Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Estatística

Avaliação de desempenho dos procedimentos de *boosting* e *bagging* em função do critério de classificação por árvore.

Eric Sato
Orientador: Prof. Dr. José Carlos Fogo

Trabalho de Conclusão de Curso

Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Estatística

Avaliação de desempenho dos procedimentos de *boosting* e *bagging* em função do critério de classificação por árvore.

Eric Sato
Orientador(a): Prof. Dr. José Carlos Fogo

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do título de Bacharel em Estatística.

São Carlos
Fevereiro de 2025

Resumo

A classificação, em sua essência, é uma atividade humana que categoriza objetos baseado em características específicas. Em algumas situações, é mais eficiente e prático delegar essa tarefa a máquinas. Historicamente, a estatística tem abordagens diferentes para classificação, com o desenvolvimento mais recente voltado para *machine learning* e reconhecimento de padrões, inspirado no funcionamento do cérebro humano.

Por vezes, dado que a classificação é crucial em várias áreas, enfrentamos desafios como modelos ineficazes ou dados desbalanceados. Os métodos estatísticos tradicionais frequentemente apresentam erros nesse contexto, levando à necessidade de melhores soluções. Combinando técnicas computacionais e estatísticas, podemos alcançar classificadores automáticos mais eficientes. O foco deste estudo é examinar o desempenho dos procedimentos de *bagging* e *boosting* usando classificação por árvore onde esses desempenhos serão avaliados a partir da curva *ROC*, taxa de erro de classificação e validação cruzada.
Palavras-chave: *classificação, machine learning, bagging, boosting, curva ROC.*

Sumário

Capítulo 1

Introdução

A área de Aprendizado de Máquina (AM) pertence ao campo da Inteligência Artificial (IA), com o propósito de desenvolver técnicas computacionais relacionadas à aprendizagem e construir sistemas que consigam captar padrões complexos e tomem decisões autônomas a partir de uma base em dados. Um sistema de aprendizado é um programa de computador que toma decisões com base em experiências acumuladas, resultantes da resolução bem-sucedida de problemas anteriores (Monard e Baranauskas, 2003).

A popularidade do aprendizado de máquina vem se destacando por conta da crescente disponibilidade de dados massivos, aliada ao poder computacional que vem sendo aprimorado ao longos dos anos. Isso tem impulsionado o interesse para um rápido desenvolvimento e adoção de técnicas de aprendizado de máquina em diversos setores.

Com esse aumento na disponibilidade dos dados, a classificação de dados vem sendo cada vez mais requisitada por ser uma tarefa presente em diversas áreas, como por exemplo: na pesquisa de mercado, sequências genéticas, mapeamento de dados, entre outros. Assim, em vários momentos, exige a habilidade de identificar e categorizar corretamente informações de mais relevância. No entanto, essa tarefa pode se tornar um desafio diante de diferentes aspectos, como os problemas nos modelos propostos ou até mesmo nos próprios dados estarem desbalanceados (LISKA *et al.*, 2012).

Há vários métodos estatísticos utilizados para realizar as classificações, uma das mais famosas é a classificação por árvore que, muitas vezes acabam apresentando taxas de erros de classificação inconsistente ao se deparar com os problemas mais complexos. Isso evidencia a necessidade de buscar algumas alternativas que consigam aprimorar o processo de classificação, tornando mais rápido e com uma boa eficiência (taxa de acerto mais altos).

Para potencializar e resolver alguns desses problemas de classificação, foi feito a com-

binação dos métodos computacionais e técnicas estatísticas que juntos entregam um classificador automático mais eficiente (LISKA *et al.*, 2012). Inserido nesse contexto, foi criados alguns métodos que utilizam a classificação por árvore como base e os métodos para aprimorar o processo de classificação mais conhecidos são: *Bagging*, *Random Forest* e *Boosting*.

A avaliação da performance em tarefas de classificação é crucial na validação e otimização de modelos, sendo um componente essencial no processo de tomada de decisões. Medir com precisão a eficácia de uma classificação permite não apenas quantificar a eficácia do modelo em identificar padrões nos dados, mas também comparar diferentes abordagens e ajustar parâmetros para melhorar seus resultados.

Dessa forma, a avaliação de desempenho de um classificador torna-se um aspecto indispensável para garantir a confiabilidade e utilidade prática dos modelos de classificação na interpretação e utilização dos resultados obtidos.

1.1 Objetivo

Este trabalho tem por objetivo, estudar o desempenho dos procedimentos de *bagging* e *boosting*, baseados na classificação por árvore, em função da probabilidade de corte do classificador base.

O estudo será feito em dois bancos de dados: o primeiro é o banco de dados do *Titanic* que no total contém 891 amostras no total e o segundo é o banco de dados do *Alzheimer* que contém uma amostra de 2149 amostras.

Para a avaliação do classificador, serão utilizados a curva *ROC*, através das medidas de especificidade e sensibilidade do classificador, a taxa de erro e outra medida de avaliação como a acurácia.

1.2 Organização do trabalho

Este trabalho está organizado da seguinte maneira. No capítulo 2, será explicado o conceito básico sobre a classificação e os 2 tipos diferentes que a compõem. Dessa forma será possível entender como foi conduzido o trabalho.

Já no capítulo 3 serão apresentados definições, conceitos básicos e exemplos sobre as metodologias utilizadas neste trabalho como classificação por árvore, *bagging*, *boosting* e

overfitting.

No capítulo 4 serão apresentados conceitos básicos de algumas medidas para avaliação do classificador. Isso irá facilitar a compreensão dos resultados e suas interpretações que foram obtidos através das aplicações das metodologias presentes neste trabalho e serão apresentados no capítulo 5.

Por fim, no capítulo 6, serão apresentadas as conclusões obtidas após a aplicação de diferentes análises em diferentes proporções para um mesmo banco de dados.

Capítulo 2

Revisão da literatura

2.1 Classificação

Naturalmente a classificação é uma tarefa atribuída a atividade humana, designando classes aos objetos através dos seus sentidos. De certa forma, quando fornecidas informações sobre um objeto, um indivíduo faz uma escolha acerca da categoria à qual esse objeto pertence, selecionando entre um grupo limitado de classes ou busca identificar agrupamentos nos quais diferentes tipos de objetos podem ser categorizados (Rubesam, 2004). Alguns exemplos de classificação feita por seres humanos são:

- reconhecer um alimento a partir do seu cheiro
- reconhecer o alimento pelo seu sabor
- reconhecer um animal pelo som que emitiu

Embora os seres humanos executem naturalmente e sem esforços as diversas tarefas de categorização, há situações que justificam a vantagem de delegar a classificação a máquinas, ou seja, para computadores. Algumas tarefas podem ser extremamente repetitivas para seres humanos onde os computadores podem fazer de forma mais eficiente.

Na estatística, há duas formas de abordagens distintas conforme citado por Michie *et al.* (1994). A primeira é a abordagem clássica, englobando técnicas baseadas no trabalho de *Fisher* de análise discriminante, que diminui de forma significante o número de classes. A segunda, mais moderna, emprega uma classe mais flexível de modelos, focando geralmente em obter uma estimativa da distribuição de probabilidade dos dados em cada atributo com o objetivo de gerar uma regra de classificação.

A comunidade de *machine learning* e os profissionais que trabalham com reconhecimento de padrões foram inicialmente motivados por tentativas de modelar o processo de aprendizagem humano, inspirados em conceitos biológicos sobre o funcionamento do cérebro. Um exemplo específico é o desenvolvimento das chamadas redes neurais artificiais, que surgiram como modelos simples para explicar o funcionamento de grupos de neurônios. No entanto, rapidamente se percebeu que essas redes possuíam um poder prático para o reconhecimento de padrões (classificação), o que levou profissionais de diversas áreas a aprimorá-las e utilizá-las de forma ampla (Rubesam, 2004).

Independentemente da abordagem empregada, o objetivo da classificação é desenvolver métodos automáticos que sejam tão eficientes quanto (ou até melhores do que) os seres humanos na tarefa de classificação e/ou que possam ser compreendidos e interpretados por seres humanos.

2.1.1 Aprendizado supervisionado

Aprendizagem supervisionada é uma abordagem muito importante no campo da aprendizado de máquina, que tem sido amplamente aplicada em diferentes tipos de problemas, sendo eles: classificar os clientes em bons e maus pagadores, mapeamento de dados, pesquisa de mercado, entre outras (Fontana, 2020). De um forma grosseira, ela treina um modelo computacional para aprender com os dados de entradas para que o modelo retorne saídas desejadas com base no conjunto de treinamento que foi previamente coletada.

Na modalidade de aprendizado supervisionado, a amostra coletada é separada em 2 conjuntos: conjunto de treinamento (normalmente 70% da amostra) e conjunto de validação (restante da amostra de 30%). Dessa forma, o modelo é fornecido a partir do conjunto de treinamento, no qual cada variável explicativa do modelo corresponde a uma característica do banco de dados e a variável resposta é a saída escolhida pelo usuário.

O objetivo do modelo é utilizar as variáveis mais significantes do banco de dados e ser capaz de fazer previsões mais corretas para os novos dados que serão coletados futuramente.

Durante o processo de treinamento, o modelo utiliza as variáveis do banco de dados para fazer a seleção do modelo, buscando encontrar padrões e relações nos dados que permitam a realização de previsões precisas. Após a escolha do modelo, é feito a aplicação de algoritmos de aprendizado que otimizam a sua capacidade em fazer inferências sobre novas amostras.

Existem diversos algoritmos de aprendizagem supervisionada, cada um com suas características e propriedades. Alguns dos mais populares incluem regressão linear, classificação por árvore, redes neurais artificiais e algoritmos de classificação, como o *k-NN* (*k*-vizinhos mais próximos) e o *SVM* (Máquinas de Vetores de Suporte).

Aprendizagem supervisionada tem sido amplamente utilizada em diversas áreas, tais como, marketing digital, medicina e finanças, entre outras. Seu potencial é imenso, permitindo que os modelos aprendam a tomar decisões e realizar boas previsões com base em dados anteriores.

2.1.2 Aprendizado não supervisionado

Na aprendizagem não supervisionada, não se associa uma característica aos resultados. A partir de um conjunto extenso de dados, esses algoritmos buscam identificar padrões e semelhanças entre os dados, possibilitando a identificação de grupos de dados semelhantes ou a semelhança de novos dados com grupos já existentes. Esses algoritmos podem ser categorizados em duas principais classes: os algoritmos de transformação e os algoritmos de agrupamento (Fontana, 2020).

- Os algoritmos de transformação, criam uma forma nova para representar o conjunto de dados que mostre ser mais conveniente que a original. Dessa forma, facilita a interpretação humana ou faz uma melhoria do desempenho de outros algoritmos de aprendizagem.

- Os algoritmos de agrupamento, conhecidos como *clustering*, separa o conjunto de dados em grupos com características similares através de critérios estabelecidos previamente, permitindo encontrar padrões entre os dados.

Ao contrário do aprendizado supervisionado, onde é essencial selecionar uma variável resposta para auxiliar na busca de padrões, o aprendizado não supervisionado não depende da seleção dessa variável resposta, permitindo a identificação de padrões inicialmente não percebidos através de outras variáveis (Bianka Tallita Passos, 2021).

Capítulo 3

Metodologia

3.1 Classificação por árvore

A classificação por árvore é uma técnica de aprendizado de máquina supervisionado que se baseia na construção de uma estrutura que é semelhante a forma da árvore, servindo para classificar dados em categorias predefinidas.

A estrutura de uma árvore é constituída por particionamentos que recebem o nome de nó e cada resultado final recebe o nome de folha (Izbicki e dos Santos, 2020). Se a partição for binárias, a partir de um determinado ponto de decisão, que chamaremos de raiz, ou nó, há sempre dois ramos direcionados à direita ou esquerda sendo que, cada ramo irá definir um valor de divisão.

A árvore de classificação é o resultado de uma sequência de divisões que, a cada passo, dependem das decisões anteriores. A divisão é determinada por uma condição sobre um atributo, dividindo aquele nó em outros nós, gerando, assim, os ramos da árvore.

O ponto de partida de uma árvore de classificação é chamado de nó raiz, que está no topo da árvore e as divisões a partir da raiz, levam a outros nós (chamados de nós filhos), que podem ser terminais ou não-terminais. Um nó não-terminal é um nó que se divide em outras classes, segundo um outro atributo, gerando nós filhos e um nó terminal, por sua vez, é aquele que não se divide e a ele é atribuída uma única classe (BARBOSA *et al.*, 2012).

Após esclarecer sobre a estrutura e a criação, será feita a explicação de forma sucinta sobre o funcionamento da classificação por árvore: o ponto de partida é chamada de nó raiz (localizada no topo da árvore), observamos se a condição do primeiro nó é satisfeita. Caso seja satisfeita, seguiremos para a esquerda do nó e caso contrário, seguiremos para

a direita do nó. Repetiremos este procedimento até chegar em uma folha ou nó terminal.

Na Figura 3.1 apresentamos um exemplo de uma árvore de classificação em que

1. O nó I é o nó raiz;
2. os nós II, III e IV são nós filhos não terminais e,
3. os nós V, VI, VII, VIII e IX são nós terminais.

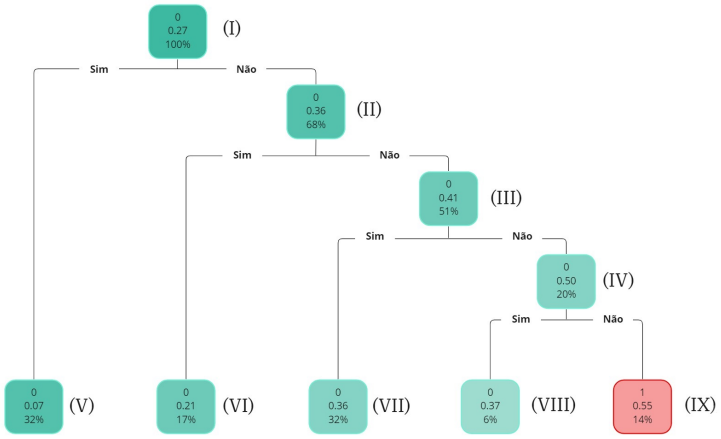

Figura 3.1: Exemplo de uma árvore de classificação.

Para um melhor entendimento sobre a classificação por árvore, há alguns exemplos de aplicação deste método em diferentes áreas: auxiliar no diagnóstico médico classificando os pacientes em diferentes grupos de risco com base em seus sintomas e histórico de saúde (Medicina); identificar grupos com maior probabilidade de responder a determinadas ofertas para definir a segmentação dos clientes para as campanhas de marketing (*Marketing*); classificar transações como legítimas ou suspeitas com base em padrões e comportamentos atípicos, detectando fraudes em transações financeiras (Finanças).

Nos dias atuais, ao se deparar com diversos métodos de classificação, temos algumas vantagens que se destacam ao utilizar a classificação por árvore:

1. Interpretabilidade: A estrutura na forma de árvore permite uma interpretação clara e intuitiva das decisões tomadas pelo modelo, tornando mais fácil compreender como determinada classificação foi feita;
2. Flexibilidade: A classificação por árvore é aplicável a problemas de classificação binária e multiclasse, além de suportar tanto atributos numéricos quanto categóricos. Dessa maneira, permite que o método seja utilizado em uma ampla gama de contextos;

3. Baixo custo computacional: A construção e a avaliação de árvores de decisão podem ser relativamente rápidas, tornando-se uma opção viável mesmo para grandes conjuntos de dados.

3.2 Classificadores múltiplos

3.2.1 *Bagging (Booststrap Aggregating)*

Diante das dificuldades que a classificação pode trazer como por exemplo, o caso de *overfitting*, em que o modelo se torna específico para os dados de treinamento e perde a capacidade de generalizar para novos dados, a dificuldade de classificação por conta da distorção que os *outliers* podem causar no resultado, entre outros, foi criado o método *bagging*, também conhecido como *bootstrap aggregating*, que consiste em diminuir a variabilidade dos dados de forma que tenha uma melhora considerável comparado com o modelo original (sem a aplicação do método) (Silva, 2023).

O método *bagging*, inicialmente proposto por Breiman (1996), foi um dos primeiros métodos que utilizavam a combinação de classificadores onde não só aprimorava o desempenho de estimadores de regressão mas também a eficácia de classificadores. Um ponto de destaque é que o sucesso do *bagging* para classificadores está associado à modelos com tendência a se ajustar intensamente aos dados de treinamento (como árvores de decisões profundas) (Breiman, 1996).

O método segue os seguintes passos: divide-se a amostra em dois conjuntos conhecidos como Treinamento e Validação; aplica-se o classificador a partir de B reamostras, com reposição, retiradas do conjunto de treinamento, utilizando o método de *bootstrap*. Após a obtenção das b amostras de treinamentos, serão aplicados os classificadores $g_i(x)$, $i = 1,2,...b$. Sendo assim, a classificação final será dada pela combinação $g(x)$ desses classificadores em que pode ser: a média ou a moda de $g_i(x)$. Após obter a combinação de $g_i(x)$, será aplicado no conjunto de validação para testar a sua eficácia (Opitz e Maclin, 1999).

Para uma melhor compreensão dos procedimentos, segue a descrição simplificada do método *bagging* (extraído de Silva, 2023):

- **Algoritmo *bagging* de aprendizagem.**

  Legenda:

  T - Conjunto de treinamento de amostra n;

B - número total de reamostras.

1. De $j = 1$ até B faça:

   $\mathbf{T}_{(j)}$ como sendo a j-ésima amostra com reposição retirada do conjunto $\mathbf{T}$;

2. O classificador individual $g^n(x)$ é treinado em cada amostra $\mathbf{T}_{(j)}$ gerada;

3. Obtém-se o classificador múltiplo para regressão ou classificação, respectivamente através de:

$$g_{final}(x) = \frac{1}{B}\sum_{j=1}^{B} g_j^n(x)$$

   ou

$$g_{final}(x) = moda[g_1^n(x), g_2^n(x), ..., g_B^n(x)]$$

Abaixo está a representação do método *bagging*.

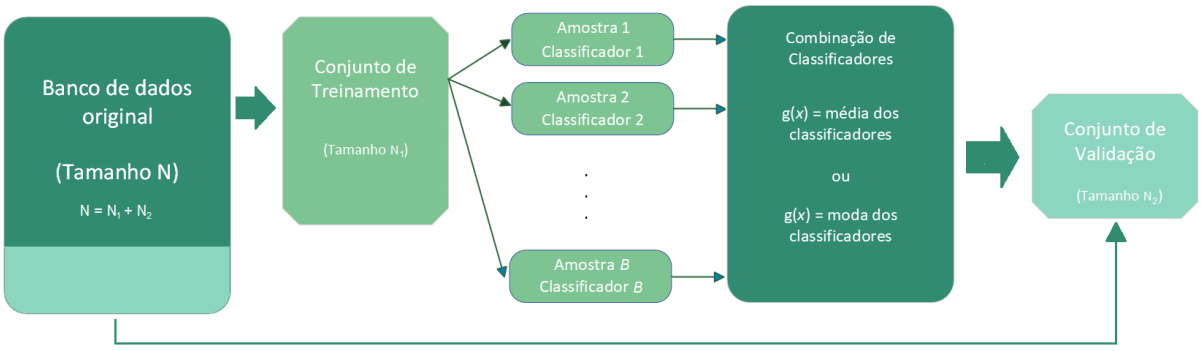

Figura 3.2: Esquema explicativo do método *bagging*.
Fonte: Silva (2023).

3.2.2 *Boosting*

Dentro da comunidade de *machine learning* havia um problema chamado de problema de *boosting* que dizia: "Suponha que existe um método de classificação que é ligeiramente melhor do que uma escolha aleatória. Esse método é chamado de *weak learner*, ou classificador fraco. A existência de um classificador fraco implica na existência de um classificador forte (*strong learner*), com erro pequeno?"(Rubesam, 2004).

O problema foi resolvido por Freund e Schapire (1995) que conseguiu mostrar que é possível se obter um classificador forte a partir de um mais fraco. Após a descoberta, foram desenvolvidos vários algoritmos dentro do contexto de *boosting*. O mais famoso entre eles é o *AdaBoost* (***Ada****ptative* ***Boost****ing*) (Rubesam, 2004).

De maneira resumida, são gerados múltiplos classificadores sequencialmente que, em cada iteração, a distribuição do conjunto se altera em função dos classificadores anteriores. Dessa forma, as observações classificadas de forma errada são penalizadas através de um peso atribuídas a elas, que é atualizada a cada iteração (Dias, 2012).

Assim como no *bagging*, o *boosting* tem os pontos iniciais muito parecidos, dividindo a amostra em 2 conjuntos: conjunto de treinamento e conjunto de validação. Porém, a partir do conjunto de treinamento, são retiradas M amostras ponderadas, onde será aplicado o classificador $f_K(x)$ em cada amostra. Ao efetuar o processo, teremos o classificador da seguinte forma

$$F(x) = \sum_{m=0}^{M} c_m f_m(x),$$

onde c_k é o peso que será atribuído em cada iteração, conforme a performance do classificador.

Para uma melhor compreensão dos procedimentos, segue a descrição simplificada do método *boosting* (extraído de Silva, 2023):

- **Algoritmo** *Adaboost* **de aprendizagem.** (LISKA *et al.*, 2012); (Alfaro *et al.*, 2013).

  Seja o conjunto de treinamento $T = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ e o conjunto de classes[1] $C = \{-1, 1\}$, o classificador do tipo *boosting* é dado por:

$$\text{sinal}\,[F(\mathbf{x})] = \text{sinal}\left[\sum_{m=1}^{M} c_m f_m(\mathbf{x})\right], \tag{3.1}$$

  em que: $\begin{cases} \text{M é o número de iterações num processo de refinamento do classificador;} \\ f_m(\mathbf{x}) \in C, \text{ é um classificador base (árvore, regressão logística, etc...);} \\ c_m, \ m = 1, 2, \ldots, M, \text{ são constantes obtidas em cada etapa do processo.} \end{cases}$

  A partir do conjunto de treinamento T, o procedimento *AdaBoost* ajusta o classificador $f_m(\mathbf{x})$ iterativamente, dando pesos menores aos itens classificados corretamente e maiores àqueles classificados errados. Os pesos são ajustados em cada etapa de forma que, ao final do processo, obtêm-se o classificador combinado, dado em (3.1) (LISKA *et al.*, 2012).

[1] O conjunto das classes pode, ainda, ser definido como $C = \{0, 1\}$

A seguir, temos os passos do algoritmo:

1. Seja o conjunto de treinamento $T = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, em que $\mathbf{x}_i \in \mathbf{R}^p$ e $y_i \in C$. Considere, inicialmente, pesos iguais para as observações $\mathbf{x}_i$:

$$w_i^m = \frac{1}{n}, \quad i = 1, 2, \ldots, n.$$

2. Fixe o número de iterações M e, para $m = 1, 2, \ldots, M$, faça:

   (a) Ajuste $f_m(\mathbf{x})$ considerando os pesos w_i^m para cada observação do conjunto de treinamento;

   (b) Calcule os erros ε_m e as constantes c_m, fazendo:

$$\varepsilon_m = \frac{\sum_{i=1}^n w_i^m I\left[y_i \neq f_m(\mathbf{x}_i)\right]}{\sum_{i=1}^n w_i^m},$$

$$c_m = \frac{1}{2} \ln\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right),$$

   O peso ε_m representa a média ponderada dos erros, com pesos $(w_1, \ldots, w_n)$.

