UNIVERSIDADE FEDERAL DO PARANÁ

ALCEU EILERT NASCIMENTO

Análise de Modelos de Previsão de Inadimplência:

Utilizando KNN e Random Forest em uma Base de Dados de Crédito

CURITIBA

2025ALCEU EILERT NASCIMENTO

Análise de Modelos de Previsão de Inadimplência

Utilizando KNN e Random Forest em uma Base de Dados de Crédito

Artigo apresentado como requisito parcial à conclusão da disciplina de Mineração de Dados e Big Data, Departamento de Ciências Econômicas, Universidade Federal do Paraná.

Orientador(a)/Professor(a): Prof(a). Dr(a). Denise Fukumi Tsunoda.

CURITIBA

2025

**Análise de Modelos de Previsão de Inadimplência  
Utilizando KNN e Random Forest em uma Base de Dados de Crédito**

Alceu Eilert Nascimento  
2025-01-31

**RESUMO**

Este texto apresenta a experiência de aplicação de diferentes modelos de classificação para prever inadimplência (default) em uma base de dados de crédito. O estudo foi conduzido com foco na comparação entre o algoritmo K-Nearest Neighbors (KNN) e o Random Forest, bem como na análise de diferentes técnicas de balanceamento de classes.

# 1. Introdução

O problema de previsão de inadimplência é de grande relevância no setor financeiro, pois permite às instituições de crédito identificar clientes com maior probabilidade de não pagamento, auxiliando na tomada de decisão e na mitigação de riscos ([Altman, 1968](#X013dc7d30b1a0659f01c84eff337e8adf6e4233)). Diversos métodos de aprendizado de máquina têm sido empregados para esse propósito, destacando-se algoritmos como *K-Nearest Neighbors* (KNN), devido à sua simplicidade e boa capacidade preditiva em cenários com número moderado de variáveis ([Cover e Hart, 1967](#ref-coverNearestNeighborPattern1967)), e *Random Forest*, reconhecido pela robustez e habilidade de lidar com dados de alta dimensionalidade ([Breiman, 2001](#ref-breimanRandomForests2001)).

Entretanto, em muitas bases de inadimplência, há desbalanceamento das classes (mais adimplentes que inadimplentes). Essa disparidade pode prejudicar o treinamento de modelos de aprendizado de máquina, pois eles tendem a aprender predominantemente o comportamento da classe majoritária ([Chawla *et al.*, 2002](#ref-chawlaSMOTESyntheticMinority2002)). Para atenuar tal problema, técnicas de reamostragem como o *Synthetic Minority Over-sampling Technique* (SMOTE) são frequentemente empregadas.

O objetivo deste artigo é comparar o desempenho de diferentes configurações de Random Forest (variando pesos de classe e aplicando SMOTE) e um modelo KNN, utilizando métricas de desempenho como acurácia, precisão, recall e F1-score ([Sokolova e Lapalme, 2009](#Xdd7736ad1534be23f9135c8855ab79024ead351)). A base de dados usada contém informações de clientes de crédito e a variável-alvo indica se o cliente está inadimplente () ou não () no mês seguinte.

# 2. Metodologia

## 2.1 Base de Dados

A base de dados utilizada foi produzida por Yeh ([2009](#ref-yehDefaultCreditCard2009)) e contém 30.000 observações com variáveis relacionadas ao limite de crédito, histórico de pagamentos, montantes devidos e valores pagos em diferentes períodos, além de informações demográficas como idade, estado civil e nível educacional. A coluna alvo, denominada originalmente default.payment.next.month, foi renomeada para , onde: é adimplente e é inadimplente.

Após o carregamento dos dados, foi removida a coluna de identificação ID, por não apresentar relevância preditiva. Em seguida, convertemos a variável-alvo em fator para adequação às funções de classificação.

