A Case Study on Fraud Detection

L. Torgo

ltorgo@fc.up.pt
Faculdade de Ciências / LIAAD-INESC TEC, LA
Universidade do Porto

Aug, 2017

Problem Description

The Context

- Fraud detection is a key activity in many organizations
- Frauds are associated with unusual activities
- These activities can be seen as outliers from a data analysis perspective

The Concrete Application

- Transaction reports of salesmen of a company
- Salesmen free to set the selling price of a series of products
- For each transaction salesmen report back the product, the sold quantity and the overall value of the transaction

The Available Data

The **sales** data set (package DMwR)

- Data frame with 401,146 rows and 5 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 transaction.
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.

NYU | STERN
MS in Business Analytics

Loading the Data

```
library(dplyr)
data(sales,package="DMwR2")  # load the data without loading the pack.
sales

## # A tibble: 401,146 × 5
##        ID   Prod Quant   Val   Insp
##    <fctr> <fctr> <int> <dbl> <fctr>
## 1      v1     p1   182  1665   unkn
## 2      v2     p1  3072  8780   unkn
## 3      v3     p1 20393 76990   unkn
## 4      v4     p1   112  1100   unkn
## 5      v3     p1  6164 20260   unkn
## 6      v5     p2   104  1155   unkn
## 7      v6     p2   350  5680   unkn
## 8      v7     p2   200  4010   unkn
## 9      v8     p2   233  2855   unkn
## 10     v9     p2   118  1175   unkn
## # ... with 401,136 more rows
```

NYU | STERN
MS in Business Analytics

Main Data Statistics

```
summary(sales)

##        ID              Prod             Quant                Val
##  v431   : 10159   p1125  :  3923   Min.   :      100   Min.   :   1005
##  v54    :  6017   p3774  :  1824   1st Qu.:      107   1st Qu.:   1345
##  v426   :  3902   p1437  :  1720   Median :      168   Median :   2675
##  v1679  :  3016   p1917  :  1702   Mean   :     8442   Mean   :  14617
##  v1085  :  3001   p4089  :  1598   3rd Qu.:      738   3rd Qu.:   8680
##  v1183  :  2642   p2742  :  1519   Max.   :473883883   Max.   :4642955
##  (Other):372409   (Other):388860   NA's   :13842       NA's   :1182
##     Insp
##  ok   : 14462
##  unkn :385414
##  fraud:  1270
##
##
##
##

nrow(filter(sales,is.na(Quant), is.na(Val)))

## [1] 888

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

##
##        ok      unkn     fraud
##  3.605171 96.078236  0.316593
```

MS in Business Analytics

Unit Price

- Looking at the stats of `Quant` and `Val` reveals high variability
  - Search for frauds should be done by product
  - A transaction can only be regarded as abnormal in the context of transactions of the same product
- Unit price might be a better way of comparing transactions of the same product
- Lets create a new column for the data set with the claimed unit price for each transaction:

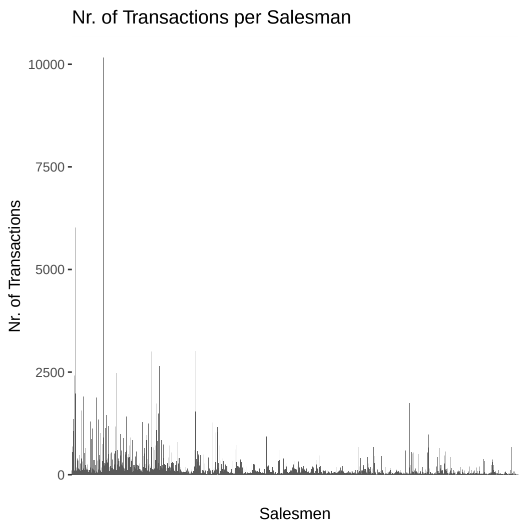
```
sales <- mutate(sales, Uprice = Val / Quant)
summary(sales$Uprice)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
##     0.00     8.46    11.89    20.30    19.11 26460.70    14136
```

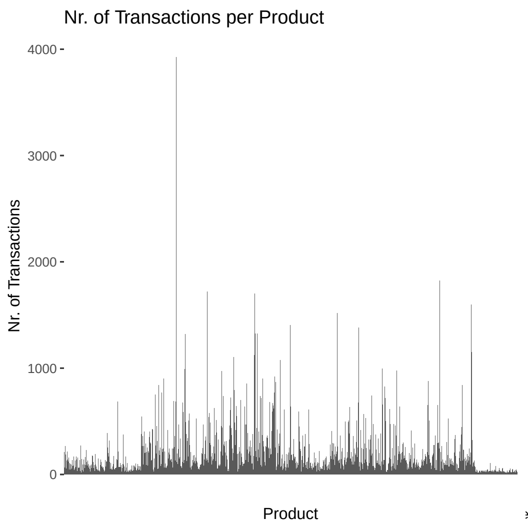
NYU | STERN
MS in Business Analytics

Variability Among Salesmen and Products

```
library(ggplot2); ids <- group_by(sales, ID)
ggplot(summarize(ids, nTrans=n()),
 aes(x=ID,y=nTrans)) + geom_bar(stat="identity") +
 theme(axis.text.x = element_blank(),
       axis.ticks.x=element_blank()) +
 xlab("Salesmen") + ylab("Nr. of Transactions") +
 ggtitle("Nr. of Transactions per Salesman")
```

```
prods <- group_by(sales, Prod)
ggplot(summarize(prods, nTrans=n()),
 aes(x=Prod,y=nTrans)) + geom_bar(stat="identity") +
 theme(axis.text.x = element_blank(),
       axis.ticks.x=element_blank()) +
 xlab("Product") + ylab("Nr. of Transactions") +
 ggtitle("Nr. of Transactions per Product")
```





The Number of Transactions of the Products

- There are products with very few transactions...
- Of the 4,548 products, 982 have less than 20 transactions
- Declaring a transaction as unusual based on comparing it to fewer than 20 other transactions seems too risky...

```
nrow(summarise(prods,nTrans=n()) %>% filter(nTrans < 20))

## [1] 982
```

NYU | STERN
MS in Business Analytics

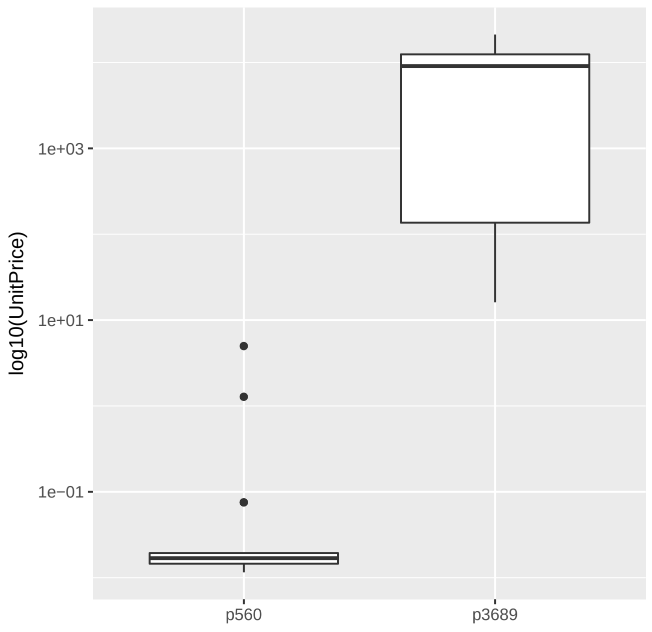
The Typical Price of each Product

```
mpProds <- summarize(prods, medianPrice=median(Uprice,na.rm=TRUE))
bind_cols(mpProds %>% arrange(medianPrice) %>% slice(1:5),
          mpProds %>% arrange(desc(medianPrice)) %>% slice(1:5))

## # A tibble: 5 x 4
##      Prod medianPrice   Prod medianPrice
##    <fctr>       <dbl> <fctr>       <dbl>
## 1    p560  0.01688455  p3689   9204.1954
## 2    p559  0.01884438  p2453    456.0784
## 3   p4195  0.03025914  p2452    329.3137
## 4    p601  0.05522265  p2456    304.8515
## 5    p563  0.05576406  p2459    283.8119
```

NYU | STERN
MS in Business Analytics

The Unit Price Distribution of the Least and Most Expensive Products

```
library(ggplot2)
library(forcats)
ggplot(filter(sales,
              Prod %in% c("p3689","p560")),
    aes(x=fct_drop(Prod), y=Uprice)) +
    geom_boxplot() +
    scale_y_log10() +
    xlab("") + ylab("log10(UnitPrice)")
```



MS in Business Analytics

The Number of Transactions per Salesmen

```
nrow(summarise(ids,nTrans=n()) %>% filter(nTrans < 10))

## [1] 1539
```

- Once again there are salesmen with very few transactions
- However, here the problem is not that serious...