   (c) Atualize os pesos w_i fazendo:

$$w_i^{m+1} = \begin{cases} \dfrac{w_i^m \mathrm{e}^{-c_m}}{z^m}, & \text{se} \quad y_i = f_m(\mathbf{x}_i) \quad \text{(classificação correta)}, \\ \\ \dfrac{w_i^m \mathrm{e}^{c_m}}{z^m}, & \text{se} \quad y_i \neq f_m(\mathbf{x}_i) \quad \text{(classificação errada)}, \end{cases}$$

   em que $z^m = \sum_{i=1}^n w_i^m \exp\{-c_m\, y_i\, f_m(\mathbf{x_i})\}$, é a constante de normalização.

   Em cada iteração, os pesos $w_i, i = 1, 2, \ldots, n$, são ajustados de forma que as observações classificadas erradas tenham uma ponderação maior.

3. Obter o classificador combinado

$$F(\mathbf{x}) = \sum_{m=1}^{M} c_m f_m(\mathbf{x}).$$

A classificação final, portanto, é dada por.

$$\hat{y}_i = \text{sinal}\left[F(\mathbf{x}_i)\right], \quad i = 1, 2, \ldots, n.$$

Na Figura 3.3 temos a representação do processo do método *boosting*:

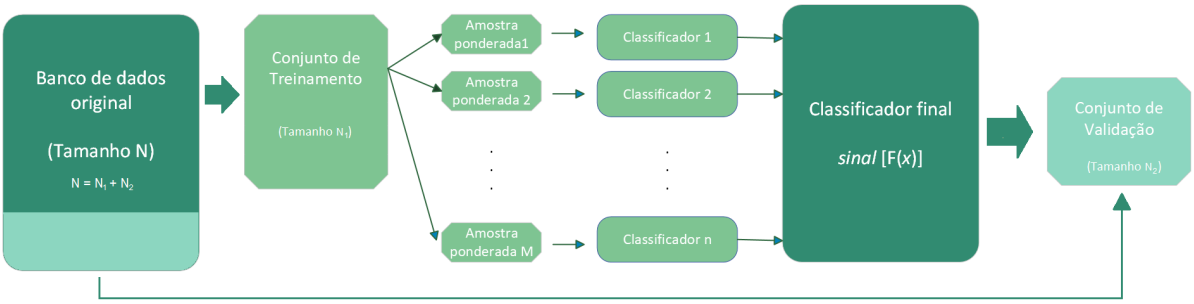

Figura 3.3: Esquema explicativo do método *Boosting*.
Fonte: Silva (2023).

3.2.3 *Overfitting*

O objetivo da classificação é encontrar resultados com o máximo de acertos possível. No entanto, quando lidamos com um grande número de variáveis, pode surgir um problema no classificador em que ele se ajusta excessivamente aos dados, resultando em um desempenho inconsistente do modelo. Isso pode levar a uma redução na taxa de acertos quando o modelo é aplicado a dados distintos no futuro. Esse fenômeno é conhecido como *overfitting* (sobreajuste), conforme discutido em James *et al.* (2013).

O *overfitting* ocorre quando um método de classificação se ajusta excessivamente ao conjunto de treinamento, perdendo a capacidade de prever com precisão resultados futuros, o que pode resultar em uma classificação inconsistente.

Para evitar este problema, é essencial avaliar se o modelo realmente requer todas as variáveis presentes no conjunto de dados para sua construção. Uma abordagem eficaz para evitar o *overfitting* é a técnica chamada *data splitting*, que envolve a divisão do conjunto de dados original em dois segmentos: o conjunto de treinamento, usado para ajustar os parâmetros do modelo, e o conjunto de validação (ou teste), usado para avaliar o desempenho do modelo.

Essa divisão é realizada de forma aleatória, usualmente com 60% dos dados destinados ao treinamento e os 40% restantes à validação. Dessa forma, ao isolar um conjunto de dados independente para validação, é possível verificar se o modelo está generalizando

bem os padrões aprendidos durante o treinamento ou se está simplesmente memorizando os dados de treinamento, o que ajudará a evitar problemas de superajuste (Silva, 2023).

Capítulo 4

Medidas de avaliação do classificador

Tendo em vista que utilizaremos o *Bagging* e *Boosting* para gerar classificadores, é do nosso interesse saber qual dos classificadores obteve a melhor performance. Dessa forma, para avaliar a performance, iremos utilizar medidas baseadas na matriz de confusão e a área sob a curva *ROC*.

4.1 Matriz de confusão

Em situações envolvendo a classificação de duas classes (classificação binária), a matriz de confusão é uma representação tabular com duas linhas e duas colunas, fornecendo informações sobre o número de verdadeiros positivos, falsos negativos, falsos positivos e verdadeiros negativos. As linhas da matriz correspondem aos valores observados das classes, enquanto as colunas representam os valores previstos, ou vice-versa.

A designação "matriz de confusão"deriva da facilidade em determinar, a partir dessa matriz, se o classificador está confundindo as duas classes durante o processo de classificação. Um exemplo de matriz de confusão é dada pela tabela 4.1.

Tabela 4.1: Matriz de confusão.

		Predição		
		0	1	
Real	0	VN	FP	POS
	1	FN	VP	NEG
		PN	PP	N

Na matriz de confusão apresentado acima, temos N é o número de unidades amostrais

presentes no conjunto de validação, PP é o número das unidades amostrais classificadas como positivas (classificados pelo modelo como 1), PN é o número de unidades amostrais classificadas como negativas (classificados pelo modelo como 0), POS é o número de unidades amostrais que pertencem à classe positivo (são reais positivos), NEG é o número de unidades amostrais que pertencem à classe negativo (são reais negativos), VP é o número de unidades amostrais classificadas corretamente como positivas (são chamados verdadeiros positivos ou VP), FN é o número de unidades amostrais positivas, classificadas de forma errada como negativas (falsos negativos ou FN), FP é o número de unidades amostrais negativas, classificadas de forma errada como positivas (falsos positivos ou FP) e por fim, VN é o número de unidades amostrais classificadas como negativas (verdadeiro negativo ou VN).

4.2 Medidas de desempenho baseadas na matriz de confusão

Na análise preditiva, a matriz de confusão é utilizada por conta da sua capacidade de proporcionar uma avaliação mais detalhada do desempenho do processo de predição. Isso ocorre porque ela não se baseia apenas na proporção de classificações corretas (acurácia). A acurácia, quando utilizada isoladamente, pode levar a interpretações equivocadas, especialmente em conjuntos de dados desbalanceados. Diante disso, apresentamos a seguir algumas métricas derivadas da matriz de confusão, as quais serão utilizadas para avaliar o desempenho do processo de classificação proposto neste estudo.

Vamos levar em consideração as mesmas nomenclaturas da seção 4.1, onde VP é o número das unidades amostrais que são verdadeiros positivos, FN é o número das unidades amostrais que são falsos negativos, FP é o número das unidades amostrais que são falsos positivos e VN é o número das unidades amostrais que são verdadeiros negativos. A partir dessas métricas, vamos definir algumas medidas de performance:

- **Sensibilidade (S):** é uma medida que traz a proporção de unidades amostrais positivas que foram classificadas de forma correta.

$$S = \frac{VP}{VP+FN} = \frac{VP}{POS}$$

- **Especificidade (E):** é uma medida que traz a proporção de unidades amostrais negativas que foram classificadas de forma correta.

$$E = \frac{VN}{VN+FP} = \frac{VN}{NEG}$$

- **Acurácia:** é a proporção de classificações corretas, tanto de casos positivos quanto negativos.

$$acurácia = \frac{VP+VN}{N} = \frac{VP+VN}{NEG}$$

- **Precisão:** é a proporção de classificações positivas corretas comparado com os casos positivos reais.

$$Precisão = \frac{VP}{VP+FN}$$

- **Escore-F:** conhecida também como *"F-mesure"* é uma média ponderada entre precisão e sensibilidade. O valor fica entre 0 a 1. Quanto mais próximo de 1, melhor o desempenho do classificador.

$$Escore\text{-}F = \frac{(1+\beta)\times precisão \times sensibilidade}{(\beta^2)\times precisão + sensibilidade}$$

onde,

$$\beta = \frac{sensibilidade}{precisão}$$

4.3 Área sob a curva *ROC (AUC)*

A capacidade de avaliar e selecionar modelos de classificação precisos sempre foi de grande importância dentro do cenário de análise de dados e tomadas de decisões. A eficiência dos modelos é frequentemente medida pela sua capacidade de distinguir corretamente entre diferentes classes, como diagnosticar doenças, identificar fraudes financeiras, entre outros. Um dos métodos mais utilizados para a avaliação desses modelos é a curva *ROC* (*Receiver Operating Characteristic*) (Prati *et al.*, 2008).

A Curva *ROC* é uma representação gráfica simples, porém robusta, que possibilita a análise das mudanças na sensibilidade e especificidade em relação a diferentes pontos de corte na probabilidade estimada (*thresholds*). A Área sob a curva (*AUC - Area Under the Curve*) é uma medida que simplifica a comparação entre diferentes Curvas *ROC*.

Para um melhor entendimento, será mostrado alguns exemplos de gráficos de curva *ROC*.

Na tabela 4.1, são feitos os cruzamentos das classificações (positivo ou negativo) das amostras reais com as classificações preditas por algum método como por exemplo, classificação por árvore.

Na avaliação de testes de diagnósticos, em epidemiologia, a tabela 4.1 representa os resultados da aplicação de testes, sendo as categorias "negativo" e "positivo" representadas, respectivamente, pelos resultados 0 e 1.

4.3.1 Sensibilidade e especificidade

Sob o enfoque epidemiológico, sensibilidade é a probabilidade de classificar corretamente um indivíduo cujo estado real seja definido como positivo em relação à condição que o teste avalia, razão pela qual também é chamada de taxa de verdadeiros positivos (TVP) (de Ullibarri Galparsoro e Fernández, 1998). No nosso contexto, na avaliação de classificadores, vamos definir a sensibilidade como sendo a probabilidade do classificador classificar corretamente um indivíduo da categoria "0" como sendo, de fato, dessa categoria. A partir da tabela 4.1, temos a seguinte fórmula:

$$\text{P(classificar na categoria 0 | pertence a categoria 0)} = \frac{TP}{POS}$$

A sensibilidade é o complementar da taxa de falsos negativos (TFN), calculada por $\frac{FN}{POS}$.

Já a especificidade é a probabilidade de classificar corretamente um indivíduo cuja categorial real seja definido como "1". A especificidade é o complementar da taxa de falsos positivos (TFP) (de Ullibarri Galparsoro e Fernández, 1998). A partir da tabela 4.1, temos que a fórmula é:

$$\text{P(classificar na categoria 1 | pertence a categoria 1)} = \frac{TN}{NEG} = 1 - TFP$$

Capítulo 5

Exemplo de aplicação

5.1 Banco de Dados - *Titanic*

O conjunto de dados utilizado nesse exemplo corresponde aos dados dos passageiros do *Titanic*, separados em conjunto de treinamento (75% da amostra) e conjunto de teste (25% da amostra). Ao todo temos uma amostra de 891 observações. Os dados foram obtidos de: https://www.kaggle.com/c/titanic/data.

Vale lembrar que esta análise feita com o banco de dados do *Titanic*, foi para entender o comportamento que o classificador e as medidas de desempenho pode trazer de resultado. Dessa maneira, será feito, no banco de dados do *Alzheimer*, uma análise com uma profundidade maior, encontrando os hiperparâmetros, variáveis para o modelo e será aplicado algumas medidas de desempenho a mais.

Seguindo com a análise com a base de dados do *Titanic*, as variáveis disponíveis nessa base de dados está listado na tabela 5.1:

O conjunto de treinamento servirá para treinar os classificadores e contém os resultados de nosso interesse, determinado pela variável *survival*, sendo 0 = "o passageiro não sobreviveu"ao naufrágio ou 1 = "o passageiro sobreviveu". Já o conjunto de teste servirá para observar o quão bom será o desempenho do nosso classificador em relação aos 25% restantes dos dados.

Para a realização do estudo, foi utilizado o modelo de regressão por árvore e foram selecionadas as variáveis sexo, idade e classe do passageiro.

Para o primeiro passo do estudo, foi observado o número de proporção dos sobreviventes (variável *survival* = 1) e não sobreviventes (variável *survival* = 0) que será utilizado mais a frente do trabalho. A proporção de sobreviventes foi de aproximadamente 38,9%

Tabela 5.1: Descrição das variáveis do conjunto de dados *Titanic*.

Variável	**Descrição**
survival	sobrevivente (0 = Não, 1 = Sim)
pclass	classe do passageiro (1 = 1ª classe, 2ª = segunda, 3 = 3ª classe)
sex	sexo
age	idade
sibsp	quantidade de irmão(s) e cônjuge
parch	quantidade de pais/filho
ticket	número da passagem
fare	valor da passagem
cabin	número da cabine
embarked	porto de embarque (C = *Cherbourg*, Q = *Queenstown*, S = *Southampton*)

e de não sobreviventes foi de 61,1%.

5.1.1 Aplicação do método *bagging*

Para a utilização de *bagging*, foram consideradas $B = 50$ amostras para o *booststrap*, sendo que o pacote utilizado foi o *adabag*. Dessa maneira, será apresentado abaixo a tabela de confundimento para ter uma ideia de como está o desempenho do nosso classificador.

Tabela 5.2: Matriz de confusão utilizando o método *bagging*.

		Predição		
		0	1	
Real	0	365	43	408
	1	76	184	260
		441	227	668

A taxa de erro de classificação observado foi de 17.81%, dada pela soma da diagonal secundária (119) dividida pelo total de amostras (668). A taxa de erros não aparenta uma classificação ruim. Após o resultado anterior, podemos observar pela Figura 5.1 a evolução das taxas de erros de classificação durante o procedimento do *bagging*.

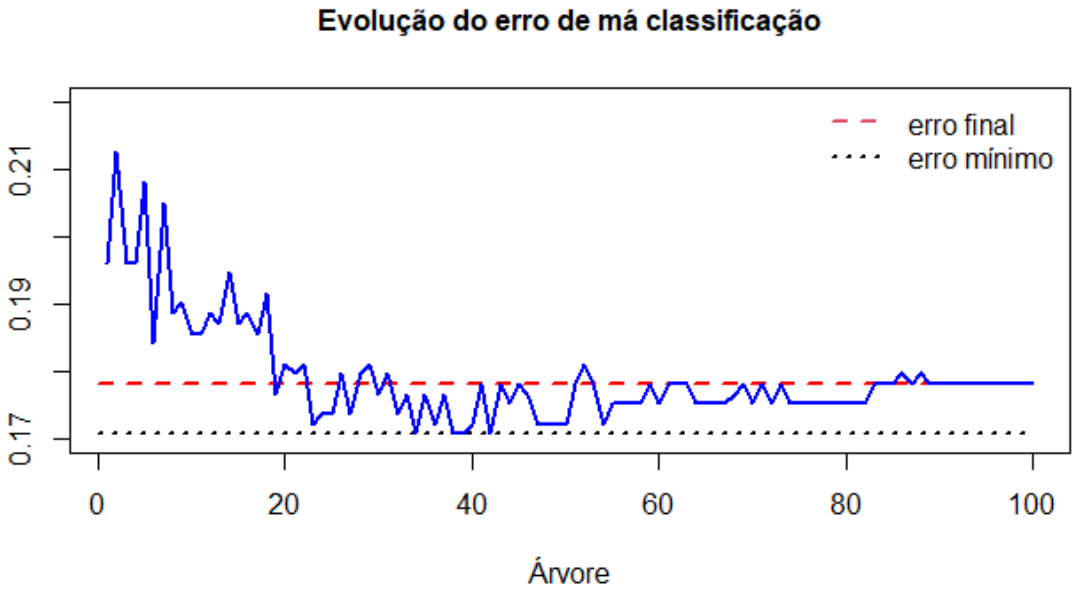

Figura 5.1: Evolução da taxa de erro do *bagging*, aplicado aos dados do *Titanic*.

É notável que há uma variação muito grande das taxas de erro de classificação ao decorrer do número de classificações feitas por árvores.

Para o estudo da curva *ROC* foram selecionados os pontos de corte (*threshold*): $0.1, 0.2, ...0.8, , 0.9$ na definição da classificação.

De modo geral, a classificação retorna os vetores contendo as probabilidades de classificação. A classificação é definida a partir do ponto de corte. Dessa maneira, foram observados os valores de sensibilidade e especificidade apresentados na Tabela 5.3.

Tabela 5.3: Valores de sensibilidade e especificidade do *bagging* aplicado aos dados do *Titanic*.

Corte	**Sensibilidade**	**Especificidade**
0.1	0.872	0.699
0.2	0.855	0.711
0.3	0.844	0.783
0.4	0.831	0.780
0.5	0.827	0.810
0.6	0.816	0.857
0.7	0.798	0.854
0.8	0.791	0.868
0.9	0.749	0.940

Como citado anteriormente, o ponto de corte usualmente utilizado é 0.5. Porém, ao observar os pares de valores (sensibilidade e especificidade) na Tabela 5.3, pode-se considerar o ponto de corte 0.6 como ideal. Para ter uma maior certeza, será apresentado

na Figura 5.2, a taxa de erro nos diferentes pontos de corte e logo em seguida, a Tabela 5.4 com os valores das taxas de erro em cada ponto de corte.

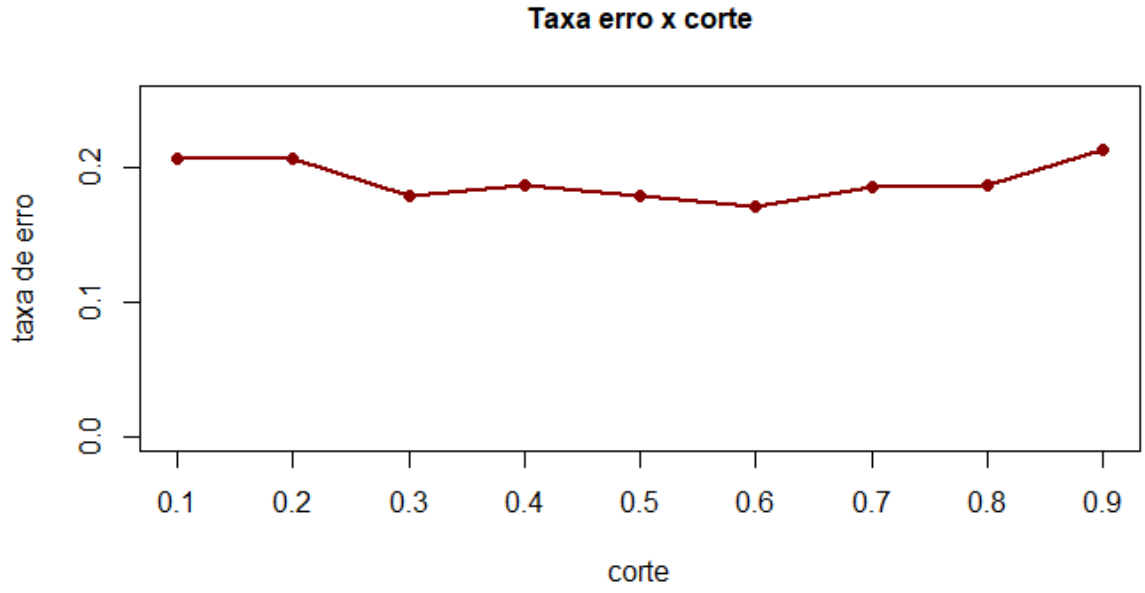

Figura 5.2: Taxa de erro em cada ponto de corte do *bagging* aplicado aos dados do *Titanic*.

Tabela 5.4: Taxa de erro de classificação para cada ponto de corte do *bagging* aplicado aos dados do *Titanic*.

Corte	**taxa de erro**
0.1	0.2065
0.2	0.2065
0.3	0.1781
0.4	0.1871
0.5	0.1781
0.6	0.1706
0.7	0.1856
0.8	0.1871
0.9	0.2125

Pela taxa de erro de classificação, o ponto de corte 0.6 apresenta a menor taxa de erro. Além das taxas de erro de má classificação, vamos considerar, também, a proporção de observações que são classificados como sobreviventes ($survival = 1$) em cada ponto de corte e compará-las com a proporção real.

Tabela 5.5: Proporção de classificação de 1 para cada ponto de corte para o *bagging* aplicando aos dados do *Titanic*.

Corte	taxa de erro	Proporção da predição de 1
0.1	0.2065	0.4580
0.2	0.2065	0.4311
0.3	0.1781	0.3727
0.4	0.1871	0.3607
0.5	0.1781	0.3398
0.6	0.1706	0.3053
0.7	0.1856	0.2874
0.8	0.1871	0.2739
0.9	0.2125	0.2005

Ao observar as proporções, o ponto de corte que mais se assemelha ao da proporção real (aproximadamente 39.8%) ocorreu no ponto de corte 0.3 com 37.27%, o que chama a atenção, pois a taxa de erro de classificação desse ponto não foi a menor.

5.1.2 Aplicação método *boosting*

Para a utilização do método *boosting*, foi selecionado o número de iterações e observadas as taxas de erro de classificação. Nas figuras 5.3 e 5.4 são apresentadas a evolução da taxa de erros para 15 e 100 iterações, respectivamente.

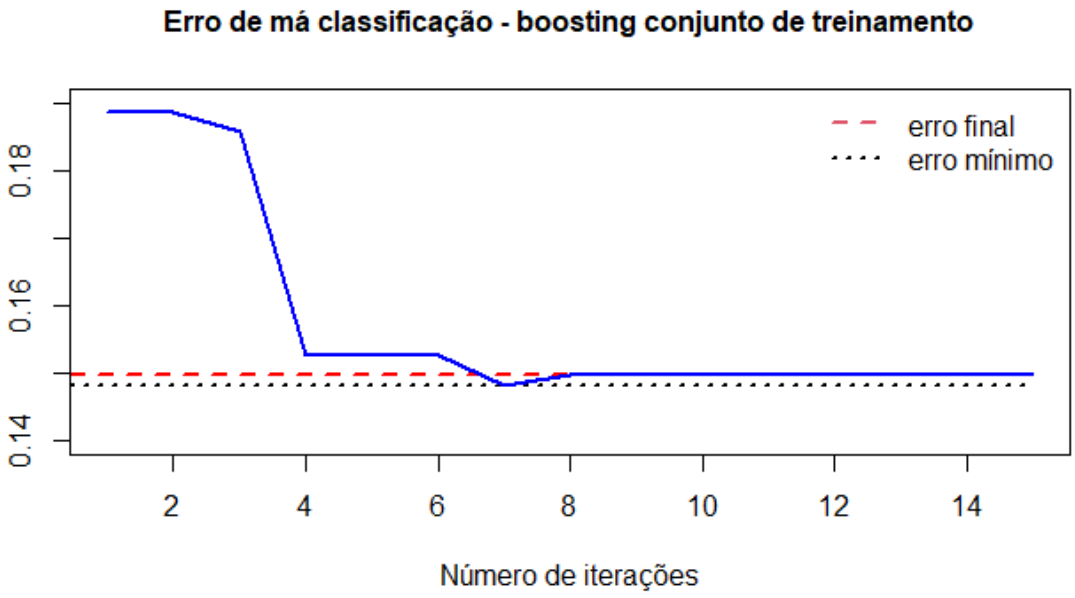

Figura 5.3: Erro de classificação para 15 iterações para o *boosting* aplicando aos dados do *Titanic*.

