PROJECT

Balanceamento de Classes em Dados de Fraudes Financeiras com ROSE (Random OverSampling Examples)

1. Working Directory

Configurando o diretório de trabalho

setwd("C:/Users/Utilizador/repos/Formacao_cientista_de_dados/big_data_analytics_R_microsoft_azure_machine_lear
getwd()

2. Imports install.packages("data.table") install.packages("C50") install.packages("ROSE") install.packages("caret") library(data.table) library(C50) library(caret) library(ROCR) library(pROC) library(ROSE)

3. Data Loading ?fread dados <- fread("creditcard.csv", stringsAsFactors = F, sep = ",", header =T)

Resumo dos dados

A coluna Class indica se a transação foi fraudulenta ou não, sendo portanto nossa variável target

0 indica que a transação não é fraudulenta

1 indica que a transação é fraudulenta

Visualizando os dados

View(dados)

dim(dados)

str(dados)

4.Data Cleaning

4.1 Check NA

Verificando se temos valores ausentes

sum(is.na(dados))

5. Pre Processing

5.1 Class distribution

Distribuição de classe: este dataset esta com as classes completamente desbalanceada;

Temos cerca de 99% de registros para a classe 0 e menos de 1% para a classe 1;

prop.table(table(dados$Class))

5.2 Plot Class

Graficamente a diferença fica clara

barplot(prop.table(table(dados$Class)))

6. Modeling

6.1 Test and Training Data

Vamos dividir os dados em treino e teste, sendo 70% para dados de treino e 30% para dados de teste

set.seed(7) linhas <- sample(1:nrow(dados), 0.7 * nrow(dados))

dataset treino: 70%

dados_treino <- dados[linhas,]

dataset teste: 30%

dados_teste <- dados[-linhas,]

Vejamos como está agora a distribuição da nossa classe nos dados de treino e de teste

Proporção dados de treino

prop.table(table(dados_treino$Class))

Proporção dados de teste

prop.table(table(dados_teste$Class))

Observação: aqui fica claro que há discrepancia nos dados;

Em geral, não devemos entregar os dados assim a um modelo de Machine Learning (a menos que tenhamos um objetivo para isso);

Neste projeto desejamos comparar os valores entre os dataset: balanceado e não balanceado;

Então, vamos primeiro criar um modelo de Machine Learning com os dados desbalanceados e comparar com o resultado depois do balanceamento de classe.

6.2 Convert Class to Factor Type

Vamos converter a classe para o tipo fator, pois isso é necessário para o treinamento do modelo de classificação;

Se não fizermos a conversão, a variável fica como tipo int e o algoritmo vai achar que queremos criar um modelo de regressão.

antes da conversão

str(dados_treino$Class)$str(dados_teste$Class)

conversão

dados_treino$Class <- as.factor(dados_treino$Class) dados_teste$Class <- as.factor(dados_teste$Class)

depois conversão

str(dados_treino$Class$)str(dados_teste$Class)

6.3 Creat Model Unbalanced

Vamos criar um modelo antes de aplicar o balanceamento de classe.

O algoritmo C5.0 cria um modelo de árvore de decisão e é estudado em detalhes no curso de Machine Learning da FCD

?C5.0

Cria o modelo com dados de treino não balanceados

modelo_v1 <- C5.0(Class ~ ., data = dados_treino)

6.3.1 Prevision Model Unbalanced

Agora fazemos previsões com o modelo usando dados de teste

previsoes_v1 <- predict(modelo_v1, dados_teste)

6.3.2 Confusion Matrix

Criamos a Confusion Matrix e analisamos a acurácia do modelo

O parâmetro positive = '1' indica que a classe 1 é a positiva, ou seja, indica que sim, a transação é fraudulenta

?caret::confusionMatrix caret::confusionMatrix(dados_teste$Class, previsoes_v1, positive = '1')

6.3.3 ROC curve

Agora criamos a Curva ROC para encontrar a métrica AUC, conforme indicado no manual em pdf

roc.curve(dados_teste$Class, previsoes_v1, plotit = T, col = "red")

6.3.4 Result

Acurácia = 0.999

Score AUC = 0.759

Como base somente na acurácia, o modelo estaria excelente;

Entretanto, a Score AUC mostra que não é bem assim;

Vamos executar a ROSE para esclarecer;

6.3.5 ROSE

Aplicando ROSE (Random OverSampling Example)

Com ROSE conseguimos balancear as clases usando a técnica de Oversampling;

?ROSE

ATENÇÃO: O IDEAL É SEMPRE APLICAR O DESBALANCEAMENTO DEPOIS DE FAZER A DIVISÃO DOS DADOS EM TREINO E TESTE.

Se fazemos antes, o padrão usado para aplicar o oversampling será o mesmo nos dados de treino e de teste e, assim,

a avaliação do modelo fica comprometida.

6.3.5.1 ROSE: TREIN

Aplicando ROSE em dados de treino e checando a proporção de classes

rose_treino <- ROSE(Class ~ ., data = dados_treino, seed = 1)$data prop.table(table(rose_treino$Class))

Resposta: Conseguimos uma proporção quase 50/50 para as duas classes. Não pecisa ser exatamente assim, mas ficou muito bom!

6.3.5.2 ROSE: TEST

Aplicando ROSE em dados de teste e checando a proporção de classes

rose_teste <- ROSE(Class ~ ., data = dados_teste, seed = 1)$dataprop.table(table($rose_teste$Class))

6.4 Creat Model Balanced

Cria o modelo com dados de treino balanceados

modelo_v2 <- C5.0(Class ~ ., data = rose_treino)

6.4.1 Prevision Model Balanced

E fazemos previsões usando dados de teste balanceados

previsoes_v2 <- predict(modelo_v2, rose_teste)

6.4.2 Confusion Matrix

Vamos verificar a acurácia

caret::confusionMatrix(rose_teste$Class, previsoes_v2, positive = '1')

6.4.3 ROC curve

Calculamos o Score AUC

roc.curve(rose_teste$Class, previsoes_v2, plotit = T, col = "green", add.roc = T)

6.4.4 Result

Acurácia = 0.993

Score AUC = 0.993

Mantivemos quase a mesma acurácia (praticamente 99% nos dois modelos), mas aumentamos o Score AUC de

forma considerável, de 76% para 99%. Isso comprova que o modelo_v2 é muito melhor e mais estável que o modelo_v1.

Isso apenas porque balanceamos as classes!