What is the Typical Total Volume of Transactions per Salesmen?

```
tvIDs <- summarize(ids,totalVal=sum(Val,na.rm=TRUE))
bind_cols(tvIDs %>% arrange(totalVal) %>% slice(1:5),
          tvIDs %>% arrange(desc(totalVal)) %>% slice(1:5))

## # A tibble: 5 x 4
##       ID totalVal     ID  totalVal
##   <fctr>    <dbl> <fctr>     <dbl>
## 1  v3355     1050   v431 211489170
## 2  v6069     1080    v54 139322315
## 3  v5876     1115    v19  71983200
## 4  v6058     1115  v4520  64398195
## 5  v4515     1125   v955  63182215
```

- Big variety!

A Closer Look at the Salesmen Performance

- The top 100 salesmen account for almost 40% of the revenue of the company
- The bottom 2,000 out of the 6,016 salesmen generate less than 2% of the revenue

```
arrange(tvIDs,desc(totalVal)) %>% slice(1:100) %>%
    summarize(t100=sum(totalVal)) /
    (summarize(tvIDs,sum(totalVal))) * 100

##       t100
## 1 38.33277

arrange(tvIDs,totalVal) %>% slice(1:2000) %>%
    summarize(b2000=sum(totalVal)) /
    (summarize(tvIDs,sum(totalVal))) * 100

##      b2000
## 1 1.988716
```

The Importance of the Unit Price

- The key assumption we are going to make to find frauds is that the transactions of the same product should not vary a lot in terms of unit price
- More precisely, we will assume that the unit price of the transactions of each product should follow a normal distribution
- This immediately provides a nice and intuitive baseline rule for finding frauds:

  Transactions with declared unit prices that do not fit the assumptions of the normal distribution are suspicious

NYU | STERN
MS in Business Analytics

The Importance of the Unit Price (2)

The Box-Plot Rule

- Values outside of the following interval are outliers:

$$[Q_1 - 1.5 \times IQR, Q_3 + 1.5 \times IQR]$$

  where $Q_1 (Q_3)$ is the 1st (3rd) quartile and IQR is the inter-quartile range

NYU | STERN
MS in Business Analytics

Using the Box-Plot Rule

```
nouts <- function(x) length(boxplot.stats(x)$out)
noutsProds <- summarise(prods,nOut=nouts(Uprice))
arrange(noutsProds,desc(nOut))

## # A tibble: 4,548 x 2
##       Prod  nOut
##     <fctr> <int>
##  1  p1125   376
##  2  p1437   181
##  3  p2273   165
##  4  p1917   156
##  5  p1918   156
##  6  p4089   137
##  7   p538   129
##  8  p3774   125
##  9  p2742   120
## 10  p3338   117
## # ... with 4,538 more rows
```

NYU | STERN
MS in Business Analytics

Using the Box-Plot Rule

- Using this very simple method, 29,446 transactions are considered outliers, which corresponds to approximately 7% of the total number of transactions,

```
summarize(noutsProds,totalOuts=sum(nOut))

## # A tibble: 1 x 1
##   totalOuts
##       <int>
## 1     29446

summarize(noutsProds,totalOuts=sum(nOut))/nrow(sales)*100

##   totalOuts
## 1   7.34047
```

Unknown Values

- As we have mentioned there are lots of unknown values
- Particularly serious are the 888 transactions with both `Quant` and `Val` unknown
- There are 3 main ways of handling unknown values in a data analysis problem:
  1. Removing the cases with unknowns
  2. Filling in the unknowns using some strategy
  3. Use only tools that can handle data sets with unknowns
- Products and salesmen involved in the 888 problematic cases:

```
nas <- filter(sales,is.na(Quant), is.na(Val)) %>% select(ID,Prod)
```

The Problematic Products and Salesmen

- The following obtains the most problematic products and salesmen

```
prop.naQandV <- function(q,v)
    100*sum(is.na(q) & is.na(v))/length(q)
summarise(ids,nProbs=prop.naQandV(Quant,Val)) %>%
  arrange(desc(nProbs))
```

```
## # A tibble: 6,016 × 2
##         ID    nProbs
##     <fctr>     <dbl>
## 1   v1237 13.793103
## 2   v4254  9.523810
## 3   v4038  8.333333
## 4   v5248  8.333333
## 5   v3666  6.666667
## 6   v4433  6.250000
## 7   v4170  5.555556
## 8   v4926  5.555556
## 9   v4664  5.494505
## 10  v4642  4.761905
## # ... with 6,006 more rows
```

```
summarise(prods,nProbs=prop.naQandV(Quant,Val)) %>%
  arrange(desc(nProbs))
```

```
## # A tibble: 4,548 × 2
##       Prod   nProbs
##     <fctr>    <dbl>
## 1   p2689 39.28571
## 2   p2675 35.41667
## 3   p4061 25.00000
## 4   p2780 22.72727
## 5   p4351 18.18182
## 6   p2686 16.66667
## 7   p2707 14.28571
## 8   p2690 14.08451
## 9   p2691 12.90323
## 10  p2670 12.76596
## # ... with 4,538 more rows
```

Removing the Problematic Products and Salesmen

- Regards the salesmen if we remove the transactions we are removing a small percentage
- In terms of the products things are not that simple because some would have more than 20% of the transactions removed...
- Still, we will see that their unit price distribution is similar to other products
- Overall, the best strategy seems to be the removal

```
sales <- filter(sales,!(is.na(Quant) & is.na(Val)))
```

Handling the Remaining Unknowns

- Let us check the transactions with either the quantity or value unknown.
- The following obtains the proportion of transactions of each product that have the quantity unknown:

```
prop.nas <- function(x) 100*sum(is.na(x))/length(x)
propNAsQp <- summarise(prods,Proportion=prop.nas(Quant))
arrange(propNAsQp,desc(Proportion))

## # A tibble: 4,548 × 2
##      Prod Proportion
##    <fctr>      <dbl>
## 1   p2442  100.00000
## 2   p2443  100.00000
## 3   p1653   90.90909
## 4   p4101   85.71429
## 5   p4243   68.42105
## 6    p903   66.66667
## 7   p3678   66.66667
## 8   p4061   66.66667
## 9   p3955   64.28571
## 10  p4313   63.63636
## # ... with 4,538 more rows
```

NYU | STERN
MS in Business Analytics

Handling the Remaining Unknowns - 2

- Two products have all their transactions (54) with unknown values of the quantity!
- Lets delete them:

```
nlevels(sales$Prod)

## [1] 4548

sales <- droplevels(filter(sales,!(Prod %in% c("p2442", "p2443"))))
nlevels(sales$Prod)

## [1] 4546
```

NYU | STERN
MS in Business Analytics

Handling the Remaining Unknowns - 3

- Are there salesmen with all transactions with unknown quantity?
- Not so serious as long as we have other transactions of the same products

```
summarise(ids,Proportion=prop.nas(Quant)) %>%
    arrange(desc(Proportion))
```

```
## # A tibble: 6,016 × 2
##         ID Proportion
##     <fctr>      <dbl>
## 1    v2925  100.00000
## 2    v4356  100.00000
## 3    v5537  100.00000
## 4    v5836  100.00000
## 5    v6044  100.00000
## 6    v6058  100.00000
## 7    v6065  100.00000
## 8    v2923   90.00000
## 9    v4368   88.88889
## 10   v2920   85.71429
## # ... with 6,006 more rows
```

NYU | STERN
MS in Business Analytics

Handling the Remaining Unknowns - 4

- Let us move our attention to the unknowns on the `Val` column
- What is the proportion of transactions of each product with unknown value in this column?
- Reasonable numbers - no need to delete these rows, we may try to fill in these unknowns

```
summarise(prods,Proportion=prop.nas(Val)) %>%
    arrange(desc(Proportion))
```

```
## # A tibble: 4,548 × 2
##       Prod Proportion
##     <fctr>      <dbl>
## 1    p2689   39.28571
## 2    p2675   35.41667
## 3    p1110   25.00000
## 4    p4061   25.00000
## 5    p2780   22.72727
## 6    p4351   18.18182
## 7    p4491   18.18182
## 8    p2707   17.85714
## 9    p1462   17.77778
## 10   p1022   17.64706
## # ... with 4,538 more rows
```

NYU | STERN
MS in Business Analytics

Handling the Remaining Unknowns - 5

- Numbers for the salesmen:
- Again reasonable numbers

```
summarise(ids,Proportion=prop.nas(Val)) %>%
    arrange(desc(Proportion))
```

```
## # A tibble: 6,016 × 2
##         ID Proportion
##     <fctr>      <dbl>
## 1   v5647   37.500000
## 2     v74   22.222222
## 3   v5946   20.000000
## 4   v5290   15.384615
## 5   v4022   13.953488
## 6   v1237   13.793103
## 7   v4472   12.500000
## 8    v975    9.574468
## 9   v4254    9.523810
## 10  v2814    9.090909
## # ... with 6,006 more rows
```