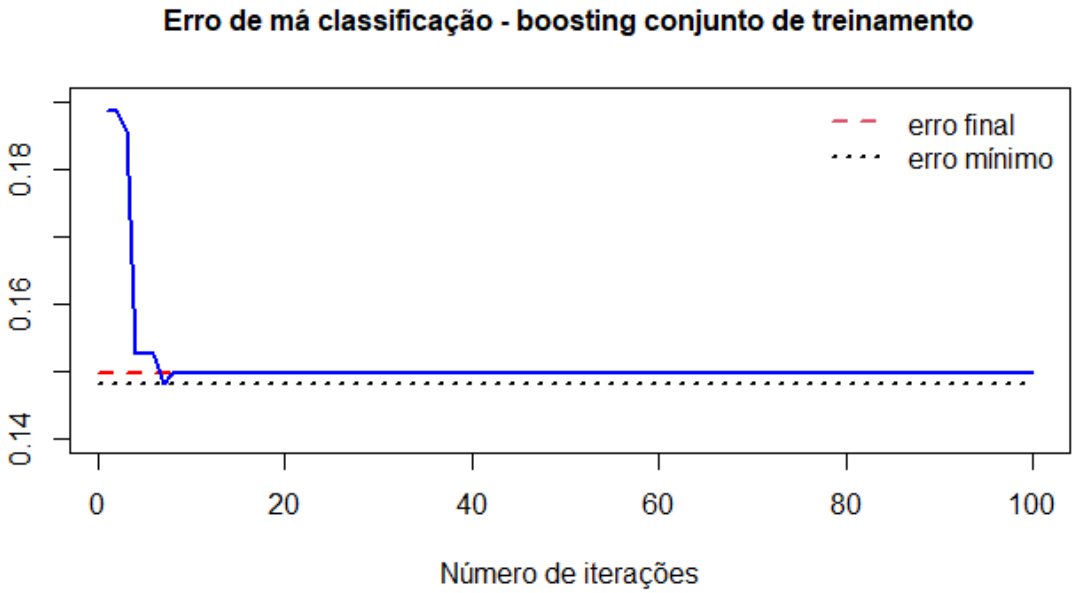

Figura 5.4: Erro de classificação para 100 iterações para o *boosting* aplicando aos dados do *Titanic*.

Ao comparar as figuras 5.3 e 5.4, pode-se observar uma estabilização do erro de classificação a partir da 8ª iteração e continua constante, mostrando uma estabilização desses erros. Portanto, seguiremos com 8 iterações.

Após os resultados anteriores, foi construída a matriz de confundimento para um melhor entendimento sobre a taxa erro de classificação de sobreviventes e não sobreviventes. Foi obtido a Tabela 5.6.

Tabela 5.6: Matriz de confusão utilizando o método *boosting*.

		Predição		
		0	1	
Real	0	373	35	408
	1	65	195	260
		438	230	668

No software *R*, foi utilizado o comando *predict.boosting()* do pacote *adabag*, onde faz o cálculo da taxa de erro de classificação e a taxa resultante foi de 14.97%. Comparando com a taxa de erro do método *bagging*, pode-se dizer que o *boosting* está realizando uma classificação ligeiramente melhor.

Agora passaremos para a curva *ROC*. Os pontos de corte utilizados para este estudo foram: $0.1, 0.2, ...0.8, , 0.9$. Dessa maneira, foram observados os seguintes valores de sensibilidade e especificidade:

Tabela 5.7: Valores de sensibilidade e especificidade do *boosting* aplicado aos dados do *Titanic*.

Corte	Sensibilidade	Especificidade
0.1	1.000	0.391
0.2	1.000	0.418
0.3	0.946	0.540
0.4	0.905	0.690
0.5	0.851	0.847
0.6	0.774	0.959
0.7	0.747	0.976
0.8	0.704	1.000
0.9	0.675	1.000

O ponto de corte mais usual utilizado é 0.5. Ao observar os valores de sensibilidade e especificidade, o melhor par observado de valores é o ponto de corte (probabilidade) 0.5, não alterando do que é utilizado usualmente.

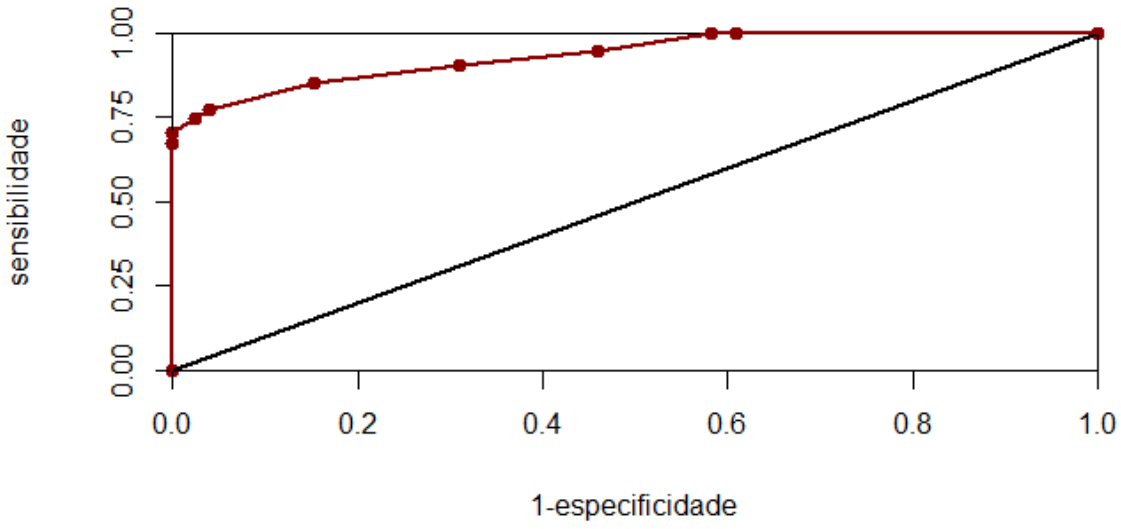

Figura 5.5: Curva *ROC* do *boosting* aplicado aos dados do *Titanic*.

Na figura 5.5, foram adicionadas os pontos de corte 0 e 1 para que a curva ficasse completa. Dessa maneira, consegue-se notar que visualmente o melhor ponto de corte é 0.4. Porém, ao se obter tal resultado, é necessário considerar, também, a taxa de erro de classificação.

Nota-se que o ponto de corte 0.4 tem uma taxa de classificação mais elevada com 20.06% o que nos leva a acreditar que o melhor ponto de corte seria o 0.5, com 14.97%.

Dessa forma, utilizando o *boosting* em um conjunto de dados com classificações de

Tabela 5.8: Taxa de erro de classificação para cada ponto de corte para o *bagging* aplicado aos dados do *Titanic*.

Corte	**taxa de erro**
0.1	0.6048
0.2	0.5419
0.3	0.3338
0.4	0.2006
0.5	0.1497
0.6	0.1841
0.7	0.2096
0.8	0.2560
0.9	0.2934

aproximadamente 40% de sobreviventes e 60% de não sobreviventes, o melhor ponto de corte é 0.5.

Neste trabalho, apresentamos as avaliações das classificações comparando a taxa de erro, curva *ROC* e proporção de classificação dos sobreviventes ($survival = 1$) para diferentes pontos de corte.

5.2 Base de dados - Mal de *Alzheimer*

Este conjunto de dados é composta por informações sobre a saúde de 2149 pacientes. Temos 34 variáveis quantitativas e a variável resposta é o *Diagnosis* (Diagnóstico), onde 0 corresponde a "Indica que não há mal de *Alzheimer*"e 1 corresponde a "Indica que há mal de *Alzheimer*". Esse conjunto de dados foi separado em conjunto de treinamento (75% da amostra) e conjunto de teste (25% da amostra) para diferentes proporções de amostra onde será feito os estudos individualmente. Os dados foram obtidos do site: https://www.kaggle.com/datasets/rabieelkharoua/alzheimers-disease-dataset

As variáveis disponíveis nessa base de dados estão listados na tabela 5.9:

Foi utilizado a função *randomForest*, que utiliza um conjunto de árvore de decisão, para selecionar o modelo para utilizar nos métodos abordados neste trabalho. Para a escolha do modelo, foi escolhido a métrica *MeanDecreaseAccuracy* que basicamente mostra quanto a variável afeta a acurácia do modelo para classificar um indivíduo como 0 (não tem *Alzheimer*) ou 1 (tem indícios de *Alzheimer*). No Figura 5.6 é possível notar que há apenas 5 variáveis que afetam significativamente a acurácia do modelo.

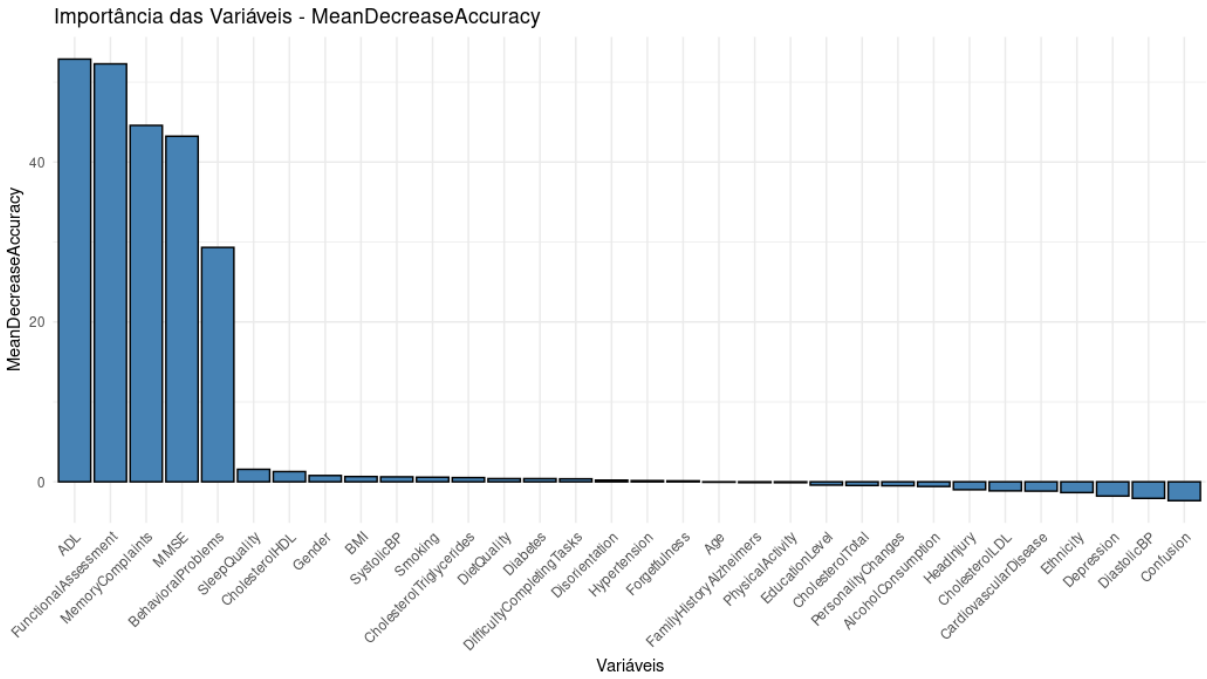

Figura 5.6: Importância das variáveis utilizando a métrica *MeanDecreaseAccuracy*.

Para evitar o super ajuste que o modelo pode gerar no conjunto de treino (*overfitting*), foram selecionados 3 variáveis dentre as 5 mais significativas para compor o modelo.

Tabela 5.9: Descrição das variáveis da base de dados do *Alzheimer*.

Variável	**Descrição**
Age	Idade
Gender	Gênero (Masculino = 0 e Feminino = 1)
Ethnicity	Etnia (0: Branco, 1: Americano Africano, 2: Asiático e 3: Outros)
EducationLevel	Nível de Educação (0: Nenhum, 1: Médio, 2: Bacharelado e 3: Pós Graduação)
BMI	Índice de massa corporal
Smoking	Fumante (0: Não e 1: Sim)
AlcoholConsumption	Consumo semanal de álcool (em unidade)
PhysicalActivity	Atividade Física Semanal (em horas)
DietQuality	Qualidade da dieta (entre 0 e 10)
SleepQuality	Qualidade do sono (entre 4 e 10)
FamilyHistoryAlzheimers	Histórico de *Alzheimer* na família (0: Sim e 1: Não)
CardiovascularDisease	Presença de doença cardiovascular (0: Sim e 1: Não)
Diabetes	Presença de Diabetes (0: Sim e 1: Não)
Depression	Presença de depressão (0: Sim e 1: Não)
HeadInjury	Histórico de lesão na cabeça (0: Sim e 1: Não)
Hypertension	Presença de hipertensão (0: Sim e 1: Não)
SystolicBP	Pressão arterial sistólica (valores entre 90 e 180)
DiastolicBP	Pressão arterial diastólica (valores entre 60 e 120)
CholesterolTotal	Nível de colesterol total (valores entre 150 e 300 mg/dL)
CholesterolLDL	Níveis de colesterol de lipoproteína de baixa densidade (valores entre 50 a 200 mg/dL)
CholesterolHDL	Níveis de colesterol de lipoproteína de alta densidade (valores entre 20 a 100 mg/dL)
CholesterolTriglycerides	Níveis de triglicerídeos (valores entre 50 a 400 mg/dL)
MMSE	Pontuação do Mini-Exame do Estado Mental (valores entre 0 a 30)
FunctionalAssessment	Pontuação de avaliação funcional (valores entre 0 a 10)
MemoryComplaints	Presença de queixas de memória (0: Não e 1: Sim)
BehavioralProblems	Presença de queixas de memória (0: Não e 1: Sim)
ADL	Pontuação de atividades de vida diária (0: Não e 1: Sim)
Confusion	Presença de confusão (0: Não e 1: Sim)
Disorientation	Presença de desorientação (0: Não e 1: Sim)
PersonalityChanges	Presença de múltipla personalidade (0: Não e 1: Sim)
DifficultyCompletingTasks	Presença de dificuldade para completar uma tarefa (0: Não e 1: Sim)
Forgetfulness	Presença de esquecimento (0: Não e 1: Sim)
Diagnosis	Status de diagnóstico para doença de *Alzheimer* (0: Não e 1: Sim)

Dessa forma, vamos seguir com o modelo que contém as seguintes variáveis: *FunctionalAssessment*, *BehavioralProblems* e *ADL*.

Para o ajuste dos hiperparâmetros dos modelos para os métodos *bagging* e *boosting*,

foram testado várias combinações de hiperparâmetros e foi observado o valor da acurácia para determinar quais hiperparâmetros seriam usados. Foi ajustado para a proporção de amostra original (65% de 0's e 35% de 1's).

Nas próximas sessões, serão feitos análises comparando o desempenho do mesmo modelo (mesmas variáveis e hiperparâmetros) em proporções diferentes de classificados como 0's e 1's. As proporções de 0's e 1's que iremos considerar são, respectivamente: $(65\%, 35\%)$, $(35\%, 65\%)$, $(80\%, 20\%)$ e $(20\%, 80\%)$. Essas proporções foram escolhidas de forma aleatória a fim de testar o modelo em amostras desbalanceadas.

Com essas diferentes proporções, vamos buscar evidências de qual é o melhor ponto de corte em cada situação, lembrando novamente que, vamos considerar o mesmo modelo ajustado.

Vale lembrar que, o conjunto de treino foi aplicado para treinar o classificador e o conjunto de teste para testar o classificador ajustado com o conjunto de treino.

5.2.1 Aplicação do método *bagging* $(65\%, 35\%)$

Foi feito a utilização de *bagging* e os hiperparâmetros ajustados foram: $B = 50$ amostras para o *booststrap*, número de árvores ajustados para 10, profundidade máxima de uma árvore foi ajustado para 4 e amostra mínima em um nó para fazer a divisão é de 100 observações, sendo que o pacote utilizado foi o *adabag*. Dessa maneira, será apresentado abaixo a tabela de confundimento 5.10 para ter uma ideia de como está o desempenho do nosso classificador.

Tabela 5.10: Matriz de confusão utilizando o método *bagging*.

		Predição		
		0	1	
Real	0	929	122	1051
	1	147	414	561
		1076	536	1612

A taxa de erro de classificação observado na matriz de confusão (Tabela 5.10), foi de 16.68%, dada pela soma da diagonal secundária (269) dividida pelo total de amostras (1612). A taxa de erro não aparenta uma classificação boa dada a importância da classificação mais assertiva por se tratar de uma amostra da área da saúde, mais especifi-

camente na área de diagnóstico. Após o resultado anterior, será mostrado na Figura 5.7 a evolução das taxas de erro de classificação.

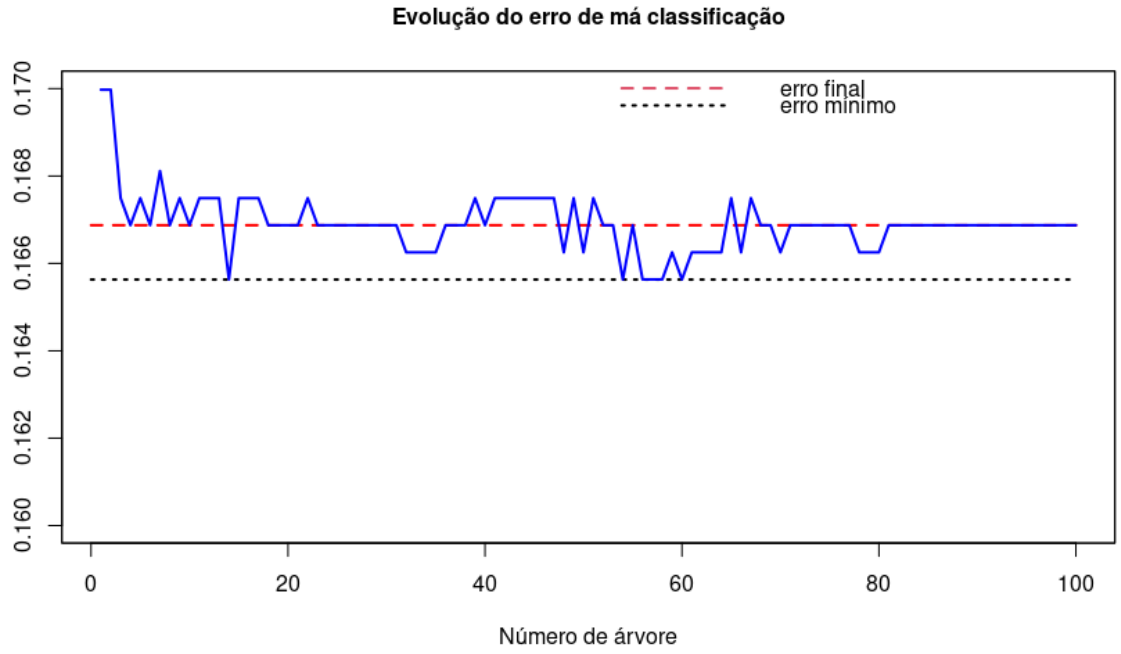

Figura 5.7: Evolução da taxa de erro do *bagging* , aplicado aos dados do *Alzheimer*.

Na Figura 5.7 é notável que há uma variação na taxa de erro que fica entorno de 16.68%, onde a taxa de erro máximo foi de 16.99% e a menor taxa de erro foi de 16.56%.

Ao aplicar o método *bagging* no conjunto de teste, foram obtidos os seguintes resultados a partir da matriz de confusão 5.11. A taxa de erro encontrado foi de 21.6%.

Tabela 5.11: Matriz de confusão utilizando o método *bagging* no conjunto de teste.

		Predição		
		0	1	
Real	0	286	52	338
	1	64	135	199
		350	187	537

O resultado apresentado na tabela 5.11 mostra que o modelo não está classificando bem, já que a questão de dar diagnósticos errados para os pacientes pode levar a situações mais sérias. Com isso, seguiremos com a análise para observar os resultados das medidas de desempenho.

Para medir o desempenho da classificação do modelo utilizando o método *bagging*, foram observados na tabela 5.12 a sensibilidade, especificidade, acurácia, precisão e a Escore-F (*F-Mesure*).

Tabela 5.12: Valores de sensibilidade e especificidade da base de *Alzheimer* com a aplicação do método *bagging* para o conjunto de teste.

Corte	**Sensibilidade**	**Especificidade**	**Acurácia**	**Precisão**	**Escore-F**
0.1	0.7136	0.8402	0.7933	0.7245	0.7190
0.2	0.7136	0.8432	0.7952	0.7282	0.7208
0.3	0.7085	0.8432	0.7933	0.7268	0.7176
0.4	0.6884	0.8432	0.7858	0.7211	0.7044
0.5	0.6784	0.8462	0.7840	0.7219	0.6995
0.6	0.6683	0.8491	0.7821	0.7228	0.6945
0.7	0.6583	0.8550	0.7821	0.7278	0.6913
0.8	0.6583	0.8698	0.7914	0.7486	0.7005
0.9	0.6332	0.8698	0.7821	0.7412	0.6829

O ponto de corte usualmente utilizado é 0.5. Para este tipo de amostra, o interessante é observar a sensibilidade e a medida-F (faz a utilização da sensibilidade) pois o interessante é acertar ao máximo na classificação dos positivos para que não haja erro no diagnóstico da doença. Na tabela 5.12, foi observado que o melhor ponto de corte foi 0.1 e 0.2 para sensibilidade e o ponto 0.2 para a medida-F. As demais medidas de desempenho foi colocado no trabalho a fim de observar o comportamento. Para ter uma maior certeza, será apresentado abaixo a tabela 5.13 contendo as taxa de erros e a proporção de classificados como 1 (pacientes com *Alzheimer*) nos diferentes pontos de corte.

Tabela 5.13: Taxa de erro de classificação e proporção de pacientes classificados como 1 para cada ponto de corte no conjunto de teste.

Corte	**taxa de erro**	**Proporção da predição de 1**
0.1	0.2067	0.3650
0.2	0.2048	0.3631
0.3	0.2067	0.3613
0.4	0.2142	0.3538
0.5	0.2160	0.3482
0.6	0.2179	0.3426
0.7	0.2179	0.3352
0.8	0.2086	0.3259
0.9	0.2179	0.3166

Pela taxa de erro de classificação que é possível observar na tabela 5.13, o ponto de

corte 0.2 apresenta a menor taxa de erro (20.48%).

Além das taxa de erro de má classificação, vamos considerar, também, a proporção que observações que são classificados como pacientes com *Alzheimer* (Doentes = 1) em cada ponto de corte.

