Sumário

Introdução 2

Resumo do relatório 2

Dataset 3

Definição da estratégia de modelagem 3

Análise exploratória dos dados 4

- Valores ausentes 4
- Correlação entre as variáveis independentes 5
- Correlação com a variável dependente 7
  - PassengerId 7
  - Name 7
  - Survived 8
  - Pclass 9
  - Sex 9
  - Embarked 9
  - SibSp e Parch 10
  - Fare 10
  - Age 11

Definindo os modelos baselines 11

Tratamento de valores ausentes 12

Identificação dos outliers 13

Feature engineering 15

Otimização dos hiper parâmetros do modelo vencedor 17

Extraindo os artefatos do modelo 17

Reporte final das métricas 17

Conclusão 19

Introdução

O objetivo deste relatório é demonstrar o processo de modelagem conduzido para resolver o desafio do Titatic, disponível em https://www.kaggle.com/c/titanic. Este é um problema de classificação binária, no qual pede-se para prever quais passageiros sobreviveram ao naufrágio do Titanic.

O modelo foi desenvolvido usando python + jupyter notebook e pode ser acessado no seguinte link: https://anaconda.org/weslleymoura/titanic_final/notebook

Resumo do relatório

No início do projeto foi definida a estratégia de amostragem dos dados (validação cruzada – kfold validation) e a métrica principal que seria usada para tomar as decisões de modelagem (balanced accuracy). O trabalho inicia-se com uma análise exploratória dos dados, revelando informações sobre valores ausentes, correlação entre variáveis independentes e correlação entre a variável dependente e cada variável preditora. Nesta fase, foram apresentados gráficos de correlação e gráficos de barras empilhados para facilitar o entendimento dos resultados. Adicionalmente, foi necessário usar o recurso de binning para que fosse possível criar os gráficos para algumas variáveis numéricas.

O objetivo da fase de análise exploratória era simplesmente exibir as características dos dados e criar alguns modelos baseline por meio de diferentes algoritmos (decision tree, gradient boosting, random forest, adaboost e logístic regression). Estes modelos competiram entre si ao longo de todo o projeto.

Na fase seguinte, os dados começaram a ser tratados. Primeiramente foi feito o preenchimento dos valores ausentes de algumas variáveis, utilizando médias de grupos. Nesta etapa foi criada a variável *título,* que acabou sendo muito importante para o modelo final. Em seguida, os outliers foram identificados por meio dos métodos z score e IQR. Os modelos baseline foram recriados, agora usando os últimos tratamentos realizados, a fim de exibir o impacto que o tratamento dos dados causou em cada algoritmo.

Dando continuidade à modelagem, foi iniciada a fase de feature engineering, na qual as principais variáveis do dataset sofreram uma série de transformações, como normalização, padronização, box-cox, one hot encoding, binning, dentre outras. A ideia era verificar quais algoritmos eram mais sensíveis à escala das variáveis. Novamente, todos os algoritmos baseline foram expostos às variáveis transformadas para análise de possíveis melhorias. Ao final desta fase foi escolhido o algoritmo mais promissor para o modelo final (o escolhido foi o gradient boosting).

Na próxima etapa, foi exibido o passo a passo para otimizar os hiper parâmetros do algoritmo vencedor (também com o objetivo de reduzir overfitting). Foram utilizados Recursive Feature Elimination (RFE) e grid search para definição do melhor conjunto de variáveis e parâmetros para o modelo final, respectivamente. Durante todo o projeto foram criadas ao redor de 50 variáveis, porém, no modelo final, apenas três variáveis concentraram todo o poder preditivo necessário para alcançarmos 82% de balanced accuracy na validação cruzada (as variáveis selecionadas foram: pclass; a variável fare transformada por meio de um standard scaler; e o título que estava incorporado no nome dos passageiros).

Finalizando o relatório, são apresentadas as conclusões e pontos de melhoria para continuação do trabalho. Foi criado um pipeline para submeter os dados de teste ao Kaggle. O modelo teve

desempenho de 77% de accuracy nos dados de teste (o que indica pouco overfitting quando comparamos os resultados da base de treino).

Dataset

Antes de tomar qualquer decisão, vamos entender o significado de cada atributo do dataset.

Variable	**Definition**	**Key**
PassengerId	Id of the passenger	
Name	Passenger Name	
Survival	Survival	0 = No, 1 = Yes
Pclass	Ticket class. A proxy for socio-economic status (SES) 1st = Upper 2nd = Middle 3rd = Lower	1 = 1st, 2 = 2nd, 3 = 3rd
Sex	Sex	male, female
Age	Age in years. Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5	
Sibsp	# of siblings / spouses aboard the Titanic. The dataset defines family relations in this way... Sibling = brother, sister, stepbrother, stepsister Spouse = husband, wife (mistresses and fiancés were ignored)	
Parch	# of parents / children aboard the Titanic. The dataset defines family relations in this way... Parent = mother, father Child = daughter, son, stepdaughter, stepson Some children travelled only with a nanny, therefore parch=0 for them.	
Ticket	Ticket number	
Fare	Passenger fare	
Cabin	Cabin number	
Embarked	Port of Embarkation	C = Cherbourg Q = Queenstown S = Southampton

Definição da estratégia de modelagem

Iniciei a modelagem do problema definido a estratégia que seria seguida durante todo o trabalho. As decisões tomadas nesta etapa (tanto explicitamente no notebook quanto implicitamente devido à natureza do problema), foram:

- A métrica utilizada para comparação e otimização dos modelos foi a **balanced accuracy**. Resolvi usar esta métrica porque o dataset não está totalmente balanceado e eu gostaria de criar um classificar capaz de acertar as duas classes (sobreviventes e não sobreviventes), ao invés de priorizar uma delas.
- Resolvi usar validação cruzada (k-fold cross-validation) porque eu não quis assumir o risco de criar uma única amostra de teste que poderia, por ventura, estar enviesada. O segundo motivo é que o próprio site do Kaggle possui um dataset de teste, então achei por melhor trabalhar apenas com cross-validation e testar o modelo somente no final, já usando os dados de teste do próprio Kaggle.
- Este é um problema supervisionado de classificação binária, logo, apenas algoritmos desta classe poderiam ser usados. Resolvi testar os seguintes algoritmos como possíveis classificadores:

- Decision tree: um modelo de árvore de decisão simples, que poderia capturar uma distribuição não linear dos dados
- Random Forest (bagging): este modelo foi incluído porque eu gostaria de verificar o poder preditivo de um ensemble neste conjunto de treino.
- Adaboost (boosting): faz contraponto ao random forest, no sentido de que ambos são modelos ensembles baseados em árvores de decisão, porém Adaboost consegue ”forçar” as observações que foram classificadas incorretamente e participarem de outros classificadores (isso é feito por meio de atribuição de pesos). Intuitivamente, esta abordagem usa árvores sequenciais, na qual a próxima árvore tenta corrigir os erros da árvore anterior. Quero analisar se esta característica do classificador traz mais poder preditivo para este conjunto de dados.
- GradientBoot: Inseri este algoritmo como desafiante do Adaboost. Gradient Boosting segue a mesma abordagem dos algoritmos baseados em boosting (como Adaboost), com a diferença de que GradientBoost tenta otimizar a função de custo de cada classificador. Ou seja, a ideia ainda é construir árvores de decisão sequenciais, porém com a premissa de que a árvore atual irá otimizar a função de custo da árvore anterior.
- Logistic Regression: a simplicidade da regressão logística poderia ser usada como fator decisivo para escolha deste modelo. Resolvi incluí-lo na lista de modelos baseline porque, caso seu desempenho seja similar aos demais algoritmos, sua facilidade de interpretação poderia garantir sua escolha como modelo final.