Filling in the Remaining Unknowns

- Basic assumption : transactions of the same products should have a similar unit price
- Lets calculate this typical price for each product
  - We will skip the transaction found fraudulent on this calculation

```
tPrice <- filter(sales, Insp != "fraud") %>%
          group_by(Prod) %>%
          summarise(medianPrice = median(Uprice,na.rm=TRUE))
```

- We can use this information for filling in either the quantity or the value, given that we do not have any transaction with both values unknown any more
- This is possible because we know that $Uprice = \frac{Val}{Quant}$

Filling in the Remaining Unknowns - 2

- The following code fills in the remaining unknowns

```
noQuantMedPrices <- filter(sales, is.na(Quant)) %>%
    inner_join(tPrice) %>% select(medianPrice)
noValMedPrices <- filter(sales, is.na(Val)) %>%
    inner_join(tPrice) %>% select(medianPrice)

noQuant <- which(is.na(sales$Quant))
noVal <- which(is.na(sales$Val))
sales[noQuant,'Quant'] <- ceiling(sales[noQuant,'Val']/noQuantMedPrices)
sales[noVal,'Val'] <- sales[noVal,'Quant'] * noValMedPrices

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

Hands On Exploratory Analysis

- Get your hands on this case study by carrying out most of the steps that were illustrated and by trying alternative paths
- Try to tick these objectives:
  1. Explore the data and understand it
  2. Understand how **dplyr** works
  3. Take care of the unknown values
  4. In the end of your steps save the resulting object in a file for later usage
     `save(sales,file="mySalesObj.Rdata")`
- **Note:** you should produce a report from your analysis using R markdown Notebooks in RStudio

Different Approaches to the Data Mining Task

- The target variable (the `Insp` column) has three values: `ok`, `fraud` and `unkn`
- The value `unkn` means that those transactions were not inspected and thus may be either frauds or normal transactions (OK)
- In a way this represents an unknown value of the target variable
- In these contexts there are essentially 3 ways of addressing these tasks:
  - Using unsupervised techniques
  - Using supervised techniques
  - Using semi-supervised techniques

Unsupervised Techniques

- Assume complete ignorance of the target variable values
- Try to find the natural groupings of the data
- The reasoning is that fraudulent transactions should be different from normal transactions, i.e. should belong to different groups of data
- Using them would mean to throw away all information contained in column `Insp`

Supervised Techniques

- Assume all observations have a value for the target variable and try to approximate the unknown function $Y = f(X_1, \cdots, X_p)$ using the available data
- Using them would mean to throw away all data with value `Insp = unkn`, which are the majority,

```
100*sum(sales$Insp == "unkn")/nrow(sales)

## [1] 96.0705
```

NYU | STERN
MS in Business Analytics

Semi-supervised Techniques

- Try to take advantage of all existing data
- Two main approaches:
  1. Use unsupervised methods trying to incorporate the knowledge about the target variable (`Insp`) available in some cases, into the criteria guiding the formation of the groups
  2. Use supervised methods trying to “guess” the target variable of some cases, in order to increase the number of available training cases
- Not too many methods exist for this type of approach...

NYU | STERN
MS in Business Analytics

Goals of the Data Mining Task

- Inspection/auditing activities are typically constrained by limited resources
- **Inspecting all cases is usually not possible**
- In this context, the best outcome from a data mining tool is an inspection ranking
- This type of result allows users to allocate their (limited) resources to the most promising cases and stop when necessary

NYU | STERN
MS in Business Analytics

How to Evaluate the Results?

- Before checking how to obtain inspection rankings let us see how to evaluate this type of output
- Our application has an additional difficulty - labeled and unlabeled cases
- If an unlabeled case appears in the top positions of the inspection ranking how to know if this is correct?

NYU | STERN
MS in Business Analytics

Precision/Recall

- Frauds are rare events
- The Precision/Recall evaluation setting allows us to focus on the performance of a model on a specific class (in this case the frauds)
- Precision will be measured as the proportion of cases the models say are fraudulent that are effectively frauds
- Recall will be measured as the proportion of frauds that are signaled by the models as such

NYU STERN
MS in Business Analytics

Precision/Recall

- Usually there is a trade-off between precision and recall - it is easy to get 100% recall by forecasting all cases as frauds
- In a setting with limited inspection resources what we really aim at is to maximize the use of these resources
- Recall is the key issue on this application - we want to capture as many frauds as possible with our limited inspection resources

NYU STERN
MS in Business Analytics

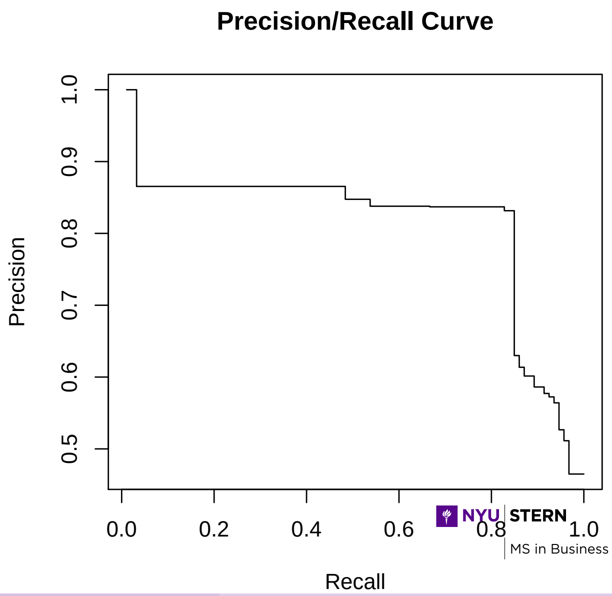
Precision/Recall Curves

- Precision/Recall curves show the value of precision for different values of recall
- Can be easily obtained for 2 class problems, where one is the target class (e.g. the fraud)
- Can be obtained with function `PRcurve()` from package `DMwR` (uses functions from package `ROCR`)
- Takes as arguments two equal-length vectors of the predicted and true positive class probabilities

Precision/Recall Curves - 2

```
library(ROCR)
data(ROCR.simple)
head(ROCR.simple$predictions)

## [1] 0.6125478 0.3642710 0.4321361 0.1402911 0.3848959 0.2444155

head(ROCR.simple$labels)

## [1] 1 1 0 0 0 1
```

```
library(DMwR)
PRcurve(ROCR.simple$predictions, ROCR.simple$labels,
        main="Precision/Recall Curve")
```



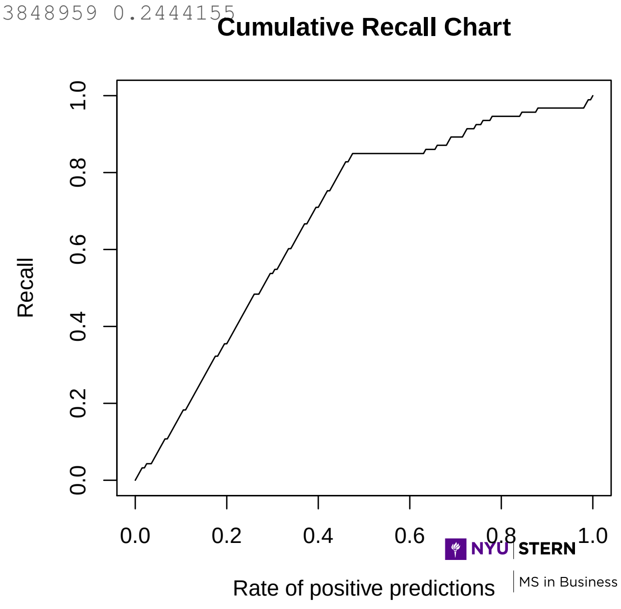
Lift Charts

- Lift charts provide a different perspective of the model predictions.
- They give more importance to the values of recall
- In the *x*-axis they have the rate of positive predictions (RPP), which is the probability that the model predicts a positive class
- In the *y*-axis they have the value of recall divided by the value of RPP
- Lift charts are almost what we want for our application
- What we want is to know the value of recall for different inspection effort levels (which is captured by RPP)
- We will call these graphs the Cumulative Recall chart, which are obtained by function `CRchart()` of package `DMwR` (again using functionalities of package `ROCR`)

Cumulative Recall Charts

```
library(ROCR)
data(ROCR.simple)
head(ROCR.simple$predictions)

## [1] 0.6125478 0.3642710 0.4321361 0.1402911 0.3848959 0.2444155

head(ROCR.simple$labels)