Ao observar as proporções, o ponto de corte que mais se assemelha ao da proporção real (aproximadamente 35%) ocorreu no ponto de corte 0.5 com 34.82%.

Abaixo, será mostrado o gráfico da curva *ROC* para observar o comportamento para o método *bagging*.

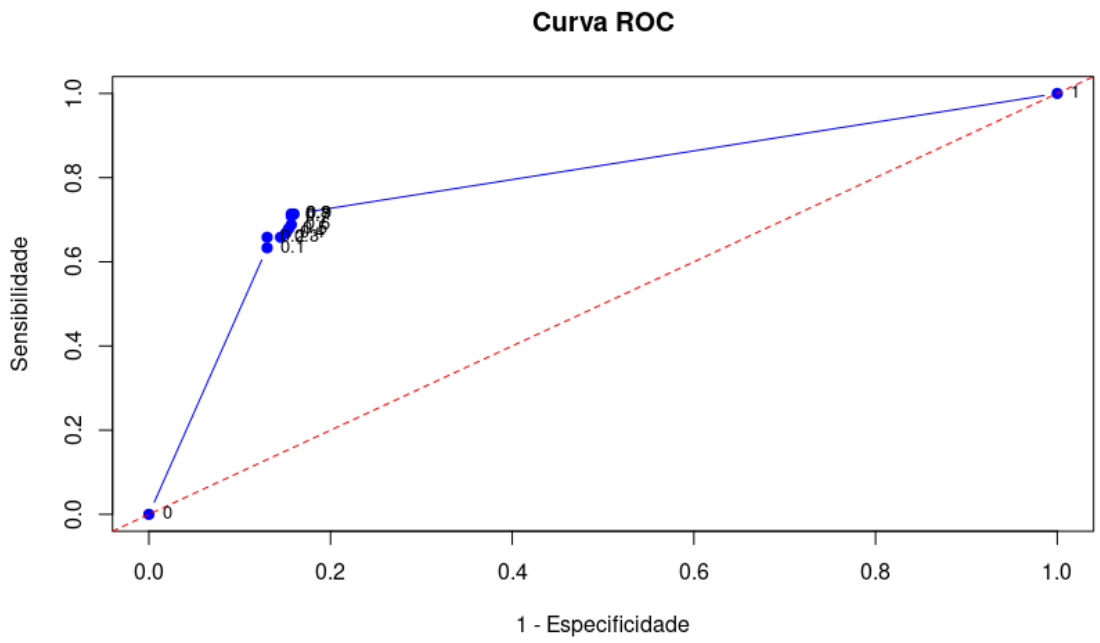

Figura 5.8: Curva *ROC* aplicado no conjunto de teste para o método *bagging*.

É possível observar uma aglomeração dos pontos na figura 5.8, dificultando um pouco a análise. Nas tabelas 5.12 e 5.13, conseguimos ter uma interpretação melhor dos resultados.

5.2.2 Aplicação método *boosting* (65%, 35%)

Para a utilização do método *boosting*, foi ajustado alguns hiperparâmetros para evitar que ocorra o *underfitting* ou *overfitting*. Os hiperparâmetros ajustados foram: a profundidade da árvore (ajustado para profundidade máximo 3) e mínimo de observações em um nó para fazer a divisão foi ajustado para 90 observações e o número de iterações foi de 300. Na figura 5.9 é apresentado a evolução da taxa de erro de classificação que será

observado para medir o desempenho do classificador ao utilizar o método *boosting*.

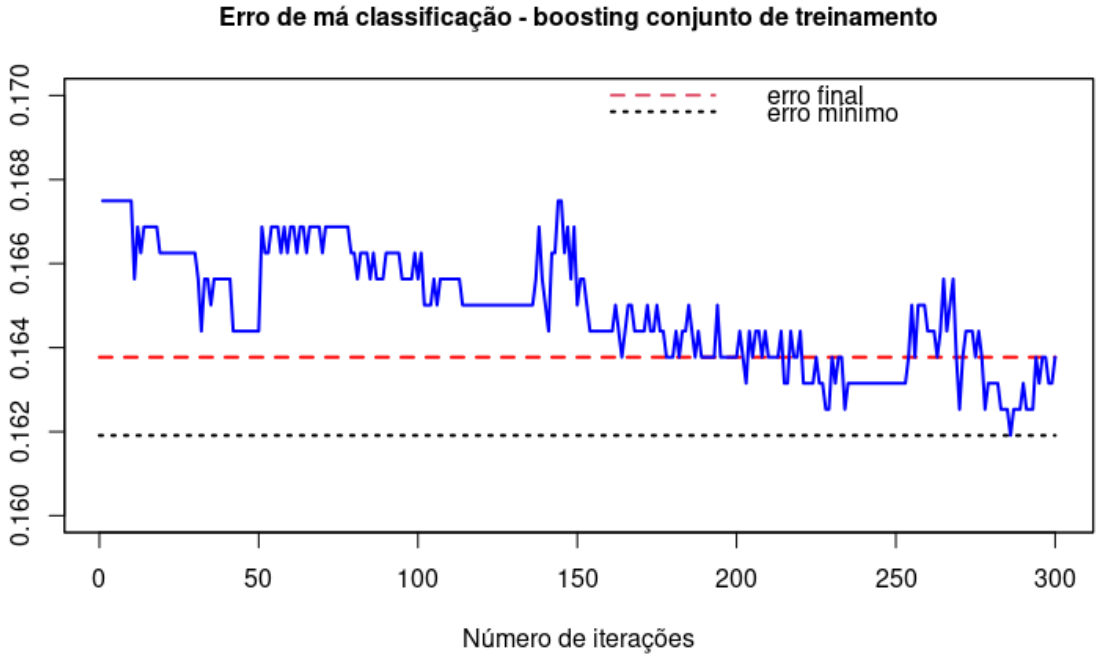

Figura 5.9: Evolução da taxa de erro do *boosting*, aplicado aos dados de treinamento do *Alzheimer*.

Ao observar a figura 5.9, pode-se observar uma melhora na taxa de erro no decorrer das iterações mas os erros acabam ficando próximos ao 16.37%. Essa queda se dá pois o método *boosting* faz uma classificação e as observações que foram classificadas de forma errada, são classificados novamente na próxima iteração com um peso maior, dando prioridade para classificar corretamente as observações com maiores pesos. Dessa forma é natural que o modelo entenda e consiga classificar sempre melhorando a classificação até um certo ponto. (ver seção 3.2.2)

Após os resultados anteriores, foi construída a matriz de confusão para um melhor entendimento sobre a taxa erro de classificação de pacientes com *Alzheimer*.

Tabela 5.14: Matriz de confusão utilizando o método *boosting* para a base de dados de treinamento do *Alzheimer*.

		Predição		
		0	1	
Real	0	935	148	1083
	1	116	413	529
		1051	561	1612

No software *R*, foi utilizado o comando *predict.boosting()* do pacote *adabag*, onde comparando com a taxa de erro do método *bagging*, pode-se dizer que o *boosting* tem um desempenho muito parecido.

Agora será aplicado o método *boosting* no conjunto de teste. O primeiro resultado que foi observado foi a tabela de confusão 5.15.

Tabela 5.15: Matriz de confusão utilizando o método *boosting* no conjunto de teste.

		Predição		
		0	1	
Real	0	286	52	338
	1	64	135	199
		350	187	537

É possível observar na tabela 5.15 uma taxa de erro de classificação de 21.6%. A taxa de erro no conjunto de teste, se comporta da mesma maneira que as taxas de erro do conjunto de treino.

Agora passaremos para a curva *ROC*. Os pontos de corte utilizados para este estudo foram: $0.1, 0.2, ...0.8, , 0.9$. Dessa maneira, foram observados os seguintes valores de sensibilidade e especificidade:

O ponto de corte usualmente utilizado é 0.5. Novamente, para este tipo de amostra, o interessante é observar a sensibilidade e a medida-F (faz a utilização da sensibilidade) pois é do nosso interesse acertar ao máximo na classificação dos positivos para que não haja erro no diagnóstico da doença.

Na tabela 5.16, foi observado que o melhor ponto de corte foi 0.1, 0.2 e 0.3 para sensibilidade (não apresenta um resultado que dê para tirar uma boa conclusão) e o ponto 0.5 para a medida-F. As demais medidas de desempenho foi colocado no trabalho a fim de observar o comportamento. Para ter uma maior certeza, será apresentado abaixo

Tabela 5.16: Tabela dos valores para medir o desempenho da classificação no conjunto de teste aplicando o método *boosting*.

Corte	Sensibilidade	Especificidade	Acurácia	Precisão	Medida-F
0.1	1.0000	0.0000	0.3480	0.3480	0.5163
0.2	1.0000	0.0000	0.3480	0.3480	0.5163
0.3	1.0000	0.1446	0.4423	0.3842	0.5552
0.4	0.9305	0.6660	0.7581	0.5979	0.7280
0.5	0.7362	0.8896	0.8362	0.7807	0.7578
0.6	0.3298	0.9905	0.7605	0.9487	0.4894
0.7	0.0000	1.0000	0.6520	0.0000	0.0000
0.8	0.0000	1.0000	0.6520	0.0000	0.0000
0.9	0.0000	1.0000	0.6520	0.0000	0.0000

a tabela 5.17 contendo as taxa de erros e a proporção de classificados como 1 (pacientes com *Alzheimer*) nos diferentes pontos de corte.

Tabela 5.17: Taxa de erro de classificação para cada ponto de corte utilizando o método *boosting* e a proporção de 1's no conjunto de teste.

Corte	taxa de erro	proporção de 1
0.1	0.6520	0.3480
0.2	0.6520	0.3480
0.3	0.5577	0.3480
0.4	0.2419	0.3238
0.5	0.1638	0.2562
0.6	0.2395	0.1148
0.7	0.3480	0.0000
0.8	0.3480	0.0000
0.9	0.3480	0.0000

Nota-se na tabela 5.17 que, realmente o ponto de corte 0.5 tem a melhor taxa de classificação com uma taxa de erro de classificação de 16.38%.

Dessa forma, utilizando o *boosting* em um conjunto de teste com classificações de aproximadamente 35% de pacientes com *Alzheimer* (classificados como 1) a 65% de pacientes sem *Alzheimer* (classificados como 0), os melhores pontos de corte são 0.1, 0.2 e 0.3 (resultados que não nos traz nenhuma conclusão).

Na figura 5.10, foram adicionadas os pontos de corte 0 e 1 para que a curva ficasse completa e foi observado uma área abaixo da curva (AUC) de aproximadamente 0.91. Dessa maneira, consegue-se notar que visualmente o melhor ponto de corte é 0.5. Porém, ao se obter tal resultado, é necessário considerar, também, a taxa de erro de classificação.

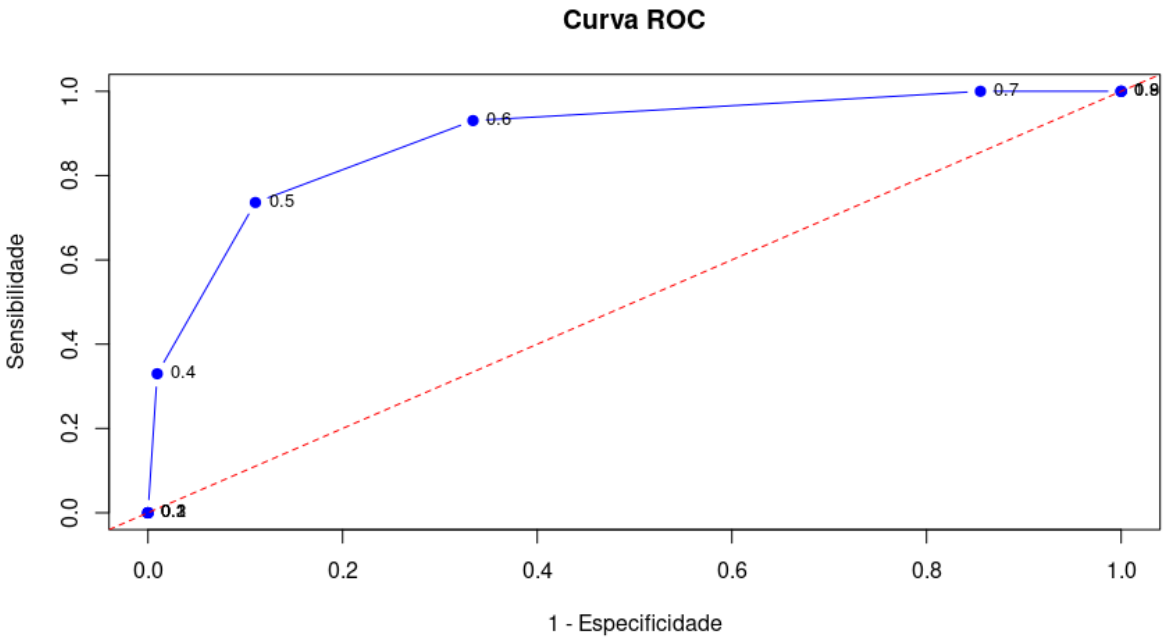

Figura 5.10: Curva *ROC* para o método *boosting* no conjunto de teste.

Mesmo com o desempenho pior do conjunto de teste, há um erro de classificação de 19.92% para pacientes com *Alzheimer*. Vale ressaltar, novamente que, a proporção da amostra desse estudo foi de aproximadamente 35% de pacientes com *Alzheimer* (Diagnóstico = 1) e 65% de pacientes sem *Alzheimer* (Diagnóstico = 0).

5.2.3 Aplicação do método *bagging* $(35\%, 65\%)$

Os hiperparâmetros ajustados foram os mesmo da análise anterior na seção 5.2.1, mudando somente o número da amostra que é de 1169 observações. Dessa maneira, será apresentado abaixo a tabela de confundimento para ter uma ideia de como está o desempenho do nosso classificador.

Tabela 5.18: Matriz de confusão utilizando o método *bagging*.

		Predição		
		0	1	
Real	0	251	55	306
	1	118	453	571
		369	508	877

A taxa de erro de classificação observado na matriz de confusão (Tabela 5.18), foi de

19.72%. A taxa de erro não aparenta uma classificação boa dada a importância da classificação mais assertiva por se tratar de uma amostra da área da saúde. Após o resultado anterior, será mostrado na Figura 5.11 com a evolução das taxas de erro de classificação.

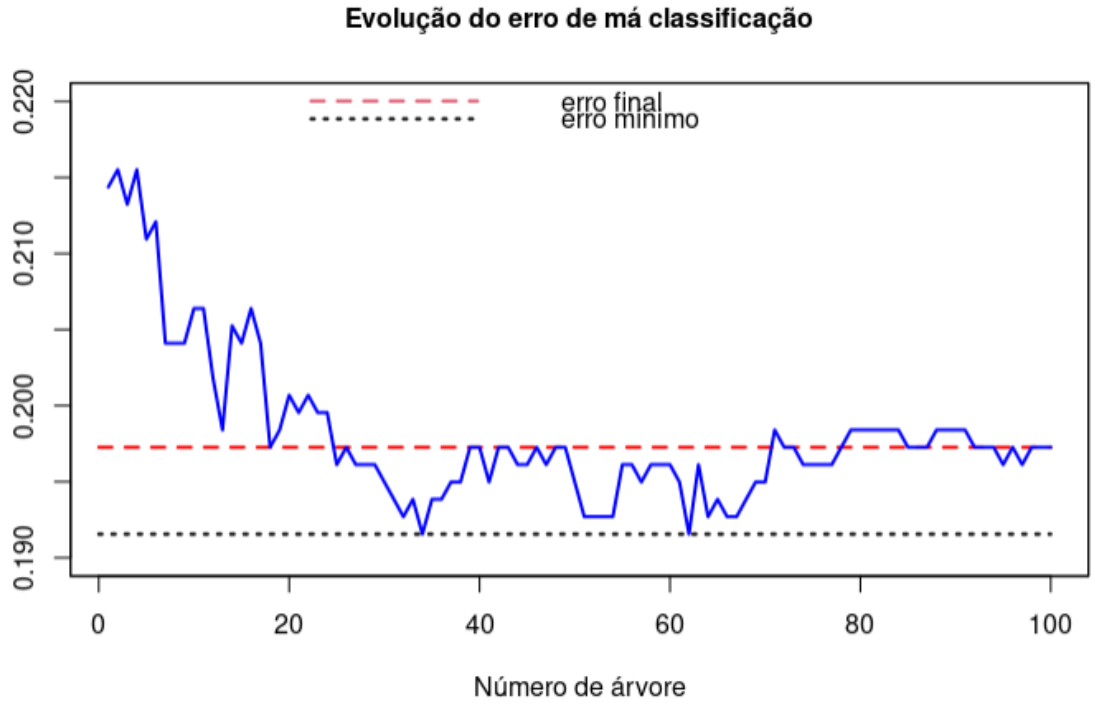

Figura 5.11: Evolução da taxa de erro do *bagging* aplicado aos dados do *Alzheimer* no conjunto de treino.

Na Figura 5.11 é notável que há uma variação na taxa de erro que fica entorno de 19.5%, onde a taxa de erro máximo foi de 21.55% e a menor taxa de erro foi de 19.15%.

Ao aplicar o método *bagging* no conjunto de teste, foram obtidos os seguintes resultados a partir da matriz de confusão 5.19. A taxa de erro encontrado foi de 24.32%.

Tabela 5.19: Matriz de confusão utilizando o método *bagging* no conjunto de teste.

		Predição		
		0	1	
Real	0	70	33	103
	1	38	151	189
		111	181	292

Esse resultado mostra que o modelo não está classificando bem, já que a questão de dar diagnósticos errados para os pacientes pode levar a situações mais sérias. Com isso,

seguiremos com a análise para observar os resultados das medidas de desempenho.

Para medir o desempenho da classificação do modelo utilizando o método *bagging*, foram observados na tabela 5.20 a sensibilidade, especificidade, acurácia, precisão e a Medida-F (*F-Mesure*).

Tabela 5.20: Valores de sensibilidade e especificidade da base de *Alzheimer* com a aplicação do método *bagging* para o conjunto de teste.

Corte	**Sensibilidade**	**Especificidade**	**Acurácia**	**Precisão**	**Medida-F**
0.1	0.9788	0.3301	0.7500	0.7283	0.8352
0.2	0.9630	0.3689	0.7534	0.7368	0.8349
0.3	0.9101	0.4369	0.7432	0.7478	0.8210
0.4	0.8783	0.5146	0.7500	0.7685	0.8198
0.5	0.7989	0.6796	0.7568	0.8207	0.8097
0.6	0.7460	0.7767	0.7568	0.8598	0.7989
0.7	0.7302	0.8252	0.7637	0.8846	0.8000
0.8	0.7249	0.8447	0.7671	0.8954	0.8012
0.9	0.6984	0.8447	0.7500	0.8919	0.7834

O ponto de corte usualmente utilizado é 0.5. Para este tipo de amostra, o interessante é observar a sensibilidade e a medida-F que faz a utilização da sensibilidade e precisão, pois o interessante é acertar ao máximo na classificação dos positivos para que não haja erro no diagnóstico da doença. Na tabela 5.20, foi observado que o melhor ponto de corte foi 0.1 para a medida-F. As demais medidas de desempenho foi colocado no trabalho a fim de observar o comportamento. Para ter uma maior certeza, será apresentado abaixo a tabela 5.21 contendo as taxa de erros e a proporção de classificados como 1 (pacientes com *Alzheimer*) nos diferentes pontos de corte.

Pela taxa de erro de classificação que é possível observar na tabela 5.21, o ponto de corte 0.8 apresenta a menor taxa de erro (23.29%).

Além das taxa de erro de má classificação, vamos considerar, também, a proporção que observações que são classificados como pacientes com *Alzheimer* (Doentes = 1) em cada ponto de corte.

Ao observar as proporções, o ponto de corte que mais se assemelha ao da proporção real (aproximadamente 65%) ocorreu no ponto de corte 0.5 com 63.01%.

Abaixo, será mostrado o gráfico da curva *ROC* para observar o comportamento para

Tabela 5.21: Taxa de erro de classificação e proporção de pacientes classificados como 1 para cada ponto de corte.

Corte	taxa de erro	Proporção da predição de 1
0.1	0.2500	0.8699
0.2	0.2466	0.8459
0.3	0.2568	0.7877
0.4	0.2500	0.7397
0.5	0.2432	0.6301
0.6	0.2432	0.5616
0.7	0.2363	0.5342
0.8	0.2329	0.5240
0.9	0.2500	0.5068

o método *bagging*.

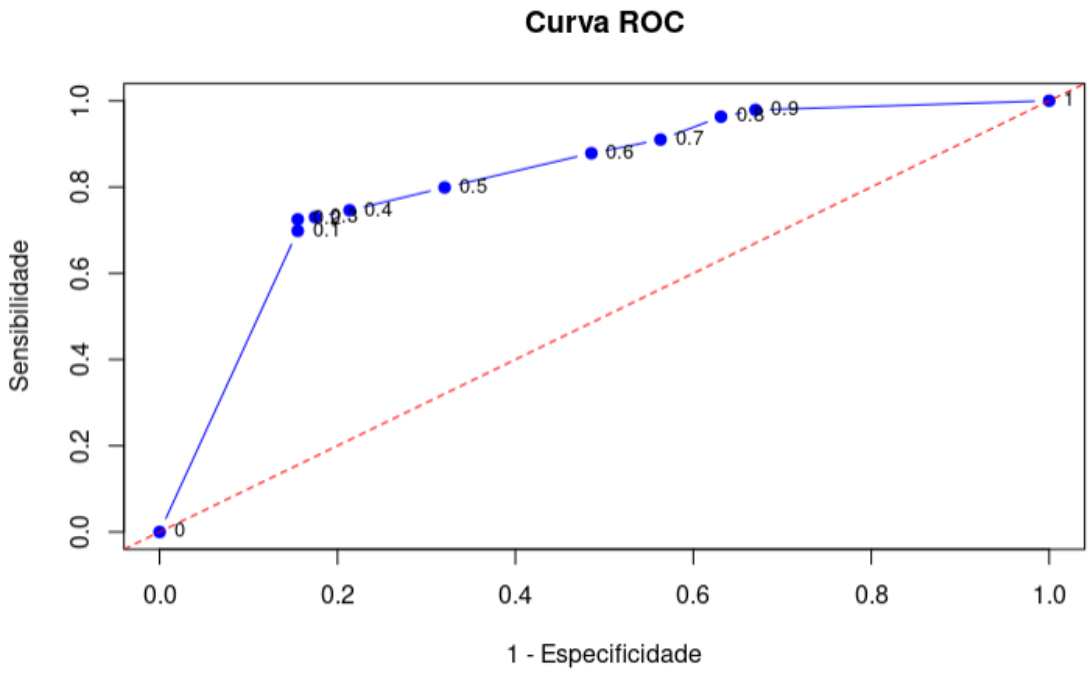

Figura 5.12: Curva *ROC* aplicado no conjunto de teste para o método *bagging*.

A área sob a curva (AUC) foi de 0.8187 e é possível observar uma aglomeração dos pontos na figura 5.12, dificultando um pouco a análise. Nas tabelas 5.20 e 5.21, conseguimos ter uma interpretação melhor dos resultados.

