Modelos de Random Forest para tarefas de classificação

Sara A. S. Dias[1], Matheus Q. Mota[2], Vinicius R. Bardelin[3]
UNICAMP, Campinas, SP

Resumo. Este trabalho propõe introduzir e sintetizar a fundamentação teórica por trás do método da *Random Forest*. Após isso, o método será aplicado em um exemplo de classificação de falência de empresas. Ademais, os resultados serão analisados, e, posteriormente, será introduzida uma técnica de *Oversampling - SMOTE*, a fim de contornar possíveis problemas com o modelo.

Palavras-chave. Random Forest, Aprendizado de Máquinas, Árvores de Decisão, CART, Classificação, Ensemble.

1 Introdução

O aumento da complexidade nos problemas no campo do aprendizado de máquinas tem impulsionado o desenvolvimento de metodologias cada vez mais robustas para tarefas de classificação e regressão. Nesse cenário, as árvores de decisão emergem como um algoritmo de aprendizado poderoso e altamente interpretável, mas apresentam limitações preditivas significativas, notadamente o problema de sobreajuste, devido à sua alta suscetibilidade à variância. Este desafio levanta a necessidade de abordagens mais robustas para maximizar o potencial das árvores de decisão.

Nesse contexto, os modelos random forest surgem como uma extensão promissora e eficaz. Ao compilar um conjunto de árvores de decisão, essa técnica busca mitigar as fragilidades das árvores individuais, proporcionando um modelo mais resistente a problemas de sobreajuste, menos suscetível à variância e mais robusto, uma vez que agrega diversos modelos para efetuar uma determinada tarefa. O objetivo deste artigo é apresentar a construção teórica, utilizando o algoritmo CART, proposto por Breiman, e uma aplicação prática de um modelo random forest. Buscamos demonstrar como a combinação de múltiplas árvores de decisão pode resultar em um modelo mais robusto, capaz de lidar com a complexidade dos problemas de aprendizado de máquinas, destacando sua eficácia em cenários práticos.

2 Fundamentações Teóricas

2.1 Árvores de Decisão

2.1.1 O básico das Árvores de Classificação

Os modelos CART são construídos particionando recursivamente o espaço preditor $\mathcal{X}$ testando uma sequência de condições booleanas e ajustando um modelo simples em cada uma das regiões particionadas [1]. Uma árvore de decisão é composta pela sua raiz, galhos e folhas. Na raiz, o modelo considera todo o espaço das entradas $\mathcal{X}$ para começar a construir a árvore. Os galhos (ou

[1]RA: 236301
[2]RA: 251495
[3]RA: 236321

nós internos) podem se dividir em dois galhos descendentes ou terminar em uma folha. Uma folha (ou nó terminal) é a atribuição de uma classe a uma instância [2]. Cada uma das divisões (ou nós) da árvore representa uma partição de $\mathcal{X}$. O modo como os modelos CART particionam o espaço das características é utilizando a divisão binária recursiva por todos os nós não terminais da árvore.

Uma árvore de decisão é um preditor $\varphi : \mathcal{X} \to \mathcal{Y}$, em que $\mathcal{Y}$ é o espaço das saídas, sendo $\mathcal{Y} = \{c_1, \ldots, c_k\}$ para tarefas de classificação. Com efeito, podemos definir uma partição do espaço das características $\mathcal{X}$ usando os preditores φ tal que $\mathcal{X} = \mathcal{X}_{c_1}^{\varphi} \cup \cdots \cup \mathcal{X}_{c_k}^{\varphi}$, onde $\mathcal{X}_{c_j}^{\varphi}$ é o conjunto dos vetores $\mathbf{x} \in \mathcal{X}$ tais que $\varphi(\mathbf{x}) = c_j$. Nesse sentido, cada nó t representa um subespaço $\mathcal{X}_t \subseteq \mathcal{X}$ do espaço das entradas, com a raiz da árvore correspondendo ao próprio $\mathcal{X}$. Um exemplo da divisão binária realizada na árvore seria: "$\mathbf{x} \in \mathcal{X}_A$?", onde $\mathcal{X}_A \subset \mathcal{X}$ é qualquer subconjunto do espaço preditor. A resposta para essa pergunta gera dois nós filhos, na esquerda estão os vetores que satisfazem essa condição, e a direita os vetores que não satisfazem. Assim, a classe predita por $\varphi(\mathbf{x})$ é o rótulo de uma folha [11]. Deste modo, o algoritmo escolhe a variável preditora e o ponto de corte que melhor ajusta o modelo por toda a árvore [1].