## [1] 1 1 0 0 0 1
```

```
CRchart(ROCR.simple$predictions, ROCR.simple$labels ,
        main='Cumulative Recall Chart')
```



Normalized Distance to Typical Price

- Previous measures only evaluate the quality of rankings in terms of the labeled transactions - supervised metrics
- The obtained rankings will surely contain unlabeled transaction reports - are these good inspection recommendations?
- **We can compare their unit price with the typical price of the reports of the same product**
- We would expect that the *difference between these prices is high*, as this **is an indication that something is wrong** with the report

NYU | STERN
MS in Business Analytics

Normalized Distance to Typical Price

- We propose to use the Normalized Distance to Typical Price ($NDTP_p$) to measure this difference

$$NDTP_p(u) = \frac{|u - \tilde{U}_p|}{IQR_p}$$

  where $\tilde{U}_p$ is the typical price of product p (measured by the median), and IQR_p is the respective inter-quartile range
- The higher the value of *NDTP* the stranger is the unit price of a transaction, and thus the better the recommendation for inspection.

NYU | STERN
MS in Business Analytics

Normalized Distance to Typical Price - 2

- The following function calculates the *NDTP*

```r
avgNDTP <- function(toInsp,train,stats) {
  if (missing(train) && missing(stats))
      stop('Provide either the training data or the product stats')
  if (missing(stats)) {
      stats <- as.matrix(filter(train,Insp != 'fraud') %>%
                         group_by(Prod) %>%
                         summarise(median=median(Uprice),iqr=IQR(Uprice)) %>%
                         select(median,iqr))
      rownames(stats) <- levels(train$Prod)
      stats[which(stats[,'iqr']==0),'iqr'] <- stats[which(stats[,'iqr']==0),'median']
  }

  return(mean(abs(toInsp$Uprice-stats[toInsp$Prod,'median']) /
                  stats[toInsp$Prod,'iqr']))
}
```

NYU | STERN
MS in Business Analytics

Experimental Methodology to Estimate the Evaluation Metrics

- Given the size of the data set, the Holdout method is the adequate methodology
- We will split the data in two partitions - 70% for training and 30% for testing
- We can repeat the random split several times to increase the statistical significance
- Our data set has a “problem” - class imbalance
- To avoid sampling bias we should use stratified sampling
- Stratified sampling consists in making sure the test sets have the same class distribution as the original data

NYU | STERN
MS in Business Analytics

Summary

- We will evaluate the inspection rankings proposed by the alternative models in terms of:
  1. Precision and Recall
  2. Normalized Distance to Typical Price (*NDTP*)
- We will obtain reliable estimates of these evaluation metrics using several repetitions of a 70%-30% Holdout with stratified sampling

Summary - 2

- The following function evaluates an inspection ranking proposal

```
evalOutlierRanking <- function(testSet,rankOrder,Threshold,statsProds,...)
{
   ordTS <- testSet[rankOrder,]
   N <- nrow(testSet)
   nF <- if (Threshold < 1) as.integer(Threshold*N) else Threshold
   cm <- table(c(rep('fraud',nF),rep('ok',N-nF)),ordTS$Insp)
   prec <- cm['fraud','fraud']/sum(cm['fraud',])
   rec <- cm['fraud','fraud']/sum(cm[,'fraud'])
   AVGndtp <- avgNDTP(ordTS[1:nF,],stats=statsProds)
   return(c(Precision=prec,Recall=rec,avgNDTP=AVGndtp))
}
```

Summary - 3

- We will call the previous function with the following information on the typical prices of each product (parameter `statsProds`)

```
globalStats <- as.matrix(filter(sales,Insp != 'fraud') %>%
                         group_by(Prod) %>%
                         summarise(median=median(Uprice),iqr=IQR(Uprice)) %>%
                         select(median,iqr))
rownames(globalStats) <- levels(sales$Prod)
globalStats[which(globalStats[,'iqr']==0), 'iqr'] <-
    globalStats[which(globalStats[,'iqr']==0), 'median']
head(globalStats,3)


##       median      iqr
## p1 11.34615 8.563580
## p2 10.87786 5.609731
## p3 10.00000 4.809092
```

The Modified Box-Plot Rule

- This simple baseline approach uses a box-plot rule to tag as suspicious transactions whose unit price is an outlier according to this rule
- This rule would tag each transaction as outlier or non-outlier - but we need outlier rankings
- We will use the values of NDTP to obtain a score of outlier and thus produce a ranking

Evaluation of the Modified Box-Plot Rule

- We will use the `performanceEstimation` package to estimate the scores of the selected metrics (precision, recall and NDTP)
- Estimates will be obtained using an Holdout experiment leaving 30% of the cases as test set
- We will repeat the (random) train+test split 3 times to increase the significance of our estimates
- We will calculate our metrics assuming that we can only audit 10% of the test set

NYU | STERN
MS in Business Analytics

The Workflow implementing the modified Box-Plot Rule

```r
BPrule.wf <- function(form,train,test,...) {
    require(dplyr, quietly=TRUE)
    ms <- as.matrix(filter(train,Insp != 'fraud') %>%
                    group_by(Prod) %>%
                    summarise(median=median(Uprice),iqr=IQR(Uprice)) %>%
                    select(median,iqr))
    rownames(ms) <- levels(train$Prod)
    ms[which(ms[,'iqr']==0),'iqr'] <- ms[which(ms[,'iqr']==0),'median']
    ORscore <- abs(test$Uprice-ms[test$Prod,'median']) /
               ms[test$Prod,'iqr']
    rankOrder <- order(ORscore,decreasing=T)
    res <- list(testSet=test,rankOrder=rankOrder,
                probs=matrix(c(ORscore,ifelse(test$Insp=='fraud',1,0)),
                             ncol=2))
    res
}
```

NYU | STERN
MS in Business Analytics

Evaluating the Modified Box-Plot Rule

```
library(performanceEstimation)
bp.res <- performanceEstimation(
    PredTask(Insp ~ .,sales),
    Workflow("BPrule.wf"),
    EstimationTask(metrics=c("Precision","Recall","avgNDTP"),
                   method=Holdout(nReps=3,hldSz=0.3,strat=TRUE),
                   evaluator="evalOutlierRanking",
                   evaluator.pars=list(Threshold=0.1,
                                       statsProds=globalStats))
    )
```

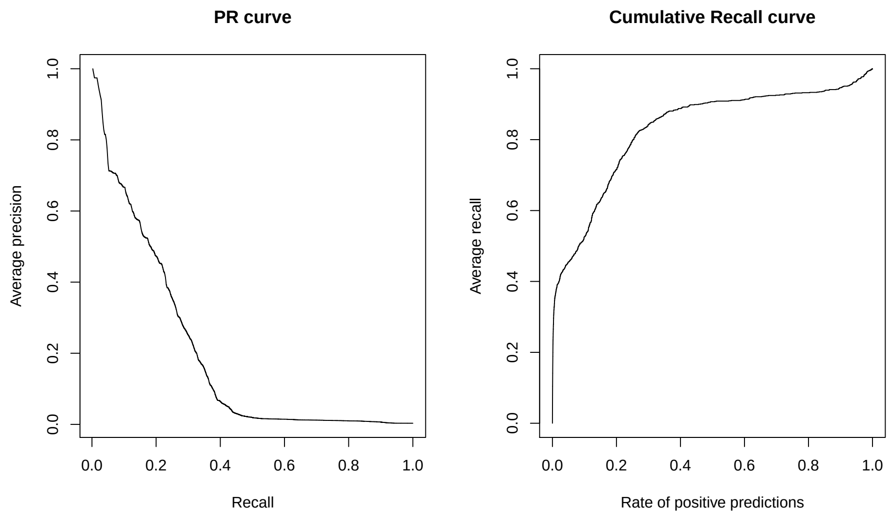
NYU | STERN
MS in Business Analytics

Inspecting the Results

```
summary(bp.res)

##
## == Summary of a  Hold Out Performance Estimation Experiment ==
##
## Task for estimating  Precision,Recall,avgNDTP  using
## Stratified  3 x 70 % / 30 % Holdout
##   Run with seed =  1234
##
## * Predictive Tasks ::  sales.Insp
## * Workflows  ::  BPrule.wf
##
## -> Task:  sales.Insp
##   *Workflow: BPrule.wf
##            Precision     Recall    avgNDTP
## avg     0.0166583375 0.52631579 11.1748886
## std     0.0006505289 0.02055329  0.9004590
## med     0.0163251707 0.51578947 10.7913476
## iqr     0.0005830418 0.01842105  0.8369579
## min     0.0162418791 0.51315789 10.5297012
## max     0.0174079627 0.55000000 12.2036171
## invalid 0.0000000000 0.00000000  0.0000000
```

Performance for Different Effort Levels