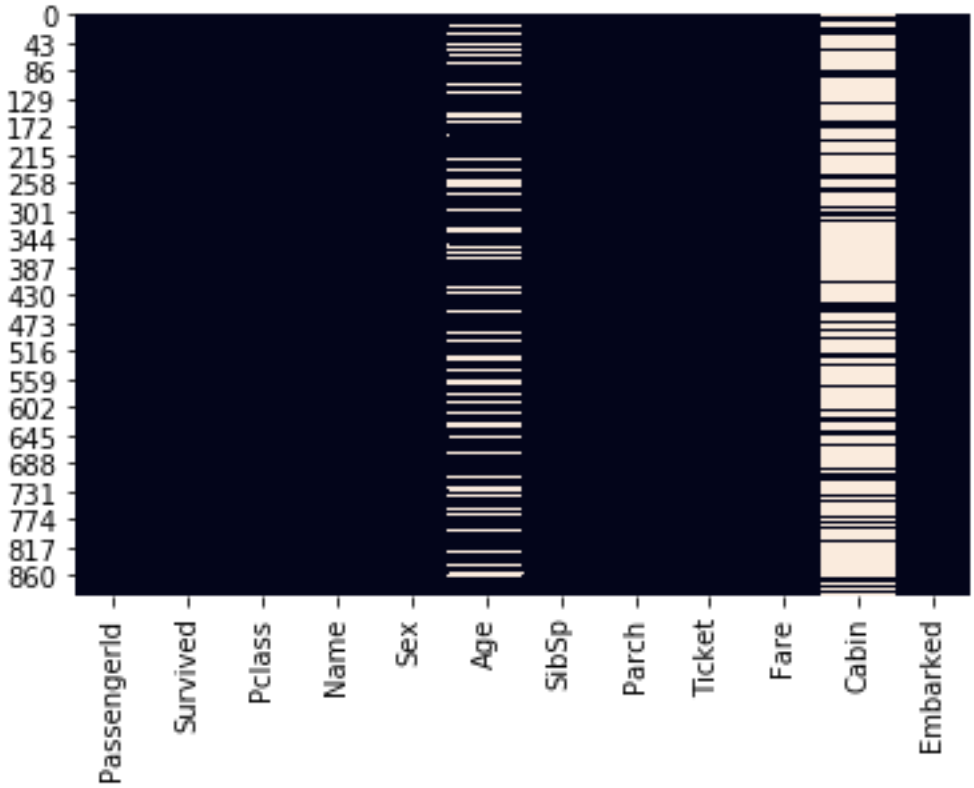
Diversas outras decisões “locais” foram tomadas durante todo o projeto, por exemplo, criação de novas variáveis, transformações, dentre outros. Estas decisões serão detalhadas no decorrer do relatório. Nesta seção foram detalhadas apenas as decisões “gerais” que impactam o projeto como um todo.

Análise exploratória dos dados

O objetivo desta fase é detalhar as características do conjunto de dados de treino.

Valores ausentes

O missing plot abaixo fornece uma ideia das variáveis que possuem valores ausentes, assim como a distribuição dos valores ausentes ao longo das observações. Inicialmente, “Cabin” é a variável que possui o maior número de valores ausentes, seguido da variável Age.



Embora o missing plot seja muito útil, também é interessante agrupar os dados do dataset para ter uma visão mais detalhada:

ausente	
Age	True
Cabin	True
Embarked	True

Na tabela acima é possível identificar valores ausentes em três variáveis: Age, Cabin e Embarked. Estes valores ausentes serão tratados no decorrer do projeto, por ora, foram apenas identificados.

Correlação entre as variáveis independentes

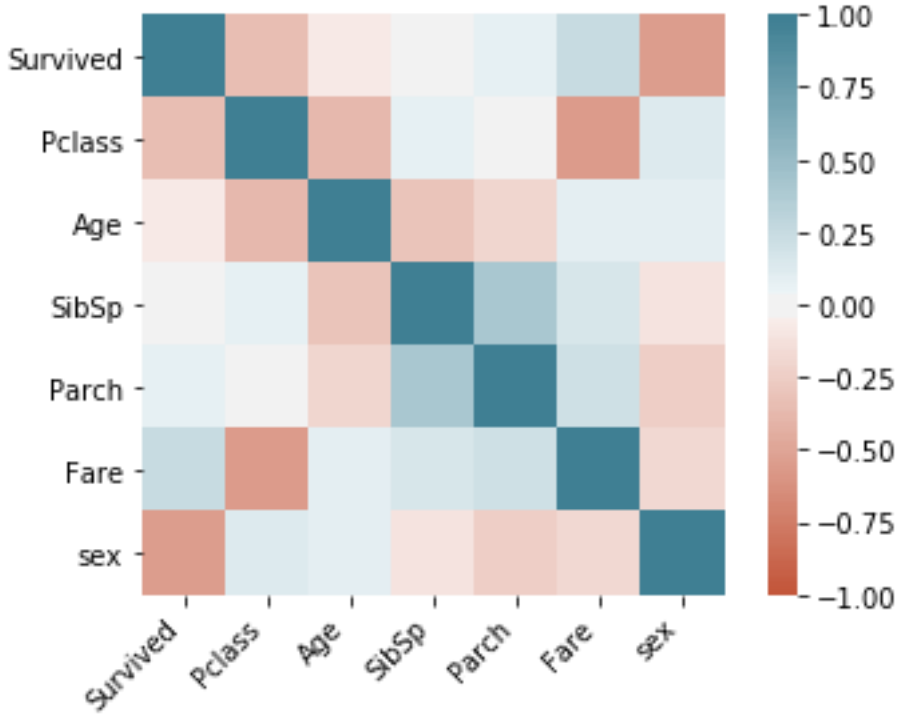
Analisar a correlação entre as variáveis independentes é importante para:

- Evitar o problema conhecido como colinearidade entre as variáveis independentes. Em teoria, variáveis correlacionadas representam a mesma informação e não precisaríamos usá-las em conjunto (já que sempre buscamos um modelo com parcimônia, simples, de fácil interpretação e bom poder preditivo).
- Criar insights sobre variáveis que se explicam. Isso pode ajudar a preencher valores ausentes, por exemplo.