5.2.4 Aplicação método *boosting* $(35\%, 65\%)$

Para a utilização do método *boosting*, foi ajustado alguns hiperparâmetros para evitar que ocorra o *underfitting* ou *overfitting*. Os hiperparâmetros ajustados foram os mesmo da análise anterior. Na figura 5.13 é apresentado a evolução da taxa de erro de classificação que será observado para medir o desempenho do classificador ao utilizar o método *boosting*.

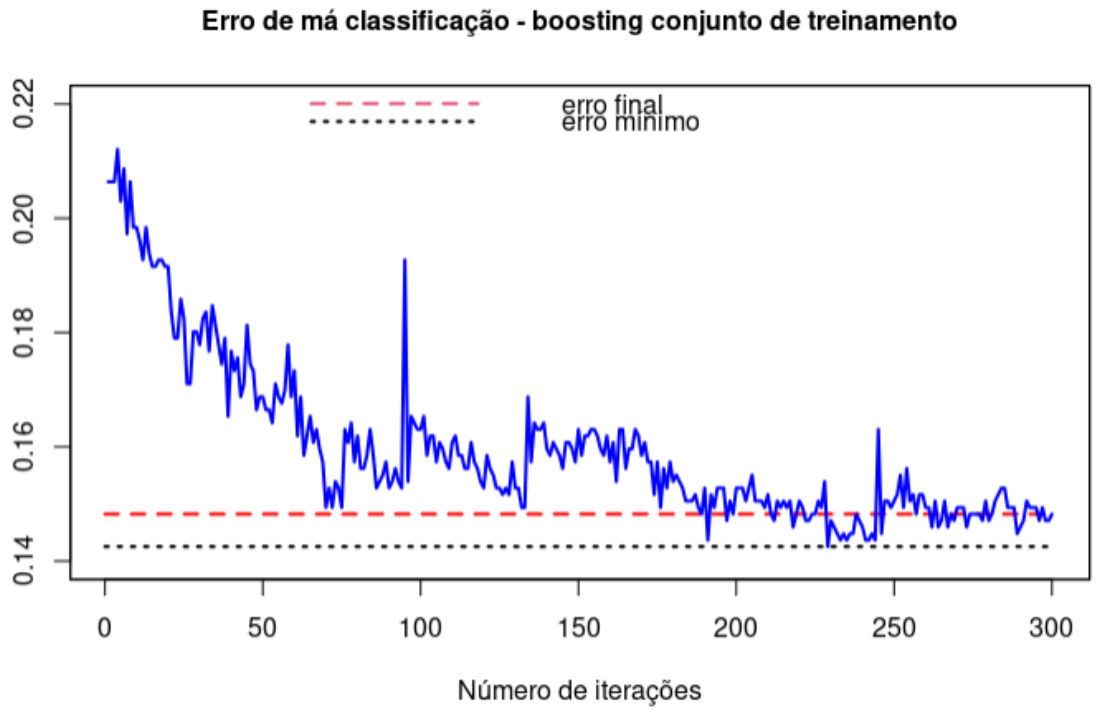

Figura 5.13: Evolução da taxa de erro do *boosting*, aplicado aos dados de treinamento do *Alzheimer*.

Ao observar a figura 5.13, pode-se observar uma melhora na taxa de erro no decorrer das iterações mas os erros acabam ficando próximos ao 15%.

Após o resultado anterior, foi construída a matriz de confusão para um melhor entendimento sobre a taxa erro de classificação de pacientes com *Alzheimer*.

Tabela 5.22: Matriz de confusão utilizando o método *boosting* para a base de dados de treinamento do *Alzheimer*.

		Predição		
		0	1	
Real	0	242	64	306
	1	66	505	571
		308	569	877

No software *R*, foi, novamente, utilizado o comando *predict.boosting()* do pacote *adabag*, onde comparando com a taxa de erro do método *bagging*, pode-se dizer que o *boosting* tem um desempenho muito parecido.

Agora será aplicado o método *boosting* no conjunto de teste. O primeiro resultado que foi observado foi a tabela de confusão 5.23.

Tabela 5.23: Matriz de confusão utilizando o método *boosting* no conjunto de teste.

		Predição		
		0	1	
Real	0	63	40	103
	1	35	154	189
		98	194	292

É possível observar na figura 5.23, uma taxa de erro de classificação de 25.68%. A taxa de erro no conjunto de teste, mostra uma piora nas taxas de erro comparado com as taxas de erro do conjunto de treino.

Agora passaremos para a curva *ROC*. Os pontos de corte utilizados para este estudo foram: $0.1, 0.2, ...0.8, , 0.9$. Dessa maneira, foram observados os seguintes valores de medidas de desempenho:

O ponto de corte usualmente utilizado é 0.5. Novamente, para este tipo de amostra, o interessante é observar a sensibilidade e a medida-F (faz a utilização da sensibilidade) pois é do nosso interesse acertar ao máximo na classificação dos positivos para que não haja erro no diagnóstico da doença.

Tabela 5.24: Tabela dos valores para medir o desempenho da classificação no conjunto de teste aplicando o método *boosting*.

Corte	Sensibilidade	Especificidade	Acurácia	Precisão	Medida-F
0.1	1.0000	0.0000	0.6511	0.6511	0.7887
0.2	1.0000	0.0000	0.6511	0.6511	0.7887
0.3	1.0000	0.0000	0.6511	0.6511	0.7887
0.4	1.0000	0.1340	0.6978	0.6830	0.8117
0.5	0.8844	0.7908	0.8518	0.8875	0.8860
0.6	0.5447	1.0000	0.7035	1.0000	0.7052
0.7	0.1331	1.0000	0.4356	1.0000	0.2349
0.8	0.0035	1.0000	0.3512	1.0000	0.0070
0.9	0.0000	1.0000	0.3489	0.0000	0.0000

Na tabela 5.24, foi observado que o melhor ponto de corte foi o 0.5 para a medida-F. As demais medidas de desempenho foi colocado no trabalho a fim de observar o comportamento. Para ter uma maior certeza, será apresentado abaixo a tabela 5.25 contendo as taxa de erros e a proporção de classificados como 1 (pacientes com *Alzheimer*) nos diferentes pontos de corte.

Tabela 5.25: Taxa de erro de classificação para cada ponto de corte utilizando o método *boosting* e a proporção de 1's no conjunto de teste.

Corte	taxa de erro	proporção de 1
0.1	0.3489	0.6511
0.2	0.3489	0.6511
0.3	0.3489	0.6511
0.4	0.3022	0.6511
0.5	0.1482	0.5758
0.6	0.2965	0.3546
0.7	0.5644	0.0867
0.8	0.6488	0.0023
0.9	0.6511	0.0000

Nota-se na tabela 5.25 que, realmente o ponto de corte 0.5 tem a melhor taxa de classificação com uma taxa de erro de classificação de 14.82%.

Dessa forma, utilizando o *boosting* em um conjunto de teste com classificações de aproximadamente 65% de pacientes com *Alzheimer* (classificados como 1) a 35% de pacientes sem *Alzheimer* (classificados como 0), os melhores pontos de corte são 0.1, 0.2, 0.3 e 0.4.

Na figura 5.14, foram adicionadas os pontos de corte 0 e 1 para que a curva ficasse completa e foi observado uma área abaixo da curva (AUC) de aproximadamente 0.9357.

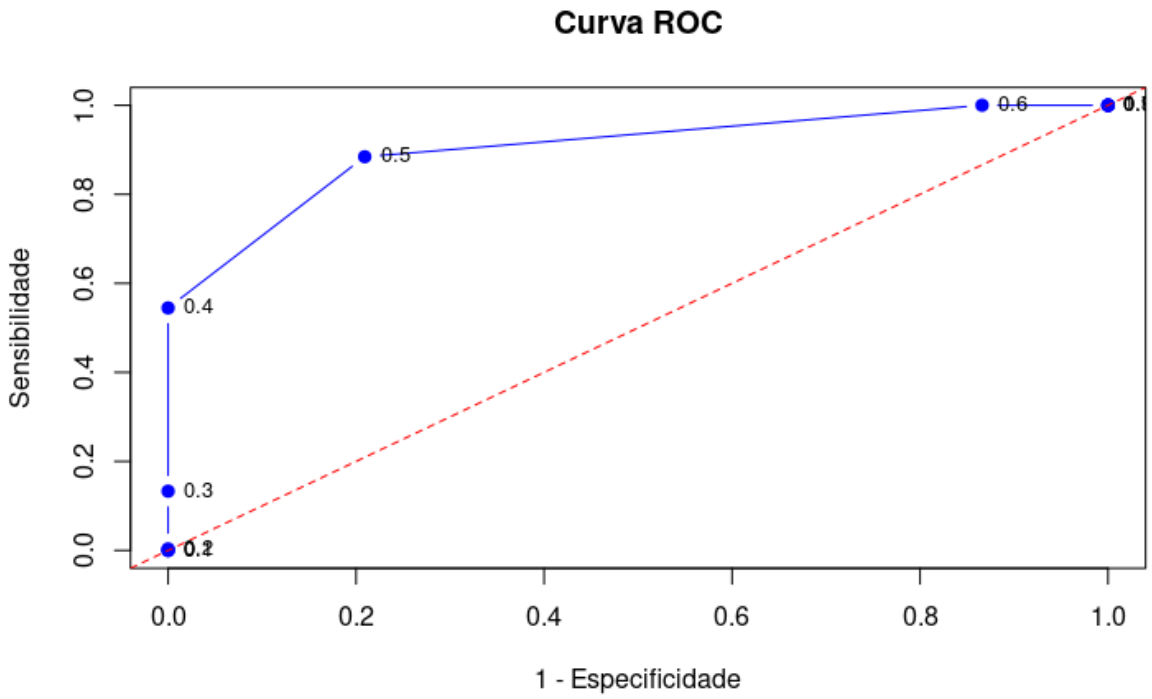

Figura 5.14: Curva *ROC* para o método *boosting* no conjunto de teste.

Dessa maneira, consegue-se notar que visualmente o melhor ponto de corte é 0.5. Porém, ao se obter tal resultado, é necessário considerar, também, a taxa de erro de classificação.

Mesmo com o desempenho pior do conjunto de teste, há um erro de classificação de 25.68% para pacientes com *Alzheimer*. Vale ressaltar, novamente que, a proporção da amostra desse estudo foi de aproximadamente 65% de pacientes com *Alzheimer* (Diagnóstico = 1) e 35% de pacientes sem *Alzheimer* (Diagnóstico = 0).

5.2.5 Aplicação do método *bagging* $(80\%, 20\%)$

Os hiperparâmetros ajustados foram os mesmo da análises anteriores, mudando somente o número da amostra que é de 1736 observações. Dessa maneira, será apresentado abaixo a tabela de confundimento para ter uma ideia de como está o desempenho do nosso classificador.

Tabela 5.26: Matriz de confusão utilizando o método *bagging*.

		Predição		
		0	1	
Real	0	969	72	1041
	1	116	145	261
		1085	217	1302

A taxa de erro de classificação observado na matriz de confusão (Tabela 5.26), foi de 14.43%. A taxa de erro não aparenta uma classificação boa dada a importância da classificação mais assertiva por se tratar de uma amostra da área da saúde. Após o resultado anterior, será mostrado na Figura 5.15 com a evolução das taxas de erro de classificação.

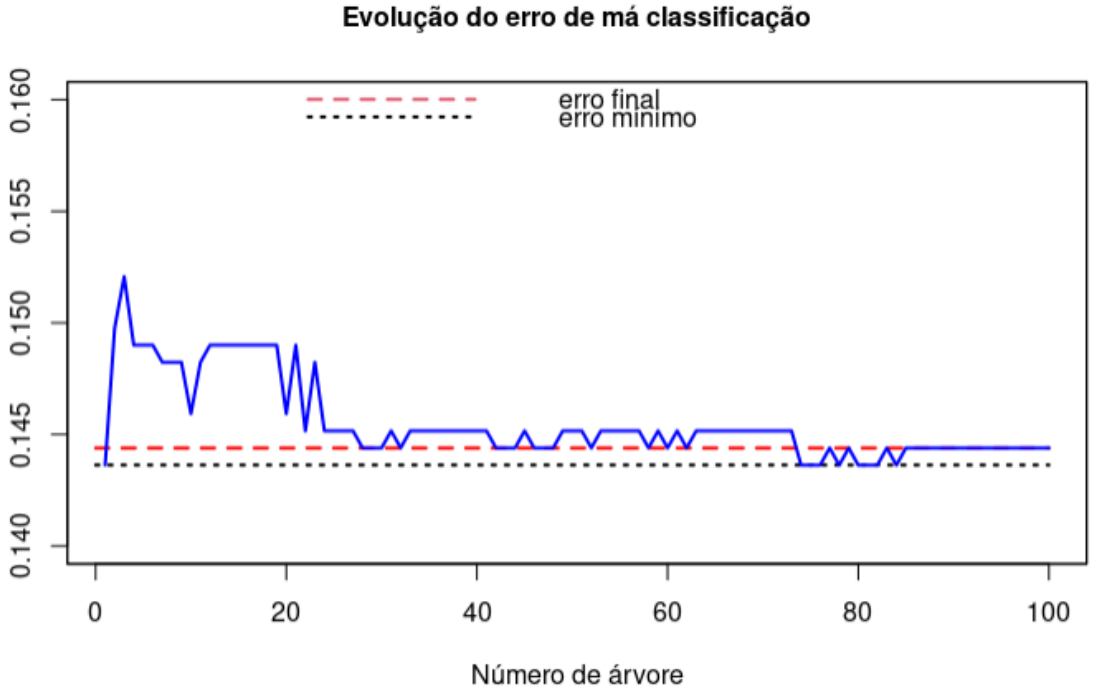

Figura 5.15: Evolução da taxa de erro do *bagging* aplicado aos dados do *Alzheimer* no conjunto de treino.

Na Figura 5.15 é notável que há uma variação na taxa de erro que fica entorno de 14.5%, onde a taxa de erro máximo foi de 15.2% e a menor taxa de erro foi de 14.36%.

Ao aplicar o método *bagging* no conjunto de teste, foram obtidos os seguintes resultados a partir da matriz de confusão 5.27. A taxa de erro encontrado foi de 15.43%.

Esse resultado mostra que o modelo não está classificando bem, já que a questão de

Tabela 5.27: Matriz de confusão utilizando o método *bagging* no conjunto de teste.

		Predição		
		0	1	
Real	0	323	25	348
	1	42	44	86
		365	69	434

dar diagnósticos errados para os pacientes pode levar a situações mais sérias. Com isso, seguiremos com a análise para observar os resultados das medidas de desempenho.

Para medir o desempenho da classificação do modelo utilizando o método *bagging*, foram observados na tabela 5.28 a sensibilidade, especificidade, acurácia, precisão e a Medida-F (*F-Mesure*).

Tabela 5.28: Valores de sensibilidade e especificidade da base de *Alzheimer* com a aplicação do método *bagging* para o conjunto de teste.

Corte	**Sensibilidade**	**Especificidade**	**Acurácia**	**Precisão**	**Medida-F**
0.1	0.7558	0.8793	0.8548	0.6075	0.6736
0.2	0.7093	0.8937	0.8571	0.6224	0.6630
0.3	0.5698	0.9167	0.8479	0.6282	0.5976
0.4	0.5465	0.9195	0.8456	0.6267	0.5839
0.5	0.5116	0.9282	0.8456	0.6377	0.5677
0.6	0.5116	0.9339	0.8502	0.6567	0.5752
0.7	0.4884	0.9368	0.8479	0.6562	0.5600
0.8	0.4535	0.9397	0.8433	0.6500	0.5342
0.9	0.3140	0.9569	0.8295	0.6429	0.4219

O ponto de corte usualmente utilizado é 0.5. Para este tipo de amostra, o interessante é observar a sensibilidade e a medida-F que faz a utilização da sensibilidade e precisão, pois o interessante é acertar ao máximo na classificação dos positivos para que não haja erro no diagnóstico da doença. Na tabela 5.28, foi observado que o melhor ponto de corte foi 0.1 para a medida-F. As demais medidas de desempenho foi colocado no trabalho a fim de observar o comportamento. Para ter uma maior certeza, será apresentado abaixo a tabela 5.29 contendo as taxa de erros e a proporção de classificados como 1 (pacientes com *Alzheimer*) nos diferentes pontos de corte.

Tabela 5.29: Taxa de erro de classificação e proporção de pacientes classificados como 1 para cada ponto de corte.

Corte	taxa de erro	Proporção da predição de 1
0.1	0.1452	0.2465
0.2	0.1429	0.2258
0.3	0.1521	0.1797
0.4	0.1544	0.1728
0.5	0.1544	0.1590
0.6	0.1498	0.1544
0.7	0.1521	0.1475
0.8	0.1567	0.1382
0.9	0.1705	0.0968

Pela taxa de erro de classificação que é possível observar na tabela 5.29, o ponto de corte 0.2 apresenta a menor taxa de erro (14.29%).

Além das taxa de erro de má classificação, vamos considerar, também, a proporção que observações que são classificados como pacientes com *Alzheimer* (Doentes = 1) em cada ponto de corte.

Ao observar as proporções, o ponto de corte que mais se assemelha ao da proporção real (aproximadamente 20%) ocorreu no ponto de corte 0.2 com 22.58%.

Abaixo, será mostrado o gráfico da curva *ROC* para observar o comportamento para o método *bagging*.

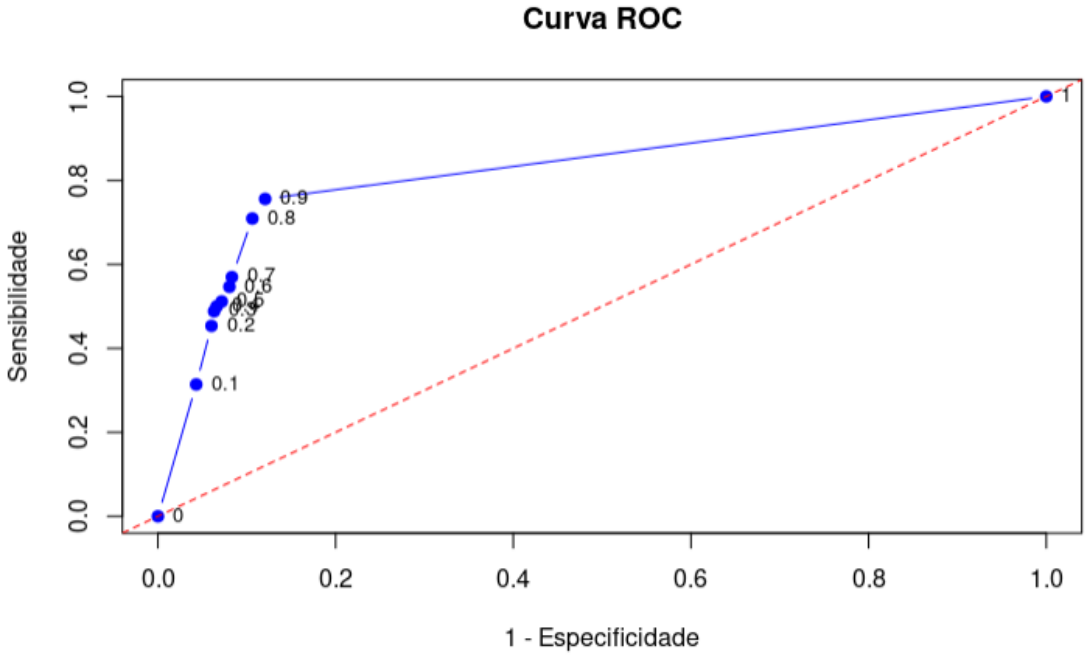

Figura 5.16: Curva *ROC* aplicado no conjunto de teste para o método *bagging*.

A área sob a curva (AUC) foi de 0.8333 e é possível observar uma aglomeração dos pontos na figura 5.16, dificultando um pouco a análise mas os pontos que mais se aproximam do canto superior esquerdo é o 0.8 ou 0.9. Nas tabelas 5.28 e 5.29, conseguimos ter uma interpretação melhor dos resultados.

5.2.6 Aplicação método *boosting* $(80\%, 20\%)$

Para a utilização do método *boosting*, foi ajustado alguns hiperparâmetros para evitar que ocorra o *underfitting* ou *overfitting*. Os hiperparâmetros ajustados foram os mesmo da análise anterior. Na figura 5.17 é apresentado a evolução da taxa de erro de classificação que será observado para medir o desempenho do classificador ao utilizar o método *boosting*.

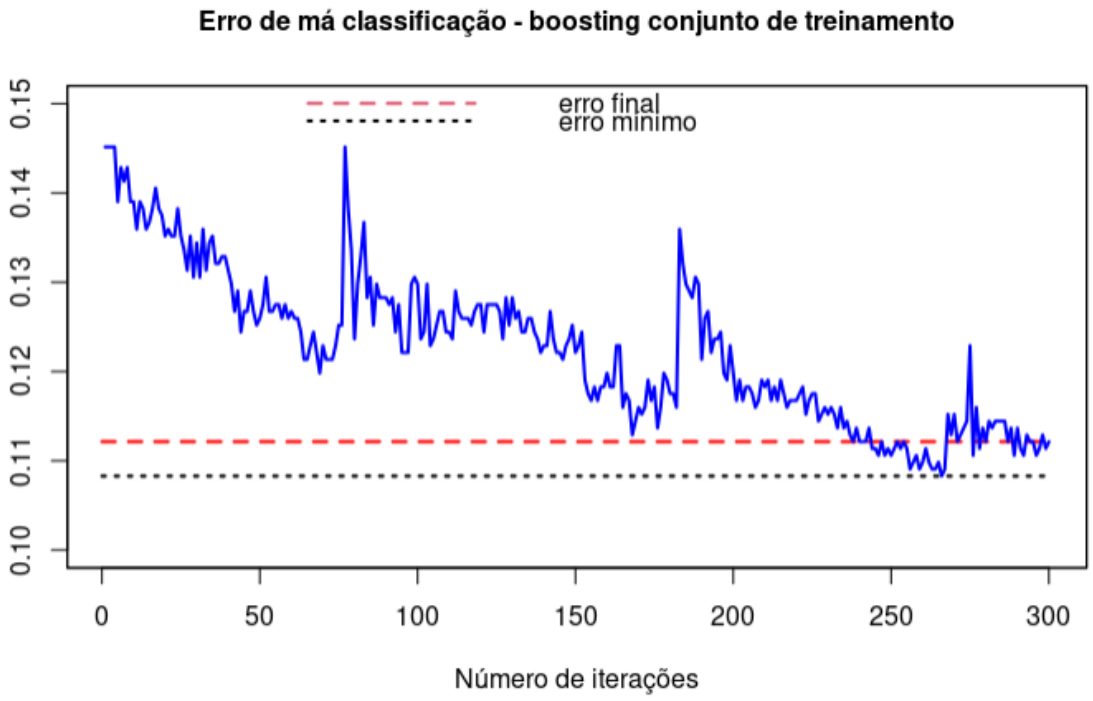

Figura 5.17: Evolução da taxa de erro do *boosting*, aplicado aos dados de treinamento do *Alzheimer*.