```
par(mfrow=c(1,2))
ps.bp <- sapply(getIterationsInfo(bp.res),function(i) i$probs[,1])
ts.bp <- sapply(getIterationsInfo(bp.res),function(i) i$probs[,2])
PRcurve(ps.bp,ts.bp,main="PR curve",avg="vertical")
CRchart(ps.bp,ts.bp,main='Cumulative Recall curve',avg='vertical')
```



The *LOF* Method

- *LOF* is probably the most well-know outlier detection algorithm.
- The result of *LOF* is a local outlier score for each case.
- This means that the result can be regarded as a kind of ranking of outlyingness of the data set
- *LOF* is based on the notion local distance of each case to its nearest neighbors
- Tries to analyze this distance in the context of the typical distance within this neighborhood
  - This way it can cope with data sets with different data densities

The *LOF* Method (2)

Key Notions in *LOF*

- The k-distance of an object o ($dist_k(o)$) is the distance from o to its kth nearest neighbor
- The k-distance neighborhood of o ($N_k(o)$) are the set of k nearest neighbors of o
- The *reachability-distance* of an object o with respect to another object p is defined as $reach.dist_k(o,p) = \max\{dist_k(p), d(o,p)\}$. If p and o are faraway from each other the reachability distance will be equal to their actual distance. If they are close to each other then this distance is substituted by $dist_k(p)$.

The *LOF* Method (3)

Key Notions in *LOF* (cont.)

- The *local reachability-distance* is the inverse of the average *reachability-distance* of its k-neighborhood,

$$lrd_k(o) = \frac{|N_k(o)|}{\sum_{p \in N_k(o)} reach.dist_k(o,p)}$$

The *LOF* Method (4)

The *LOF* score

The *LOF* score of an observation captures the degree to which we can consider it an outlier. It is given by,

$$LOF_k(o) = \frac{\sum_{p \in N_k(o)} \frac{lrd_k(o)}{lrd_k(p)}}{N_k(o)}$$

Breunig, M., Kriegel, H., Ng, R., and Sander, J. (2000). “LOF: identifying density-based local outliers”. In ACM Int. Conf. on Management of Data, pages 93-104.

NYU STERN
MS in Business Analytics

Using *LOF* to Obtain the Rankings

- The function `lofactor()` in package `DMwR` implements *LOF*
- The following implements a workflow using this function

```
LOF.wf <- function(form, train, test, k, ...) {
  require(DMwR2, quietly=TRUE)
  ntr <- nrow(train)
  all <- as.data.frame(rbind(train,test))
  N <- nrow(all)
  ups <- split(all$Uprice,all$Prod)
  r <- list(length=ups)
  for(u in seq(along=ups))
      r[[u]] <- if (NROW(ups[[u]]) > 3)
                  lofactor(ups[[u]],min(k,NROW(ups[[u]]) %/% 2))
                else if (NROW(ups[[u]])) rep(0,NROW(ups[[u]]))
                else NULL
  all$lof <- vector(length=N)
  split(all$lof,all$Prod) <- r
  all$lof[which(!(is.infinite(all$lof) | is.nan(all$lof)))] <-
      SoftMax(all$lof[which(!(is.infinite(all$lof) | is.nan(all$lof)))])

  res <- list(testSet=test,
              rankOrder=order(all[(ntr+1):N,'lof'],decreasing=TRUE),
              probs=as.matrix(cbind(all[(ntr+1):N,'lof'],
                                    ifelse(test$Insp=='fraud',1,0))))
  res
}
```

Evaluating the *LOF* Method

```
lof.res <- performanceEstimation(
    PredTask(Insp ~ .,sales),
    Workflow("LOF.wf",k=7),
    EstimationTask(metrics=c("Precision","Recall","avgNDTP"),
                   method=Holdout(nReps=3,hldSz=0.3,strat=TRUE),
                   evaluator="evalOutlierRanking",
                   evaluator.pars=list(Threshold=0.1,
                                       statsProds=globalStats))
    )
```

NYU | STERN
MS in Business Analytics

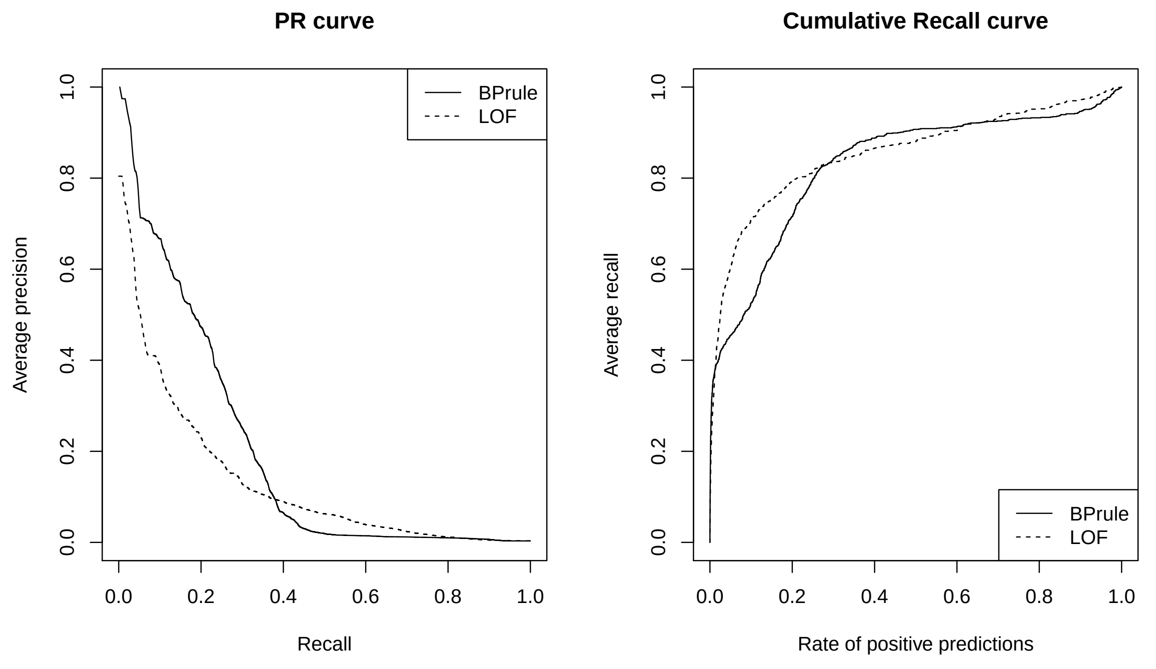
Inspecting the Results of *LOF*

```
summary(lof.res)

##
## == Summary of a  Hold Out Performance Estimation Experiment ==
##
## Task for estimating  Precision,Recall,avgNDTP  using
## Stratified  3 x 70 % / 30 % Holdout
##   Run with seed =  1234
##
## * Predictive Tasks ::  sales.Insp
## * Workflows  ::  LOF.wf
##
## -> Task:  sales.Insp
##   *Workflow: LOF.wf
##              Precision     Recall   avgNDTP
## avg     0.0221000611 0.69824561 8.7661376
## std     0.0006251502 0.01975146 0.9362724
## med     0.0220722972 0.69736842 8.4358879
## iqr     0.0006246877 0.01973684 0.8915197
## min     0.0214892554 0.67894737 8.0397428
## max     0.0227386307 0.71842105 9.8227821
## invalid 0.0000000000 0.00000000 0.0000000
```

LOF vs the Box-Plot Rule

```
par(mfrow=c(1,2))
ps.lof <- sapply(getIterationsInfo(lof.res), function(i) i$probs[,1])
ts.lof <- sapply(getIterationsInfo(lof.res), function(i) i$probs[,2])

PRcurve(ps.bp,ts.bp,main="PR curve",lty=1,
        xlim=c(0,1),ylim=c(0,1),avg="vertical")
PRcurve(ps.lof,ts.lof,add=T,lty=2,avg='vertical')
legend('topright',c('BPrule','LOF'),lty=c(1,2))

CRchart(ps.bp,ts.bp,main='Cumulative Recall curve',
        lty=1,xlim=c(0,1),ylim=c(0,1),avg='vertical')
CRchart(ps.lof,ts.lof,add=T,lty=2,avg='vertical')
legend('bottomright',c('BPrule','LOF'),lty=c(1,2))
```

LOF vs the Box-Plot Rule



Hands On Unsupervised Approaches

- Explore the solutions using unsupervised approaches that were described
- Try the following variants of these approaches
  1. Repeat the experiments with the BP rule and LOF, this time using 1% has the available inspection effort. Comment on the results.
  2. Create a new workflow using the boxplot rule that uses the mean and the standard deviation for calculating the outlier scores instead of the median and IQR. Run the experiments and check the results.
  3. Run the experiments with the LOF workflow using different values for the *k* parameter (e.g. 5 and 10). Check the impact on the results.

Classification Approaches

- If we consider only the `OK` and `fraud` cases we have a supervised classification task
- The particularity of this task is that we have a highly unbalanced class distribution

```
table(sales[sales$Insp != 'unkn','Insp'])/
    nrow(sales[sales$Insp != 'unkn',])