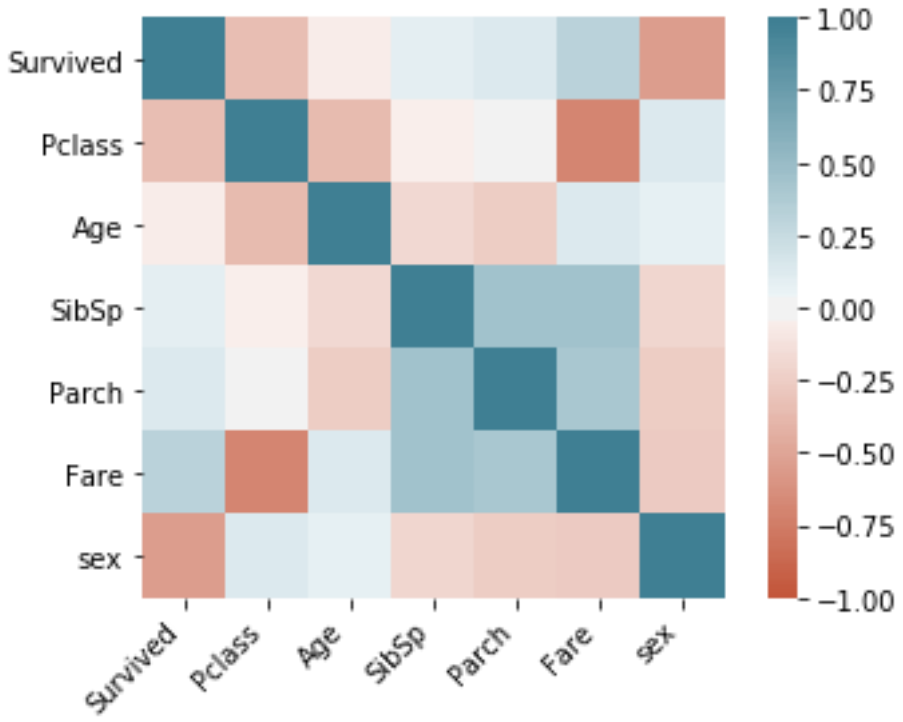
O gráfico de correlação abaixo fornece alguns insights sobre o dataset, por exemplo: as variáveis Pclass, Sex e Fare parecem bons preditores, pois possuem um nível de correlação maior com a variável target (Survived). Correlação positiva para Fare (quanto maior o valor pago pelo ticket,

mais chances de sobrevivência) e correlação negativa para Sex e Pclass (quanto menor o valor da variável, mais chances de sobrevivência). Lembrando que Sex e Pclass são variáveis qualitativas, portanto é preciso cuidado quanto à interpretação desta correlação:

- Sex: feminino = 0 e masculino = 1. Portanto, mulheres têm mais chances de sobrevivência do que homens.
- Pclass: 1 = Primeira classe, 2 = Segunda classe e 3 = Terceira classe. Portanto, quanto mais próximo da primeira classe (pessoas com maior poder aquisitivo), mais chances de sobrevivência.



O gráfico anterior assumiu o índice de correlação de Pearson (que assume uma correlação linear entre as variáveis). Nem sempre as variáveis possuem uma correlação linear, nestes casos, pode ser interessante analisar a correlação de Spearman:



O índice de correlação de Spearman confirmou as descobertas feitas sobre as variáveis com maior poder preditivo e, adicionalmente, revelou que existe mais correlação entre as variáveis Fare vs SibSp e Parch do que imaginávamos.

Correlação com a variável dependente

A seguir serão feitas análises bivariadas entre cada variável independente com a variável dependente. Algumas variáveis são podem ser utilizadas no modelo e, para estas variáveis, serão detalhados os motivos de sua exclusão. Outras variáveis necessitam de tratamentos antes de serem analisadas.

PassengerId

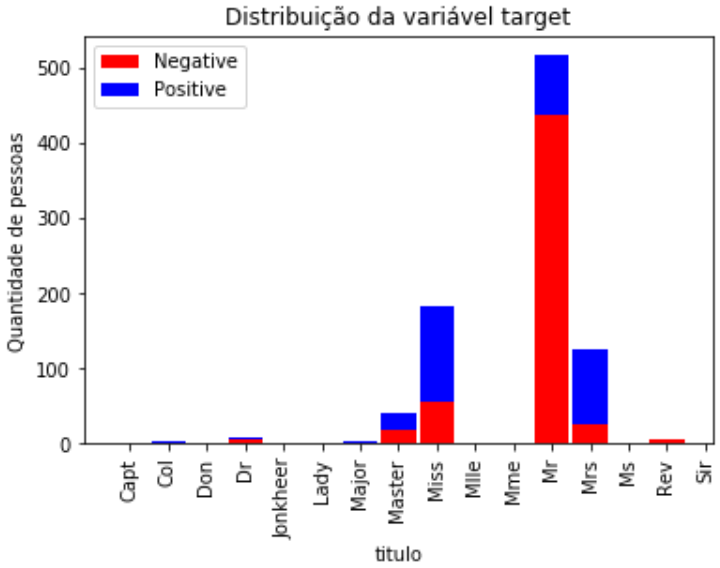
PassengerId é um identificador de cada pessoa (variável numérica e sequencial). Essa variável é única para cada observação, não trás nenhuma informação em comum ou adicional às observações e, portanto, não faz sentido usá-la como variável independente já que nenhuma informação contribuiria para a generalização de um modelo (pelo contrário, traria overfit para o conjunto de treino).

Name

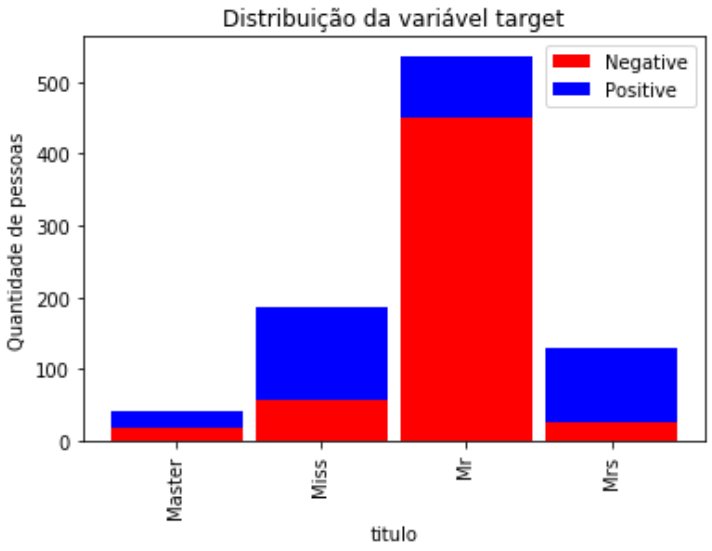
A variável name, em sua essência, não traria informações relevantes para o modelo. No entanto, este atributo contém mais informações do que apenas o nome do passageiro. Por meio do prefixo do nome (exemplo: Mr., Miss, etc) é possível inferir o gênero, estado civil ou até mesmo alguma informação sobre nível de formação da pessoa.

Notei também que alguns nomes possuem um segundo nome entre parênteses. Isso pode ter algum significado não capturado nas demais variáveis, porém, sem ter um conhecimento de domínio mais profundo, é muito arriscado inserir um tipo de informação deste tipo no modelo.

Em resumo, resolvi extrair apenas o título do nome dos passageiros. O gráfico abaixo mostra a distribuição da variável target em cada título de passageiro.



Claramente esta variável consegue separar sobreviventes de não sobreviventes. No entanto, alguns títulos parecem duplicados ou possuem poucas observações. Por este motivo, resolvi agrupar alguns valores da seguinte forma:



As conclusões são as seguintes: Mr possuem menos chances de sobrevivência, enquanto Mrs possuem mais chances de sobrevivência. O poder preditivo desta variável será posto à prova no decorrer do projeto.