## 2.2 Pré-Processamento e Particionamento

Para avaliar o desempenho dos modelos, os dados foram divididos em treino (%) e teste (%), utilizando a função createDataPartition do pacote **caret** ([Kuhn, 2008](#ref-kuhnBuildingPredictiveModels2008)). Foi definida uma semente aleatória (set.seed(123)) para garantir a reprodutibilidade dos resultados.

Em seguida, aplicou-se normalização (método range) nas variáveis preditoras, mantendo-se a escala entre 0 e 1. Esse processo foi realizado por meio do preProcess do **caret**. A normalização busca evitar que variáveis em escalas muito diferentes influenciem de forma desproporcional a distância (principalmente no KNN) e o critério de divisão de nós (no caso de algumas variantes de árvores).

## 2.3 Modelagem

A modelagem estatística e de aprendizado de máquina para previsão de inadimplência envolve a escolha de algoritmos apropriados, a configuração de hiperparâmetros e a validação dos modelos treinados. Nesta seção, serão apresentados os modelos utilizados, justificando sua aplicação e comparando seu desempenho em termos de métricas padronizadas. Os algoritmos foram selecionados com base na capacidade de capturar padrões não lineares nos dados e na eficiência em cenários de alto desequilíbrio de classes, como é o caso da previsão de inadimplência.

Inicialmente, implementaram-se métodos supervisionados de classificação, incluindo KNN e Random Forest, cada um com características distintas na forma de aprendizado e na capacidade de generalização. O desempenho dos modelos foi avaliado considerando métricas como acurácia, sensibilidade, especificidade e F1-score, buscando identificar a abordagem mais eficaz para detectar clientes inadimplentes com maior precisão e reduzir falsos negativos, fator crítico em contextos de risco de crédito.

2.3.1 K-Nearest Neighbors (KNN)

O KNN é um método baseado em instâncias, em que a classificação é decidida pela maioria dos rótulos dos vizinhos mais próximos de uma nova instância ([Cover e Hart, 1967](#ref-coverNearestNeighborPattern1967)). Definiu-se para o treinamento, avaliando-se seu desempenho no conjunto de teste. As métricas de avaliação foram obtidas por meio da confusionMatrix do **caret**.

2.3.2 Random Forest

O Random Forest, proposto por Breiman ([2001](#ref-breimanRandomForests2001)), consiste em um conjunto de árvores de decisão geradas a partir de amostras aleatórias com reposição (*bootstrapping*) dos dados e da seleção aleatória de subconjuntos de variáveis em cada nó. Foram construídos três cenários distintos:

TABELA 1 – PERFIL DOS MODELOS DE RANDOM FOREST

| **Cenário** | **Descrição** |
| --- | --- |
| **RF Padrão** | **ntree** = ; **mtry** = (onde é o número de variáveis preditoras). Nenhum ajuste de pesos de classe |
| **RF com Pesos** | **ntree** = ; **classwt** = c(, ), atribuindo maior peso à classe inadimplente () |
| **RF com SMOTE** | Aplicação da técnica **SMOTE** no conjunto de treino, via step\_smote do pacote themis - **ntree** = 1000 para maior robustez do modelo |

Fonte: O Autor (2025)

A avaliação foi realizada com base na matriz de confusão e nas métricas de acurácia, precisão, recall e F1-score.

# 3. Resultados e Discussões

A análise dos resultados permite avaliar o desempenho dos modelos em prever a inadimplência, comparando diferentes abordagens e suas respectivas métricas. O foco principal está na capacidade de detectar corretamente os clientes inadimplentes, minimizando falsos negativos. Além disso, discute-se o impacto do balanceamento dos dados e da escolha dos hiperparâmetros na eficácia dos algoritmos.

## 3.1 Desempenho do Modelo KNN

O modelo KNN apresentou, no conjunto de teste, uma acurácia de , demonstrando boa capacidade de classificação geral, embora ainda haja algumas limitações quanto à correta identificação da classe minoritária. A precisão para a classe positiva (definida como ) foi de , indicando que, entre todas as instâncias classificadas como adimplentes, cerca de % eram efetivamente da classe . Já o recall de reforça a habilidade do modelo em recuperar a maior parte dos adimplentes. Por fim, o F1-Score de reflete o equilíbrio entre precisão e recall, denotando um desempenho satisfatório para o objetivo de classificação proposto.

