Relatório de Análise e Detecção de Fraude

Lucas Carvalho da Luz Moura

13 de setembro de 2025

Contents

1 1. Objetivo e Metodologia

Este relatório detalha o processo de análise de dados para detecção de fraude em transações comerciais. O objetivo principal é desenvolver e avaliar modelos que possam produzir um ranking de transações ordenadas pela probabilidade de serem fraudulentas [fonte: 9, 10].

O conjunto de dados apresenta um desafio comum em detecção de fraude:

- **Dados Rotulados (Minoria):** Um pequeno subconjunto de transações foi inspecionado e rotulado como 'OK' (legítimo) ou 'Fraud' (fraudulento) [fonte: 13, 14].
- **Dados Não Rotulados (Maioria):** A grande maioria das transações tem o status 'unkn' (desconhecido) [fonte: 16].

- **Desequilíbrio de Classe:** Dentro dos dados rotulados, os casos de fraude são eventos raros em comparação com os casos normais [fonte: 341].

Três abordagens metodológicas foram exploradas (conforme *Fraudulent-Transactions-2.R*):

1. **Não Supervisionadas (Detecção de Anomalia):** Fraudes como anomalias ou outliers [fonte: 26-28].

2. **Supervisionadas (Classificação):** Apenas dados rotulados para treinar um classificador binário [fonte: 30-32].

3. **Semissupervisionadas (Self-Training):** Combinação de dados rotulados e não rotulados [fonte: 351-355].

Os modelos **Árvore de Decisão (ctree)** [fonte: 999] e **Random Forest (randomForest)** [fonte: 11, 1001] formam a base para essas abordagens.

---

2 2. Carregamento de Pacotes e Dados

```
# Instalar pacotes (se necessário)
# install.packages("party")
# install.packages("randomForest")
# install.packages("ROCR")
# install.packages("DMwR")
# install.packages("e1071")
# install.packages("RWeka")

# Carregar bibliotecas
library(party)
library(randomForest)
library(ROCR)
library(DMwR)
library(e1071)
library(RWeka)

# Carregar dados pré-processados
load("workspace-june-27.RData")
```

---

3 3. Métricas de Avaliação e Funções Auxiliares

Devido ao desequilíbrio de classe, focamos em **Precisão** e **Recall**.
Os scripts fornecem funções auxiliares para métricas de negócio e gráficos.

```r
# Curva PR (Precision-Recall)
PRcurve <- function(preds, trues, ...) {
  require(ROCR, quietly = TRUE)
  pd <- prediction(preds, trues)
  pf <- performance(pd, 'prec', 'rec')
  pf@y.values <- lapply(pf@y.values, function(x) rev(cummax(rev(x))))
  plot(pf, ...)
}

# Recall Cumulativo (Lift Chart)
CRchart <- function(preds, trues, ...) {
  require(ROCR, quietly = TRUE)
  pd <- prediction(preds, trues)
  pf <- performance(pd, 'rec', 'rpp')
  plot(pf, ...)
}

# Distância Normalizada Média do Preço Típico (NDTP)
avgNDTP <- function(toInsp, stats) {
  stats[which(stats[, 'iqr'] == 0), 'iqr'] <- stats[which(stats[, 'iqr'] == 0), 'median']
  mdtp <- mean(abs(toInsp$Uprice - stats[toInsp$Prod, 'median']) / stats[toInsp$Prod, 'iqr'])
  return(mdtp)
}

# Avaliação de ranking
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 <- if ("fraud" %in% colnames(cm)) cm['fraud', 'fraud'] / sum(cm['fraud', ]) else 0
  rec <- if ("fraud" %in% colnames(cm)) cm['fraud', 'fraud'] / sum(cm[, 'fraud']) else 0

  AVGndtp <- avgNDTP(ordTS[1:nF, ], stats = statsProds)

  return(c(Precision = prec, Recall = rec, avgNDTP = AVGndtp))
}
```

---

4 4. Modelos de Exemplo (Iris Dataset)

4.1 4.1 Árvore de Decisão Condicional (ctree)

```r
set.seed(1234)
set <- sample(2, nrow(iris), replace = TRUE, prob = c(0.7, 0.3))
trainData.iris <- iris[set == 1,]
testData.iris <- iris[set == 2,]
```

```r
iris.ctree <- ctree(Species ~ ., data = trainData.iris)
table(predict(iris.ctree), trainData.iris$Species)

plot(iris.ctree)
```

4.2 4.2 Random Forest

```r
tree.set <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
trainData.rf <- iris[tree.set==1,]
testData.rf <- iris[tree.set==2,]

rf <- randomForest(Species ~ ., data = trainData.rf,
                   ntree = 100, proximity = TRUE)

print(rf)
plot(rf)

irisPred <- predict(rf, newdata = testData.rf)
table(irisPred, testData.rf$Species)
```

---

5 5. Modelos de Detecção de Fraude

Usamos a função `holdOut` do **DMwR** para comparar Recall e avgNDTP.

