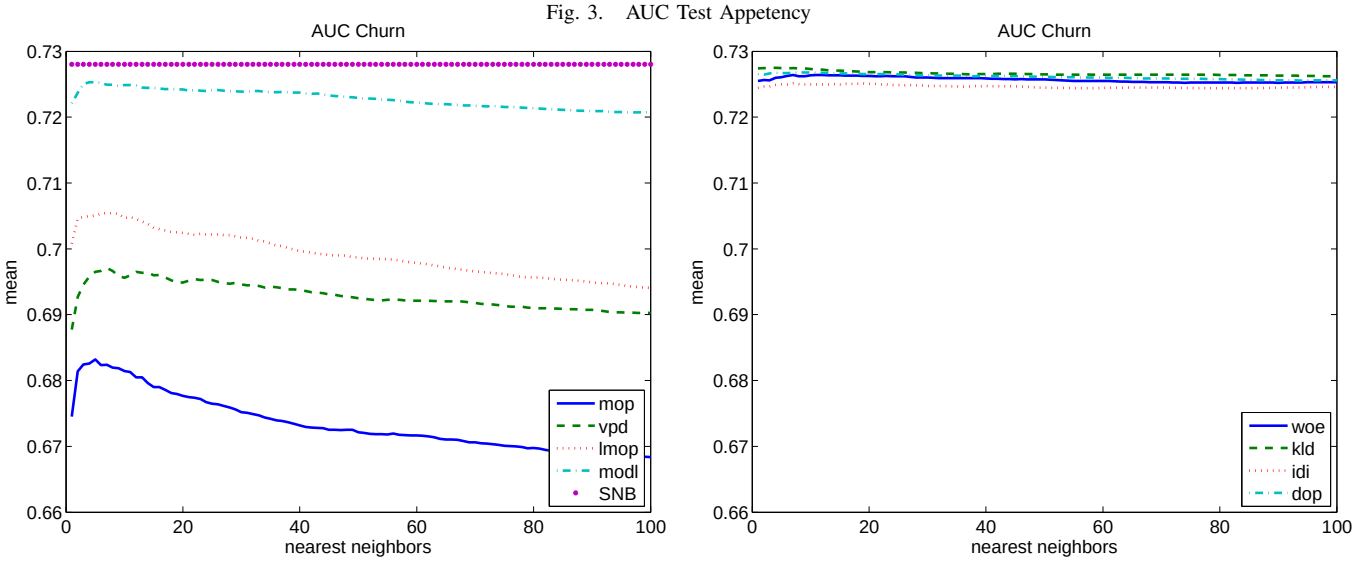

Fig. 4. AUC Test Churn

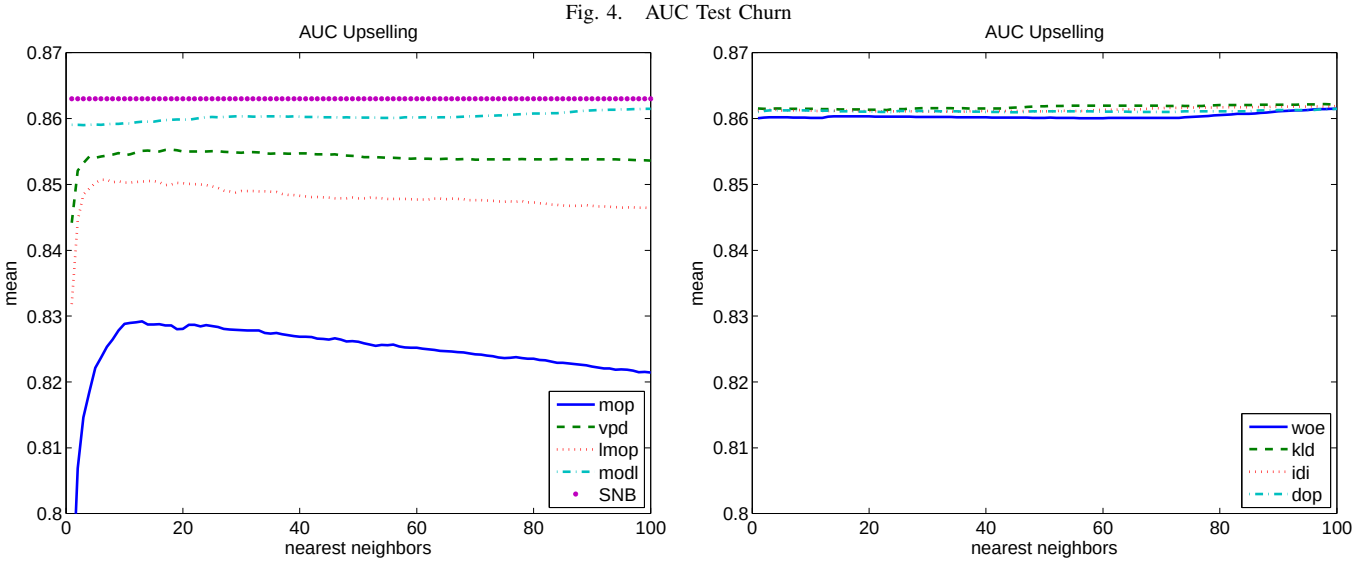

Fig. 5. AUC Test Upselling

classical naive Bayes (NB : naive Bayes classifier without variable selection and model averaging). The Table III presents simultaneously the results obtained by this NB classifier, the SNB classifier and those obtained using a KNN which uses the representations MOP, LMOP and VPD. The gap between the results with the direct deployment of the SNB and KNN-base deployment using LMOP indicates that the geometry given by a naïve Bayes classifier is not equal to the geometry of the dedicated model. The gap between the results with the direct deployment of the NB and KNN-base deployment using LMOP indicates that the geometry given by a naïve Bayes classifier combined by the scoring of the SNB is interesting. Further research on this point will be investigated.