Ao observar a figura 5.17, pode-se observar uma melhora na taxa de erro no decorrer das iterações mas os erros acabam ficando próximos ao 10.82%.

Após o resultado anterior, foi construída a matriz de confusão 5.30 para um melhor entendimento sobre a taxa erro de classificação de pacientes com *Alzheimer*.

Tabela 5.30: Matriz de confusão utilizando o método *boosting* para a base de dados de treinamento do *Alzheimer*.

		Predição		
		0	1	
Real	0	988	53	1041
	1	93	168	261
		1081	221	1302

No software *R*, foi, novamente, utilizado o comando *predict.boosting()* do pacote *adabag*, onde comparando com a taxa de erro do método *bagging*, pode-se dizer que o *boosting* tem um desempenho muito parecido.

Agora será aplicado o método *boosting* no conjunto de teste. O primeiro resultado que foi observado foi a tabela de confusão 5.31.

Tabela 5.31: Matriz de confusão utilizando o método *boosting* no conjunto de teste.

		Predição		
		0	1	
Real	0	321	27	348
	1	38	48	86
		359	86	434

É possível observar na tabela 5.31, uma taxa de erro de classificação de 14.97%. A taxa de erro no conjunto de teste, mostra uma piora nas taxas de erro comparado com as taxas de erro do conjunto de treino.

Agora passaremos para a curva *ROC*. Os pontos de corte utilizados para este estudo foram: $0.1, 0.2, ...0.8, , 0.9$. Dessa maneira, foram observados os seguintes valores de medidas de desempenho:

Tabela 5.32: Tabela dos valores para medir o desempenho da classificação no conjunto de teste aplicando o método *boosting*.

Corte	Sensibilidade	Especificidade	Acurácia	Precisão	Medida-F
0.1	1.0000	0.0000	0.2005	0.2005	0.3340
0.2	1.0000	0.0000	0.2005	0.2005	0.3340
0.3	1.0000	0.0682	0.2550	0.2120	0.3499
0.4	0.9732	0.6609	0.7235	0.4185	0.5853
0.5	0.6437	0.9491	0.8879	0.7602	0.6971
0.6	0.0077	1.0000	0.8011	1.0000	0.0152
0.7	0.0000	1.0000	0.7995	0.0000	0.0000
0.8	0.0000	1.0000	0.7995	0.0000	0.0000
0.9	0.0000	1.0000	0.7995	0.0000	0.0000

O ponto de corte usualmente utilizado é 0.5. Novamente, para este tipo de amostra, o interessante é observar a sensibilidade e a medida-F (faz a utilização da sensibilidade) pois é do nosso interesse acertar ao máximo na classificação dos positivos para que não haja erro no diagnóstico da doença.

Na tabela 5.32, foi observado que o melhor ponto de corte foi o 0.5 para a medida-F. As demais medidas de desempenho foi colocado no trabalho a fim de observar o comportamento. Para ter uma maior certeza, será apresentado abaixo a tabela 5.33 contendo as taxa de erros e a proporção de classificados como 1 (pacientes com *Alzheimer*) nos diferentes pontos de corte.

Tabela 5.33: Taxa de erro de classificação para cada ponto de corte utilizando o método *boosting* e a proporção de 1's no conjunto de teste.

Corte	taxa de erro	proporção de 1
0.1	0.7995	0.2005
0.2	0.7995	0.2005
0.3	0.7450	0.2005
0.4	0.2765	0.1951
0.5	0.1121	0.1290
0.6	0.1989	0.0015
0.7	0.2005	0.0000
0.8	0.2005	0.0000
0.9	0.2005	0.0000

Nota-se na tabela 5.33 que, realmente o ponto de corte 0.5 tem a melhor taxa de classificação com uma taxa de erro de classificação de 11.21%.

Dessa forma, utilizando o *boosting* em um conjunto de teste com classificações de aproximadamente 20% de pacientes com *Alzheimer* (classificados como 1) a 80% de pa-

cientes sem *Alzheimer* (classificados como 0), os melhores pontos de corte são 0.1, 0.2 e 0.3.

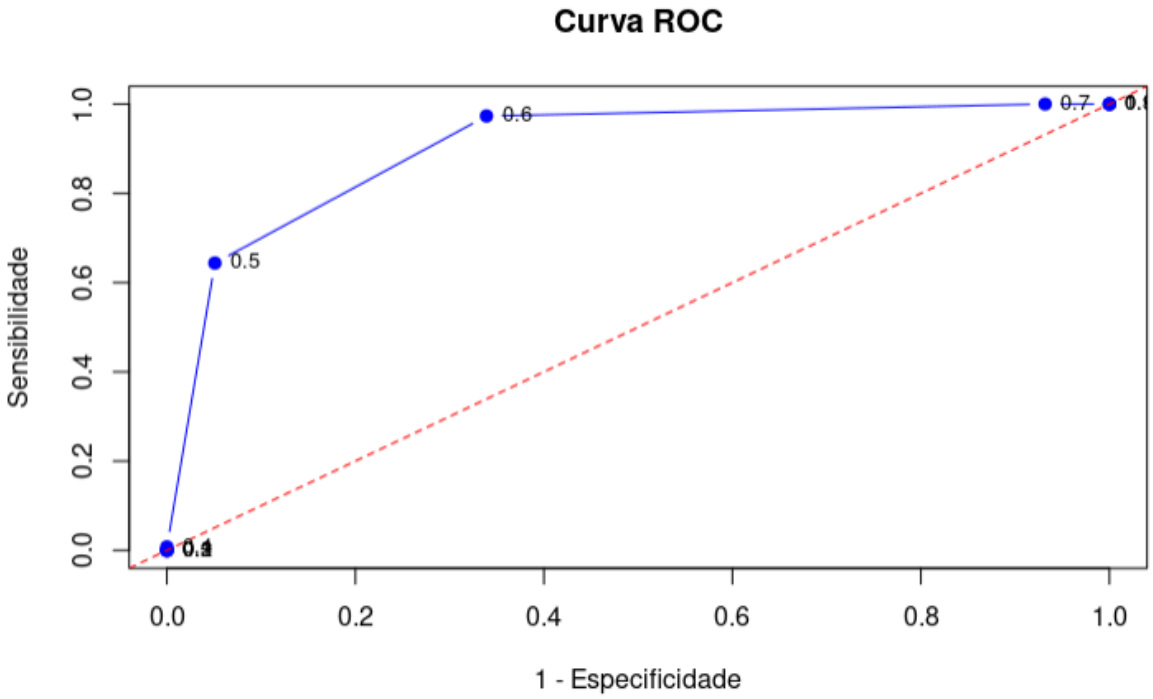

Figura 5.18: Curva *ROC* para o método *boosting* no conjunto de teste.

Na figura 5.18, foram adicionadas os pontos de corte 0 e 1 para que a curva ficasse completa e foi observado uma área abaixo da curva (AUC) de aproximadamente 0.9325. Dessa maneira, consegue-se notar que visualmente o melhor ponto de corte é 0.5 pois é o ponto que mais se aproxima do canto esquerdo superior. Porém, ao se obter tal resultado, é necessário considerar, também, a taxa de erro de classificação.

Mesmo com o desempenho pior do conjunto de teste, há um erro de classificação de 14.97% para pacientes com *Alzheimer*. Vale ressaltar, novamente que, a proporção da amostra desse estudo foi de aproximadamente 20% de pacientes com *Alzheimer* (Diagnóstico = 1) e 80% de pacientes sem *Alzheimer* (Diagnóstico = 0).

5.2.7 Aplicação do método *bagging* $(20\%, 80\%)$

Os hiperparâmetros ajustados foram os mesmo da análises anteriores, mudando somente o número da amostra que é de 950 observações. Dessa maneira, será apresentado abaixo a tabela de confundimento para ter uma ideia de como está o desempenho do nosso

classificador.

Tabela 5.34: Matriz de confusão utilizando o método *bagging*.

		Predição		
		0	1	
Real	0	57	88	145
	1	19	548	567
		76	636	712

A taxa de erro de classificação observado na matriz de confusão (Tabela 5.34), foi de 15.02%. A taxa de erro não aparenta uma classificação boa dada a importância da classificação mais assertiva por se tratar de uma amostra da área da saúde. Após o resultado anterior, será mostrado na Figura 5.19 com a evolução das taxas de erro de classificação.

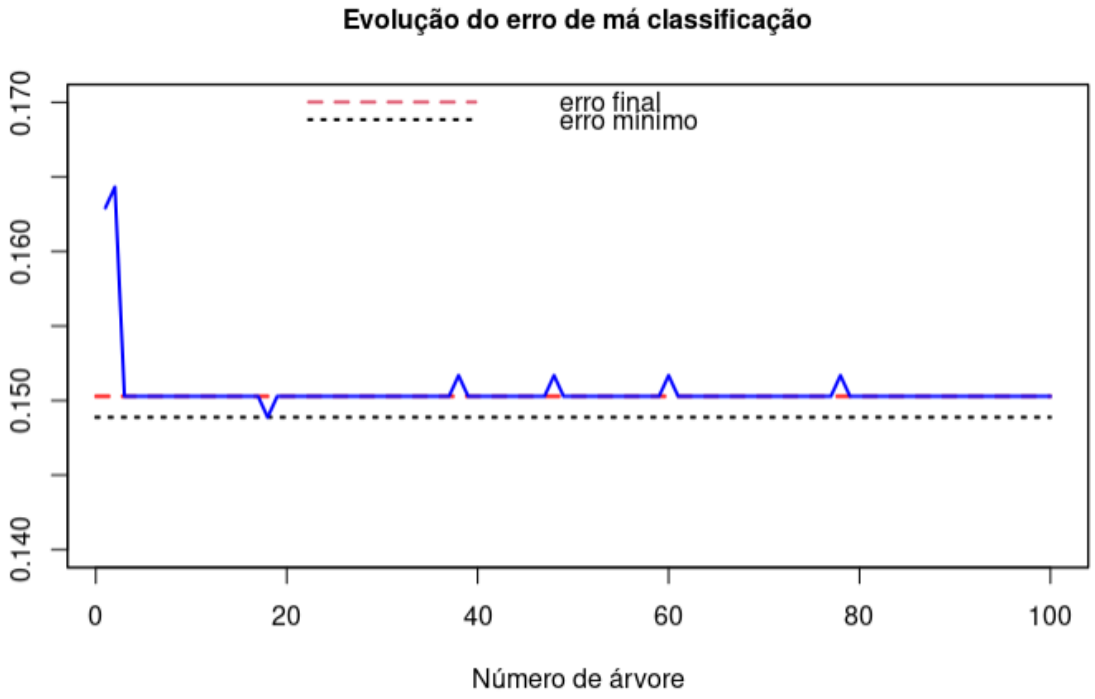

Figura 5.19: Evolução da taxa de erro do *bagging* aplicado aos dados do *Alzheimer* no conjunto de treino.

Na Figura 5.11 é notável que há uma variação na taxa de erro que fica entorno de 15%, onde a taxa de erro máximo foi de 16.43% e a menor taxa de erro foi de 14.88%.

Ao aplicar o método *bagging* no conjunto de teste, foram obtidos os seguintes resultados a partir da matriz de confusão 5.35. A taxa de erro encontrado foi de 15.43%.

Tabela 5.35: Matriz de confusão utilizando o método *bagging* no conjunto de teste.

		Predição		
		0	1	
Real	0	15	30	45
	1	4	189	193
		19	219	238

Esse resultado mostra que o modelo não está classificando bem, já que a questão de dar diagnósticos errados para os pacientes pode levar a situações mais sérias. Com isso, seguiremos com a análise para observar os resultados das medidas de desempenho.

Para medir o desempenho da classificação do modelo utilizando o método *bagging*, foram observados na tabela 5.36 a sensibilidade, especificidade, acurácia, precisão e a Medida-F (*F-Mesure*).

Tabela 5.36: Valores de sensibilidade e especificidade da base de *Alzheimer* com a aplicação do método *bagging* para o conjunto de teste.

Corte	**Sensibilidade**	**Especificidade**	**Acurácia**	**Precisão**	**Medida-F**
0.1	0.9845	0.3111	0.8571	0.8597	0.9179
0.2	0.9793	0.3333	0.8571	0.8630	0.9175
0.3	0.9793	0.3333	0.8571	0.8630	0.9175
0.4	0.9793	0.3333	0.8571	0.8630	0.9175
0.5	0.9793	0.3333	0.8571	0.8630	0.9175
0.6	0.9793	0.3556	0.8613	0.8670	0.9197
0.7	0.9741	0.3556	0.8571	0.8664	0.9171
0.8	0.9534	0.4000	0.8487	0.8720	0.9109
0.9	0.8912	0.5556	0.8277	0.8958	0.8935

O ponto de corte usualmente utilizado é 0.5. Sempre reforçando que para este tipo de amostra, o interessante é observar a sensibilidade e a medida-F (que faz a utilização da sensibilidade e precisão), pois o interessante é acertar ao máximo na classificação dos positivos para que não haja erro no diagnóstico da doença. Na tabela 5.36, foi observado que o melhor ponto de corte foi 0.6 para a medida-F. As demais medidas de desempenho foi colocado no trabalho a fim de observar o comportamento. Para ter uma maior certeza, será apresentado abaixo a tabela 5.37 contendo as taxa de erros e a proporção de classificados como 1 (pacientes com *Alzheimer*) nos diferentes pontos de corte.

Tabela 5.37: Taxa de erro de classificação e proporção de pacientes classificados como 1 para cada ponto de corte.

Corte	taxa de erro	Proporção da predição de 1
0.1	0.1429	0.9286
0.2	0.1429	0.9202
0.3	0.1429	0.9202
0.4	0.1429	0.9202
0.5	0.1429	0.9202
0.6	0.1387	0.9160
0.7	0.1429	0.9118
0.8	0.1513	0.8866
0.9	0.1723	0.8067

Pela taxa de erro de classificação que é possível observar na tabela 5.29, o ponto de corte 0.6 apresenta a menor taxa de erro (13.87%).

Além das taxa de erro de má classificação, vamos considerar, também, a proporção que observações que são classificados como pacientes com *Alzheimer* (Doentes = 1) em cada ponto de corte.

Ao observar as proporções, o ponto de corte que mais se assemelha ao da proporção real (aproximadamente 80%) ocorreu no ponto de corte 0.9 com 80.67%.

Abaixo, será mostrado o gráfico da curva *ROC* para observar o comportamento para o método *bagging*.

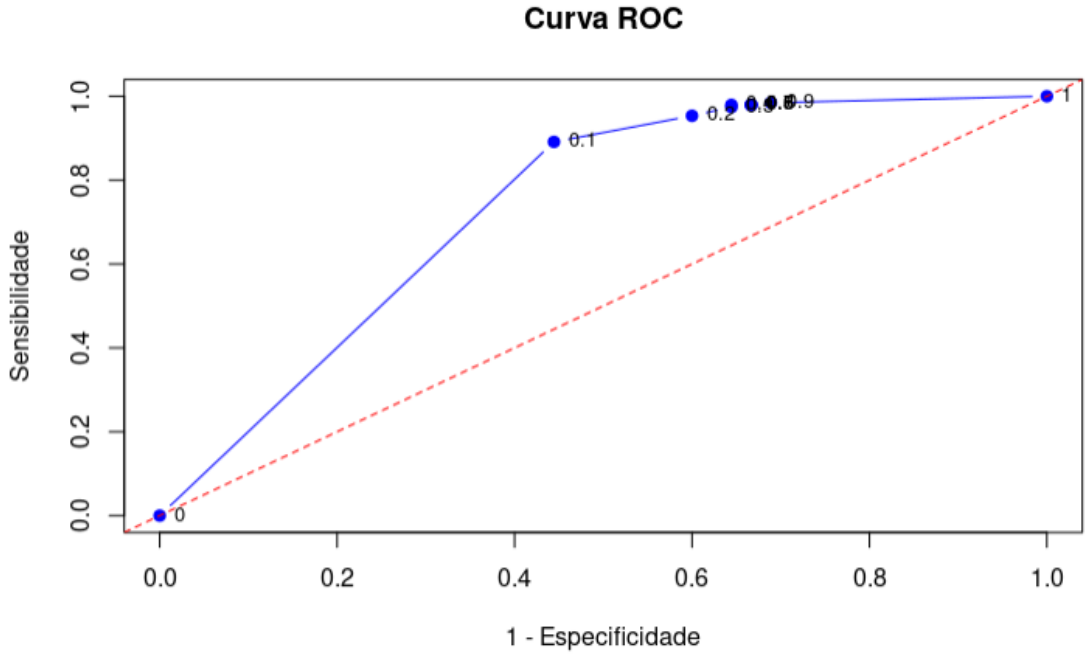

Figura 5.20: Curva *ROC* aplicado no conjunto de teste para o método *bagging*.

A área sob a curva (AUC) foi de 0.8458 e é possível observar uma aglomeração dos pontos na figura 5.20, dificultando um pouco a análise mas os pontos que mais se aproximam do canto superior esquerdo é o 0.1. Nas tabelas 5.36 e 5.37, conseguimos ter uma interpretação melhor dos resultados.

5.2.8 Aplicação método *boosting* $(20\%, 80\%)$

Para a utilização do método *boosting*, foi ajustado alguns hiperparâmetros para evitar que ocorra o *underfitting* ou *overfitting*. Os hiperparâmetros ajustados foram os mesmo da análise anterior. Na figura 5.21 é apresentado a evolução da taxa de erro de classificação que será observado para medir o desempenho do classificador ao utilizar o método *boosting*.

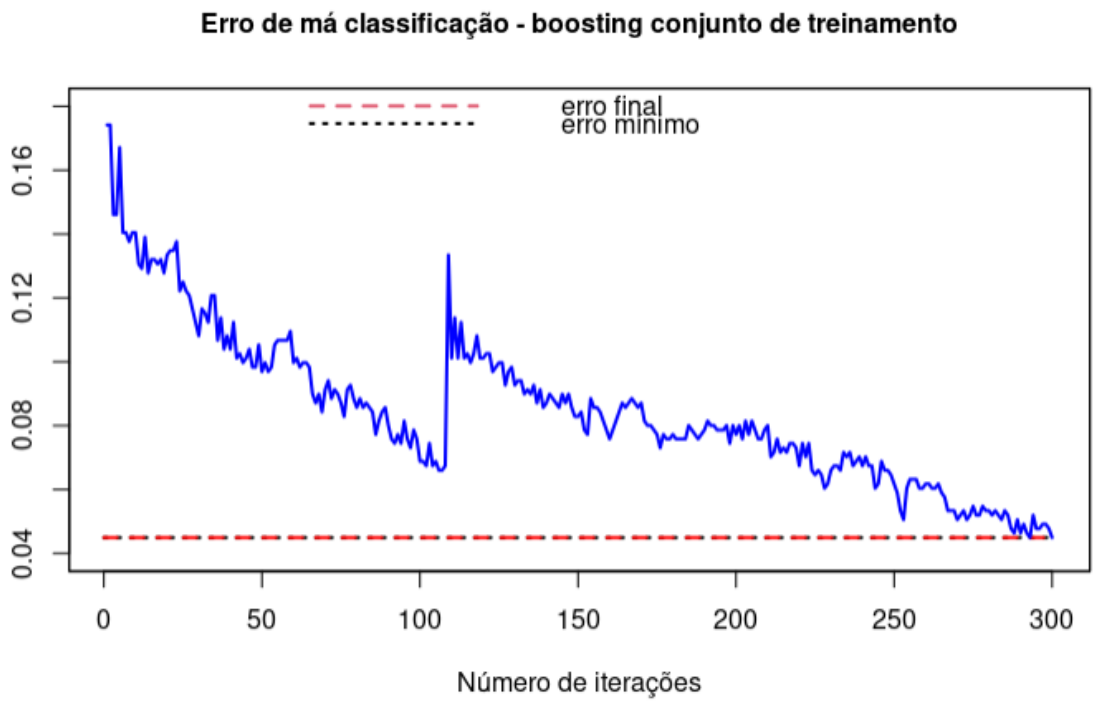

Figura 5.21: Evolução da taxa de erro do *boosting*, aplicado aos dados de treinamento do *Alzheimer*.

Ao observar a figura 5.21, pode-se observar uma melhora na taxa de erro no decorrer das iterações mas os erros acabam ficando próximos ao 4.49%.

Após o resultado anterior, foi construída a matriz de confusão 5.38 para um melhor entendimento sobre a taxa erro de classificação de pacientes com *Alzheimer*.

Tabela 5.38: Matriz de confusão utilizando o método *boosting* para a base de dados de treinamento do *Alzheimer*.

		Predição		
		0	1	
Real	0	120	25	145
	1	7	560	567
		127	585	712

No software *R*, foi, novamente, utilizado o comando *predict.boosting()* do pacote *adabag*, onde comparando com a taxa de erro do método *bagging*, pode-se dizer que o *boosting* tem um desempenho muito parecido.

Agora será aplicado o método *boosting* no conjunto de teste. O primeiro resultado que foi observado foi a tabela de confusão 5.39.

Tabela 5.39: Matriz de confusão utilizando o método *boosting* no conjunto de teste.

		Predição		
		0	1	
Real	0	18	27	45
	1	17	176	193
		35	203	238

É possível observar na tabela 5.39, uma taxa de erro de classificação de 18.48%. A taxa de erro no conjunto de teste, mostra uma piora nas taxas de erro comparado com as taxas de erro do conjunto de treino.

Agora passaremos para a curva *ROC*. Os pontos de corte utilizados para este estudo foram: $0.1, 0.2, ...0.8, , 0.9$. Dessa maneira, foram observados os seguintes valores de medidas de desempenho:

O ponto de corte usualmente utilizado é 0.5. Novamente, para este tipo de amostra, o interessante é observar a sensibilidade e a medida-F (faz a utilização da sensibilidade) pois é do nosso interesse acertar ao máximo na classificação dos positivos para que não haja erro no diagnóstico da doença.

Tabela 5.40: Tabela dos valores para medir o desempenho da classificação no conjunto de teste aplicando o método *boosting*.

Corte	**Sensibilidade**	**Especificidade**	**Acurácia**	**Precisão**	**Medida-F**
0.1	1.0000	0.0000	0.7963	0.7963	0.8866
0.2	1.0000	0.0000	0.7963	0.7963	0.8866
0.3	1.0000	0.0000	0.7963	0.7963	0.8866
0.4	1.0000	0.0000	0.7963	0.7963	0.8866
0.5	0.9877	0.8276	0.9551	0.9573	0.9722
0.6	0.4568	1.0000	0.5674	1.0000	0.6271
0.7	0.0247	1.0000	0.2233	1.0000	0.0482
0.8	0.0000	1.0000	0.2037	0.0000	0.0000
0.9	0.0000	1.0000	0.2037	0.0000	0.0000

Na tabela 5.40, foi observado que o melhor ponto de corte foi o 0.5 para a medida-F. As demais medidas de desempenho foi colocado no trabalho a fim de observar o comportamento. Para ter uma maior certeza, será apresentado abaixo a tabela 5.41 contendo as taxa de erros e a proporção de classificados como 1 (pacientes com *Alzheimer*) nos diferentes pontos de corte.

