Capstone Project

Machine Learning Engineer Nanodegree – Udacity

Rhaissa Lorrany Souza Araújo

11 de março, 2018

I. DEFINIÇÃO

Visão geral do projeto

A empresa Porto Seguro é uma das maiores empresas de seguros de automóveis do Brasil, e responsável por seguros em diversos serviços.

O foco do problema proposto são os seguros de automóveis, onde um dos maiores desafios é a atribuição correta de valores para diversos tipos de clientes considerando quem de fato necessitará do seguro. A partir de informações dos clientes e de todo o histórico de sinistros é possível encontrar padrões no perfil de quem utiliza o produto, de modo que seja possível realizar melhores estratégias dentro da empresa, com preços mais condizentes com a probabilidade de sinistro dos clientes.

Descrição do problema

O projeto aqui proposto se relaciona com o desafio da empresa Porto Seguro na plataforma Kaggle (*https://www.kaggle.com/c/porto-seguro-safe-driver-prediction*). A Porto Seguro apresenta uma base com diversas informações de seus clientes e tem interesse em conhecer a probabilidade de sinistro dos respectivos clientes no ano seguinte.

A solução proposta para o problema usa algoritmos de *machine learning* cuja resposta é uma classificação (nesse caso, ter ocorrido ou não um sinistro; duas classes), onde as variáveis explicativas são informações estruturadas das quais não sabemos seus respectivos significados. A qualidade do resultado pode ser avaliada como sugerido na proposta da própria empresa: utilizando o índice de Gini normalizado. Considerando que as informações utilizadas para os algoritmos continuarão a ser coletadas pela empresa, o algoritmo é facilmente replicável.

Métricas

A métrica de avaliação proposta pela Porto Seguro é o índice de Gini normalizado. Essa métrica permite avaliar como está a distribuição de probabilidade do modelo de forma geral. Assim, deve-se esperar que casos de sinistro estejam concentrados em altas probabilidades do modelo, e os casos que não ocorreram sinistro devem estar concentrados em baixas probabilidades. A métrica se baseia na AUC (*area under the curve*), que considera o acerto do público de sinistro (taxa de verdadeiros positivos) e o erro do público que não foi sinistro (taxa de falsos positivos) em diversos pontos da curva de probabilidades.

Representação matemática:

$$Gini = 2 * AUC - 1$$

II. ANÁLISE

Exploração dos dados

O conjunto de dados possui 595.212 observações e 59 variáveis, das quais uma representa um campo de identificação e outra a variável resposta representada por ‘0’s e ‘1’s.

Os campos que são dados faltantes estão representados pelo valor -1. Nos casos das variáveis categóricas esses dados faltantes serão tratados como uma nova categoria e, no caso das variáveis contínuas substituídos pela mediana. Como pode ser observado na figura 1 há duas variáveis categóricas com alta quantidade de dados faltantes: *ps_car_03_cat* e *ps_car_05_cat.*

Os nomes das variáveis não representam seu significado por escolha da empresa, mas foram organizados de forma que aquelas que possuem ‘bin’ no nome são binárias (14), ‘cat’, são categóricas (17) e as demais são contínuas ou categóricas ordinais (26). Estão também organizadas por similaridade de informação: há 4 “grupos”, representados pelos conjuntos ‘ind’, ‘reg’, ‘car’ e ‘calc’.

O sinistro ocorre em 3,64% das observações e está representado pelo ‘1’ na variável *target.*

De todas as variáveis somente 13 possuem dados faltantes: *ps_car_03_cat, ps_car_05_cat, ps_reg_03, ps_car_14, ps_car_07_cat, ps_ind_05_cat, ps_car_09_cat, ps_ind_02_cat, ps_car_01_cat, ps_ind_04_cat, ps_car_11, ps_car_02_cat, ps_car_12*.