Observa-se uma boa capacidade de identificar corretamente a classe majoritária (adimplentes), com recall de cerca de %, mas isso se deve em parte ao desbalanceamento do conjunto de dados. O baixo valor de Kappa () revela que, apesar de o modelo identificar bem a classe , há limitações na distinção mais equilibrada das classes.

## 3.2 Desempenho do Random Forest

O modelo Random Forest foi aplicado com o objetivo de superar as limitações observadas no KNN, especialmente no que se refere ao desbalanceamento das classes. Sua abordagem baseada em múltiplas árvores permite capturar relações mais complexas entre as variáveis preditoras, resultando em um modelo mais robusto e generalizável.

A seguir, são apresentados os resultados obtidos com a configuração padrão e com ajustes nos hiperparâmetros para otimização do desempenho.

3.2.1 RF Padrão

O modelo Random Forest padrão (rf\_model.1) apresentou um desempenho robusto no conjunto de teste, evidenciado por uma acurácia de , o que reflete uma elevada taxa de classificações corretas. A precisão de indica que a maioria das instâncias classificadas como pertencentes à classe positiva corresponde efetivamente aos casos de adimplência. Além disso, o recall de demonstra a capacidade do modelo em identificar corretamente a maior parte dos verdadeiros positivos, enquanto o F1-Score de evidencia um equilíbrio satisfatório entre precisão e recall. Tais resultados corroboram a eficácia do Random Forest em tarefas de classificação, especialmente em contextos onde a distinção entre classes é desafiadora, como nos problemas de previsão de inadimplência.

O Random Forest padrão supera o KNN em termos de acurácia, atingindo . O recall (%) indica que o modelo consegue identificar a maioria dos adimplentes corretamente, mas ainda mantém uma especificidade relativamente baixa (%), o que implica em dificuldade para identificar corretamente os inadimplentes (clique moderado de falsos negativos/falsos positivos).

3.2.2 RF com Pesos

O modelo Random Forest com Pesos (rf\_model.2) apresentou um desempenho sólido no conjunto de teste, registrando uma acurácia de , o que significa que aproximadamente % das classificações foram realizadas de forma correta. A precisão de indica que, dentre as instâncias classificadas como positivas, % correspondem efetivamente à classe prevista. Além disso, o recall de evidencia que o modelo identificou corretamente % dos casos positivos, demonstrando uma elevada capacidade de detecção. Por fim, o F1-Score de reflete um equilíbrio adequado entre precisão e recall, sugerindo que o ajuste de pesos empregado na modelagem contribuiu para uma melhora na identificação dos casos de inadimplência, mesmo diante do desbalanceamento presente na base de dados. Tais resultados corroboram a eficácia da estratégia de ponderação das classes na mitigação dos vieses inerentes a conjuntos de dados desbalanceados, conforme discutido por ([Breiman, 2001](#ref-breimanRandomForests2001)).

A atribuição de maior peso à classe inadimplente () ligeiramente reduziu a acurácia para , mas potencialmente melhora a identificação da classe minoritária no treino. Apesar disso, a especificidade ainda não apresentou ganho substancial no conjunto de teste. Esses resultados confirmam que o trade-off entre as métricas de desempenho deve ser avaliado conforme a estratégia do negócio (provável priorização de redução de inadimplência).

