Aprendizado de Máquinas

Igor Patrício Michels

12 de junho de 2022

O presente trabalho se refere a disciplina de Aprendizado de Máquinas, do sétimo período da Graduação em Matemática Aplicada da FGV/EMAp.

1 Introdução

O objetivo do mesmo é utilizar alguns dos métodos vistos no curso para tentar classificar os clientes de hotéis quanto ao status da sua reserva, isso é, se esses clientes irão ou não cancelar a reserva. Os dados aqui utilizados foram encontrados no Kaggle [1].

2 Limpeza dos Dados

A base de dados escolhida era muito grande (com quase 120 mil observações e 36 colunas). Dessas colunas, algumas estavam quase totalmente nulas e algumas outras apresentavam valores faltantes. Para lidar com isso optei por remover a coluna que estava praticamente toda incompleta e remover as linhas que, após isso, tinham algum registro nulo.

Feito isso, acabei com um dataset semi balanceado, com cerca de 60% dos dados sendo de clientes que não cancelaram a reserva e os demais com clientes que acabaram cancelando.

3 Análise Exploratória

O primeiro passo para o trabalho foi olhar para os dados em busca de algum insight. Para tanto, olhamos para algumas métricas, sendo as mais interessantes o histórico dos clientes, suas requisições e o cruzamento de tempo de antecedência na reserva e valor da diária, disponíveis nas Figura 1, Figura 2 e Figura 3, respectivamente.

Podemos ver que todos esses gráficos nos dão informação sobre o comportamento do cliente como, aparentemente, ao ter mais de duas reservas efetivadas um cliente não cancela mais sua reserva. Além disso, quanto mais pedidos o cliente realizou (ou se realizou reserva de vagas de estacionamento), o cliente acaba sendo menos propício a cancelamentos. Por fim, notamos que, quanto mais próxima a data de chegada o cliente efetua a reserva, menos propenso ele está de desistir da reserva.

Tendo feito uma recapitulação da primeira parte, podemos focar na segunda parte do trabalho.

4 Tratamento dos Dados

O primeiro passo para a modelagem é tratar os dados para possibilitar que os mesmos possam ser modelados. No dataset havia algumas colunas categóricas, para tratar tais dados foi necessária a criação de variáveis dummies que indicam se o cliente faz parte daquela categoria ou não. Para dados como mês de chegada e dia de chegada, que podem ser interpretadas como categóricas, optei por manter numa mesma classe, uma vez que há uma ordenação intrínseca nesses dados, assim, para essas informações, a ordem cronológica que introduz a escala.