2.1.2 Construindo Árvores de Classificação

Os critérios utilizados para divisão binária recursiva (dbr) nos modelos CART baseiam-se em dividir os nós de modo a minimizar a impureza de cada nó. Então, para efetuar a dbr, selecionamos um preditor X_j e um limiar s que dividem o espaço preditor nas regiões $\{X|X_j < s\}$ e $\{X|X_j \geq s\}$ que levam ao menor índice de impureza. Nesse caso, considerando todos os preditores de um conjunto de dados, e todos os possíveis limiares s para cada um desses preditores, selecionamos o preditor e o limiar s que vão resultar no nó mais puro possível [9]. Por isso, o modelo CART é dito ter uma abordagem gananciona, pois procura dentre muitas possibilidades o melhor caso por toda a árvore.

Formalmente, uma árvore divide o espaço das características em regiões distintas e mutuamente exclusivas $\{R\}_1^m$. Com efeito, a predição de uma resposta dada uma instância $\mathbf{x}$ em uma região R_j é dada pela função moda $\xi(\mathbf{x}) = moda\{c_i : \mathbf{x}_i \in R_j\}$ [8]. Em palavras, se na região R_j a classe mais predominante é a classe c_j, então a classe predita na região R_j será c_j [8, 9].

Para a seleção da melhor divisão binária dentre todas as covariáveis, é necessário, inicialmente, identificar a melhor divisão para uma única variável. Faremos isso fixando uma medida de eficiência de divisão. Seja $c_1, \ldots, c_k$ as k classes de um problema de classificação. Para um nó t, definimos a função de impuridade do nó $i(t)$ como:

$$i(t) = \phi(p(1|t), \ldots, p(k|t))$$

onde $p(j|t)$ é uma estimativa de $\mathbb{P}[\mathbf{X} \in c_j|t]$, a probabilidade condicional de uma observação $\mathbf{X}$ ser da classe c_j dado que ela está no t-ésimo nó. Uma função ϕ frequentemente utilizada é o índice Gini, dado por:

$$i(t) = \sum_{k \neq k'} p(k|t)p(k'|t) = 1 - \sum_{k} \{p(k|t)\}^2.$$

Outra função comumente empregada é a entropia, expressa por:

$$i(t) = -\sum_{k=1}^{k} p(k|t) \log p(k|t).$$

Considere que em uma divisão s efetuada em um nó t, obtemos duas proporções p_L e p_R, onde p_L é a proporção das instâncias que satisfazem $\{X|X_j < s\}$, e vão para o nó descendente da esquerda, e p_R as instâncias que satisfazem $\{X|X_j \geq s\}$ que vão para o nó descendente da direita. Então, a eficiência de uma divisão s em um nó t é dada pela redução da impuridade obtida dividindo o nó pai t em dois nós filhos t_L e t_R, que é expressa por:

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R).$$

Portanto, a melhor divisão para uma única covariável X_j é aquela com o maior valor de $\Delta i(s,t)\ \forall s \in \mathcal{S}_j$, onde $\mathcal{S}_j$ é o conjunto de todas as possíveis divisões para X_j.

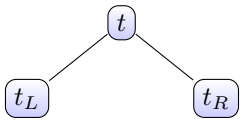

Figura 1: Divisão binária resursiva

2.1.3 Podando Árvores de Classificação

Os modelos baseados em árvores frequentemente se ajustam muito bem aos dados de treino e apresentam alta variância. Nesse sentido, é preciso restringir o tamanho da árvore para mitigar a possibilidade de sobre-ajuste. A ideia do algoritmo CART é construir uma árvore, inicialmente sem restrições, e depois poda-la até que ela possua um "tamanho ideal". No entanto, existem diversas possibilidades de podar a árvore original, e cada poda origina uma sub-árvore. Estamos interessados em encontrar a melhor sub-árvore, e faremos isso utilizando uma medida relacionada a taxa de erro de classificação, denotada por $R(\mathbb{T})$, onde $\mathbb{T}$ é a árvore original sem poda.