3.2.3 RF com SMOTE

No cenário de Random Forest com SMOTE (rf\_model.3), o modelo apresentou uma acurácia de , o que significa que aproximadamente % das previsões foram corretas no conjunto de teste. A precisão de indica que a grande maioria das instâncias classificadas como positivas efetivamente correspondem à classe prevista, enquanto o recall de demonstra a elevada capacidade do modelo em identificar os casos positivos. O F1-Score de evidencia um equilíbrio robusto entre precisão e recall, refletindo a eficácia da estratégia de balanceamento implementada por meio da técnica SMOTE, que gera amostras sintéticas da classe minoritária para mitigar o desbalanceamento dos dados ([Chawla *et al.*, 2002](#ref-chawlaSMOTESyntheticMinority2002)). Apesar da acurácia ter sofrido uma leve queda em comparação com os modelos Random Forest sem SMOTE, os demais indicadores sugerem uma melhora na detecção dos casos positivos, aspecto crucial em problemas de previsão de inadimplência, onde a identificação correta dos casos de risco é prioritária. Assim, a incorporação do SMOTE mostrou-se uma estratégia promissora para otimizar o desempenho preditivo do modelo, corroborando evidências presentes na literatura ([Breiman, 2001](#ref-breimanRandomForests2001)).

A aplicação de SMOTE no conjunto de treino foi benéfica para aumentar o número de exemplos sintéticos da classe minoritária, buscando um equilíbrio maior entre as classes. Observou-se uma redução na acurácia global () em comparação ao modelo básico, mas houve melhora considerável na sensibilidade em relação à classe menos representada (ainda que a métrica exata de recall para a classe 1 não esteja diretamente visível na tabela — possivelmente indicando um maior esforço de identificação da classe inadimplente). Em cenários de avaliação de crédito, esse resultado pode ser preferível, pois reduzir falsos negativos (inadimplentes não identificados) é frequentemente mais crítico do que garantir a mais alta acurácia global ([Baesens *et al.*, 2003](#X5a82b13b2fdfcb42c39cd6350eaf8d0ab6fa292)).

## 3.3 Comparação Geral

Ao comparar KNN e Random Forest, verifica-se que o Random Forest obteve consistentemente melhor desempenho em termos de acurácia e manteve F1-score mais equilibrado, ainda que todos os modelos apresentem desafios na distinção da classe minoritária. Em aplicações de classificação de risco de crédito, o custo de prever indevidamente que um cliente é adimplente (falso negativo) costuma ser maior do que o oposto (falso positivo). Assim, a escolha final do modelo deve considerar as estratégias de mitigação de perdas e o custo-benefício de errar em cada classe.

# 4. Conclusão

Este estudo ilustra a aplicação de dois algoritmos populares em Mineração de Dados — KNN e Random Forest — para a previsão de inadimplência em uma base de crédito. Os resultados indicam que, embora o modelo KNN tenha apresentado boa sensibilidade para a classe majoritária, ele foi superado pelo Random Forest em termos de acurácia global. Entre as variações do Random Forest, o modelo padrão destacou-se com uma acurácia de , evidenciando um desempenho superior de forma geral. Ademais, a aplicação de técnicas de balanceamento, seja através do SMOTE ou pela atribuição de pesos, influenciou a distribuição dos acertos e erros, ocasionando uma redução na acurácia global, mas potencialmente melhorando a sensibilidade para a classe minoritária. Essa melhoria na detecção dos casos de inadimplência é especialmente relevante no contexto de risco de crédito, onde a minimização de falsos negativos é crucial para a mitigação de perdas.

Portanto, a decisão sobre qual abordagem utilizar deve ser guiada por métricas que reflitam a estratégia de negócio (por exemplo, priorizar a detecção de inadimplentes) e considerar os custos associados a cada tipo de erro. Pesquisas futuras podem incluir técnicas avançadas de ensemble e tuning mais aprofundado dos hiperparâmetros para refinar ainda mais o desempenho preditivo.

# Referências bibliográficas

ALTMAN, E. I. [Financial ratios, discriminant analysis and the prediction of corporate bankruptcy](https://doi.org/10.1111/j.1540-6261.1968.tb00843.x). **The Journal of Finance**, v. 23, n. 4, p. 589–609, set. 1968.