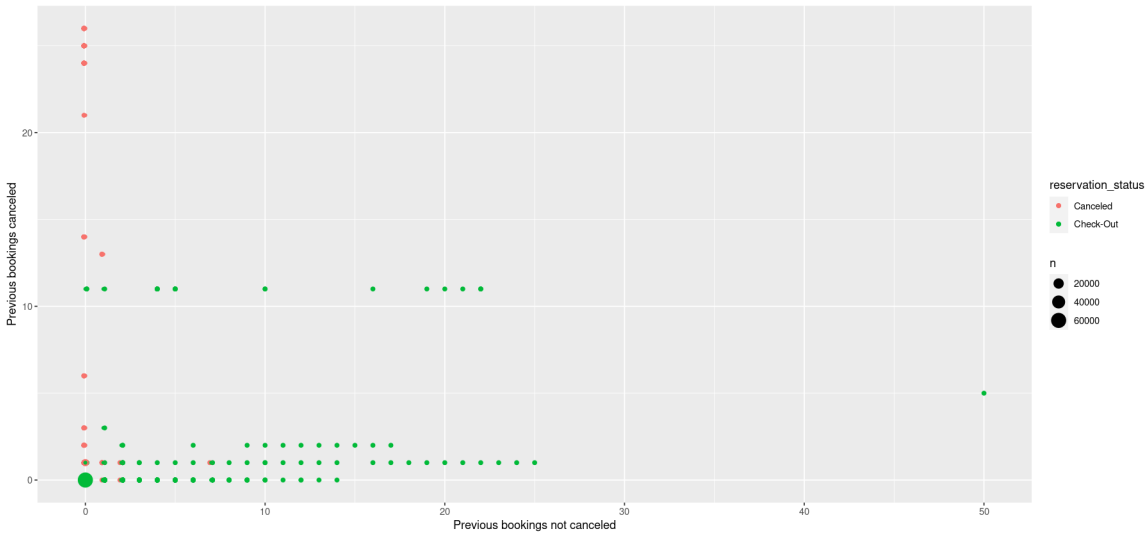

Figura 1: Histórico do Cliente

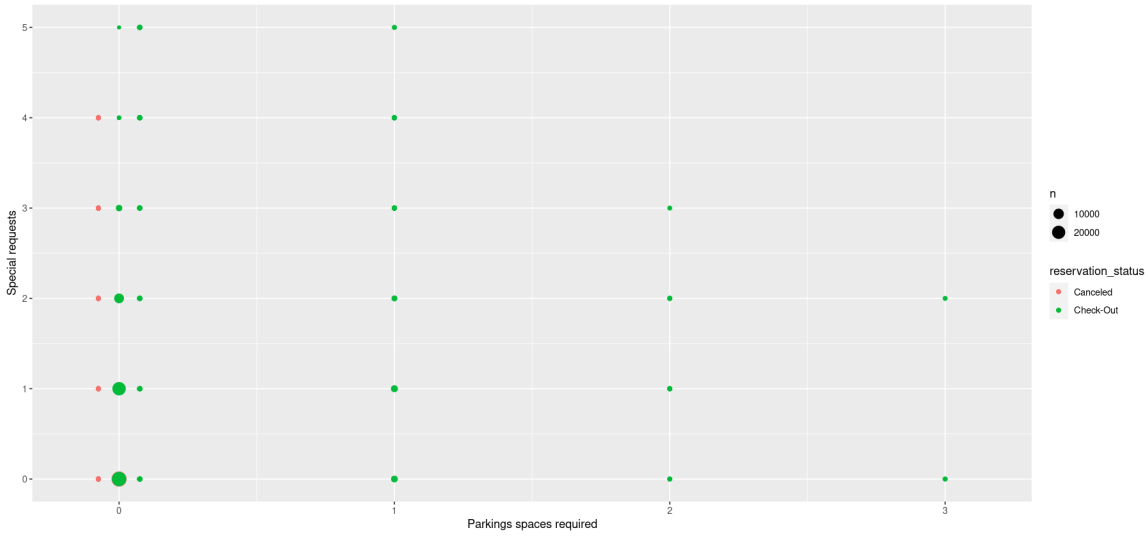

Figura 2: Requisições dos clientes

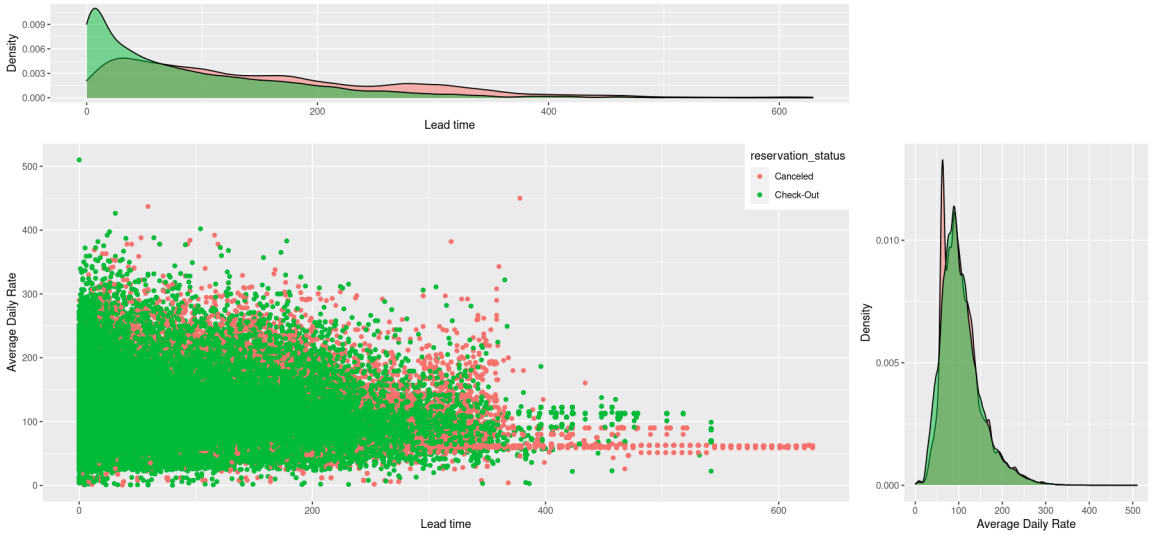

Figura 3: Tempo de Antecedência x Valor da Diária

Além dessa transformação, retirei dos dados informações como status da reserva e data do status da reserva, uma vez que a primeira é referente a categoria da variável que estamos modelando (is_canceled) e a segunda pode derivar o status diretamente por meio da data prevista para a chegada do cliente.

Com essas transformações o dataset está pronto para ser modelado. Assim, separei o dataset em dois conjuntos de dados, um com 70% dos dados, utilizado para treino, e o outro com 30%, utilizado para teste.

5 Modelagem

Para a modelagem do problema eu optei por testar o desempenho de quatro modelos:

- Regressão Logística;
- Árvore de Decisão;
- Random Forest e;
- Gradient Boosting.

Como muito trecho de código se repetiria, tanto na implementação dos modelos quanto no teste, criei funções que recebem o modelo e os dados necessários e retornam algumas análises ou os parâmetros ótimos para aquele modelo. Tais funções de métricas devolvem a matriz de confusão para o modelo e algumas métricas, enquanto que a outra realiza o plot da curva ROC.