Survived

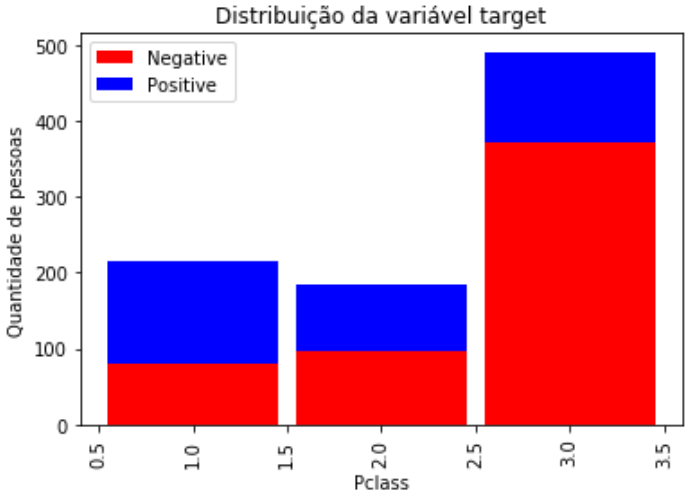
Esta é a variável dependente. Esta é uma variável do tipo nominal (binária) e não possui valores nulos. O dataset inteiro possui 891 casos, dos quais 38.38% pertencem à classe positiva (passageiro sobreviveu). Existe um pequeno desbalanceamento dos dados, mas não creio que o suficiente para pensar em undersampling ou oversampling nesse primeiro momento.

Outra informação relevante para levar em consideração é que, se criamos um modelo que prevê 100% dos passageiros como não sobreviventes, este modelo nos daria uma acurácia de 62%. É

certo que este não é o modelo ideal, mas vale a pena levar em considerar este número de 62% como baseline no decorrer da modelagem.

Pclass

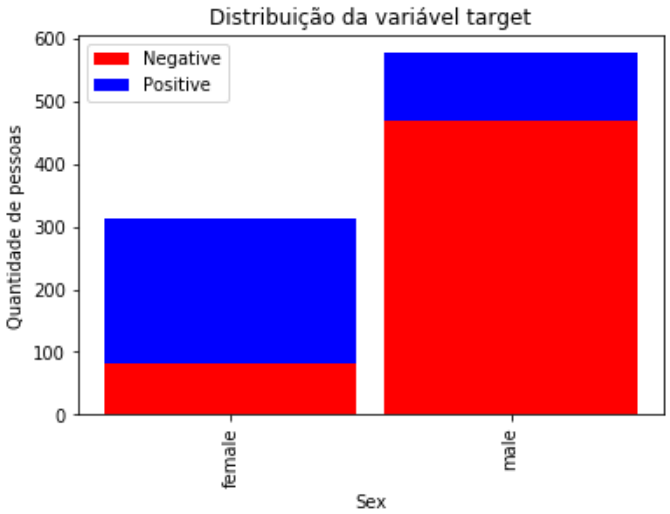
Esta é uma variável ordinal que exibe a classe econômica associada ao ticket. Não encontrei nenhum valor ausente.



Como já observado no gráfico de correlação anteriormente, pessoas com maior poder aquisitivo possuem mais chances de sobrevivência.

Sex

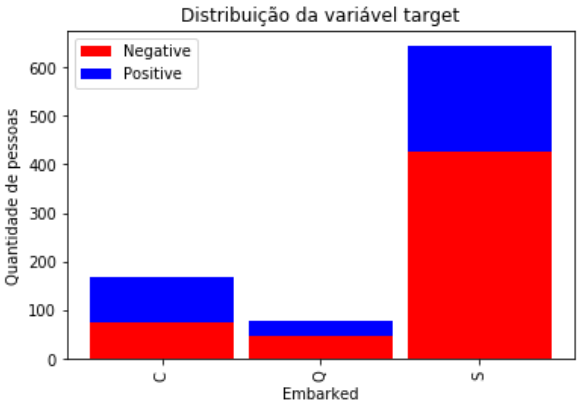
Esta é uma variável binária e classifica o gênero do passageiro como feminino (0) ou masculino (1).



Mulheres possuem mais chances de sobrevivência do que homens.

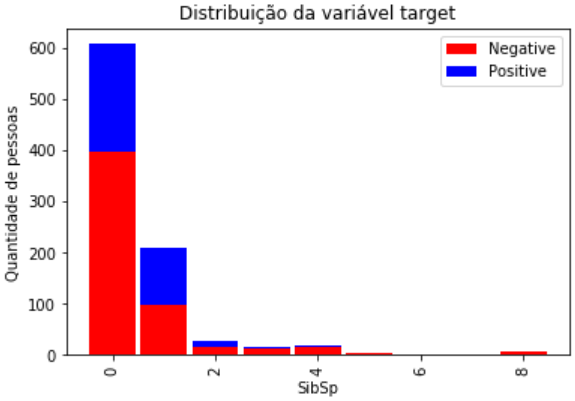
Embarked

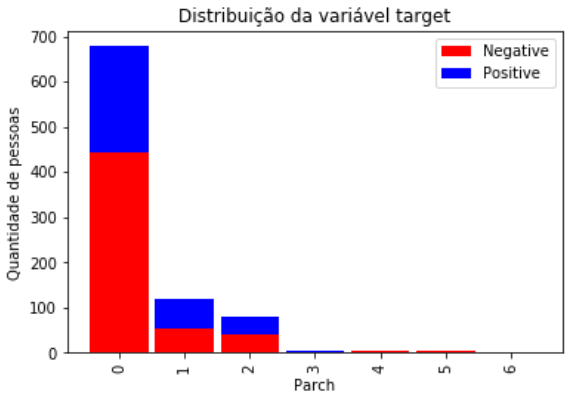
Esta é uma variável nominal usada para especificar o porto de embarque do passageiro. Me parece que passageiros que embarcaram no porto C possuem um pouco mais de chances de sobrevivência.



SibSp e Parch

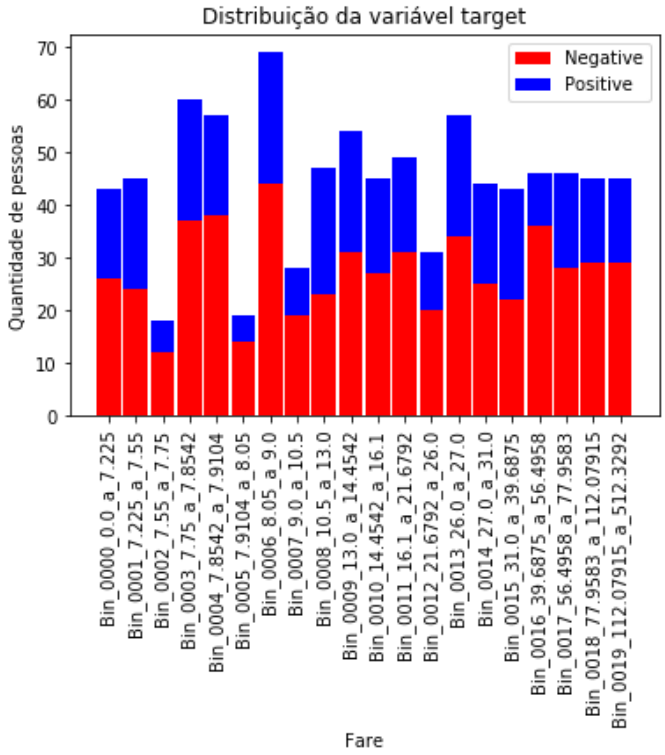
Estas duas variáveis referem-se à quantidade de familiares a bordo de cada passageiro. Mais detalhes sobre os tipos de familiares que se enquadram em cada variável estão disponíveis na seção “dataset” deste documento. Estas variáveis estão correlacionadas entre si e possuem uma leve correlação positiva com a variável target.