TABLE III
PERFORMANCES (AUC) OF THE "NAÏVE BAYES (NB)" ON THE TEST SET VERSUS THE KNN WHICH USES THE MOP, LMOP OR VPD REPRESENTATION ; THE NUMBER BETWEEN () IS THE VALUE OF K.

	Appetency	Churn	Upselling
SNB	0817 $\pm$ 0.009	0.728 $\pm$ 0.018	0.863 $\pm$ 0.007
NB	0.785 $\pm$ 0.008	0.672 $\pm$ 0.020	0.754 $\pm$ 0.004
MOP	0.809 $\pm$ 0.011(2)	0.680 $\pm$ 0.020 (9)	0.828 $\pm$ 0.008 (5)
LMOP	0.814 $\pm$ 0.011 (8)	0.705 $\pm$ 0.019 (8)	0.850 $\pm$ 0.009 (8)
VPD	0.816 $\pm$ 0.009 (9)	0.690 $\pm$ 0.015 (2)	0.855 $\pm$ 0.008 (6)

VI. CONCLUSION

We presented in this article a method to build a data representation which allows a KNN to reach the performance of a beforehand built classifier to be deployed. It was shown that the proposed method is generic : (1) the instances can be projected using a sensitivity analysis not dedicated to the beforehand built classifier (2) there are no restrictions on the type of (probabilistic) classifier. It was shown on three marketing problems that it is possible to create an effective representation in terms of nearest neighbors with performance similar to the classifier directly deployed. The industrial problem presented in introduction, to calculate the Index Table, receives an extremely effective solution.

The concept of instance selection of the train set or reduction of the size of the train set represents a direction for future researches. The eight representations used in this article are integrated into the software Kawab, available as a shareware.

RÉFÉRENCES

[1] R. Féraud, M. Boullé, F. Clérot, and F. Fessant. Vers l'exploitation de grandes masses de données. In *Extraction et Gestion des Connaissances (EGC)*, pages 241–252, 2008.

[2] R. Féraud, M. Boullé, F. Clérot, F. Fessant, and V. Lemaire. The orange customer analysis platform. In *Industrial Conference on Data Mining (ICDM)*, Berlin, July 2010.

[3] J.S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Software*, 11(1) :37–57, 1985.

[4] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB Conference*, 1999.

[5] Lui Yang and Rong Jin. Contents distance metric learning : A comprehensive survey, 2006.

[6] A.N. Gorban, B. Kégl, D.C. Wunsch, and A. Zinovyev, editors. *Principal Manifolds for Data Visualization and Dimension Reduction*, volume 58. Series : Lecture Notes in Computational Science and Engineering, 2008. Online version available.

[7] J. Shawe-Taylor and N. Cristianini. *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[8] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[9] V. Pisetta and D. A. Zighed. Similarity and kernel matrix evaluation based on spatial autocorrelation analysis. In *International Symposium on Foundations of Intelligent Systems (ISMIS)*, pages 422 – 430, 2009.

[10] I. Guyon. *Feature extraction, foundations and applications*. Elsevier, 2005.

[11] M. Robnik-Sikonja and I. Kononenko. Explaining classifications for individual instances. *IEEE TKDE*, 20(5) :589–600, 2008.

[12] V. Lemaire and C. Clérot. An input variable importance definition based on empirical data probability and its use in variable selection. In *International Joint Conference on Neural Networks (IJCNN)*, 2004.

[13] P. Langley, W. Iba, and K. Thompson. An analysis of bayesian classifiers. In *International conference on Artificial Intelligence, AAAI*, pages 223–228, 1992.

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

[15] M. Boullé. a bayes optimal discretization method for continuous attributes. *Machine Learning*, 65(1) :131–165, 2006.

[16] M. Boullé. A bayes optimal approach for partitioning the values of categorical attributes. *Journal of Machine Learning Research*, 6 :1431–1452, 2005.

[17] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. *Machine Learning*, pages 194–202, 1995.

[18] D. Randall Wilson and Tony R. Martinez. Supervised and unsupervised discretization of continuous features. *Journal of Artificial Intelligence Research*, 6 :1–34, 1997.

[19] I. Guyon, V. Lemaire, M. Boullé, G. Dror, and D. Vogel. Analysis of the kdd cup 2009 : Fast scoring on a large orange customer database. *JMLR Workshop and Conference Proceedings*, 7 :1–22, 2009.

[20] T. Fawcett. Roc graphs : Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, HP Labs, 2003., 2003.