Já a função que retorna os parâmetros ótimos testa os modelos propostos para um conjunto de parâmetros definidos pelo usuário e retorna o modelo definido pelos melhores parâmetros dentre os dados. Para essa avaliação é realizado o Cross-Validation com cinco grupos, assim, tomamos os dados de treino, separamos em cinco grupos, excluímos um e fitamos o modelo com os outros quatro grupos e calculamos o desempenho desse modelo no grupo excluído, variando esse grupo e retornando o modelo com o melhor desempenho médio.

5.1 Regressão Logística

O modelo de classificação mais simples que vimos. Para esse modelo realizo a escolha do modelo conforme citado acima variando o parâmetro de regularização, o qual testo para os valores no conjunto $\{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$. Após o Cross-Validation, chego que o melhor parâmetro de regularização é 0.1 que, ao fitar e aplicar para o conjunto de teste, dá uma precisão de 0.76 (para ambas as classes) e, consequentemente, uma acurácia de 0.76.

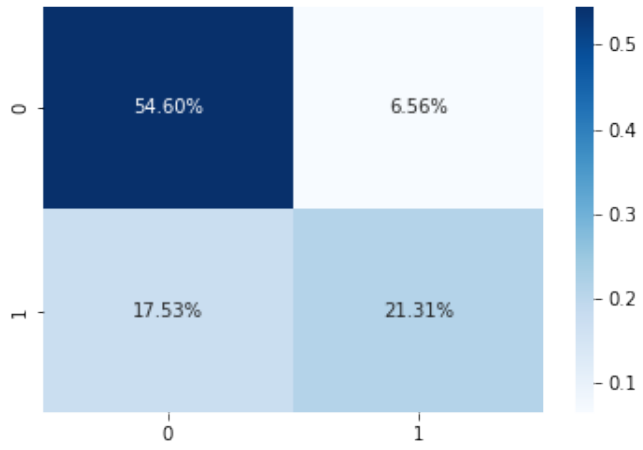

Figura 4: Matriz de Confusão para a Regressão Logística

A Figura 4 mostra a Matriz de Confusão para esse modelo. Note que ainda existe muitos clientes que acabaram cancelando mas que o modelo classifica como sendo um cliente que chegou a se hospedar nos hotéis (erro tipo II).