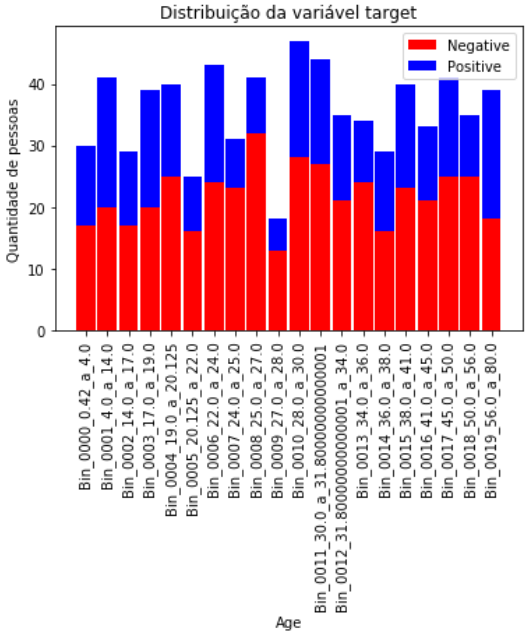
Fare

Valor pago pelo ticket. Esta é uma variável importante, já que possui uma correlação um pouco maior coma variável target. Para conseguir visualizar o histograma desta variável foi preciso criar bins, já que se trata de uma variável contínua.



Age

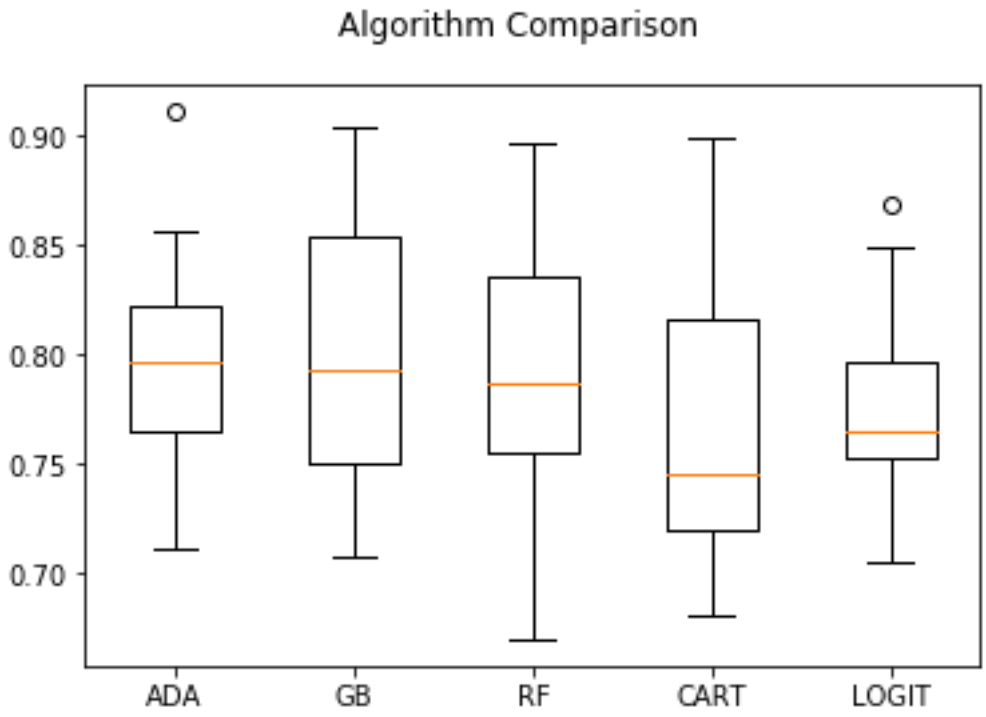
Outra variável continua onde foram criados bins para visualização do histograma.



Definindo os modelos baselines

Antes de realizar os tratamentos dos dados, resolvi criar os modelos baseline para futura comparação. A ideia é conseguir analisar, ao longo do projeto, como as decisões de modelagem estão interferindo nos resultados dos modelos. Na seção “estratégia de modelagem”, presente no início deste documento, já foi explicado os motivos para a escolha dos algoritmos testados. O gráfico seguinte exibe os resultados dos modelos (reportados pela balanced accuracy), com

tratamentos mínimos na base de treino (de fato, o único tratamento feito foi o preenchimento de valores ausentes por médias).




```
ADA: 0.796048 (0.045971) GB: 0.801082 (0.058313) RF: 0.790875
(0.058868) CART: 0.769072 (0.065699) LOGIT: 0.777280 (0.042782)
```

Este é o resultado de um cross-validation, por este motivo foi possível criar um box-plot para cada algoritmo. O box-plot mostra como a métrica de avaliação se comportou ao longo das validações feitas. A princípio, e possível tomar nota de alguns pontos interessantes:

- O algoritmo Gradient Boosting (GB resultou na melhor balanced accuracy.
- O algoritmo mais instável foi a árvore de decisão (CART), já que apresenta o maior desvio padrão entre todos os algoritmos (indicante instabilidade ao longo das validações)
- AdaBoost e Logistic regression, apresentaram resultados muito bons (outliers) em algumas amostras de validação.
- RandomForest parece tão bom quanto o Gradient Boosting, porém, em determinadas amostras, o valor de balanced accuracy foi o menor entre todos os algoritmos.

Por enquanto me parece que todos os algoritmos possuem defeitos, mas minha primeira impressão é que o Gradient Boosting é a opção mais promissora.

Tratamento de valores ausentes

Os valores ausentes da variável Age foram tratados por meio da média de grupos. Foram utilizadas duas variáveis para agrupar os dados: título e pclass. Utilizei estas duas variáveis porque estão mais correlacionadas com a idade.