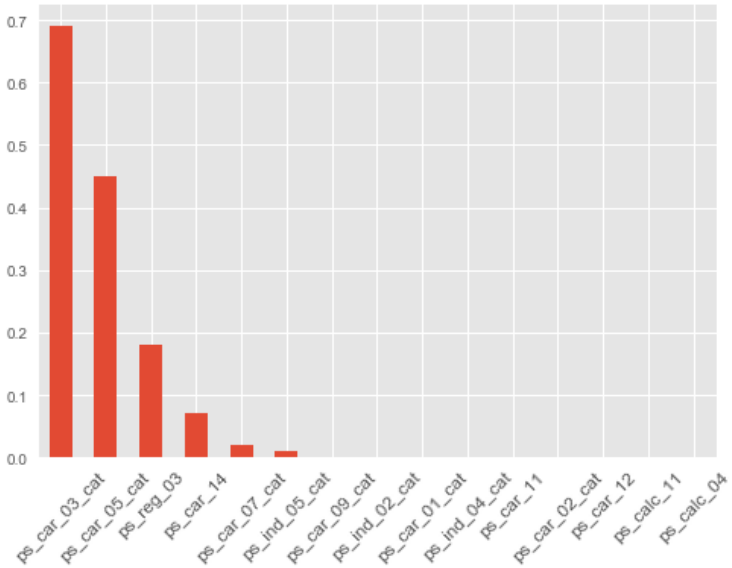


Fig.1: Percentual de *missing* por variável

Algumas variáveis categóricas possuem grande quantidade de categorias: *ps_car_06_cat*, *ps_car_11_cat*, com 18 e 104, respectivamente. Essas não terão tratamento para indicadores *dummy*, por aumentar muito a quantidade de colunas e prejudicar o processamento dos algoritmos de *machine learning.*

Visualização exploratória

Os gráficos abaixo mostram a relação marginal entre as variáveis categóricas e binárias em relação a variável resposta.

Legendas dos gráficos:

- % observações na categoria
- % da variável resposta na categoria
- ---- % da variável resposta em toda a base (3,64%)

A figura 2 abaixo mostra a variação do percentual da variável *target* entre as categorias, trazendo alguma intuição de quais podem ser relevantes na modelagem. Apesar do baixo volume de algumas categorias na *ps_ind_05_cat*, há várias categorias como a 2, 6 e *missing* com percentual de sinistro mais alto que a média geral. No caso da *ps_car_01_cat* isso acontece também com algumas categorias se destacando em relação à média, como a 6, 7, 9 e *missing,* mas devido ao pouco volume desta última é possível que se trate de uma informação que não se destaque.

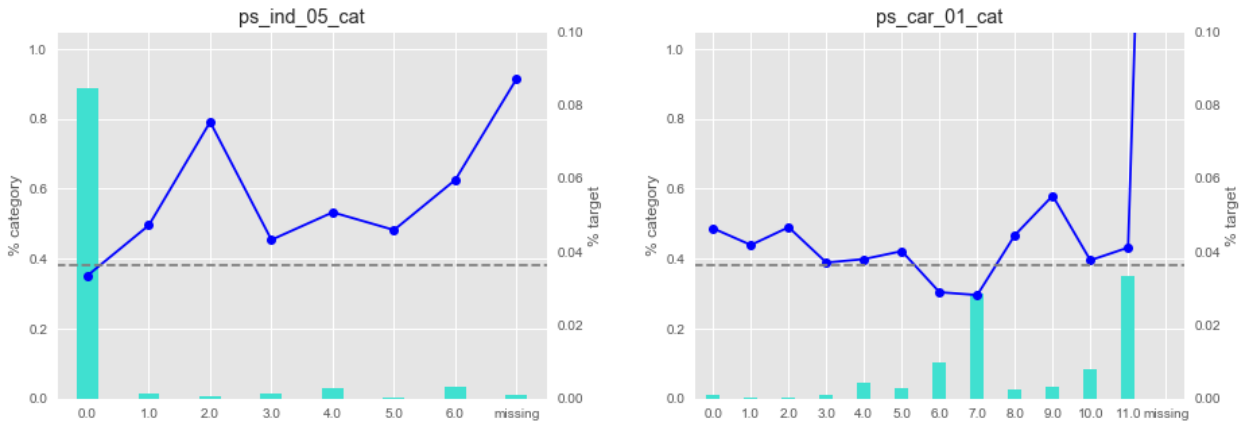


Fig.2: variação do percentual da *target* nas variáveis ps_ind_05_cat e ps_car_01_cat