5.2 Árvore de Decisão

A Árvore de Decisão é outro modelo simples que vimos em aula e se assemelha muito ao processo de tomada de decisão do ser humano, isso é, olha para uma característica e, dada certa condição, avalia outra e assim por diante até chegar em uma conclusão. Dentro da Árvore de Decisão os parâmetros que variei foram a profundidade máxima, com o conjunto $\{1, 5, 10, 15, 20, 25\}$ e a quantidade mínima de amostras em um nó para haver uma nova divisão, estabelecendo o conjunto $\{50, 100, 200, 400\}$ para isso.

Após o processo de Cross-Validation os parâmetros definidos foram a profundidade máxima sendo 15 e a quantidade mínima para dividir um nó sendo 50. Realizando a fitagem do modelo com tais parâmetros, chego em uma acurácia de 0.86, com uma precisão de 0.88 para classificar hóspedes e de 0.83 para classificar os cancelamentos.

A Figura 5 mostra a Matriz de Confusão para esse modelo. Podemos ver que agora a quantidade de erro do tipo II diminuiu consideravelmente. Além disso, o erro do tipo I também diminuiu um pouco, de 6.56% para 6.44%.

5.3 Random Forest

A Random Forest pode ser vista como um conjunto de árvores de decisão, assim, a ideia desse classificar é treinar algumas árvores de decisão e utilizar tais árvores para a classificação do cliente. É bem similar a Árvore de Decisão, mas com uma vantagem de podermos ter uma visão mais probabilística das classes, possibilitando inferir as probabilidades de cada classe por meio da razão de árvores que classificaram o cliente em cada uma das classes. A desvantagem, no entanto, se dá no tempo de treinamento, pois agora ao invés de fitarmos uma árvores devemos realizar esse procedimento várias vezes.

Para esse modelo estou variando a quantidade de árvores no conjunto $\{10, 50, 100\}$ e a profundidade máxima e o número mínimo de amostrar para uma nova divisão do nó sendo dadas pelos mesmos conjuntos da Árvore de Decisão. Após o Cross-Validation, os parâmetros ótimos encontrados foram de 100 árvores, com profundidade máxima igual a 25 e a quantidade mínima de amostras para dividir um nó sendo igual a 50.

A Figura 6 mostra a Matriz de Confusão para esse modelo. Podemos ver que houve pouca variação entre a Random Forest e a Árvore de Decisão. A principal diferença é que esse modelo tem desempenho melhor