Então, seja $r(t) = 1 - \max_k p(k|t)$ a estimativa do erro de classificação $R(t)$. Para o conjunto $\tilde{\mathbb{T}} = \{\tau_1, \ldots, \tau_L\}$ dos nós terminais de $\mathbb{T}$, podemos estimar a taxa do erro de classificação por $R(\mathbb{T}) = \sum_{\tau \in \tilde{\mathbb{T}}} R(\tau)P(\tau)$, onde $P(\tau)$ é a probabilidade de uma amostra estar em um nó τ. Assim, o algoritmo da poda é expresso por:

1. Construa uma árvore $\mathbb{T}$;
2. Calcule uma estimativa de $R(\tau)$ em cada nó terminal $\tau \in \mathbb{T}$;
3. Podar $\mathbb{T}$, de baixo para cima, i.e, dos nós terminais em direção a raíz, de modo que em cada estágio da poda, a estimativa de $R(\mathbb{T})$ é minimizada.

Dessa forma, a sub-árvore escolhida é aquela em que a quantidade $R(\mathbb{T})$ é mínima e que ainda mantém um poder preditivo satisfatório nos dados. [2, 4]

2.2 Métodos Ensemble, Bagging e Random Forest

Uma outra forma de reduzir a chance de sobre-ajuste é construir um conjunto (ou ensemble) de árvores de decisão. A ideia da utilização de métodos de aprendizado conjunto é construir um

modelo preditivo combinando modelos de base mais simples. Os métodos ensemble apresentados nesta seção, bagging e random forests, são métodos ensemble que para classificação, um conjunto (ou comitê) de árvores vota na classe predita.

O bootstrap aggregating, conhecido pelo acrônimo Bagging, é um método para gerar várias versões de um preditor e utilizá-las para obter um estimador agregado mais robusto. [3] Para árvores de classificação, o bagging criará B árvores utilizando B amostras bootstrap do conjunto de treinamento. Cada amostra bootstrap é obtida efetuando extrações com reposição do conjunto de treinamento. Dessa forma, são construídos B preditores $\hat{f}^1(x), \ldots, \hat{f}^B(x)$ em cada uma das B amostras bootstrap e a predição feita por cada árvore é interpretada como o voto daquela árvore em determinada classe. O estimador agregado classifica uma amostra $\mathbf{x}$ como pertencente à classe c_k se essa classe for a mais votada entre as B árvores construídas. Denotamos esse procedimento de classificação como regra do voto da maioria. [2, 8, 9]

Um modelo random forest é um classificador (ou regressor) construído como um ensemble de árvores de decisão, em que cada árvore é treinada utilizando um subconjunto distinto do conjunto de treinamento, geralmente via bagging, construída de acordo com o que foi apresentado na seção **2.1**. [2, 8, 10]. Os modelos random forests possuem uma vantagem em relação aos modelos bagging no sentindo de descorrelacionar as árvores. Além das árvores de decisão serem construídas com base em B amostras bootstrap, em cada divisão feita nas árvores, apenas uma amostra aleatória de m preditores é considerada para aquela divisão do total de p preditores. Assim, aplicando a seleção aleatória de características, o modelo considera sempre diferentes preditores, até eventualmente aqueles que são menos fortes, driblando o problema de alta correlação entre as árvores, presente no bagging, que considerará utilizar sempre os preditores mais fortes, construindo árvores bastante semelhantes entre si.

2.3 SMOTE

A técnica denominada Synthetic Minority Oversampling Technique, ou SMOTE, é um método de oversampling eficaz que tem sido amplamente adotado em diversas aplicações, principalmente ao abordar conjuntos de dados desbalanceados. O método consiste em gerar dados sintéticos aplicando interpolação linear entre um ponto da classe minoritária e um de seus k vizinhos mais próximos.

Ao aplicar o SMOTE a acurácia melhora à medida que a quantidade de amostras da classe minoritária n é maior. A razão para isso é que, para valores mais altos de n, os padrões dos k vizinhos mais próximos ficam ainda mais próximos uns dos outros. Isso significa que estamos lidando com uma região de valores de função de densidade semelhantes. Ir muito longe significa alcançar regiões com valores de densidade nitidamente diferentes e, portanto, padrões gerados com menos representatividade para a análise. [5–7]

Ao estudarmos o efeito do SMOTE no desempenho de um classificador, tem-se que a técnica proporciona um benefício devido ao impulso que oferece para padrões da classe minoritária. No entanto, esse benefício varia de um tipo de classificador para outro. O SMOTE foi apresentado brevemente já que será utilizada na seção prática a seguir.