Nas figuras 3, 4 e 5 abaixo, há variação do percentual da variável *target* nas variáveis binárias. No caso da figura 3 abaixo, observa-se que cada categoria tem tendências opostas quanto ao sinistro.

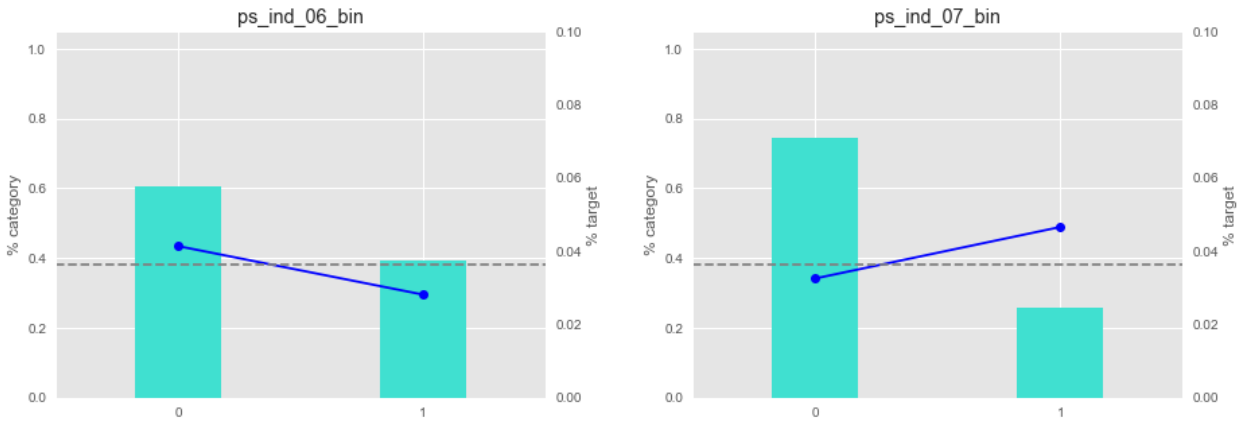


Fig. 3: Percentual da *target* nas variáveis ps_ind_06_bin e ps_ind_07_bin

Na figura 4 abaixo, observa-se uma situação comum em outras variáveis: a variável é predominante (quase 100%) em uma das categorias. O mesmo ocorre com *ps_ind_12_bin* e *ps_ind_13_bin.* Devido a essa característica de ter variância próxima de zero, elas devem ser excluídas dado que não contribuem na modelagem.