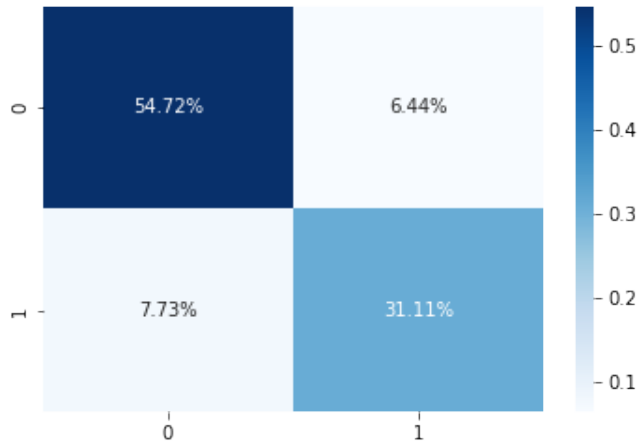

Figura 5: Matriz de Confusão para a Árvore de Decisão

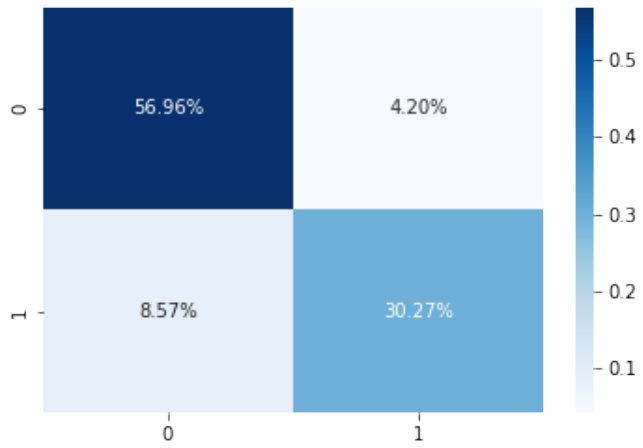

Figura 6: Matriz de Confusão para a Random Forest

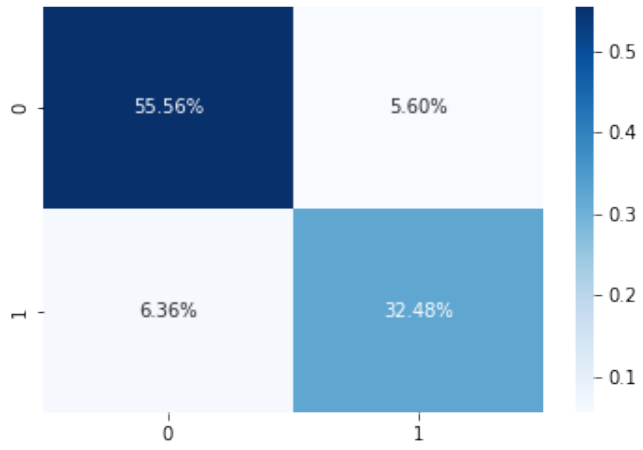

Figura 7: Matriz de Confusão para o Gradient Boosting

para classificar os cancelamentos (precisão de 0.88) e um pouco menor para classificar os hóspedes (0.87), com acurácia de 0.87.

5.4 Gradient Boosting

O Gradient Boosting é uma forma de melhorar o desempenho de algoritmos de regressão e classificação. Toma como base alguns modelos mais fracos e busca, passo a passo, melhorar sua performance, o que acaba sendo uma vantagem, pois os resultados devem ser melhores, entretanto o tempo de treino e validação acabam aumentando, devido a complexidade do modelo.

Aqui utilizei como base a Árvore de Decisão, assim, os parâmetros a serem verificados com o Cross-Validation foram o número de estimadores, no conjunto $\{1, 5, 10, 50\}$, a profundidade máxima, dentre $\{1, 5, 10\}$ e a taxa de aprendizado, variando entre $\{0.1, 0.5, 1, 5, 10\}$. Os valores ótimos foram de 50 estimadores, com profundidade máxima igual a 10 e taxa de aprendizado igual a 0.5.

A Figura 7 mostra a Matriz de Confusão para esse modelo. A acurácia desse modelo foi a melhor, chegando a 0.88, com precisão de 0.90 para classificar hóspedes e de 0.85 para predizer cancelamentos. Essa melhora era esperada, tendo em vista que se trata de um modelo que melhora a Random Forest.

5.5 Comentários Gerais

Nos modelos desenvolvidos acima, muitos dos parâmetros retornados pelo Cross-Validation ficaram nos limites dos conjuntos dados como entrada, o que significa que, se abrisse mais opções para os parâmetros, poderíamos ter outro parâmetro que gerasse um melhor desempenho na validação. Ponderei aumentar o conjunto de parâmetros a serem utilizados para essa validação, entretanto, vendo que os resultados faziam sentido com o que vimos em aula (os modelos mais complexos estão performando melhor) e o fato dos resultados já estarem muito bons, vejo que aumentar o conjunto de parâmetros para serem validados acarretaria numa pequena melhora de desempenho e num alto aumento de run-time, como veremos a seguir.

5.5.1 Run-time e Performance

A Tabela 1 mostra as informações gerais sobre os tempos de execução e os resultados gerais de cada modelo, agrupando os dados citados acima e possibilitando a comparação entre os modelos.

Já na Figura 8 temos as Curvas ROCs para os modelos analisados acima. Nesse gráfico podemos ver que o Gradient Boosting apresenta o melhor resultado, com a Random Forest muito próxima a ele.