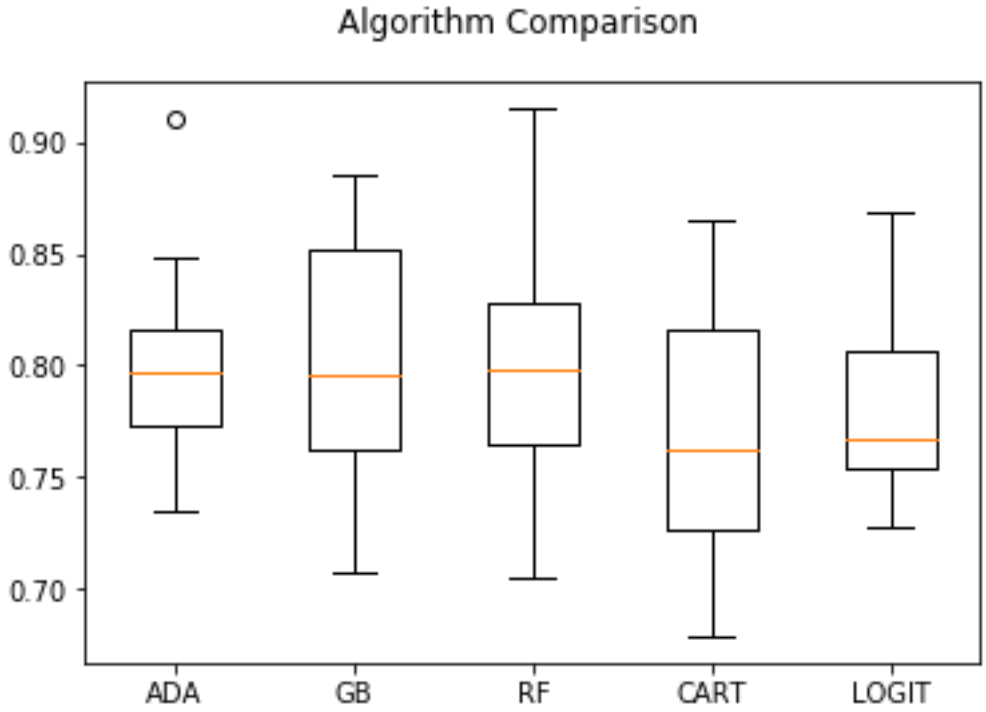
Fiz um comparativo de ganho na base de treino, aplicando a lógica da média dos grupos versus a média geral e o resultado foi o seguinte;

```
Usando a média geral, o erro médio da coluna idade é: 11.322944471906405 anos. Já aplicando o média dos grupos, o erro médio da coluna idade é de 8.648221288515407 anos
```

Este teste foi feito apenas na base de treino para me dar uma ideia do ganho entre as duas abordagens de preenchimento de valores ausentes. Em teoria, quanto mais variáveis fossem adicionadas aos grupos, mais próximo da idade real do passageiro seria o resultado final. No entanto, ao adicionar mais variáveis também estaria condicionando a validade dos resultados apenas aos dados de treino (overfitting).

Já a coluna Embarked possuía apenas dois valores ausentes, que foram preenchidos pelo valor de Embarked que mais se repetia na base.

Seguindo com a estratégia de analisar o impacto deste tratamento nos modelos finais, gerei o gráfico a seguir para analisar a balances accuracy de cada modelo após o preenchimento dos valores ausentes.



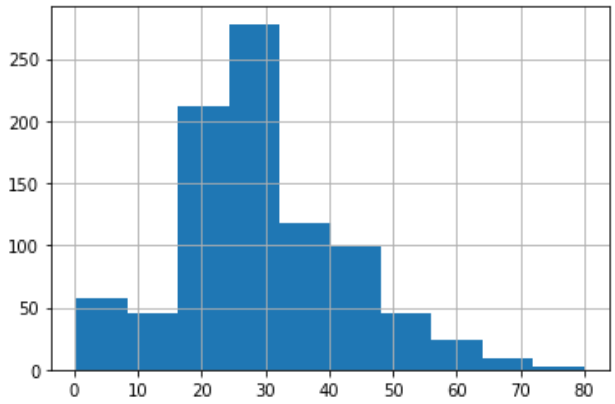

```
ADA: 0.799094 (0.039965) GB: 0.803966 (0.053188) RF: 0.796658 (0.053432) CART: 0.766982 (0.056130) LOGIT: 0.780952 (0.040732)
```

Embora pouco, nota-se melhorias nos algoritmos. Inclusive no modelo Gradient Boosting, o qual eu havia mencionado anteriormente como mais promissor.

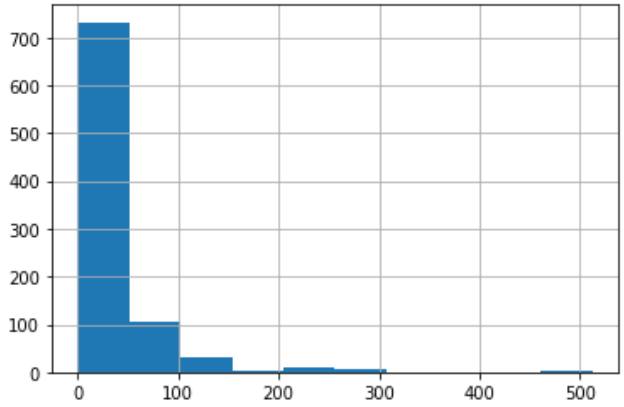
Identificação dos outliers

Ao invés de suavizar os outliers, resolvi apenas identificá-los nas variáveis Age e Fare. Mais à frente, durante o treino dos modelos, esta identificação será transmitida aos algoritmos como uma feature. Desta forma, os algoritmos terão a oportunidade de tratar os outliers de forma adequada (seja por meio de novos nós, no caso das árvores de decisão, ou por meio de novos interceptors).

Para identificação dos outlier na variável Age, utilizei o método z score. Este método assume que a variável segue uma distribuição normal e confia na média e desvio padrão dos dados para identificar outliers. A variável Age não parece muito distante de uma distribuição normal:



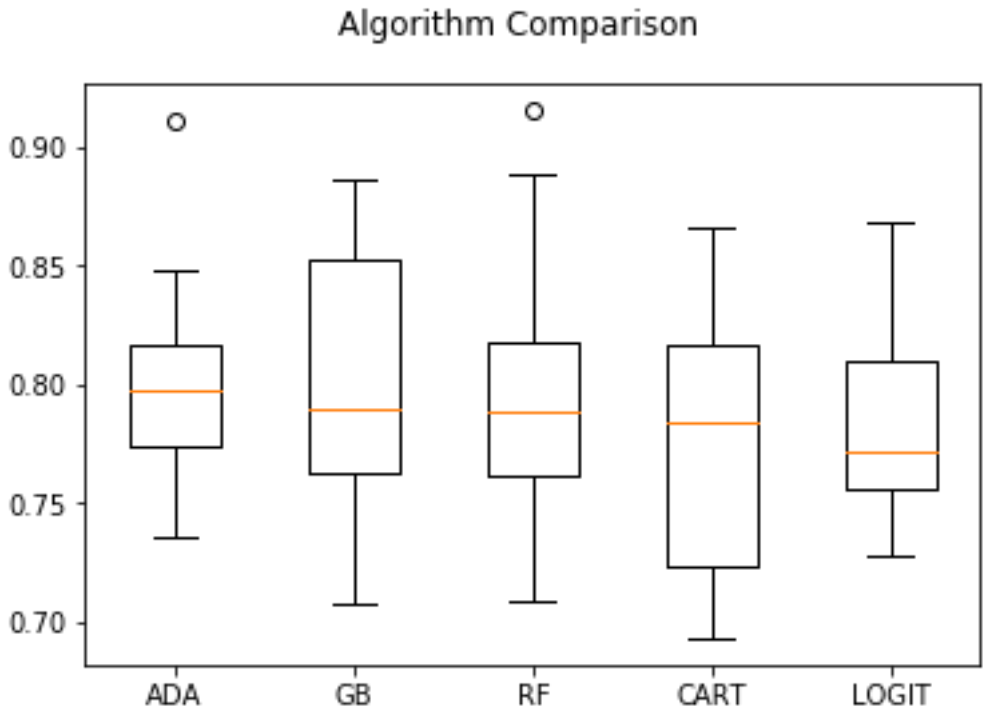
Já a variável Fare, definitivamente, não segue uma distribuição normal.