Tabela 5.41: Taxa de erro de classificação para cada ponto de corte utilizando o método *boosting* e a proporção de 1's no conjunto de teste.

Corte	**taxa de erro**	**proporção de 1**
0.1	0.2037	0.7963
0.2	0.2037	0.7963
0.3	0.2037	0.7963
0.4	0.2037	0.7963
0.5	0.0449	0.7865
0.6	0.4326	0.3638
0.7	0.7767	0.0197
0.8	0.7963	0.0000
0.9	0.7963	0.0000

Nota-se na tabela 5.41 que, realmente o ponto de corte 0.5 tem a melhor taxa de classificação com uma taxa de erro de classificação de 4.49%.

Dessa forma, utilizando o *boosting* em um conjunto de teste com classificações de aproximadamente 80% de pacientes com *Alzheimer* (classificados como 1) a 20% de pacientes sem *Alzheimer* (classificados como 0), os melhores pontos de corte são 0.1, 0.2, 0.3 e 0.4.

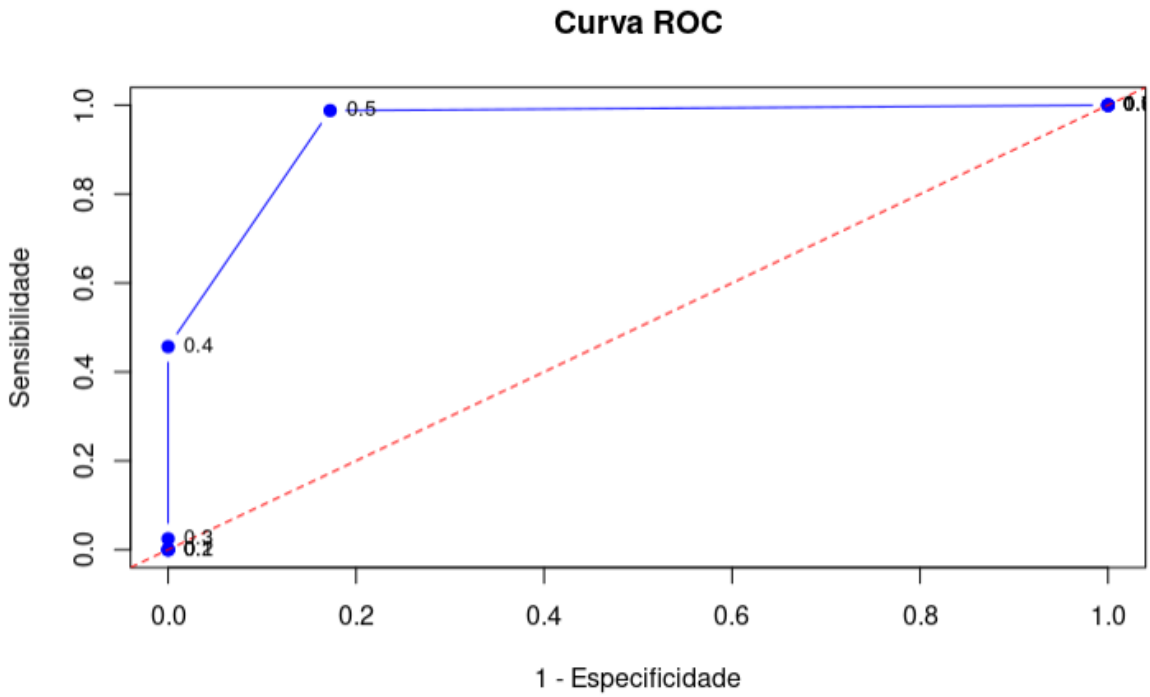

Figura 5.22: Curva *ROC* para o método *boosting* no conjunto de teste.

Na figura 5.22, foram adicionadas os pontos de corte 0 e 1 para que a curva ficasse completa e foi observado uma área abaixo da curva (AUC) de aproximadamente 0.9877. Dessa maneira, consegue-se notar que visualmente o melhor ponto de corte é 0.5 pois é o ponto que mais se aproxima do canto esquerdo superior. Porém, ao se obter tal resultado, é necessário considerar, também, a taxa de erro de classificação.

Mesmo com o desempenho pior do conjunto de teste, há um erro de classificação de 18.48% para pacientes com *Alzheimer*. Vale ressaltar, novamente que, a proporção da amostra desse estudo foi de aproximadamente 80% de pacientes com *Alzheimer* (Diagnóstico = 1) e 20% de pacientes sem *Alzheimer* (Diagnóstico = 0).

Capítulo 6

Considerações Finais

O objetivo deste trabalho foi medir o desempenho do classificador, utilizando diversas métricas de avaliação, a partir dos métodos *bagging* e *boosting* em diferentes pontos de corte, que, na prática, definem novas classificações. Além disso, buscou-se analisar se o ponto de corte 0.5 realmente se apresenta como a melhor escolha para a classificação.

A metodologia empregada consistiu na aplicação dos métodos de *bagging* e *boosting*, ambos baseados em árvores de decisão. As bases de dados previamente separados em conjuntos de treino e teste, assim como o modelo que tiveram as variáveis e seus hiperparâmetros selecionados antes da aplicação dos métodos de *bagging* e *boosting*. Para a avaliação do desempenho dos classificadores, foram utilizadas métricas baseadas na matriz de confusão: acurácia, sensibilidade, especificidade e a área sob a curva *ROC (AUC)* .

Os resultados indicaram que ambos os métodos, *bagging* e *boosting*, melhoram o desempenho da classificação em comparação a classificação que utiliza somente a árvore de decisão. O *boosting* apresentou menor taxa de erro em alguns cenários comparado com o *bagging*, especialmente em bases de dados menores, onde sua capacidade de aprendizado iterativo se mostrou eficaz ao atribuir pesos diferenciados às observações mal classificadas. No entanto, observou-se que o *boosting* pode ser mais suscetível ao *overfitting*, impactando a capacidade de generalização do modelo para dados bem desbalanceados. Por outro lado, o *bagging* demonstrou ser mais estável e menos sensível a variações nos dados, tornando-se uma boa alternativa para bases menores.

Em relação ao ponto de corte ideal, os testes mostraram que, na maioria dos casos, o valor padrão de 0.5 foi de fato a melhor escolha, equilibrando sensibilidade, especificidade e também a medida-F. Contudo, em alguns contextos específicos, valores alternativos poderiam ser vantajosos dependendo da necessidade de reduzir falsos positivos ou fal-

sos negativos, especialmente em aplicações como em diagnóstico médico (diagnóstico de *Alzheimer*), onde a minimização de erros tem um impacto crítico.

Dessa forma, o estudo reforça a importância da escolha criteriosa do método de classificação e do ponto de corte adequado para cada aplicação. Além disso, evidencia a relevância de utilizar múltiplas métricas para avaliar o desempenho de um classificador, garantindo que a análise não seja enviesada por uma única medida de desempenho.

Para os estudos futuros, pode ser explorado algumas técnicas adicionais para otimizar o desempenho do *boosting*, bem como a aplicação de outros métodos para aprimorar ainda mais a eficácia dos classificadores em diferentes contextos. Seria interessante trazer o mesmo estudo para contextos diferentes a fim de entender o comportamento de cada método de classificação.

Referências Bibliográficas

Alfaro, E., Gamez, M. e Garcia, N. (2013). adabag: An r package for classification with boosting and bagging. *Journal of Statistical Software*, **54**, 1–35.

BARBOSA, M., CARNEIRO, T. e Tavares, A. I. (2012). Métodos de classificação por árvores de decisão disciplina de projeto e análise de algoritmos. *UFOP–Universidade Federal de Ouro Preto Ouro Preto, Minas Gerais–MG*.

Bianka Tallita Passos (2021). Conhecendo os tipos de aprendizado de máquina: supervisionado e não supervisionado. Disponível em: https://ateliware.com/blog/aprendizado-de-maquina-tipos. Acesso em: Novembro de 2023.

Breiman, L. (1996). Bagging predictors. *Machine learning*, **24**, 123–140.

de Ullibarri Galparsoro, L. e Fernández, P. (1998). Curvas roc. *Atención Primaria en la Red*, **5**(4), 229–35.

Dias, A. A. D. (2012). *Previsão do incumprimento no crédito a empresas com classificadores múltiplos*. Tese de doutorado, Universidade Tecnica de Lisboa (Portugal).

Fontana, É. (2020). Introdução aos algoritmos de aprendizagem supervisionada. *Departamento de Engenharia Química, Universidade Federal do Paraná*.

Freund, Y. e Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. Em *Computational Learning Theory: Second European Conference, EuroCOLT'95 Barcelona, Spain, March 13–15, 1995 Proceedings 2*, páginas 23–37. Springer.

Izbicki, R. e dos Santos, T. M. (2020). *Aprendizado de máquina: uma abordagem estatística*. Rafael Izbicki.

James, G., Witten, D., Hastie, T., Tibshirani, R. *et al.* (2013). *An introduction to statistical learning*, volume 112. Springer.

LISKA, G. R., de Menezes, F. S., Cirillo, M. Â. e Vivanco, M. J. F. (2012). Classificação de dados em modelos com resposta binária via algoritmo boosting e regressão logística. *Revista Matemática e Estatística em Foco-ISSN*, **2318**, 0552.

Michie, D., Spiegelhalter, D. J. e Taylor, C. C. (1994). *Machine learning, neural and statistical classification*. Ellis Horwood.

Monard, M. C. e Baranauskas, J. A. (2003). Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, **1**(1), 32.

Opitz, D. e Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, **11**, 169–198.

Prati, R. C., Batista, G., Monard, M. C. *et al.* (2008). Curvas roc para avaliação de classificadores. *Revista IEEE América Latina*, **6**(2), 215–222.

Rubesam, A. (2004). *Estimação não parametrica aplicada a problemas de classificação via Bagging e Boosting*. Tese de doutorado, [sn].

Silva, L. A. (2023). Utilização de classificadores múltiplos com modelo de regressão em sobrevivência, na classificação de clientes. *Trabalho de conclusão de curso - DEs (UFSCar)*.