Modelo	Estimação dos Parâmetros	Fitting	Precisão C_0	Precisão C_1	Acurácia
Regressão Logística	3 min 14 s	6.36 s	0.76	0.76	0.76
Árvore de Decisão	47.5 s	665 ms	0.88	0.83	0.86
Random Forest	12 min 47 s	8.05 s	0.87	0.88	0.87
Gradient Boosting	19 min 43 s	28.8 s	0.90	0.85	0.88

Tabela 1: Tempos e Precisões de cada modelo.
C_0 representa a classe de clientes que se hospedou e C_1 a classe de clientes que cancelaram a reserva.

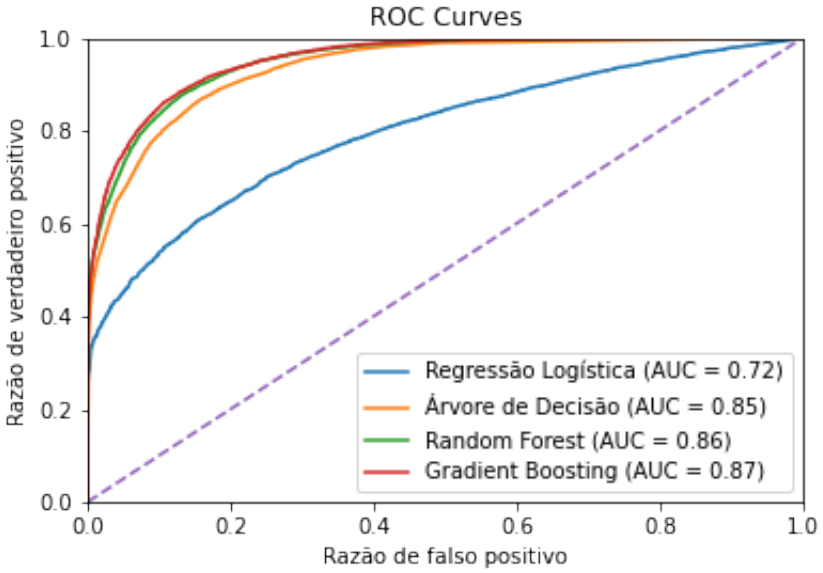

Figura 8: Curvas ROC

Entretanto, apesar de tanto a Tabela 1 quanto a Figura 8 mostrarem que o Gradient Boosting ser o modelo com o melhor desempenho, seguido pela Random Forest, podemos ver que a mesma Tabela 1 nos mostra que esses dois modelos tem um run-time muito alto. Assim, cruzando performance com run-time, podemos ver que o modelo de Árvore de Decisão acaba sendo muito bom, uma vez que, quando comparado ao Gradient Boosting, tem acurácia um pouco menor, 2% menor, e um run-time 25 vezes menor. Já ao comparar com a Random Forest, a acurácia da Árvore de Decisão é 1% menor e seu run-time é 16 vezes menor.

6 Conclusão

Nesse trabalho utilizamos alguns modelos estatísticos para tentar classificar reservas de hotéis entre os clientes que de fato se hospedariam e os que viriam a cancelar a reserva, além de reaver (e aplicar) o Cross-Validation.

Pudemos ver que os modelos baseados em árvores se saíram melhor que o modelo de Regressão Logística, apesar de, em alguns casos, terem um run-time elevado, assim, olhando para tempo de execução e cruzando com o desempenho, o modelo de Árvore de Decisão se mostra bem interessante ao passo que a quantidade de dados cresce.

7 Reprodutibilidade

Ao contrário da primeira parte, essa segunda parte do trabalho foi realizada em Python, em virtude da facilidade de implementação dos modelos e do Cross-Validation por meio da biblioteca sklearn [2]. Além disso, os códigos utilizados estão disponíveis em Jupyter Notebook no repositório do trabalho [3].

Referências

[1] Hotel Booking. *Kaggle*. `https://www.kaggle.com/datasets/mojtaba142/hotel-booking`.

[2] Scikit-Learn. *Scikit-Learn*. `https://scikit-learn.org/stable/`.

[3] Hotel Booking. *Igor Michels*. `https://github.com/IgorMichels/MachineLearn`