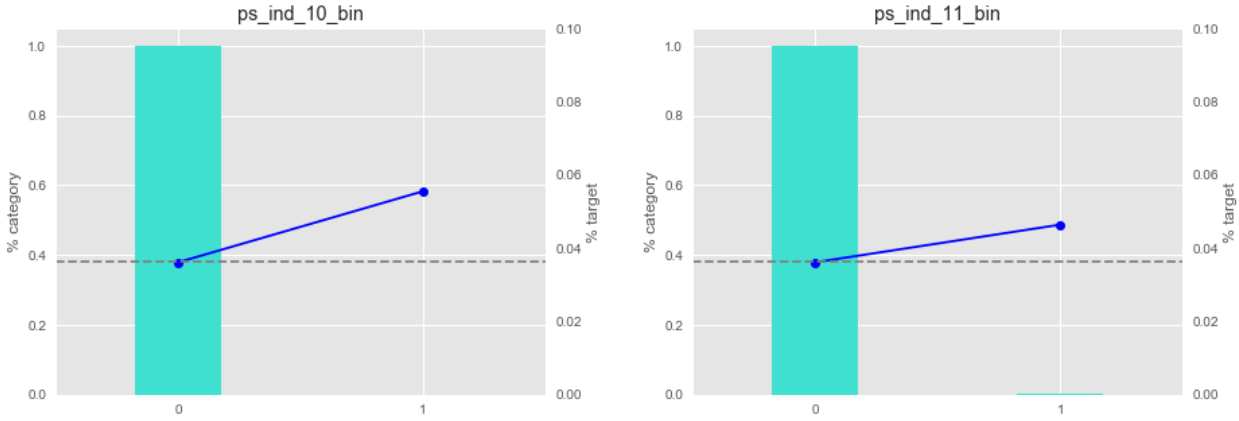


Fig. 4: Percentual da *target* nas variáveis ps_ind_10_bin e ps_ind_11_bin

Na figura 5 abaixo, observa-se uma outra situação comum em outras variáveis: o percentual da *target* não difere por categoria. Isso se repete nas variáveis *ps_ind_18_bin, ps_calc_15_bin, ps_calc_16_bin, ps_calc_17_bin, ps_calc_18_bin, ps_calc_19_bin, ps_calc_20_bin.*

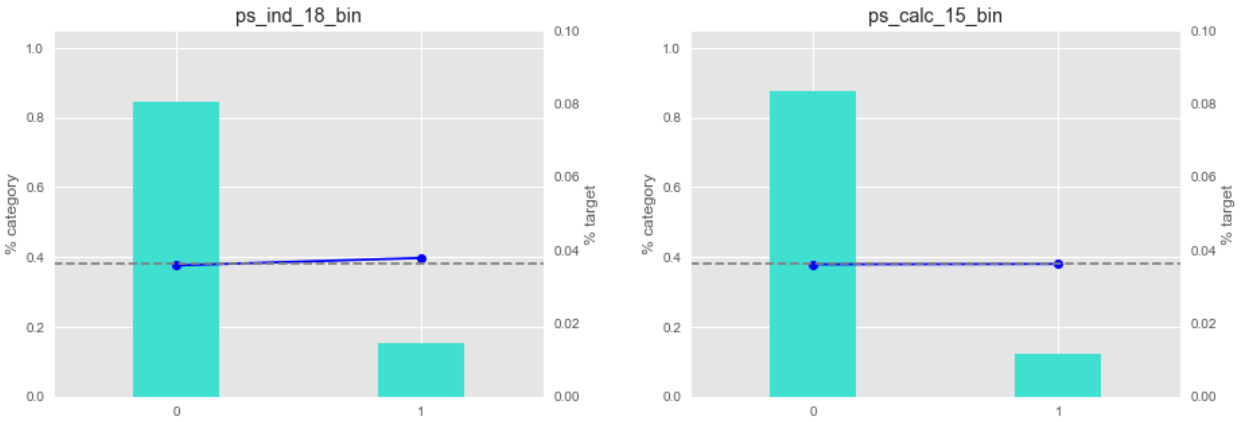


Fig. 5: Percentual da *target* nas variáveis ps_ind_18_bin e ps_calc_15_bin

As variáveis contínuas foram segmentadas em faixas para que sejam analisadas descritivamente da mesma forma, segundo sua proporção da *target.* As faixas possuem aproximadamente o mesmo volume de observações.

Na figura 6, ilustrando a variável *ps_car_13*, observa-se que com o aumento do seu valor a taxa de sinistro aumenta também. Na figura 7, a ilustração por faixas pequenas é interessante pois observa-se que existe uma relação entre as variáveis, porém não linear, entre a variável *ps_car_14* e a *target.* Ao longo de sua distribuição é possível observar grupos com maior percentual de sinistro tanto em valores menores, quanto maiores da variável. Os algoritmos de árvores são capazes de capturar tais relações.