##
##        ok      unkn     fraud
## 0.9193692 0.0000000 0.0806308
```

Classification Approaches (2)

- This type of problems creates serious difficulties to most classification algorithms
- They will tend to focus on the prevailing class value
- The problem is that this class is the least important on this application !
- The are essentially 2 ways of addressing these problems
  1. Change the learning algorithms, namely their preference bias criteria
  2. Change the distribution of the data to balance it

Sampling Methods

SMOTE

- Sampling methods change the distribution of the data by sampling over the available data set
- The goal is to make the training sample more adequate for our objectives
- SMOTE is amongst the most successful sampling methods
  - SMOTE under-samples the majority class
  - SMOTE also over-samples the minority class creating several new cases by clever interpolation on the provided cases
  - The result is a less unbalanced data set
- Package **UBL** implements several sampling approaches including SMOTE, through function `SmoteClassif`
- The result of this function is a new (more balanced) data set

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. JAIR, 16:321-357.
P. Branco, L. Torgo and R. Ribeiro (2016). A Survey of Predictive Modeling on Imbalanced Domains. ACM Computing Surveys, 49 (2), 31.

P. Branco, R. Ribeiro and L. Torgo (2016). A UBL: an R package for Utility-based Learning. CoRR abs/1604.08079.

Bayesian Classification

- Bayesian classifiers are statistical classifiers - they predict the probability that a case belongs to a certain class
- Bayesian classification is based on the Bayes Theorem (next slide)
- A particular class of Bayesian classifiers - the Naive Bayes Classifier - has shown rather competitive performance on several problems even when compared to more “sophisticated” methods
- Naive Bayes is available in R on package **e1071**, through function `naiveBayes()`

The Bayes Theorem (1)

- Let D be a data set formed by n cases $\{\langle \mathbf{x}, y\rangle\}_{i=1}^{n}$, where $\mathbf{x}$ is a vector of p variable values and y is the value on a target nominal variable $Y \in \mathcal{Y}$
- Let H be a hypothesis that states that a certain test cases belongs to a class $c \in \mathcal{Y}$
- Given a new test case $\mathbf{x}$ the goal of classification is to estimate $P(H|\mathbf{x})$, i.e. the probability that H holds given the evidence $\mathbf{x}$
- More specifically, if $\mathcal{Y}$ is the domain of the target variable Y we want to estimate the probability of each of the possible values given the test case (evidence) $\mathbf{x}$

The Bayes Theorem (2)

- $P(H|\mathbf{x})$ is called the **posterior probability**, or *a posteriori probability*, of H conditioned on $\mathbf{x}$
- We can also talk about $P(H)$, the **prior probability**, or *a priori probability*, of the hypothesis H
- Notice that $P(H|\mathbf{x})$ is based on more information than $P(H)$, which is independent of the observation $\mathbf{x}$
- Finally, we can also talk about $P(\mathbf{x}|H)$ as the posterior probability of $\mathbf{x}$ conditioned on H

Bayes' Theorem

$$P(H|\mathbf{x}) = \frac{P(\mathbf{x}|H)P(H)}{P(\mathbf{x})}$$

The Naive Bayes Classifier

How it works?

- We have a data set D with cases belonging to one of m classes $c_1, c_2, \cdots, c_m$
- Given a new test case $\mathbf{x}$ this classifier produces as prediction the class that has the highest estimated probability, i.e. $\max_{i \in \{1,2,\cdots,m\}} P(c_i|\mathbf{x})$
- Given that $P(\mathbf{x})$ is constant for all classes, according to the Bayes Theorem the class with the highest probability is the one maximizing the quantity $P(\mathbf{x}|c_i)P(c_i)$

NYU | STERN
MS in Business Analytics

The Naive Bayes Classifier (2)

How it works? (cont.)

- The class priors $P(c_i)$'s are estimated from the training data as $|D_{c_i}|/|D|$, where $|D_{c_i}|$ is the number of cases belonging to class c_i
- Regards the quantities $P(\mathbf{x}|c_i)$'s the correct computation would be computationally very demanding. The Naive Bayes classifier simplifies this task by naively assuming *class condition independence*. This essentially resumes to assuming that there is no dependence relationship among the predictors of the problem. This independence allows us to use,

$$P(\mathbf{x}|c_i) = \prod_{k=1}^{p} P(x_k|c_i)$$

- Note that the quantities $P(x_1|c_i), P(x_2|c_i), \cdots, P(x_p|c_i)$ can be easily estimated from the training data

Using Naive Bayes for Our Task together with SMOTE

```
NBsm.wf <- function(form,train,test,C.perc="balance",dist="HEOM",...) {
    require(e1071,quietly=TRUE)
    require(UBL,quietly=TRUE)

    sup <- which(train$Insp != 'unkn')
    data <- as.data.frame(train[sup,c('ID','Prod','Uprice','Insp')])
    data$Insp <- factor(data$Insp,levels=c('ok','fraud'))
    newData <- SmoteClassif(Insp ~ .,data,C.perc=C.perc,dist=dist,...)
    model <- naiveBayes(Insp ~ .,newData)
    preds <- predict(model,test[,c('ID','Prod','Uprice','Insp')],type='raw')
    rankOrder <- order(preds[,'fraud'],decreasing=T)
    rankScore <- preds[,'fraud']

    res <- list(testSet=test,
                rankOrder=rankOrder,
                probs=as.matrix(cbind(rankScore,
                                      ifelse(test$Insp=='fraud',1,0))))
    res
}
```

MS in Business Analytics

Evaluating the Naive Bayes model

```
nbs.res <- performanceEstimation(
    PredTask(Insp ~ ., sales),
    Workflow("NBsm.wf"),
    EstimationTask(metrics = c("Precision","Recall","avgNDTP"),
                   method = Holdout(nReps=3, hldSz=0.3, strat=TRUE),
                   evaluator = "evalOutlierRanking",
                   evaluator.pars = list(Threshold=0.1,
                                         statsProds=globalStats))
    )
```

Inspecting the Results

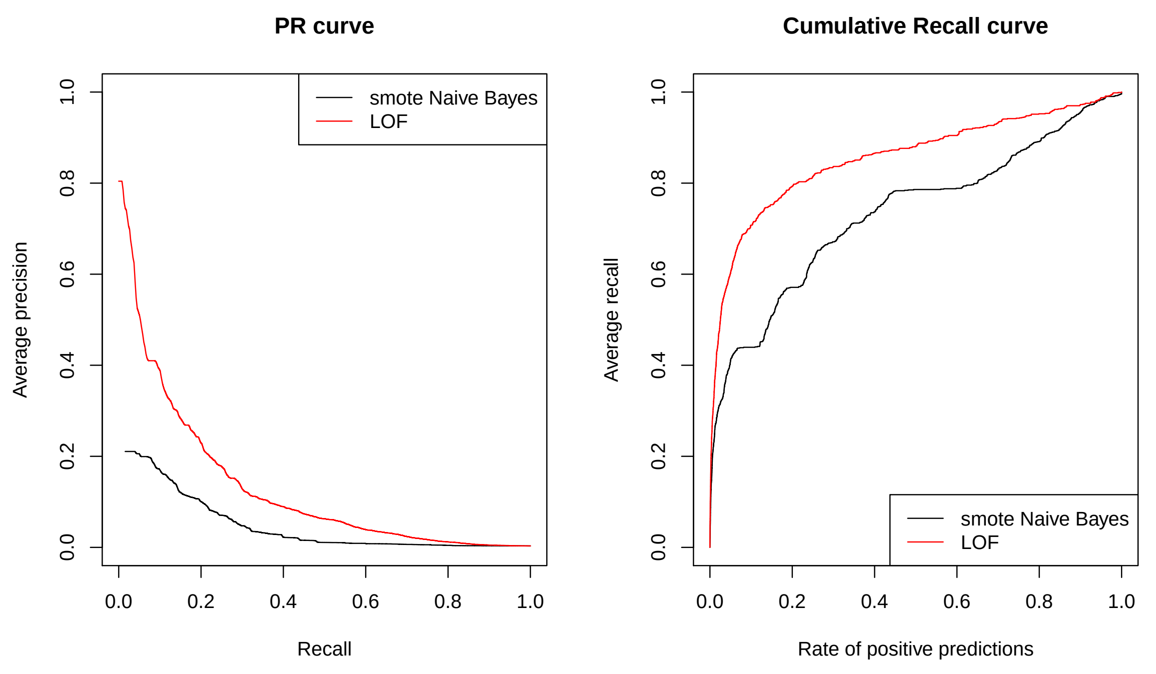
```
summary(nbs.res)

##
## == Summary of a  Hold Out Performance Estimation Experiment ==
##
## Task for estimating  Precision,Recall,avgNDTP  using
## Stratified  3 x 70 % / 30 % Holdout
##   Run with seed =  1234
##
## * Predictive Tasks ::  sales.Insp
## * Workflows  ::  NBsm.wf
##
## -> Task:  sales.Insp
##   *Workflow: NBsm.wf
##            Precision     Recall   avgNDTP
## avg     0.013909712 0.43947368 6.4017351
## std     0.001332667 0.04210526 0.8850754
## med     0.013909712 0.43947368 6.0882813
## iqr     0.001332667 0.04210526 0.8424183
## min     0.012577045 0.39736842 5.7160437
## max     0.015242379 0.48157895 7.4008803
## invalid 0.000000000 0.00000000 0.0000000
```

Performance for Different Effort Levels