BAESENS, B. *et al.* [Benchmarking state-of-the-art classification algorithms for credit scoring](https://doi.org/10.1057/palgrave.jors.2601545). **Journal of the Operational Research Society**, v. 54, n. 6, p. 627–635, jun. 2003.

BREIMAN, L. [Random Forests](https://doi.org/10.1023/A:1010933404324). **Machine Learning**, v. 45, n. 1, p. 5–32, 2001.

CHAWLA, N. V. *et al.* [SMOTE: Synthetic Minority Over-sampling Technique](https://doi.org/10.1613/jair.953). **Journal of Artificial Intelligence Research**, v. 16, p. 321–357, 1 jun. 2002.

COVER, T.; HART, P. [Nearest neighbor pattern classification](https://doi.org/10.1109/TIT.1967.1053964). **IEEE Transactions on Information Theory**, v. 13, n. 1, p. 21–27, jan. 1967.

KUHN, M. [Building Predictive Models in *R* Using the **caret** Package](https://doi.org/10.18637/jss.v028.i05). **Journal of Statistical Software**, v. 28, n. 5, 2008.

SOKOLOVA, M.; LAPALME, G. [A systematic analysis of performance measures for classification tasks](https://doi.org/10.1016/j.ipm.2009.03.002). **Information Processing & Management**, v. 45, n. 4, p. 427–437, jul. 2009.

YEH, I.-C. [**Default of Credit Card Clients**](https://doi.org/10.24432/C55S3H), 2009.

APÊNDICE  
(script utilizado)

*# SETUP ----*  
  
*# Pacotes utilizados no modelo de previsão de inadimplência*  
  
*# dplyr: Manipulação eficiente de dados com funções do tidyverse*  
**library**(dplyr)

*# ggplot2: Criação de gráficos avançados e customizáveis*  
**library**(ggplot2)

*# gridExtra: Combinação de múltiplos gráficos em um único layout*  
**library**(gridExtra)

*# recipes: Pré-processamento de dados, incluindo normalização, imputação e balanceamento*  
**library**(recipes)

*# caret: Utilizado para particionar os dados, treinar modelos e calcular métricas de desempenho*  
**library**(caret)

*# themis: Técnicas de balanceamento de classes, incluindo SMOTE*  
**library**(themis)

*# class: Implementação do algoritmo KNN (K-Nearest Neighbors)*  
**library**(class)  
  
*# randomForest: Implementação do algoritmo de Random Forest para classificação e regressão*  
**library**(randomForest)

*# xgboost: Implementação otimizada de Gradient Boosting para aprendizado supervisionado*  
**library**(xgboost)

*# ELT ----*  
*# Carregar os dados (substituir pelo caminho correto)*  
data <- **read.csv2**("C:/Users/DELL/OneDrive/R/Rprojetos/ufpr\_ppgecon/mineracao\_dados/projeto\_final/data/credit\_data.csv")  
  
  
*# Remover a coluna ID, pois não é relevante para a modelagem*  
data <- data **%>%** **select**(**-**ID)  
  
*# Renomear a coluna alvo corretamente*  
**colnames**(data)[**colnames**(data) **==** "default.payment.next.month"] <- "Y"  
  
*# Converter a variável Y para fator*  
data**$**Y <- **as.factor**(data**$**Y)  
  
*# Verificar a estrutura corrigida dos dados*  
**str**(data)