Por este motivo, ao invés de usar o método z score, foi aplicada a regra do box plot (IQR interquartile range). Esta regra consiste nas seguintes etapas:

- Ordenar os dados de forma crescente
- Calcular Q1 e Q3
- Encontrar IQR (Q3-Q1)
- Definir o limite inferior Q1*1.5
- Definir o limite superior Q3*1.5
- Outlier é qualquer ponto fora dos limites

Mais uma vez os modelos foram retreinados após a identificação dos valores ausentes, conforme resultados abaixo:




```
ADA: 0.799094 (0.039965) GB: 0.802495 (0.053753) RF: 0.797060
(0.054441) CART: 0.774393 (0.053665) LOGIT: 0.782002 (0.038554)
```

O que mais me chamou a atenção foi o ganho que o tratamento trouxe para o modelo de regressão logística. A balanced accuracy passou de .78 para .80. Os demais modelos apresentaram pouca ou nenhuma melhoria, o que indica que não são tão sensíveis ao outliers quanto a regressão logística.

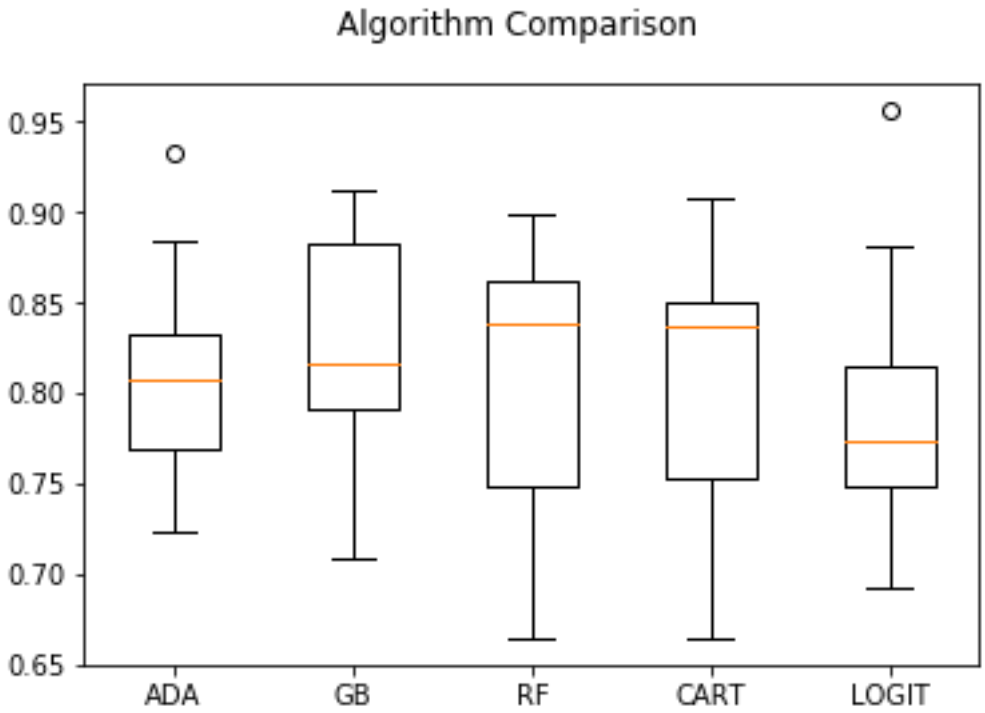
Feature engineering

Nesta fase foram realizados dois tipos de tarefas: 1) criação de novas variáveis e; 2) transformação de variáveis já existentes. As novas variáveis criadas foram n_familia (quantidade total de familiares) e criança (identifica se o passageiro possui menos de 14 anos).

Em termos de transformação de variáveis, as seguintes tarefas foram realizadas:

- As variáveis categóricas passaram pela transformação one hot encoding
- As variáveis numéricas passaram por um processo de binning
- As variáveis numéricas também foram padronizadas e normalizadas

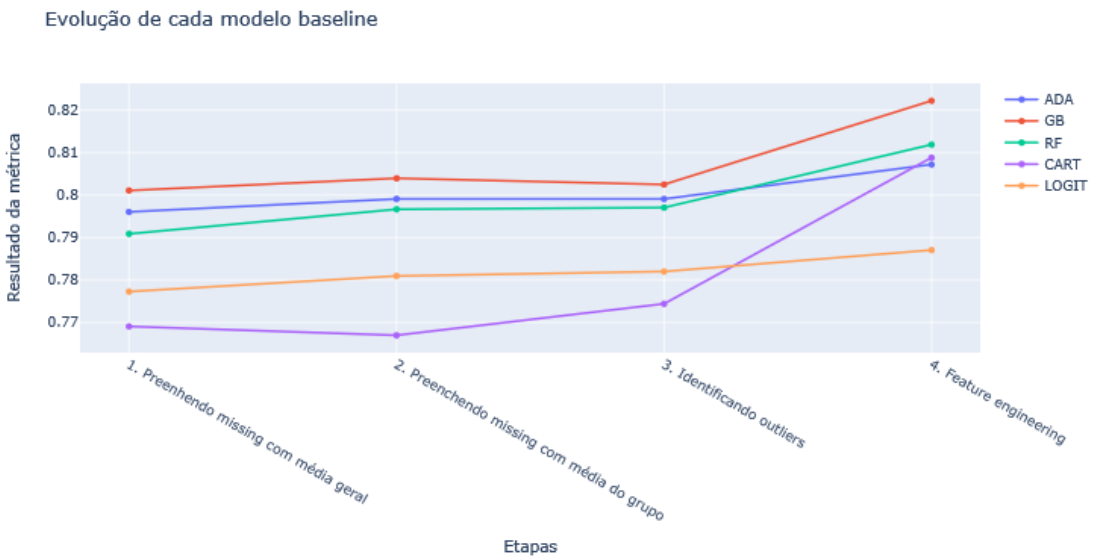
Em posse de todas as variáveis, os modelos baselines foram executados novamente. No entanto, desta vez, antes da execução de cada algoritmo, as variáveis foram pré-selecionadas por meio do método Recursive Feature Elimination (RFE). Como o processo de feature engineering resultou em inúmeras variáveis, RFE foi adicionado ao processo para permitir que somente as variáveis mais relevantes fossem testadas em cada algoritmo.




```
ADA: 0.807220 (0.049823) GB: 0.822225 (0.058677) RF: 0.811902
(0.066141) CART: 0.808835 (0.061107) LOGIT: 0.787051 (0.061384)
```

Em geral, todos os algoritmos se beneficiaram da fase de feature engineering. Mais uma vez, o modelo de logistic regression se destacou em termos de melhoria quando comparado com as fases anteriores. O principal motivo foi a normalização das variáveis.

A decisão final foi tomada com base no gráfico de linhas abaixo, o qual apresenta o desempenho de cada modelo ao longo de cada atividade de modelagem.