```r
par(mfrow=c(1,2))
ps.nbs <- sapply(getIterationsInfo(nbs.res), function(i) i$probs[,1])
ts.nbs <- sapply(getIterationsInfo(nbs.res), function(i) i$probs[,2])
PRcurve(ps.nbs,ts.nbs,main="PR curve",lty=1,xlim=c(0,1),
        ylim=c(0,1),avg="vertical")
PRcurve(ps.lof,ts.lof,add=T,lty=1,col="red",avg='vertical')
legend('topright',c('smote Naive Bayes','LOF'),lty=c(1,1),col=c("black","red"))
CRchart(ps.nbs,ts.nbs,main='Cumulative Recall curve',
        lty=1,xlim=c(0,1),ylim=c(0,1),avg='vertical')
CRchart(ps.lof,ts.lof,add=T,lty=1,col="red",avg='vertical')
legend('bottomright',c('smote Naive Bayes','LOF'),lty=c(1,1),col=c("black","red"))
```

NYU | STERN
MS in Business Analytics

Performance for Different Effort Levels (2)



NYU | STERN
MS in Business Analytics

The AdaBoost Algorithm

- The AdaBoost method is a boosting algorithm that belongs to the class of Ensemble methods
- Boosting was developed with the goal of answering the question: Can a set of weak learners create a single strong learner?
- In the above question a “weak” learner is a model that alone has very poor predictive performance

Rob Schapire (1990). Strength of Weak Learnability. Machine Learning Vol. 5, pages 197–227.

NYU | STERN
MS in Business Analytics

The AdaBoost Algorithm (2)

- Boosting algorithms work by iteratively creating a strong learner by adding at each iteration a new weak learner to make the ensemble
- Weak learners are added with weights that reflect the learner’s accuracy
- After each addition the data is re-weighted such that cases that are still poorly predicted gain more weight for the next iteration
- This means that each new weak learner will focus on the errors of the previous ones

Rob Schapire (1990). Strength of Weak Learnability. Machine Learning Vol. 5, pages 197–227.

NYU | STERN
MS in Business Analytics

The AdaBoost Algorithm (3)

- AdaBoost (Adaptive Boosting) is an ensemble algorithm that can be used to improve the performance of a base algorithm
- It consists of an iterative process where new models are added to form an ensemble
- It is adaptive in the sense that at each new iteration of the algorithm the new models are built to try to overcome the errors made in the previous iterations
- At each iteration the weights of the training cases are adjusted so that cases that were wrongly predicted get their weight increased to make new models focus on accurately predicting them
- AdaBoost was created for classification although variants for regression exist

Y. Freund and R. Schapire (1996). Experiments with a new boosting algorithm, in Proc. of 13th International Conference on Machine Learning

The AdaBoost Algorithm

- The goal of the algorithm is to reach a form of additive model composed of k weak models

$$H(\mathbf{x}_i) = \sum_k w_k h_k(\mathbf{x}_i)$$

  where w_k is the weight of the weak model $h_k(\mathbf{x}_i)$
- All training cases start with a weight equal to $d_1(\mathbf{x}_i) = 1/n$, where n is the training set size

The AdaBoost Algorithm

- AdaBoost is available in several R packages
- The package `adabag` is an example with the function `boosting()`
- The packages `gbm` and `xgboost` include implementations of boosting for regression tasks
- We will use the package `RWeka` that provides the function `AdaBoost.M1()`

Using AdaBoost for Our Task

```r
ab.wf <- function(form,train,test,ntrees=100,...) {
    require(RWeka,quietly=TRUE)
    sup <- which(train$Insp != 'unkn')
    data <- as.data.frame(train[sup,c('ID','Prod','Uprice','Insp')])
    data$Insp <- factor(data$Insp,levels=c('ok','fraud'))
    model <- AdaBoostM1(Insp ~ .,data,
                        control=Weka_control(I=ntrees))
    preds <- predict(model,test[,c('ID','Prod','Uprice','Insp')],
                     type='probability')
    rankOrder <- order(preds[,"fraud"],decreasing=TRUE)
    rankScore <- preds[,"fraud"]

    res <- list(testSet=test,
                rankOrder=rankOrder,
                probs=as.matrix(cbind(rankScore,
                                      ifelse(test$Insp=='fraud',1,0))))
    res
}
```

Evaluating the AdaBoost model

```
ab.res <- performanceEstimation(
    PredTask(Insp ~ .,sales),
    Workflow("ab.wf"),
    EstimationTask(metrics=c("Precision","Recall","avgNDTP"),
                   method=Holdout(nReps=3,hldSz=0.3,strat=TRUE),
                   evaluator="evalOutlierRanking",
                   evaluator.pars=list(Threshold=0.1,
                                       statsProds=globalStats))
    )
```

NYU STERN
MS in Business Analytics

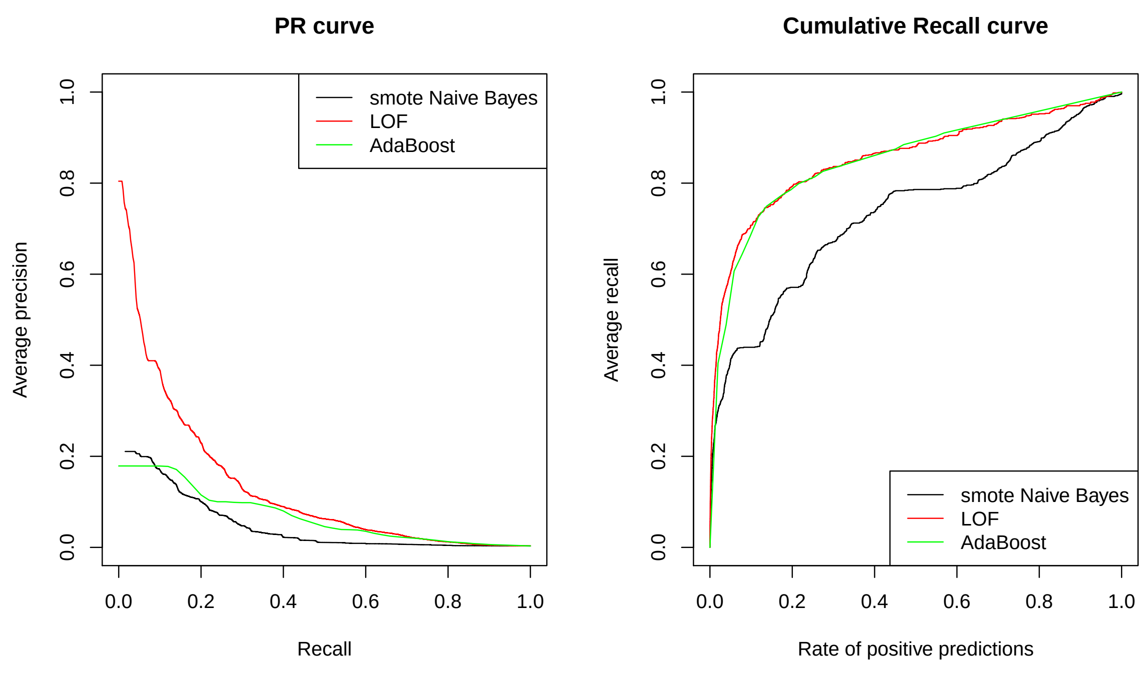
Inspecting the Results

```
summary(ab.res)

##
## == Summary of a  Hold Out Performance Estimation Experiment ==
##
## Task for estimating  Precision,Recall,avgNDTP  using
## Stratified  3 x 70 % / 30 % Holdout
##   Run with seed =  1234
##
## * Predictive Tasks ::  sales.Insp
## * Workflows  ::  ab.wf
##
## -> Task:  sales.Insp
##   *Workflow: ab.wf
##            Precision     Recall   avgNDTP
## avg     0.0204897551 0.64736842 7.0543305
## std     0.0004997501 0.01578947 0.3744884
## med     0.0204897551 0.64736842 6.9309492
## iqr     0.0004997501 0.01578947 0.3589210
## min     0.0199900050 0.63157895 6.7571001
## max     0.0209895052 0.66315789 7.4749422
## invalid 0.0000000000 0.00000000 0.0000000
```

Performance for Different Effort Levels