## 'data.frame': 30000 obs. of 24 variables:  
## $ LIMIT\_BAL: int 20000 120000 90000 50000 50000 50000 500000 100000 140000 20000 ...  
## $ SEX : int 2 2 2 2 1 1 1 2 2 1 ...  
## $ EDUCATION: int 2 2 2 2 2 1 1 2 3 3 ...  
## $ MARRIAGE : int 1 2 2 1 1 2 2 2 1 2 ...  
## $ AGE : int 24 26 34 37 57 37 29 23 28 35 ...  
## $ PAY\_0 : int 2 -1 0 0 -1 0 0 0 0 -2 ...  
## $ PAY\_2 : int 2 2 0 0 0 0 0 -1 0 -2 ...  
## $ PAY\_3 : int -1 0 0 0 -1 0 0 -1 2 -2 ...  
## $ PAY\_4 : int -1 0 0 0 0 0 0 0 0 -2 ...  
## $ PAY\_5 : int -2 0 0 0 0 0 0 0 0 -1 ...  
## $ PAY\_6 : int -2 2 0 0 0 0 0 -1 0 -1 ...  
## $ BILL\_AMT1: int 3913 2682 29239 46990 8617 64400 367965 11876 11285 0 ...  
## $ BILL\_AMT2: int 3102 1725 14027 48233 5670 57069 412023 380 14096 0 ...  
## $ BILL\_AMT3: int 689 2682 13559 49291 35835 57608 445007 601 12108 0 ...  
## $ BILL\_AMT4: int 0 3272 14331 28314 20940 19394 542653 221 12211 0 ...  
## $ BILL\_AMT5: int 0 3455 14948 28959 19146 19619 483003 -159 11793 13007 ...  
## $ BILL\_AMT6: int 0 3261 15549 29547 19131 20024 473944 567 3719 13912 ...  
## $ PAY\_AMT1 : int 0 0 1518 2000 2000 2500 55000 380 3329 0 ...  
## $ PAY\_AMT2 : int 689 1000 1500 2019 36681 1815 40000 601 0 0 ...  
## $ PAY\_AMT3 : int 0 1000 1000 1200 10000 657 38000 0 432 0 ...  
## $ PAY\_AMT4 : int 0 1000 1000 1100 9000 1000 20239 581 1000 13007 ...  
## $ PAY\_AMT5 : int 0 0 1000 1069 689 1000 13750 1687 1000 1122 ...  
## $ PAY\_AMT6 : int 0 2000 5000 1000 679 800 13770 1542 1000 0 ...  
## $ Y : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...

*# Separar em treino (70%) e teste (30%)*  
**set.seed**(123)  
index <- **createDataPartition**(data**$**Y, p = 0.7, list = FALSE)  
train\_data <- data[index, ]  
test\_data <- data[**-**index, ]  
  
*# Normalizar os dados (exceto a variável resposta)*  
norm\_params <- **preProcess**(train\_data[,**-ncol**(train\_data)], method = **c**("range"))  
train\_data[,**-ncol**(train\_data)] <- **predict**(norm\_params, train\_data[,**-ncol**(train\_data)])  
test\_data[,**-ncol**(train\_data)] <- **predict**(norm\_params, test\_data[,**-ncol**(train\_data)])  
  
*# Definir os preditores e a variável resposta*  
train\_x <- train\_data[, **-ncol**(train\_data)]  
train\_y <- train\_data**$**Y  
test\_x <- test\_data[, **-ncol**(test\_data)]  
test\_y <- test\_data**$**Y  
  
  
*# MODELOS ----*  
***## KNN ----***  
*# Treinar modelo KNN (k=5)*  
k\_value <- 5  
knn\_model <- **knn**(train = train\_x, test = test\_x, cl = train\_y, k = k\_value)  
  
*# Avaliação do modelo*  
conf\_matrix <- **confusionMatrix**(knn\_model, test\_y)  
**print**(conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 6407 1309  
## 1 602 681  
##   
## Accuracy : 0.7876   
## 95% CI : (0.779, 0.7961)  
## No Information Rate : 0.7789   
## P-Value [Acc > NIR] : 0.02269   
##   
## Kappa : 0.2937   
##   
## Mcnemar's Test P-Value : < 2e-16   
##   
## Sensitivity : 0.9141   
## Specificity : 0.3422   
## Pos Pred Value : 0.8304   
## Neg Pred Value : 0.5308   
## Prevalence : 0.7789   
## Detection Rate : 0.7120   
## Detection Prevalence : 0.8574   
## Balanced Accuracy : 0.6282   
##   
## 'Positive' Class : 0   
##

