TP2: Detecting Fraudulent Transactions

1 Problem Description and Objectives

Fraud detection is an important area for potential application of data mining techniques given the economic and social consequences that are usually associated with these illegal activities. From the perspective of data analysis, frauds are usually associated with unusual observations as these are activities that are supposed to be deviations from the norm. These deviations from normal behavior are frequently known as outliers in several data analysis disciplines. In effect, a standard definition of an outlier is that it is "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" (Hawkins, 1980).

The data we will be using in this case study refers to the transactions reported by the salespeople of some company. These salespeople sell a set of products of the company and report these sales with a certain periodicity. The data we have available concerns these reports over a short period of time. The salespeople are free to set the selling price according to their own policy and market. At the end of each month, they report back to the company their transactions. The goal of this data mining application is to help in the task of verifying the veracity of these reports given past experience of the company that has detected both errors and fraud attempts in these transaction reports. The help we provide will take the form of a ranking of the reports according to their probability of being fraudulent. This ranking will allow to allocate the limited inspection resources of the company to the reports that our system signals as being more "suspicious".

2 The Available Data

The data we have available is of an undisclosed source and has been anonymized. Each of the 401 146 rows of the data table includes information on one report by some salesman. This information includes his ID, the product ID, and the quantity and total value reported by the salesman. This data has already gone through some analysis at the company. The result of this analysis is shown in the last column, which has the outcome of the inspection of some transactions by the company. Summarizing, the dataset we will be using has the following columns:

- **ID** - a factor with the ID of the salesman.
- **Prod** - a factor indicating the ID of the sold product.
- **Quant** - the number of reported sold units of the product.
- **Val** - the reported total monetary value of the sale.
- **Insp** - a factor with three possible values: `ok` if the transaction was inspected and considered valid by the company, `fraud` if the transaction was found to be fraudulent, and `unkn` if the transaction was not inspected at all by the company.

3 Loading and exploring the dataset

The dataset is available in the book package. If you use the book package data, then you should proceed as follows:

```
library(DMwR)
data(sales)
```

To get an initial overview of the statistical properties of the data, we use the function `summary()`.

```
summary(sales)
```

An interesting alternative can be obtained using the function `describe()` from the extra package `Hmisc`. Try it!

We have a significant number of products and salespeople, as we can confirm using the function `nlevels()`:

```
c(nlevels(sales$ID), nlevels(sales$Prod))
```

The result of the `summary()` function reveals several relevant facts on this data. First there are a considerable number of unknown values in the columns `Quant` and `Val`. This can be particularly problematic if both happen at the same time, as this would represent a transaction report without the crucial information on the quantities involved in the sale. We can easily check if there are such situations:

```
length(which(is.na(sales$Quant) & is.na(sales$Val)))
```

As you can see, this is a reasonable number of transactions. Given the large total amount of transactions, one can question whether it would not be better to simply delete these reports. We will consider this and other alternatives later on.

Another interesting observation from the results of the `summary()` function is the distribution of the values in the inspection column. In effect, and as expected, the proportion of frauds is relatively low, even if we only take into account the reports that were inspected, which are also a small proportion overall:

```
table(sales$Insp)/nrow(sales) * 100
```

- ♦ **En utilisant les commandes `table()` et `barplot()`, afficher les diagrammes en bâton représentants (a) le nombre de rapports par vendeur et (b) le nombre de rapports par produit.**

The descriptive statistics of `Quant` and `Val` show a rather marked variability. This suggests that the products may be rather different and thus it may make sense to handle them separately. If the typical prices of the products are too different, then a transaction report can only be considered abnormal in the context of the reports of the same product. Still, these two quantities may not be the ideal ones to draw this conclusion. Given the different quantity of products that are sold on each transaction, it is more correct to carry out this analysis over the unit price instead. This price can be added as a new column of our data frame:

```
sales$Uprice <- sales$Val/sales$Quant
```

The unit price should be relatively constant over the transactions of the same product. When analyzing transactions over a short period of time, one does not expect strong variations of the unit price of the products.

- ♦ **La situation peut être très différente lorsqu'on considère les prix unitaires de tous les produits. Tracer le boxplot qui permet de visualiser les variations de cette variable.**

In the light of previous results, it seems inevitable to analyze the set of transactions of each product individually, looking for suspicious transactions on each of these sets. One problem with this approach is that some products have very few transactions: 982 out of 4 548 products have less than 20 transactions. Declaring a report as unusual based on a sample of less then 20 reports may be too risky.

The computation of the number of transactions and the average unit price for each product can be done as follows:

```
attach(sales)
num.prod = as.numeric(table(Prod))
sum(num.prod < 20)                # nb de produits ayant <20 transactions
av.uprice = tapply(Uprice, Prod, mean, na.rm=T)
```

4 The boxplot rule for outlier detection

One of the main assumptions we will be making in our analysis to find abnormal transaction reports is that the unit price of any product should follow a near-normal distribution. This means that we expect that the transactions of the same product will have roughly the same unit price with some small variability, possibly caused by some strategies of the salespeople to achieve their commercial goals. In this context, there are some basic statistical tests that can help us in finding deviations from this normality assumption. An example is the box plot rule. This rule serves as the basis of outlier identification.