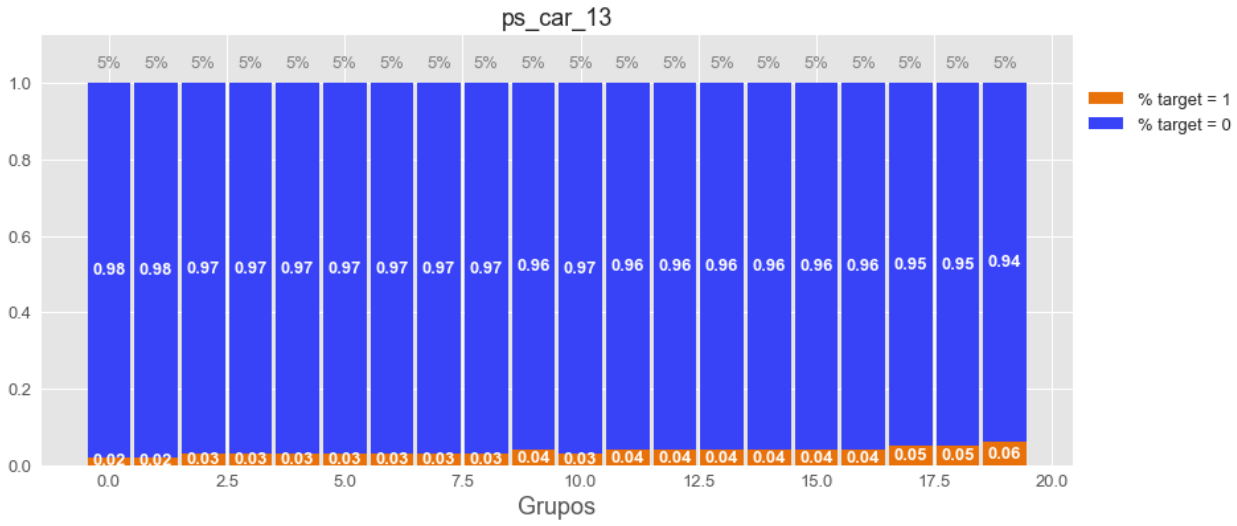


Fig. 6: Percentual da *target* na variável contínua ps_car_13

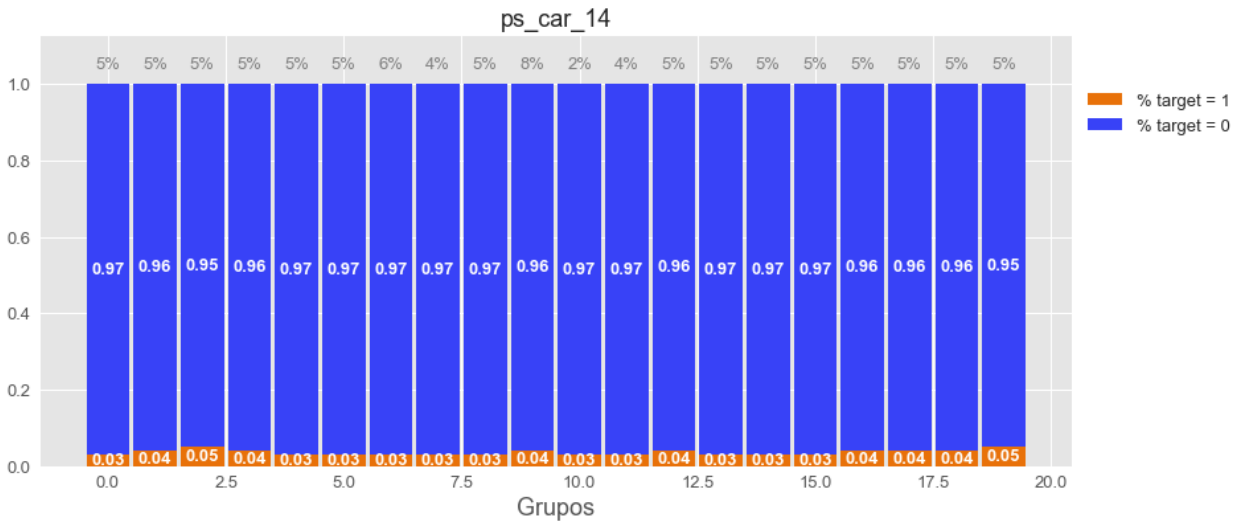


Fig. 7: Percentual da *target* na variável contínua ps_car_14

No gráfico de correlação abaixo observa-se que dentre as variáveis com *'car'* no nome há alta correlação, assim como entre *ps_reg_02* e *ps_reg_03*. No caso do grupo de variáveis com 'calc' no nome a correlação é próxima de zero.

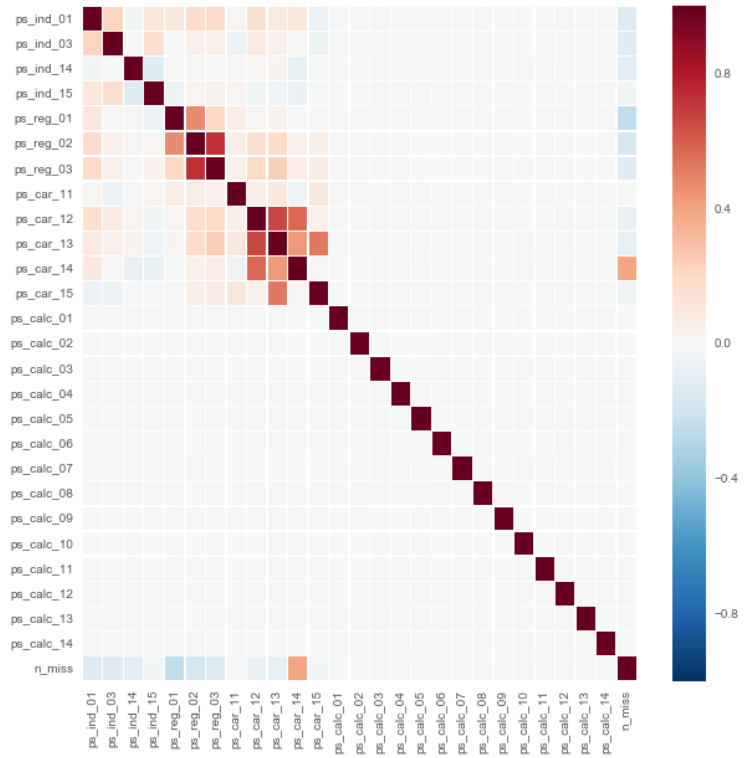


Fig. 8: *Heatmap* com correlação das variáveis contínuas

Algoritmos e técnicas

Foram aplicados algoritmos diversos de *machine learning* cuja resposta pode ser uma classificação (nesse caso, ter ocorrido ou não um sinistro: duas classes) e onde se considera um conjunto de variáveis explicativas estruturadas. Em geral, são algoritmos que são a combinação de vários classificadores *(ensembles)*, eles performam melhor que um único pois ajudam a reduzir o *overfitting* e são mais estáveis.

Os algoritmos de *boosting* se caracterizam por se basear em outros algoritmos mais simples, como árvores de decisão, onde cada árvore se baseia no resultado da anterior para formar um classificador robusto: na primeira árvore todas observações possuem o mesmo peso, mas a árvore posterior se baseia no resultado da primeira, dando um peso maior para as observações classificadas erradas, e assim isso se repete até uma determinada quantidade de árvores ou acurácia desejada. *O Gradient Boosting* tem esse nome pois utiliza o gradiente descendente no processo de minimização do erro. Tanto o *Extreme Gradient Boosting* quanto o *LightGBM* tem o mesmo conceito por trás, porém possuem melhorias no algoritmo: menor uso da memória, então são muito mais rápidos, possuem modificações no algoritmo que aumentam a acurácia dos modelos e com escalabilidade para grandes conjuntos de dados.