5.1 5.1 Abordagens Não Supervisionadas

```r
# Modelo 1: Box Plot Rule
bp.res <- holdOut(learner('ho.BPrule',
                          pars=list(Threshold=0.1, statsProds=globalStats)),
                  dataset(Insp ~ .,sales),
                  hldSettings(3,0.3,1234,T), itsInfo=TRUE)
summary(bp.res)

# Modelo 2: LOF
lof.res <- holdOut(learner('ho.LOF',
                           pars=list(k=7, Threshold=0.1, statsProds=globalStats)),
                   dataset(Insp ~ .,sales),
                   hldSettings(3,0.3,1234,T), itsInfo=TRUE)
summary(lof.res)

# Modelo 3: Clustering (ORh)
orh.res <- holdOut(learner('ho.ORh',
                           pars=list(Threshold=0.1, statsProds=globalStats)),
                   dataset(Insp ~ .,sales),
                   hldSettings(3,0.3,1234,T), itsInfo=TRUE)
summary(orh.res)
```

5.2 5.2 Supervisionadas e SMOTE

```
# Modelo 4: Naive Bayes
nb.res <- holdOut(learner('ho.nb',
                          pars=list(Threshold=0.1, statsProds=globalStats)),
                  dataset(Insp ~ .,sales),
                  hldSettings(3,0.3,1234,T), itsInfo=TRUE)
summary(nb.res)

# Modelo 5: Naive Bayes + SMOTE
nbs.res <- holdOut(learner('ho.nbs',
                           pars=list(Threshold=0.1, statsProds=globalStats)),
                   dataset(Insp ~ .,sales),
                   hldSettings(3,0.3,1234,T), itsInfo=TRUE)
summary(nbs.res)

# Modelo 6: AdaBoostM1
ab.res <- holdOut(learner('ho.ab',
                          pars=list(Threshold=0.1, statsProds=globalStats)),
                  dataset(Insp ~ .,sales),
                  hldSettings(3,0.3,1234,T), itsInfo=TRUE)
summary(ab.res)
```

5.3 5.3 Semissupervisionadas (Self-Training)

```
# Modelo 7: NB com Self-Training
nb.st.res <- holdOut(learner('ho.nb.st',
                             pars=list(Threshold=0.1, statsProds=globalStats)),
                     dataset(Insp ~ .,sales),
                     hldSettings(3,0.3,1234,T), itsInfo=TRUE)
summary(nb.st.res)

# Modelo 8: AdaBoost com Self-Training
ab.st.res <- holdOut(learner('ho.ab.st',
                             pars=list(Threshold=0.1, statsProds=globalStats)),
                     dataset(Insp ~ .,sales),
                     hldSettings(3,0.3,1234,T), itsInfo=TRUE)
summary(ab.st.res)
```

---

6 6. Abordagem Alternativa: IA Moderna (Pipeline Python)

Fluxo de trabalho proposto com **Scikit-learn e XGBoost**:

1. Engenharia de características (incluir NDTP como feature).
2. Pré-processamento unificado via `ColumnTransformer`.

3. Pipeline com **SMOTE** dentro da validação cruzada.

4. Modelo central: **XGBClassifier (XGBoost)**.

5. Avaliação principal: **AUC-PR (Average Precision)**.

---

7 7. Comparação de Abordagens

- O script em R explora paradigmas separados: **anomalias (LOF, BPrule)**, **supervisionados (Naive Bayes, AdaBoost)** e **semissupervisionados (Self-Training)**.

- A abordagem moderna em Python concentra-se em **um pipeline otimizado único**, geralmente baseado em **Gradient Boosting (XGBoost/LightGBM)**, considerado **estado da arte** em dados tabulares.