The rule states that an observation should be tagged as an anomaly high (low) value if it is above (below) the high (low) whisker, defined as `Q3 + 1.5 x IQR` (`Q1 - 1.5 x IQR`), where `Q1` is the first quartile, `Q3` the third quartile, and `IQR = (Q3 - Q1)` the inter-quartile range. This simple rule works rather well for normally distributed variables, and it is robust to the presence of a few outliers being based in robust statistics like the quartiles.

The following command allows us to compute the boxplot statistics for the distributions of unit prices for each product:

```
BP.uprice = tapply(Uprice, Prod, function(x) boxplot.stats(x)$stats)
```

♦ **Soit x un vecteur numérique. Que représente la valeur n calculée par la commande** `n = sum(boxplot.stats(x)$out)`**?**

1. **En utilisant la commande** `tapply` **définir un vecteur nommé** `out.uprice` **dont les composants sont les nombres d'outliers par produit.**
2. **Quel est le nombre total d'outliers détectés par cette méthode ?**
3. **Quel est le pourcentage de transactions signalées comme anormale ?**
4. **Quels sont les 10 produits ayant le plus de transactions douteuses ?**

One might question whether this simple rule for identifying outliers would be sufficient to provide the kind of help we want in this application. In what follows we will evaluate the performance of a small variant of this rule adapted to our application.

There is a caveat to some of the conclusions we have drawn in this section. We have been using the data independently of the fact that some of the reports were found to be fraudulent and some other may also be fraudulent although not yet detected. This means that some of these "conclusions" may be biased by data that is wrong. The problem is that for the transactions that are tagged as frauds, we do not know the correct values. Theoretically, the only transactions that we are sure to be correct are the ones for which the column `Insp` has the value `OK`, but these are just

3.6% of the data. So, although the analysis is correct, the conclusions may be impaired by low-quality data. This should be taken into account in a real-world situation not to provide advice to the company based on data that includes errors.

Another thing one can do is present the results to the company and if some result is unexpected to them, carry out a closer analysis of the data that leads to that surprising result. This means that this sort of analysis usually requires some form of interaction with the domain experts.

5 Defining the Data Mining Tasks

The main goal of this application is to use data mining to provide guidance in the task of deciding which transaction reports should be considered for inspection as a result of strong suspicion of being fraudulent.

The available dataset has a column (`Insp`) that has information on previous inspection activities. The main problem we have is that the majority of the available reports have not been inspected. This means that we have two types of observations in our dataset. We have a (small) set of labeled observations for which we have the description of their characteristics plus the result of their inspection. We have another (large) set of unlabeled observations that have not been inspected. In this context, there are different types of modeling approaches that can be applied to these data, depending on which observations we use for obtaining the models:

Unsupervised Techniques We completely ignore the column `Insp`, since in the majority of cases its value is noninformative. We are thus facing a descriptive data mining task as opposed to predictive tasks, which are the goal of supervised methods.

Clustering is an example of a descriptive data mining technique. Clustering methods try to find the "natural" groupings of a set of observations by forming clusters of cases that are similar to each other. The notion of similarity usually requires the definition of a metric over the space defined by the variables that describe the observations. Cases that are near each other are usually considered part of the same natural group.

Outlier detection can also be viewed as a descriptive data mining task. Some outlier detection methods assume a certain expected distribution of the data, and tag as outliers any observations that deviate from this distribution. Another common outlier detection strategy is to assume a metric over the space of variables and use the notion of distance to tag as outliers observations that are "too far" from others.

Supervised Techniques The set of transactions that were labeled normal or fraudulent (i.e., have been inspected) can be used with other types of modeling approaches. Supervised learning methods use this type of labeled data. The task of the modeling technique is to obtain the model parameters that optimize a certain selected criterion, for example, minimize the prediction error of the model.

In the case of our dataset, the target variable is the result of the inspection task and can take two possible values: ok and fraud. We are thus facing a classification problem. The transactions that were not inspected cannot be used in these tasks. This means that we can only use 15 732 of the 401 146 available reports as the training sample.

Semi-Supervised Techniques Semi-supervised methods are motivated by the observation that for many applications it is costly to find labeled dataŮthat is, cases for which we have the value of the target variable. In this context, one frequently faces problems with a large proportion of data that is unlabeled, together with a small amount of labeled data.

6 Experimental Methodology

The dataset we are using has a very reasonable size. In this context, it makes sense to select the Hold Out method for our experimental comparisons. This method consists of randomly splitting the available dataset into two partitions (typically in 70%/30% proportions). One of the partitions is used for obtaining the models, while the other is used for testing them.