```
par(mfrow=c(1,2))
ps.ab <- sapply(getIterationsInfo(ab.res), function(i) i$probs[,1])
ts.ab <- sapply(getIterationsInfo(ab.res), function(i) i$probs[,2])
PRcurve(ps.nbs,ts.nbs,main="PR curve",lty=1,xlim=c(0,1),
        ylim=c(0,1),avg="vertical")
PRcurve(ps.lof,ts.lof,add=T,lty=1,col="red",avg='vertical')
PRcurve(ps.ab,ts.ab,add=T,lty=1,col="green",avg='vertical')
legend('topright',c('smote Naive Bayes','LOF','AdaBoost'),
       lty=1,col=c("black","red","green"))
CRchart(ps.nbs,ts.nbs,main='Cumulative Recall curve',
        lty=1,xlim=c(0,1),ylim=c(0,1),avg='vertical')
CRchart(ps.lof,ts.lof,add=T,lty=1,col="red",avg='vertical')
CRchart(ps.ab,ts.ab,add=T,lty=1,col="green",avg='vertical')
legend('bottomright',c('smote Naive Bayes','LOF','AdaBoost'),
       lty=1,col=c("black","red","green"))
```

Performance for Different Effort Levels (2)



Hands On Classification Approaches

- Explore the solutions using classification approaches that were described
- Try the following variants of these approaches
  1. The workflow using Naive Bayes generates a more balanced distribution through SMOTE. Explore different settings of SMOTE and check the impact on the results. Suggestion: check the help page of the function `SmoteClassif()`
  2. Modify the Naive Bayes workflow to completely eliminate SMOTE, i.e. use the original unbalanced data. Check and comment on the results.
  3. Change the number of trees used in the ensemble obtained with the adaboost workflow. Check the results. Tip: try a large number!

Semi-Supervised Approaches

Self Training

- Self training is a generic semi-supervised method that can be applied to any classification method
- It is an iterative process consisting of the following main steps:
  1. Build a classifier with the current labeled data
  2. Use the model to classify the unlabeled data
  3. Select the highest confidence classifications and add the respective cases with that classification to the labeled data
  4. Repeat the process until some criteria are met
- The last classifier of this iterative process is the result of the semi-supervised process
- Function `SelfTrain()` on package `DMwR2` implements these ideas for any probabilistic classifier

Trying to Improve AdaBoost with Self Training

```
pred.ada <- function(m,d) {
    p <- predict(m,d,type='probability')
    data.frame(cl=colnames(p)[apply(p,1,which.max)],
               p=apply(p,1,max)
               )
}
ab.st.wf <- function(form,train,test,ntrees=100,...) {
    require(RWeka,quietly=TRUE)
    require(DMwR2,quietly=TRUE)
    train <- as.data.frame(train[,c('ID','Prod','Uprice','Insp')])
    train[which(train$Insp == 'unkn'),'Insp'] <- NA
    train$Insp <- factor(train$Insp,levels=c('ok','fraud'))
    model <- SelfTrain(form,train,
                       learner='AdaBoostM1',
                       learner.pars=list(control=Weka_control(I=ntrees)),
                       pred='pred.ada')
    preds <- predict(model,test[,c('ID','Prod','Uprice','Insp')],
                     type='probability')

    rankOrder <- order(preds[,'fraud'],decreasing=T)
    rankScore <- preds[,"fraud"]

    res <- list(testSet=test,
              rankOrder=rankOrder,
              probs=as.matrix(cbind(rankScore,
                                    ifelse(test$Insp=='fraud',1,0))))
    res
}
```

MS in Business Analytics

Evaluating the Self Trained AdaBoost model

```
ab.st.res <- performanceEstimation(
    PredTask(Insp ~ .,sales),
    Workflow("ab.st.wf"),
    EstimationTask(metrics=c("Precision","Recall","avgNDTP"),
                   method=Holdout(nReps=3,hldSz=0.3,strat=TRUE),
                   evaluator="evalOutlierRanking",
                   evaluator.pars=list(Threshold=0.1,
                                       statsProds=globalStats))
    )
```

NYU | STERN
MS in Business Analytics

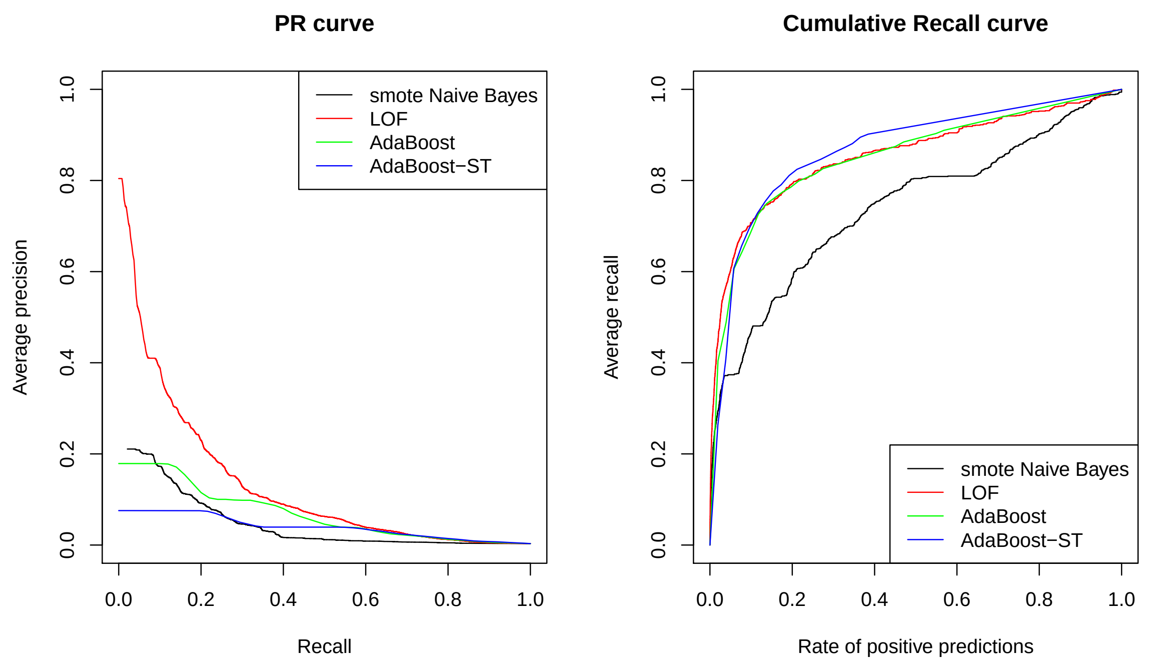
Inspecting the Results

```
summary(ab.st.res)

##
## == Summary of a  Hold Out Performance Estimation Experiment ==
##
## Task for estimating  Precision,Recall,avgNDTP  using
## Stratified  3 x 70 % / 30 % Holdout
##   Run with seed =  1234
##
## * Predictive Tasks ::  sales.Insp
## * Workflows  ::  ab.st.wf
##
## -> Task:  sales.Insp
##   *Workflow: ab.st.wf
##            Precision     Recall   avgNDTP
## avg     0.021294908 0.67280702 7.9596032
## std     0.001087055 0.03434521 0.8194148
## med     0.021655839 0.68421053 7.8507389
## iqr     0.001041146 0.03289474 0.8139729
## min     0.020073297 0.63421053 7.2000624
## max     0.022155589 0.70000000 8.8280083
## invalid 0.000000000 0.00000000 0.0000000
```

Performance for Different Effort Levels

```
par(mfrow=c(1,2))

ps.ab.st <- sapply(getIterationsInfo(ab.st.res), function(i) i$probs[,1])
ts.ab.st <- sapply(getIterationsInfo(ab.st.res), function(i) i$probs[,2])

PRcurve(ps.nbs,ts.nbs,main="PR curve",lty=1,xlim=c(0,1),
        ylim=c(0,1),avg="vertical")
PRcurve(ps.lof,ts.lof,add=T,lty=1,col="red",avg='vertical')
PRcurve(ps.ab,ts.ab,add=T,lty=1,col="green",avg='vertical')
PRcurve(ps.ab.st,ts.ab.st,add=T,lty=1,col="blue",avg='vertical')
legend('topright',c('smote Naive Bayes','LOF','AdaBoost','AdaBoost-ST'),
       lty=1,col=c("black","red","green","blue"))

CRchart(ps.nbs,ts.nbs,main='Cumulative Recall curve',
        lty=1,xlim=c(0,1),ylim=c(0,1),avg='vertical')
CRchart(ps.lof,ts.lof,add=T,lty=1,col="red",avg='vertical')
CRchart(ps.ab,ts.ab,add=T,lty=1,col="green",avg='vertical')
CRchart(ps.ab.st,ts.ab.st,add=T,lty=1,col="blue",avg='vertical')
legend('bottomright',c('smote Naive Bayes','LOF','AdaBoost','AdaBoost-ST'),
       lty=1,col=c("black","red","green","blue"))
```

Performance for Different Effort Levels



Summary

- We have seen a concrete example of a relevant data mining application domain - fraud detection
- We have covered a few relevant data mining topics for fraud detection:
  - Outlier detection and ranking
  - Imbalanced class distributions and methods to handle this
  - Semi-supervised classification methods

Hands On Semi-Supervised Approaches

- Explore the solutions using the self trained AdaBoost model
- Try the following variants
  1. Explore the help page of function `SelfTrain()` and try different settings on the workflow of AdaBoost