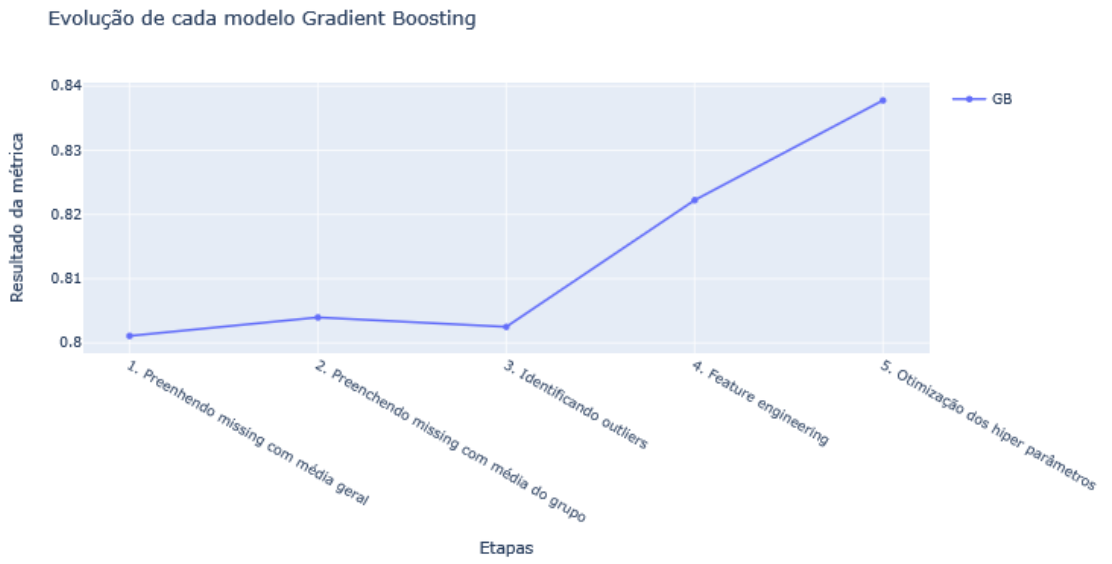


Minha escolha foi o modelo baseado no algoritmo Gradient Boosting. Além de apresentar a melhor balanced accuracy (conforme exibido no gráfico de linhas), também mostrou boa estabilidade (conforme último box-plot apresentado).

Otimização dos hiper parâmetros do modelo vencedor

A fase final da modelagem consistiu em calibrar o algoritmo Gradient Boosting. Os seguintes parâmetros foram calibrados: n_estimators, learning_rate e max_depth. Os valores encontrados foram 200, 0.3 e 2, respectivamente.

O modelo calibrado resultou em balanced accuracy de .837. O gráfico abaixo exibe a evolução histórica de todas as tentativas de melhoria do modelo final.



Extraindo os artefatos do modelo

Com o modelo já criado e treinado, todos os artefatos foram extraídos para futura utilização no pipeline de produção. Estes artefatos incluem: arquivo binário do modelo treinado, encoders responsáveis por transformações dos dados, estatísticas dos dados de treino para tratamento de valores ausentes e outliers nos novos dados. Em caráter informativo, as seguintes variáveis e índices de importância foram inseridas no modelo final:

	variavel	importancia
2	titulo__Mr	0.507671
1	standard_scale__Fare	0.321216
0	Pclass	0.171113

Reporte final das métricas

Durante todo o projeto foi utilizada a métrica balanced accuracy para tomada de decisões. Já com o modelo pronto, estou reportando métricas adicionais na tabela abaixo (cross-validation):

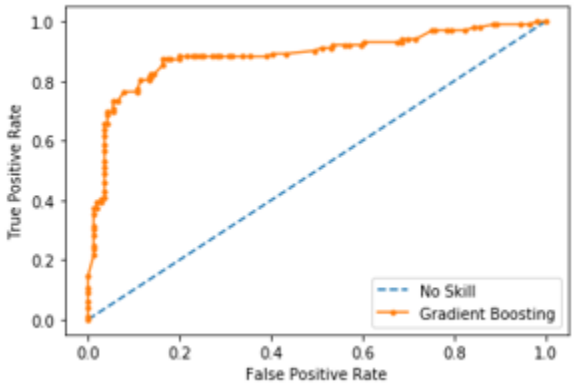
MÉTRICA	VALOR

BALANCED ACCURACY	.837
ACCURACY	.854
PRECISION	.821
RECALL	.773
AUC	.889

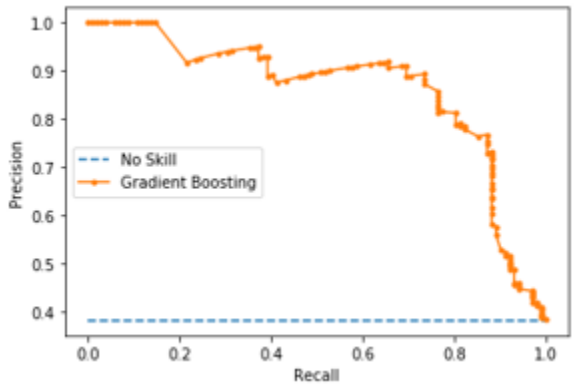
Adicionalmente, estou compartilhando a curva roc e a curva precision-recall. Na minha estratégia de modelagem decidi não criar uma base de teste, já que optei por aplicar cross-validation em todo o projeto e usar a própria base de teste do kaggle para testar o classificador final. O problema é que a base de teste do Kaggle não está disponível para eu usar na construção destas curvas.

Por este motivo, usei uma amostra aleatória simples com 70% dos dados para *retreinar* o modelo vencedor e as curvas foram construídas nos 30% restante dos dados. Isso acaba sendo uma proxy para os resultados que apresentei anteriormente no cross-valdiation (ficaram muito próximos).

Curva ROC



Curva precision-recall



O modelo final foi avaliado nos dados de teste pelo próprio kaggle, usando a métrica accuracy. O resultado foi de 77%.

Conclusão

Neste relatório tentei explorar os pontos que julguei mais relevantes para a modelagem deste problema. No notebook compartilhado, automatizei a maior parte das atividades para que fosse possível, rapidamente, testar novas ideias sem muito esforço. Por fim, também desenvolvi o código de forma que fosse possível criar um pipeline de produção para fazer predições em dados nunca vistos antes (por exemplo, o conjunto de treino do kaggle).

Ressalto aqui algumas direções para continuar a modelagem deste problema, as quais não tive tempo de implementar no notebook:

- Acredito que é possível explorar mais as variáveis Name e Ticket para identificar pessoas da mesma família ou que estavam viajando juntas
- Creio que seria uma boa tentativa usar um ensemble entre um dos modelos de árvore de decisão e o modelo de regressão logística. Talvez aplicar stacking ou blending destes dois modelos
- Na fase de feature engineering ainda é possível continuar explorando as transformações das variáveis (testar outros bins, agrupamentos, etc)
- Seria interessante substituir o método Recursive Feature Elimination por outros formas de identificação das melhores variáveis
- No grid search poderia usar logspace para definir os valores de teste de cada parâmetros
- Em termos de automação, poderia usar pipelines
- Creio que existe abertura para tentar usar under/over sampling com o objetivo de balancear os dados de treino