We observed above that the products are rather different, and that some products have, in effect, few transactions. In this context, we may question whether it makes sense to analyze the transactions of all products together. An argument in favor of checking them together is that there is a variable (the product ID) that can be used to discriminate among the products, and thus the modeling techniques can use the variable if necessary. Moreover, by putting all transactions together, the models can take advantage of some eventual relationships among products. Nevertheless, an alternative would be to analyze each product in turn, ranking its transactions by some outlier score.

7 Supervised techniques

To apply supervised techniques, we first need to remove from the dataset all the data corresponding to unlabeled examples. This may be done by:

```
sales1 <- sales[sales$Insp != "unkn",]
sales1$Insp=factor(sales1$Insp)        # permet de supprimer la modalité "unkn"
sales1 <- na.omit(sales1)              # supprime les exemples avec valeurs manquantes
```

Pour la tâche de classification, on peut très bien répéter ce qu'on a fait lors du TP précédent. On propose ici une alternative très intéressante basée sur l'utilisation de boosting et, plus précisément, l'algorithme XGboost (e**X**treme **G**radient **Boost**ing). Pour cela,

- commencez par installer et charger le package correspondant:
  ```
  install.packages("xgboost")
  library(xgboost)
  ```
- lisez la documentation se trouvant sur la page
  https://xgboost.readthedocs.io/en/latest/R-package/xgboostPresentation.html
- définissez l'échantillon d'entrainement et l'échantillon de test dans le problème des transactions froduleuses qu'on est en train de considérer:
  ```
  sales1 <- sales1[, c("ID", "Prod", "Uprice", "Insp")]
  N <- nrow(sales1)
  Index <- sample(1:N)
  K=round(0.7*N)
  sales.train <- sales1[Index[1:K],]
  sales.test <- sales1[Index[(K+1):N],]
  ```

♦ **On souhaite utiliser la méthode XGBoost avec des arbres de décision de profondeur 2 pour prévoir la nature de la transaction ("ok" versus "fraud").**

1. **Ecrire un petit programme qui pour un nombre de tours (round) variant entre 1 et 10 effectue les prédictions par XGBoost et affiche les taux d'erreurs commises sur l'échantillon de test et l'échantillon d'apprentissage.**
2. **Quel est le nombre de tours optimal selon l'erreur d'entraînement, celui selon l'erreur de test ? Commentez le résultat.**
3. Comparez ces résultats à ceux obtenus par un arbre de décision et un modèle logistique.

8 The Class Imbalance Problem

Our dataset has a very imbalanced proportion of normal and fraudulent reports. The latter are a clear minority, roughly 8.1% of the inspected reports. Problems of this type can create all sorts of difficulties in the task of obtaining predictive models. In effect, for our application it would be easy to obtain around 90% accuracy by predicting that all reports are normal. Given the prevalence of this class, this would get us to this apparently very high accuracy level.

Another problem with class imbalance is that it has a strong impact on the performance of the learning algorithms that tend to disregard the minority class given its lack of statistical support. This is particularly problematic in situations where this minority class is exactly the most relevant class, as is the case in our domain.

There are several techniques that have been developed with the purpose of helping the learning algorithms overcome the problems raised by class imbalance. They generally group in two families: (1) methods that bias the learning process by using specific evaluation metrics that are more sensitive to minority class examples; and (2) sampling methods that manipulate the training data to change the class distribution. In our attempt to use supervised classification methods in our problem, we will use a method belonging to this second group.

Several sampling methods have been proposed to change the class imbalance of a dataset. A successful example is the SMOTE method (Chawla et al., 2002). The general idea is to artificially generate new examples of the minority class using the nearest neighbors of these cases. Furthermore, the majority class examples are also under-sampled, leading to a more balanced dataset. This method has been implemented in a function called `SMOTE()` included the book package. Given an imbalanced sample, this function generates a new data set with a more balanced class distribution.

```
newData <- SMOTE(Insp ~ ., sales.train, perc.under = 500)
```

To see the result:

```
table(newData$Insp)
table(sales.train$Insp)
```

On peut maintenant appliquer les différents algorithmes de classification à ces données `newData` et tester la précision des classifieurs obtenus en utilisant les données `sales.test`.

- ♦ **Faites des prédictions en utilisant la méthode XGBoost et les arbres de décision sur ces données équilibrées. Présentez les résultats et commentez les.**

A faire pour le compte-rendu du TP2

Utilisez votre éditeur de texte préféré (Word, LaTeX, Open Office) pour rédiger le compte-rendu. Convertissez le fichier final en pdf avant de le déposer dans le répertoire partagé (le même que pour le TP 1).

Répondez à toutes les questions de cette fiche indiquées par un losange.

La date limite pour l'envoi du compte-rendu est le 13 novembre.