*# Métricas de performance*  
accuracy <- conf\_matrix**$**overall["Accuracy"]  
precision <- conf\_matrix**$**byClass["Precision"]  
recall <- conf\_matrix**$**byClass["Recall"]  
F1\_score <- 2 **\*** (precision **\*** recall) **/** (precision **+** recall)  
  
**cat**("**\n**Desempenho do modelo KNN:**\n**")

##   
## Desempenho do modelo KNN:

**cat**("Acurácia:", accuracy, "**\n**")

## Acurácia: 0.7876431

**cat**("Precisão:", precision, "**\n**")

## Precisão: 0.8303525

**cat**("Recall:", recall, "**\n**")

## Recall: 0.9141104

**cat**("F1-score:", F1\_score, "**\n**")

## F1-score: 0.8702207

***## RANDOM FOREST ----***  
  
*# Treinar modelo Random Forest*  
**set.seed**(123)  
rf\_model.1 <- **randomForest**(Y **~** .,   
 data = train\_data,   
 ntree = 100,   
 mtry = **sqrt**(**ncol**(train\_data) **-** 1),   
 importance = TRUE)  
  
*# Fazer previsões*  
rf\_pred.1 <- **predict**(rf\_model.1, test\_data, type = "class")  
  
*# Avaliação do modelo*  
conf\_matrix.1 <- **confusionMatrix**(rf\_pred.1, test\_data**$**Y)  
**print**(conf\_matrix.1)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 6596 1253  
## 1 413 737  
##   
## Accuracy : 0.8149   
## 95% CI : (0.8067, 0.8228)  
## No Information Rate : 0.7789   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3669   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9411   
## Specificity : 0.3704   
## Pos Pred Value : 0.8404   
## Neg Pred Value : 0.6409   
## Prevalence : 0.7789   
## Detection Rate : 0.7330   
## Detection Prevalence : 0.8722   
## Balanced Accuracy : 0.6557   
##   
## 'Positive' Class : 0   
##

*# Métricas de performance*  
accuracy.1 <- conf\_matrix.1**$**overall["Accuracy"]  
precision.1 <- conf\_matrix.1**$**byClass["Precision"]  
recall.1 <- conf\_matrix.1**$**byClass["Recall"]  
F1\_score.1 <- 2 **\*** (precision **\*** recall) **/** (precision **+** recall)  
  
**cat**("**\n**Desempenho do modelo Random Forest:**\n**")

##   
## Desempenho do modelo Random Forest:

**cat**("Acurácia:", accuracy.1, "**\n**")

## Acurácia: 0.8148683

**cat**("Precisão:", precision.1, "**\n**")

## Precisão: 0.8403618

**cat**("Recall:", recall.1, "**\n**")

## Recall: 0.9410758

**cat**("F1-score:", F1\_score.1, "**\n**")

## F1-score: 0.8702207

*# Importância das variáveis*  
**varImpPlot**(rf\_model.1)

*# Random Forest ajustado para dar mais peso para inadimplentes (classwt)*  
rf\_model.2 <- **randomForest**(Y **~** .,   
 data = train\_data,   
 ntree = 100,   
 classwt = **c**(0.5, 1.5))  
  
*# Fazer previsões*  
rf\_pred.2 <- **predict**(rf\_model.2, test\_data, type = "class")  
  
*# Avaliação do modelo*  
conf\_matrix.2 <- **confusionMatrix**(rf\_pred.2, test\_data**$**Y)  
**print**(conf\_matrix.2)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 6504 1259  
## 1 505 731  
##   
## Accuracy : 0.804   
## 95% CI : (0.7956, 0.8121)  
## No Information Rate : 0.7789   
## P-Value [Acc > NIR] : 3.171e-09   
##   
## Kappa : 0.3416   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9279   
## Specificity : 0.3673   
## Pos Pred Value : 0.8378   
## Neg Pred Value : 0.5914   
## Prevalence : 0.7789   
## Detection Rate : 0.7227   
## Detection Prevalence : 0.8627   
## Balanced Accuracy : 0.6476   
##   
## 'Positive' Class : 0   
##