Apêndice A

Código

```
## Pacotes e Bibliotecas
library(readr)
library(corrplot)
library(rsample)
library(ipred)
library(e1071)
library(caret)
library(ROCR)
library(ada)
library(adabag)
library(rpart)
library(rpart.plot)
library(rattle)
library(MASS)
library(randomForest)
library(pROC)
library(ggplot2)

#####################################################################
######Estudo de Classificação para a Base de Dados de Titanic
#####################################################################

## Lendo os dados
```

```
####################
Titanic <- read.csv("Titanic_Simulações.csv")
Titanic[1:5,]
attach(Titanic)

## Separar treino (75%) e teste (25%)
########################################
train <- Titanic[1:668,]
train$Survived <- as.factor(train$Survived)
teste <- Titanic[669:891,]
teste$Survived <- as.factor(teste$Survived)

teste <- read.csv("Titanic_test.csv")
teste[1:5,]
attach(teste)



### Análises com o pacote
### bagging
#########################
#Bagging no conjunto treinamento
m.bag <- 20
train_bagg <- bagging(Survived ~ Sex + Age + Pclass,
                      data=train, coob=T, nbagg=m.bag, ns=591)

pred_train_bagg <- predict(train_bagg, newdata=train)
pred_train_bagg$confusion
pred_train_bagg$error

classe.estimada <- pred_train_bagg$class
classe.estimada[1:20]
tab_bagg <- table(classe.estimada, train$Survived)
tab_bagg
```

```
## evolução do erro de má classificação
evol_train_bagg <- errorevol(train_bagg, train)

k.min.bag <- which(evol_train_bagg$error==min(evol_train_bagg$error))[1]
min.er.bag <- round(evol_train_bagg$error[k.min.bag],5)
mn.y <- round(min.er.bag-0.005,2)
mx.y <- round(max(evol_train_bagg$error)[1]+0.005,2)
mx.x <- length(evol_train_bagg$error)
erro.final <- pred_train_bagg$error

plot(evol_train_bagg$error, type = "n",
     ylim = c(mn.y, mx.y), main = "Evolução do erro de má classificação",
     xlab = "Árvore",ylab = " ", col = "blue", lwd = 2, cex.main=1)
lines(c(0,mx.x), c(min.er.bag,min.er.bag), lty = 3, lwd=2)
lines(c(0,mx.x), c(erro.final,erro.final), lty = 2, col="red", lwd=2)
lines(evol_train_bagg$error, col = "blue", lwd = 2)
box()
legend("topright", legend=c("erro final", "erro mínimo"),
       lty=c(2,3), col=c(2,1), lwd=2, bty="n")


########################################
## Predição do conjunto de teste ##
########################################
pred_teste_bagg <- predict(train_bagg, newdata=teste)
pred_teste_bagg$confusion
pred_teste_bagg$error
errorevol(train_bagg, teste)

Pred_teste_bagg <- cbind(teste$PassengerId,
                         Status.teste, pred_teste_bagg$class)
dimnames(Pred_teste_bagg)[[1]] <- rownames(Pred_teste_boost,
```

```
                                                do.NULL = F, prefix = "Pred.")
dimnames(Pred_teste_bagg)[[2]] <- c("Passageiro",
                                    "Sobreviv?ncia","Sobreviv.predita")
Pred_teste_boost[1:20,]

## evolução do erro de má classificação
evol_teste_bagg <- errorevol(train_bagg, teste)

k.min.bag <- which(evol_teste_bagg$error==min(evol_teste_bagg$error))[1]
min.er.bag <- round(evol_teste_bagg$error[k.min.bag],5)
mn.y <- round(min.er.bag-0.005,2)
mx.y <- round(max(evol_teste_bagg$error)[1]+0.005,2)
mx.x <- length(evol_teste_bagg$error)
erro.final <- pred_teste_bagg$error

plot(evol_teste_bagg$error, type = "n", ylim = c(mn.y, mx.y),
     main = "Evolução do erro de má classificação na testeação",
     xlab = "Árvore",ylab = " ", col = "blue", lwd = 2, cex.main=1)
lines(c(0,mx.x), c(min.er.bag,min.er.bag), lty = 3, lwd=2)
lines(c(0,mx.x), c(erro.final,erro.final), lty = 2, col="red2", lwd=2)
lines(evol_teste_bagg$error, col = "blue3", lwd = 2)
box()
legend("topright", legend=c("erro final", "erro mínimo"),
       lty=c(2,3), col=c(2,1), lwd=2, bty="n")



################################
## Predição do conjunto teste ##
################################
pred_teste_bagg <- predict(train_bagg, newdata=teste)
pred_teste_bagg
```

```
Pred_Sobrev_bagg <- cbind(teste$PassengerId,pred_teste_bagg$class)
dimnames(Pred_Sobrev_bagg)[[1]] <- rownames(Pred_Sobrev_bagg, do.NULL = F,
                                            prefix = "Pred.")
dimnames(Pred_Sobrev_bagg)[[2]] <- c("Num.passageiro","Sobreviv.predita")
Pred_Sobrev_bagg[1:20,]




#####################################
#Análise BOOSTING conjunto de treino#
#####################################
m.bst <- 11
train_boost <- boosting(Survived ~ Sex + Age + Pclass, data=train,
                        boos=FALSE, mfinal = m.bst, coeflearn = "Freund")

train_boost$weights
pred_train_boost <- predict.boosting(train_boost, newdata=train)
tab.boost <- pred_train_boost$confusion
tab.boost

pred_train_boost$error

## evolução do erro de má classificação
evol_train_boost <- errorevol(train_boost, train)
evol_train_boost$error
k.min.bst <- which(evol_train_boost$error==min(evol_train_boost$error))[1]
min.er.bst <- round(evol_train_boost$error[k.min.bst],5)
mnt.y <- round(min.er.bst-0.005,2)
mxt.y <- round(max(evol_train_boost$error)[1]+0.005,2)
mxt.x <- length(evol_train_boost$error)
erro.final <- pred_train_boost$error
```

```
plot(evol_train_boost$error, type = "n", ylim = c(mnt.y, mxt.y),
     main = "Erro de má classificação - boosting conjunto de treinamento",
     xlab = "Número de iterações",ylab = " ", col = "blue", lwd = 2, cex.main=1)
lines(c(0,mxt.x), c(min.er.bst,min.er.bst), lty = 3, lwd=2)
lines(c(0,mxt.x), c(erro.final,erro.final), lty = 2, col="red", lwd=2)
lines(evol_train_boost$error, col = "blue", lwd = 2)
box()
legend("topright", legend=c("erro final", "erro mínimo"),
       lty=c(2,3), col=c(2,1), lwd=2, bty="n")


Status.train <- rep(1,dim(train)[1])
Status.train[train$Survived==0] <- 0
escores_t1 <- pred_train_boost$prob[,2]
matriz_escores_t1 <- cbind(escores_t1, Status.train)
dimnames(matriz_escores_t1)[[1]] <- rownames(matriz_escores_t1, do.NULL = F,
                                             prefix = "Passag.")
dimnames(matriz_escores_t1)[[2]] <- c("Prob_Sobrev","Status_sobrev.")
matriz_escores_t1[1:20,]

## Curva ROC - predição
## conjunto treinamento
########################
Sobrev <- train$Survived
prob <- pred_train_boost$prob[,2]
brier <- sum((prob-Sobrev)^2)
corte <- seq(0.1,0.9,by=0.1)
k <- length(corte)
sens <- numeric(k)
espec <- numeric(k)
tx.erro <- numeric(k)
pred.um <- numeric(k)
for(i in 1:k){
```

```
  classif <- rep(0,668)
  classif[which(prob>=corte[i])] <- 1
  tabela <- table(Sobrev, classif)
  #  if(dim(tabela)[2]==1){tabela <- cbind(tabela,c(0,0))}
  sens[i] <- sensitivity(tabela)
  espec[i] <- specificity(tabela)
  tx.erro[i] <- (tabela[1,2]+tabela[2,1])/sum(tabela)
  pred.um[i] <- sum(tabela[,2])/sum(tabela)
}

cbind(corte,sens,espec)

cbind(corte, tx.erro, pred.um)

plot(corte,tx.erro,pch=19,type="l",lwd=2, col="red4")

p1 <- length(which(Sobrev==1))/length(Sobrev)
plot(pred.um,tx.erro,pch=19,type="b",lwd=2)
lines(c(p1,p1),c(0,1), col="red4",lwd=2)



sens.g <- c(1,sens,0)
espec.g <- c(0,espec,1)

## Cálculo da área sob a curva
##############################
na <- length(sens.g)
area <- numeric(na-1)
for(j in 1:(na-1)){
  area[j] <- (sens.g[j+1]+sens.g[j])*(espec.g[j+1]-espec.g[j])/2
}
area
auc <- sum(area)
```

```
auc

PP <- 2*auc-1
PP


plot(xlim=c(0,1), ylim=c(0,1), axes=F, 1-espec.g, sens.g,
     xlab="1-especificidade", ylab="sensibilidade", pch=19, lwd=2, col="red4")
axis(2,seq(0,1,0.25), pos=0)
axis(1,seq(0,1,0.2), pos=0)
lines(c(0,1),c(1,1))
lines(c(1,1),c(1,0))
lines(c(0,1),c(0,1), lwd=2)
lines(1-espec.g, sens.g, lwd=2, col="red4")

########################################
## Predição no conjunto de teste ##
########################################
pred_teste_boost <- predict.boosting(train_boost, newdata=teste)
pred_teste_boost$confusion
pred_teste_boost$error
Status.teste <- rep(1,dim(teste)[1])
Status.teste[teste$Survived==0] <- 0

Pred_teste_boost <- cbind(teste$PassengerId, Status.teste,
                          pred_teste_boost$class)
dimnames(Pred_teste_boost)[[1]] <- rownames(Pred_teste_boost,
                                            do.NULL = F, prefix = "Pred.")
dimnames(Pred_teste_boost)[[2]] <- c("Passageiro","Sobrevivência",
                                     "Sobreviv.predita")
Pred_teste_boost[1:20,]

## evolução do erro de má classificação
```

```
evol_teste_boost <- errorevol(train_boost, teste)
evol_teste_boost$error
k.min.bst <- which(evol_teste_boost$error==min(evol_teste_boost$error))[1]
min.er.bst <- round(evol_teste_boost$error[k.min.bst],5)
mnv.y <- round(min.er.bst-0.005,2)
mxv.y <- round(max(evol_teste_boost$error)[1]+0.005,2)
mxv.x <- length(evol_teste_boost$error)
erro.final <- pred_teste_boost$error

plot(evol_teste_boost$error, type = "n", ylim = c(mnv.y, mxv.y),
     main = "Erro de má classificação - boosting conjunto de teste",
     xlab = "Número de iterações",ylab = " ", col = "blue", lwd = 2,
     cex.main=1)
lines(c(0,mxv.x), c(min.er.bst,min.er.bst), lty = 3, lwd=2)
lines(c(0,mxv.x), c(erro.final,erro.final), lty = 2, col="red", lwd=2)
lines(evol_teste_boost$error, col = "blue", lwd = 2)
box()
legend("topright", legend=c("erro final", "erro mínimo"),
       lty=c(2,3), col=c(2,1), lwd=2, bty="n")


################################
## Predição do conjunto teste ##
################################
pred_teste_boost <- predict.boosting(train_boost, newdata=teste)

Pred_Sobrev_boost <- cbind(teste$PassengerId,pred_teste_boost$class)
dimnames(Pred_Sobrev_boost)[[1]] <- rownames(Pred_Sobrev_boost, do.NULL = F,
                                             prefix = "Pred.")
dimnames(Pred_Sobrev_boost)[[2]] <- c("Num.passageiro","Sobreviv.predita")
Pred_Sobrev_boost[1:20,]

#################################################################
```

```
######Estudo de Classificação para a Base de Dados de Alzheimer
#####################################################################

####Ler a base de dados
##Base de dados contém 2149 obs. e 35 variáveis
dados_alz1 <- read.csv("alzheimers_disease_data.csv")
attach(dados_alz1)
head(dados_alz1)


#Excluir as colunas quanlitativas da base
dados_alz <- dados_alz1[,-c(1,35)]

head(dados_alz)

#Variáveis selecionadas para o modelo
dados_alz[c(24,26,27)]

#Proporção real de 0's e 1's dos dados originais
length(which(dados_alz$Diagnosis==1))/nrow(dados_alz)

length(which(dados_alz$Diagnosis==0))/nrow(dados_alz)

nrow(dados_alz)


####Alterar a base de dados para a proporção desejada (x% de 0 e y% de 1)
# Definir a proporção desejada
x <- 0.20
y <- 0.80


prop_0 <- x
```

```
prop_1 <- y

# Separar os dados por valores de Diagnosis
data_0 <- subset(dados_alz, Diagnosis == 0)
data_1 <- subset(dados_alz, Diagnosis == 1)

# Determinar o máximo de amostras sem perder dados desnecessariamente
n_max_0 <- nrow(data_0)
n_max_1 <- nrow(data_1)

# Calcular o maior tamanho possível mantendo a proporção desejada
n_0 <- min(n_max_0, round(n_max_1 * (prop_0 / prop_1)))
n_1 <- min(n_max_1, round(n_max_0 * (prop_1 / prop_0)))

# Ajustar novamente para garantir que a soma seja consistente
total_final <- n_0 + n_1 #número total de amostra
n_0 <- round(total_final * prop_0)
n_1 <- round(total_final * prop_1)

# Selecionar amostras aleatórias respeitando os tamanhos calculados
set.seed(729739)         # Definir a semente para garantir a reprodutibilidade
sampled_0 <- data_0[sample(nrow(data_0), n_0, replace = FALSE), ]
sampled_1 <- data_1[sample(nrow(data_1), n_1, replace = FALSE), ]

# Combinar as amostras em um novo conjunto de dados
base_ajustada <- rbind(sampled_0, sampled_1)

# Verificar a nova proporÃ§Ã£o
table(base_ajustada$Diagnosis) / nrow(base_ajustada)


## Separar treino (75%) e teste (25%)
########################################
```

```
## com escolha aleatória
########################

#Para proporção 65% e 35% (Proporção Original)

amostra_75 <- 1612
amostra_25 <- 2149 - amostra_75

base_ajustada <- dados_alz

## Separar treino (75%) e teste (25%)
#Quatidade correta da amostra
amostra_75 <- round(total_final*0.75,0)
amostra_25 <- total_final - amostra_75



########################################
set.seed(729739)
amostra <- sort(sample(1:total_final, amostra_75, replace=F))
train <- base_ajustada[amostra,]
train$Diagnosis <- as.factor(train$Diagnosis)
teste <- base_ajustada[-amostra,]
teste$Diagnosis <- as.factor(teste$Diagnosis)



####Ajustar várias árvores para definir as variáveis do modelo
#############################################################
modelo_rf <- randomForest(factor(train$Diagnosis)~., data=train, ntree=300,
                          mtry=4, importance=TRUE)
plot(modelo_rf)

#Importância da variável
```

```
var_importance <- importance(modelo_rf)


# Selecionar apenas a coluna "MeanDecreaseAccuracy"
mean_decrease_accuracy <- var_importance[, "MeanDecreaseAccuracy"]

# Exibir os valores ordenados pela importância (em ordem decrescente)
mean_decrease_accuracy <- sort(mean_decrease_accuracy, decreasing = TRUE)

importance_df <- data.frame(
  Variable = names(mean_decrease_accuracy),
  MeanDecreaseAccuracy = mean_decrease_accuracy
)


#Essa medida diz o quanto cada variável impacta na acurácia da classificação
ggplot(importance_df, aes(x = reorder(Variable, -MeanDecreaseAccuracy),
                          y = MeanDecreaseAccuracy)) +
  geom_bar(stat = "identity", fill = "steelblue", color = "black") +
  labs(title = "Importância das Variáveis - MeanDecreaseAccuracy",
       x = "Variáveis",
       y = "MeanDecreaseAccuracy") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


#Hiperparametros para o método Bagging -----------------


# Garantir que Diagnosis está como fator binário com níveis bem definidos
train$Diagnosis <- factor(train$Diagnosis, levels = c(0, 1))
teste$Diagnosis <- factor(teste$Diagnosis, levels = c(0, 1))
```

```
# Definir os hiperparâmetros a serem testados
nbagg_values <- c(10, 50, 100, 200)
maxdepth_values <- c(3, 4, 5)
minsplit_values <- c(90, 100, 150, 200)
ns_values <- c(50,100,200,300)


param_grid <- expand.grid(nbagg = nbagg_values,
                          maxdepth = maxdepth_values,
                          minsplit = minsplit_values,
                          ns = ns_values)

acuracia <- is.numeric(192)
n <- 1
#descobrir hiperparametros
for (i in 1:length(nbagg_values)) {
  for (j in 1:length(maxdepth_values)) {
    for (k in 1:length(minsplit_values)) {
      for (l in 1:length(ns_values)) {


      modelo <- bagging(Diagnosis~
                          FunctionalAssessment + ADL + BehavioralProblems,
                        data=train,
                        nbagg = nbagg[i],
                        ns = ns_values,
                        control=rpart.control(maxdepth_values[j],
                                              minsplit=minsplit_values[k]))

      pred_train_bagg <- predict(modelo, newdata=train)

      pred_teste_bagg <- predict(modelo, newdata=teste)
```

```
      acuracia[n] <- (
        pred_teste_bagg$confusion[1,1]+
          pred_teste_bagg$confusion[2,2])/length(teste$Diagnosis)

      n <- n+1
      }

    }
  }
}
tabela_hiper <- cbind(param_grid,round(acuracia,4))

which(acuracia == max(acuracia))


#nbagg = 10, maxdepth = 4, minsplit = 100, ns = 50, acuracidade: 78.96%
tabela_hiper[17,]


#Hiperparametros para o método Boosting -----------------

# Garantir que Diagnosis está como fator binário com níveis bem definidos
train$Diagnosis <- factor(train$Diagnosis, levels = c(0, 1))
teste$Diagnosis <- factor(teste$Diagnosis, levels = c(0, 1))

# Definir os hiperparâmetros a serem testados
mfinal_values <- c(10, 50, 100, 200,300)
maxdepth_values <- c(3, 4, 5)
minsplit_values <- c(90, 100, 150, 200)

param_grid <- expand.grid(mfinal = mfinal_values,
                          maxdepth = maxdepth_values,
                          minsplit = minsplit_values)
```

```
acuracia <- is.numeric(60)
n <- 1
#descobrir hiperparametros
for (i in 1:length(mfinal_values)) {
  for (j in 1:length(maxdepth_values)) {
    for (k in 1:length(minsplit_values)) {
      modelo <- boosting(Diagnosis~
                           FunctionalAssessment + ADL + BehavioralProblems,
                         data=train,
                         boos=TRUE,
                         mfinal = mfinal_values[i],
                         control=rpart.control(maxdepth_values[j],
                                               minsplit=minsplit_values[k]))

      pred_train_boost <- predict(modelo, newdata=train)

      pred_teste_boost <- predict(modelo, newdata=teste)

      acuracia[n] <- (
        pred_teste_boost$confusion[1,1]+
          pred_teste_boost$confusion[2,2])/length(teste$Diagnosis)

      n <- n+1
    }

  }
}

tabela_hiper <- cbind(param_grid,round(acuracia,4))

which(acuracia == max(acuracia))
```

```
#mfinal = 300, maxdepth = 3, minsplit = 90, acuracidade: 78.96%
tabela_hiper[5,]


#nbagg = 10, maxdepth = 4, minsplit = 100, ns = 50, acuracidade: 78.96%
# Aplicação do método Bagging ------------------------------------------------
#Bagging no conjunto treinamento (atualizar valores de nbag=número de árvores
#e ns = tamanho da amostra booststrap)
# proporção de 0 e 1
set.seed(729739)
m.bag <- 10 #qtd de árvores
train_bagg <- bagging(Diagnosis ~ FunctionalAssessment + ADL +
                        BehavioralProblems,
                      data=train, coob=TRUE, nbagg = m.bag, ns=50,
                      control = rpart.control(
                        maxdepth = 4,
                        minsplit = 100))

## evolução do erro de má classificação
evol_train_bagg <- errorevol(train_bagg,train)
evol_train_bagg$error

#taxa mínima de erro
max(evol_train_bagg$error)
min(evol_train_bagg$error)

#Predição utilizando método bagging, sua matriz de confusão e taxa de erro
pred_train_bagg <- predict(train_bagg, newdata=train)
pred_train_bagg$confusion
pred_train_bagg$error

####Plotar gráfico da evolução da taxa de erro conjunto de treino
```

```
k.min.bag <- which(evol_train_bagg$error==min(evol_train_bagg$error))[1]
min.er.bag <- round(evol_train_bagg$error[k.min.bag],5)
mn.y <- round(min.er.bag-0.005,2)
mx.y <- round(max(evol_train_bagg$error)[1]+0.005,2)
mx.x <- length(evol_train_bagg$error)
erro.final <- pred_train_bagg$error

plot(evol_train_bagg$error, type = "n", ylim = c(mn.y, mx.y),
     main = "Evolução do erro de má classificação",
     xlab = "Número de árvore",ylab = " ", col = "blue", lwd = 2, cex.main=1)
lines(c(0,mx.x), c(min.er.bag,min.er.bag), lty = 3, lwd=2)
lines(c(0,mx.x), c(erro.final,erro.final), lty = 2, col="red", lwd=2)
lines(evol_train_bagg$error, col = "blue", lwd = 2)
box()
legend("topright", legend=c("erro final", "erro mínimo"),
       lty=c(2,3), col=c(2,1), lwd=2, bty="n")




########################################
## Predição do conjunto de teste ##
########################################
pred_teste_bagg <- predict(train_bagg, newdata=teste)
pred_teste_bagg$confusion #matriz de confusão
pred_teste_bagg$error #Taxa de erro
errorevol(train_bagg, teste) #evolução do erro

## evolução do erro de má classificação
evol_teste_bagg <- errorevol(train_bagg, teste)



####Plotar gráfico da evolução da taxa de erro conjunto de testeação
```

```
k.min.bag <- which(evol_teste_bagg$error==min(evol_teste_bagg$error))[1]
min.er.bag <- round(evol_teste_bagg$error[k.min.bag],5)
mn.y <- round(min.er.bag-0.005,2)
mx.y <- round(max(evol_teste_bagg$error)[1]+0.005,2)
mx.x <- length(evol_teste_bagg$error)
erro.final <- pred_teste_bagg$error

plot(evol_teste_bagg$error, type = "n",
     ylim = c(mn.y, mx.y),
     main = "Evolução do erro de má classificação no teste",
     xlab = "Árvore",ylab = " ", col = "blue", lwd = 2, cex.main=1)
lines(c(0,mx.x), c(min.er.bag,min.er.bag), lty = 3, lwd=2)
lines(c(0,mx.x), c(erro.final,erro.final), lty = 2, col="red2", lwd=2)
lines(evol_teste_bagg$error, col = "blue3", lwd = 2)
box()
legend("topright", legend=c("erro final", "erro mínimo"),
       lty=c(2,3), col=c(2,1), lwd=2, bty="n")



##Estudo das medidas de desempenho para o conjunto de teste##
#################################METODO BAGGING#####################
####################################################################
Sobrev <- teste$Diagnosis
prob <- pred_teste_bagg$prob[,2]
brier <- sum((prob-Sobrev)^2)
corte <- seq(0.1,0.9,by=0.1)
k <- length(corte)
sens <- numeric(k)
espec <- numeric(k)
acur <- numeric(k) #Acuracidade
prec <- numeric(k) #precisão
medidaf <- numeric(k) #F-mesure
```

```
tx.erro <- numeric(k) #taxa de erro
pred.um <- numeric(k) #predição de 1's
for(i in 1:k){
  classif <- rep(0,amostra_25)
  classif[which(prob>=corte[i])] <- 1
  tabela <- table(Sobrev, classif)
  if(dim(tabela)[2]==1){tabela <- cbind(tabela,c(0,0))}
  if(dim(tabela)[1]==1){tabela <- rbind(tabela,c(0,0))}
  sens[i] <- tabela[2,2]/(tabela[2,2]+tabela[2,1])
  espec[i] <- tabela[1,1]/(tabela[1,1]+tabela[1,2])
  acur[i] <- ((tabela[1,1]+tabela[2,2])/(tabela[1,1]+tabela[1,2]+
                                          tabela[2,1]+tabela[2,2]))
  prec[i] <- tabela[2,2]/(tabela[2,2]+tabela[1,2])
  medidaf[i] <- (2*sens[i]*prec[i])/(sens[i]+prec[i])
  tx.erro[i] <- (tabela[1,2]+tabela[2,1])/sum(tabela)
  pred.um[i] <- sum(tabela[,2])/sum(tabela)
}

#tabela contendo as medidas (apenas 4 casas decimais)
cbind(round(corte,4),round(sens,4),round(espec,4),round(acur,4),round(prec,4),
      round(medidaf,4))

#tabela com taxa de erro e predição de 1's (apenas 4 casas decimais)
cbind(round(corte,4), round(tx.erro,4), round(pred.um,4))




##############################
##########Curva ROC##########
##############################

teste$Diagnosis <- factor(teste$Diagnosis, levels = c(0, 1))
```

```
curva_roc <- roc(teste$Diagnosis,pred_teste_bagg$prob[,2],levels = c(0,1),
                 direction = '<')

#Valor da área sob a curva
auc_value <- auc(curva_roc)
print(paste("AUC:", round(auc_value, 4)))

#Definindo pontos de corte (thresholds)
thresholds <- seq(0, 1, by = 0.1)

# Garantir que a variável Diagnosis esteja no formato binário (0 ou 1)
true_labels <- as.numeric(as.factor(teste$Diagnosis)) - 1

# Inicializar vetores para Sensibilidade e Especificidade
sensitivities <- numeric(length(thresholds))
specificities <- numeric(length(thresholds))

for (i in seq_along(thresholds)) {
  # Aplicar o threshold para converter probabilidades em classes binárias
  pred_class <- ifelse(pred_teste_bagg$prob[, 2] >= thresholds[i], 1, 0)

  # Criar matriz de confusão com classes 0 e 1
  conf_mat <- table(
    factor(pred_class, levels = c(0, 1)),
    factor(true_labels, levels = c(0, 1))
  )

  # Extrair valores da matriz de confusão (garantindo que seja 2x2)
  VP <- conf_mat[2, 2]  # Verdadeiro Positivo
  FN <- conf_mat[1, 2]  # Falso negativo
  VN <- conf_mat[1, 1]  # Verdadeiro Negativo
  FP <- conf_mat[2, 1]  # Falso Positivo
```

```
  # Calcular sensibilidade e especificidade
  sensitivities[i] <- ifelse((VP + FN) > 0, VP / (VP + FN), 0)
  specificities[i] <- ifelse((VN + FP) > 0, VN / (VN + FP), 0)
}




# Criar uma tabela com os resultados
result_table <- data.frame(
  Threshold = thresholds,
  Sensitivity = round(sensitivities, 4),
  Specificity = round(specificities, 4)
)

#Alterar os valores de forma manual
result_table[result_table$Threshold == 0.0, c("Sensitivity",
                                              "Specificity")] <- c(1, 0)
result_table[result_table$Threshold == 1.0, c("Sensitivity",
                                              "Specificity")] <- c(0, 1)

# Mostrar a tabela
print(result_table)

#########################
####Gráfico curva ROC####
#########################
plot(1 - result_table$Specificity,
     result_table$Sensitivity,
     type = "b", col = "blue",
     pch = 19, xlab = "1 - Especificidade", ylab = "Sensibilidade",
     main = "Curva ROC", xlim = c(0, 1), ylim = c(0, 1))
abline(a = 0, b = 1, col = "red", lty = 2)
```

```
text(
  1 - result_table$Specificity, result_table$Sensitivity,
  labels = round(1-result_table$Threshold, 2), pos = 4, cex = 0.8
)
```

```
#mfinal = 300, maxdepth = 3, minsplit = 90, acuracidade: 78.96%
# Aplicação do método Boosting ------------------------------------------------
set.seed(729739)
m.bst <- 300 #qtd iterações
train_boost <- boosting(Diagnosis ~
                          FunctionalAssessment + ADL + BehavioralProblems,
                        data = train, boos=TRUE, mfinal = m.bst,
                        coeflearn = "Freund",
                        rpart.control(maxdepth = 3, minsplit = 90))
```

```
#predições com modelo boosting para conjunto de treino
pred_train_boost <- predict.boosting(train_boost, newdata=train)
tab.boost <- pred_train_boost$confusion
tab.boost #matriz de confusão
```

```
#Probabilidades de classificação para cada observação
pred_train_boost$prob
```

```
#Pesos atrbuídos em cada iteração
train_boost$weights

#Erro de classificação
pred_train_boost$error

## evolução do erro de má classificação
evol_train_boost <- errorevol(train_boost, train)
min(evol_train_boost$error)
k.min.bst <- which(evol_train_boost$error==min(evol_train_boost$error))[1]
min.er.bst <- round(evol_train_boost$error[k.min.bst],5)
mnt.y <- round(min.er.bst-0.005,2)
mxt.y <- round(max(evol_train_boost$error)[1]+0.005,2)
mxt.x <- length(evol_train_boost$error)
erro.final <- pred_train_boost$error

plot(evol_train_boost$error, type = "n",
     ylim = c(mnt.y, mxt.y),
     main = "Erro de má classificação - boosting conjunto de treinamento",
     xlab = "Número de iterações",ylab = " ",
     col = "blue", lwd = 2,
     cex.main=1)
lines(c(0,mxt.x), c(min.er.bst,min.er.bst), lty = 3, lwd=2)
lines(c(0,mxt.x), c(erro.final,erro.final), lty = 2, col="red", lwd=2)
lines(evol_train_boost$error, col = "blue", lwd = 2)
box()
legend("topright", legend=c("erro final", "erro mínimo"),
       lty=c(2,3), col=c(2,1), lwd=2, bty="n")

#descobrir o número da iteração da evolução de má classificação
which(evol_train_boost$error==min(evol_train_boost$error))
min(evol_train_boost$error)
```

```
#######################################
## Predição no conjunto de teste ##
#######################################
pred_teste_boost$prob
pred_teste_boost <- predict.boosting(train_boost, newdata=teste)
pred_teste_boost$confusion
pred_teste_boost$error
Status.teste <- rep(1,dim(teste)[1])
Status.teste[teste$Diagnosis==0] <- 0
evol_teste_boost <- errorevol(train_boost, teste)


###medidas de desempenho
###conjunto teste
#######################
Sobrev <- teste$Diagnosis
prob <- pred_teste_boost$prob[,2]
corte <- seq(0.1,0.9,by=0.1)
k <- length(corte)
sens <- numeric(k) #sensibilidade
espec <- numeric(k) #especificidade
acur <- numeric(k) #Acuracidade
prec <- numeric(k) #precisão
medidaf <- numeric(k) #F-mesure
tx.erro <- numeric(k)
pred.um <- numeric(k)
for(i in 1:k){
  classif <- rep(0,amostra_25)
  classif[which(prob>=corte[i])] <- 1
```

```
  tabela <- table(Sobrev, classif)
  if(dim(tabela)[2]==1){tabela <- cbind(tabela,c(0,0))}
  if(dim(tabela)[1]==1){tabela <- rbind(tabela,c(0,0))}
  sens[i] <- tabela[2,2]/(tabela[2,2]+tabela[2,1])
  espec[i] <- tabela[1,1]/(tabela[1,1]+tabela[1,2])
  acur[i] <- ((tabela[1,1]+tabela[2,2])/(tabela[1,1]+tabela[1,2]+
                                          tabela[2,1]+tabela[2,2]))
  prec[i] <- tabela[2,2]/(tabela[2,2]+tabela[1,2])
  medidaf[i] <- (2*sens[i]*prec[i])/(sens[i]+prec[i])
  #sens[i] <- sensitivity(tabela)
  #espec[i] <- specificity(tabela)
  tx.erro[i] <- (tabela[1,2]+tabela[2,1])/sum(tabela)
  pred.um[i] <- sum(tabela[,2])/sum(tabela)
}

#tabela contendo as medidas (apenas 4 casas decimais)
cbind(round(corte,4),round(sens,4),round(espec,4),round(acur,4),
      round(prec,4),round(medidaf,4))

#tabela com taxa de erro e predição de 1's (apenas 4 casas decimais)
cbind(round(corte,4), round(tx.erro,4), round(pred.um,4))




######################
######Curva ROC#######
######################

train$Diagnosis <- factor(train$Diagnosis, levels = c(0, 1))

curva_roc <- roc(train$Diagnosis,pred_train_boost$prob[,2],
                 levels = c(0,1), direction = '<')
```

```
#Valor da área sob a curva
auc_value <- auc(curva_roc)
print(paste("AUC:", round(auc_value, 4)))

#Definindo pontos de corte (thresholds)
thresholds <- seq(0, 1, by = 0.1)

# Garantir que a variável Diagnosis esteja no formato binário
true_labels <- as.numeric(as.factor(train$Diagnosis)) - 1

# Inicializar vetores para Sensibilidade e Especificidade
sensitivities <- numeric(length(thresholds))
specificities <- numeric(length(thresholds))
acuracia <- numeric(length(thresholds))
precisao <- numeric(length(thresholds))
medidaf <- numeric(length(thresholds))
prop_um <- numeric(length(thresholds))
tx_erro <- numeric(length(thresholds))

for (i in seq_along(thresholds)) {
  pred_class <- ifelse(pred_train_boost$prob[, 2] >= thresholds[i], 1, 0)
  conf_mat <- table(
    factor(pred_class, levels = c(0, 1)),
    factor(true_labels, levels = c(0, 1))
  )

  VP <- conf_mat[2, 2]  # Verdadeiro Positivo
  FN <- conf_mat[1, 2]  # Falso Negativo
  VN <- conf_mat[1, 1]  # Verdadeiro Positivo
  FP <- conf_mat[2, 1]  # Falso Negativo

  # Calcular sensibilidade e especificidade
  sensitivities[i] <- ifelse((VP + FN) > 0, VP / (VP + FN), 0)
```

```
  specificities[i] <- ifelse((VN + FP) > 0, VN / (VN + FP), 0)
  acuracia[i] <- ifelse((VP + VN + FP + FN) > 0,
                        (VP + VN) / (VP + VN + FP + FN), 0)
  precisao[i] <- ifelse((VP + FP) > 0, VP / (VP + FP), 0)
  medidaf[i] <- ifelse((2 * VP + FP + FN) > 0,
                       (2 * VP) / (2 * VP + FP + FN), 0)
  tx_erro[i] <- ifelse((FP + FN) > 0, ((FP + FN) / (VP + VN + FP + FN)), 0)
  prop_um[i] <- ifelse((VP) > 0, (VP) / (VP + VN + FP + FN), 0)
}

# Criar uma tabela com os resultados
result_table <- data.frame(
  Threshold = thresholds,
  Sensitivity = round(sensitivities, 4),
  Specificity = round(specificities, 4),
  Acuracia = round(acuracia,4),
  Precisao = round(precisao,4),
  MedidaF = round(medidaf,4),
  Taxa_Erro = round(tx_erro,4),
  prop_um = round(prop_um,4)
)

#Alterar os valores de forma manual
result_table[result_table$Threshold == 0.0, c("Sensitivity",
                                              "Specificity")] <- c(1, 0)
result_table[result_table$Threshold == 1.0, c("Sensitivity",
                                              "Specificity")] <- c(0, 1)

# Mostrar a tabela
print(result_table)


#########################
```

```
####Gráfico curva ROC####
#########################
plot(1 - result_table$Specificity,
     result_table$Sensitivity,
     type = "b", col = "blue",
     pch = 19, xlab = "1 - Especificidade", ylab = "Sensibilidade",
     main = "Curva ROC", xlim = c(0, 1), ylim = c(0, 1))
abline(a = 0, b = 1, col = "red", lty = 2)
text(
  1 - result_table$Specificity, result_table$Sensitivity,
  labels = round(1-result_table$Threshold, 2), pos = 4, cex = 0.8
)

#taxa de erro máximo e mínimo
max(evol_train_boost$error)
min(evol_train_boost$error)
```