3 Exemplo Prático

No contexto da análise de predição de falências de empresas, este artigo utiliza o conjunto de dados "Company Bankruptcy Prediction", obtido no site Kaggle. O objetivo principal é avaliar a eficácia do algoritmo random forest na tarefa de prever se uma empresa está em risco de falência ou não. Esta análise visa ampliar a compreensão da aplicação do algoritmo random forest em um cenário real de previsão de falências corporativas de Taiwan de 1999 a 2009.

Uma observação inicial fundamental é o desbalanceamento de classes em nosso conjunto de dados. Esse desequilíbrio pode resultar em um modelo que tende a classificar a maioria das empresas como não falidas, ignorando a detecção de falências, o que é obviamente indesejável. No contexto desta análise, enfrentamos a situação em que a classe "a empresa não faliu"(classe 0) é aproximadamente 30 vezes mais numerosa que a classe “empresa faliu” (classe 1):

```
Bankrupt?
0 6599
1 220
```

Aplicamos o modelo a fim de testar seu desempenho, mesmo com os dados desbalanceados. Com isso, separamos o conjunto de dados, onde 80% do conjunto é para treino e o restante para o conjunto de teste. Vale ressaltar que os hiperparâmetros do modelo são os padrões do scikit-learn, de modo que o único argumento fixado foi o *random_state*.

Destarte, os seguintes resultados foram obtidos:

Tabela 1: Resultados para o modelo *random forest*, sem aplicar *oversampling*.

Classe	Precisão	Recall	$f1$-score
0	0.97	1.00	0.98
1	0.82	0.18	0.29
Acurácia	0.97		

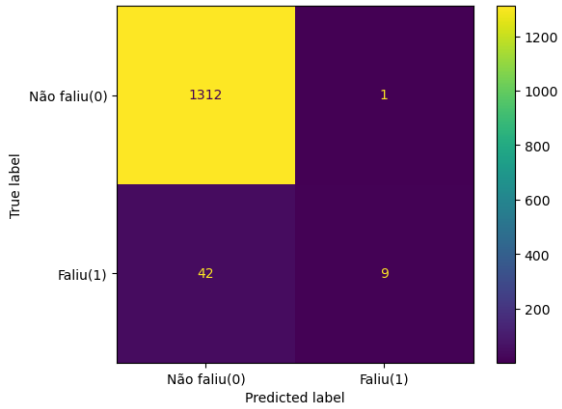

Figura 2: Matriz de Confusão para o modelo sem *oversamping*. Fonte: Autoria própria.

É evidente que este modelo, apesar de possuir uma acurácia de 97%, não fornece um bom desempenho na prática. Isso vem do fato de que a premissa deste modelo é classificar se uma empresa vai falir ou não, todavia, devido ao desbalanceamento de dados, o algoritmo tende a predizer a classe majoritária. Como resultado, observamos um número considerável de falsos negativos em comparação com os falsos positivos. Isso leva ao pior erro possível: o modelo prevê que a empresa não vai falir quando, na verdade, os dados reais mostram que ela deveria falir.

Para contornar tal problemática, é necessário implementar técnicas que possam reduzir os ruídos no conjunto de dados, sendo assim, reduzir o erro associado a tais problemas. Deste modo, uma das possíveis soluções é utilizar a técnica de *oversampling: SMOTE*.

Assim, por meio da técnica citada acima, obtivemos os seguintes resultados:

Tabela 2: Resultados para o modelo *random forest*, com *SMOTE*.