*# Métricas de performance*  
accuracy.2 <- conf\_matrix.2**$**overall["Accuracy"]  
precision.2 <- conf\_matrix.2**$**byClass["Precision"]  
recall.2 <- conf\_matrix.2**$**byClass["Recall"]  
F1\_score.2 <- 2 **\*** (precision **\*** recall) **/** (precision **+** recall)  
  
**cat**("**\n**Desempenho do modelo Random Forest:**\n**")

##   
## Desempenho do modelo Random Forest:

**cat**("Acurácia:", accuracy.2, "**\n**")

## Acurácia: 0.8039782

**cat**("Precisão:", precision.2, "**\n**")

## Precisão: 0.8378204

**cat**("Recall:", recall.2, "**\n**")

## Recall: 0.9279498

**cat**("F1-score:", F1\_score.2, "**\n**")

## F1-score: 0.8702207

*# Importância das variáveis*  
**varImpPlot**(rf\_model.2)

*# SMOTE*  
*# equilibrar as classes.*  
  
train\_data.3 <- train\_data **%>%**   
 **recipe**(Y **~** .) **%>%** *# pré-processamento para a variável resposta `Y` com todas as outras variáveis como preditoras*  
 **step\_smote**(Y) **%>%** *# SMOTE para balancear as classes da variável `Y` (gera novos exemplos sintéticos da classe minoritária)*  
 **prep**() **%>%** *# Prepara a receita, estimando os parâmetros necessários para a transformação*  
 **juice**() *# Extrai os dados transformados para serem usados no modelo*  
  
  
rf\_model.3 <- **randomForest**(Y **~** .,   
 data = train\_data.3,   
 ntree = 1000)  
  
  
*# Fazer previsões*  
rf\_pred.3 <- **predict**(rf\_model.3,   
 test\_data,   
 type = "class")  
  
*# Avaliação do modelo*  
conf\_matrix.3 <- **confusionMatrix**(rf\_pred.3, test\_data**$**Y)  
**print**(conf\_matrix.3)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 6185 1025  
## 1 824 965  
##   
## Accuracy : 0.7945   
## 95% CI : (0.786, 0.8028)  
## No Information Rate : 0.7789   
## P-Value [Acc > NIR] : 0.0001606   
##   
## Kappa : 0.3811   
##   
## Mcnemar's Test P-Value : 3.301e-06   
##   
## Sensitivity : 0.8824   
## Specificity : 0.4849   
## Pos Pred Value : 0.8578   
## Neg Pred Value : 0.5394   
## Prevalence : 0.7789   
## Detection Rate : 0.6873   
## Detection Prevalence : 0.8012   
## Balanced Accuracy : 0.6837   
##   
## 'Positive' Class : 0   
##

*# Métricas de performance*  
accuracy.3 <- conf\_matrix.3**$**overall["Accuracy"]  
precision.3 <- conf\_matrix.3**$**byClass["Precision"]  
recall.3 <- conf\_matrix.3**$**byClass["Recall"]  
F1\_score.3 <- 2 **\*** (precision **\*** recall) **/** (precision **+** recall)  
  
**cat**("**\n**Desempenho do modelo Random Forest:**\n**")

##   
## Desempenho do modelo Random Forest:

**cat**("Acurácia:", accuracy.3, "**\n**")

## Acurácia: 0.7945327

**cat**("Precisão:", precision.3, "**\n**")

## Precisão: 0.8578363

**cat**("Recall:", recall.3, "**\n**")

## Recall: 0.8824369

**cat**("F1-score:", F1\_score.3, "**\n**")

## F1-score: 0.8702207

*# Importância das variáveis*  
**varImpPlot**(rf\_model.3)