Classe	Precisão	Recall	$f1$-score
0	0.99	0.96	0.98
1	0.42	0.71	0.53
Acurácia	0.95		

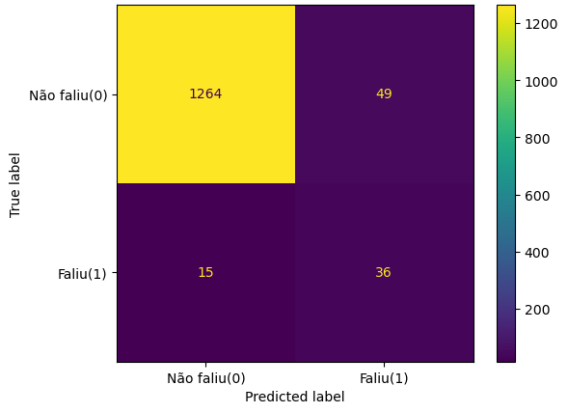

Figura 3: Matriz de Confusão para o modelo com *SMOTE*. Fonte: Autoria própria.

Como é possível observar, temos que o número de falsos negativos foi reduzido de forma considerável, apesar do aumento dos falsos positivos. Todavia, dado ao contexto que este modelo deve responder, temos que o erro falso positivo não seria grave comparado ao caso do falso negativo, tendo em vista que o investidor não estaria investindo em uma empresa com características de prosperidade, mas o mesmo poderia contornar esta situação buscando outra empresa.

4 Discussão e Conclusão

A partir da exploração das árvores de decisão até o modelo random forest, considera-se o bom desempenho desses algoritmos em aplicações práticas. Ao construir um ensemble a partir da base das árvores de decisão, a flexibilidade oferecida nos permite escolher entre técnicas como random

forest ou bagging, que podem resolver problemas complexos, por exemplo, o sobre-ajuste enfrentado pelas árvores de decisão individuais. No contexto abordado, optamos pelo algoritmo Random Forest, destacando especialmente sua capacidade de seleção aleatória de características.

Ao realizar um exemplo prático, observou-se que o modelo de random forest obteve o melhor desempenho, mesmo com o desbalanceamento presente no conjunto de dados. Por outro lado, a abordagem da random forest mostrou-se menos suscetível ao viés de prever que as empresas não estariam suscetíveis a falência, principalmente utilizando a técnica de Oversampling, o que contribuiu significativamente para sua eficácia no cenário analisado.

Este trabalho não apenas evidencia a eficácia dessa abordagem, mas também proporciona uma compreensão aprofundada da sua construção teórica, e a aplicação prática em cenários complexos de aprendizado de máquinas.

Referências

[1] Hastie et al. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. New York: Springer, 2009. ISBN: 978-0-387-84857-0.

[2] Izenman Alan J. **Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning**. New York: Springer, 2008. ISBN: 978-0-387-78189-1.

[3] Leo Breiman. "Bagging predictors". Em: **Machine Learning** 24 (1996), pp. 123–140. DOI: 10.1007/BF00058655. URL: https://doi.org/10.1007/BF00058655.

[4] Leo Breiman et al. "CART: Classification and Regression Trees". Em: 1984. URL: https://api.semanticscholar.org/CorpusID:59814698.

[5] Amir F. Atiya Dina Elreedy. "A Comprehensive Analysis of Synthetic Minority Oversampling Technique (SMOTE) for handling class imbalance". Em: **Information Sciences** 505 (2019), pp. 32–64. URL: https://www.sciencedirect.com/science/article/pii/S0020025519306838.

[6] Firuz Kamalov Dina Elreedy Amir F. Atiya. "A theoretical distribution analysis of synthetic minority oversampling technique (SMOTE) for imbalanced learning". Em: **Machine Learning** 112 (2023). DOI: https://doi.org/10.1007/s10994-022-06296-4.

[7] Firuz Kamalov Dina Elreedy Amir F. Atiya. "A theoretical distribution analysis of synthetic minority oversampling technique (SMOTE) for imbalanced learning". Em: **Machine Learning** 112 (2023). DOI: https://doi.org/10.1007/s10994-022-06296-4.

[8] Rafael Izbicki. **Aprendizado de máquina : uma abordagem estatística**. 1th. São Carlos, SP: Springer, 2020. ISBN: 978-65-00-02410-4.

[9] Gareth James et al. **An Introduction to Statistical Learning with Applications in Python**. 7th. New York, NY: Springer, 2013. ISBN: 978-1-4614-7137-0.

[10] Shai Shalev-Shwartz e Shai Ben-David. **Understanding Machine Learning: From Theory to Algorithms**. New York: Cambridge University Press, 2014. ISBN: 978-1-107-05713-5.

[11] "Understanding Random Forests: from theory to practice". Tese de Doutorado. University of Liège, 2014.