O *Random Forest* se baseia no conceito de *bagging,* no qual cada árvore de decisão utiliza uma parte aleatória do conjunto de dados, e também um conjunto específico de variáveis é sorteado dentre todos para cada quebra da árvore.

O Naive Bayes é um algoritmo simples e rápido, baseado no Teorema de Bayes, onde assume a independência entre todas as variáveis.

Benchmark

Considerando que se trata de um desafio já encerrado do *Kaggle*, já é conhecido que a sua melhor solução teve um índice de Gini de 0,29698 na sua base teste. De acordo com o fórum de discussão, o primeiro lugar foi a implementação de um modelo que combinava outros seis, um deles sendo *LightGBM* e outros cinco de redes neurais (https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/discussion/44629).

Aparentemente, outros bons resultados foram obtidos com *XGBoost e LightGBM.*

III. METODOLOGIA

Pré-processamento de dados

Dado que a maioria dos algoritmos que serão testados se baseiam em árvores de decisão e não são influenciados pela escala e *outliers*, os dados só foram normalizados para aplicação do *Naive Bayes*. O tratamento de dados faltantes foi realizado conforme citado anteriormente, com a mediana para os casos de variáveis contínuas e ordinais e como uma categoria à parte no caso das categóricas e binárias.

Implementação

A implementação envolveu os seguintes processos utilizando o *jupyter notebook*:

- Importação das principais bibliotecas necessárias: modelagem, *grid search*, validação cruzada, transformação de escala;
- Leitura dos arquivos;
- Separação de base treino e teste utilizando uma semente, sempre preservando as mesmas bases para qualquer algoritmo e em qualquer momento que o programa for executado;
- Criação de nova variáveis (ex.: quantidade de informações faltantes por indivíduo);
- Tratamento dos dados faltantes;
- Função com métrica de Gini normalizado criada para ser utilizada no processo de busca dos melhores hiperparâmetros;

Referente aos algoritmos de *machine learning*: foram realizados os modelos a seguir com um processo de validação cruzada *k fold*, com k = 5 no caso do *Random Forest, Gradient Boosting, Extreme Gradient Boosting*, principalmente por motivos de tempo de processamento e limitações de *hardware*.

1. *Gradient Boosting*
   Parâmetros variados:
   - Quantidade de estimadores (n_*estimators*)
   - Taxa de aprendizado (*learning_rate*)
   - Quantidade de variáveis sorteadas por quebra da árvore (*max_features*)
   - Profundidade máxima da árvore (*max_depth*)

2. *Random Forest*
   Parâmetros variados:
   - Quantidade de estimadores (n_*estimators*)
   - Quantidade de variáveis sorteadas por quebra da árvore (*max_features*)
   - Profundidade máxima da árvore (*max_depth*)
   - Quantidade mínima necessária de casos para realizar uma quebra no nó *(min_samples_split)*

3. *Extreme Gradient Boosting*
   Parâmetros variados:
   - Quantidade de estimadores (n_*estimators*)
   - Quantidade de variáveis sorteadas por quebra da árvore (*max_features*)
   - Profundidade máxima da árvore (*max_depth*)
   - Quantidade mínima necessária de casos para realizar uma quebra no nó *(min_samples_split)*

4. Gaussian Naive Bayes

5. *LightGBM*
   Parâmetros:
   - Número de folhas *(num_leaves)*

- Percentual de features selecionadas a cada iteração *(feature_fraction)*
- Percentual das observações selecionados (*bagging_fraction*)
- Taxa de aprendizado *(learning_rate)*
- Profundidade máxima da árvore *(max_depth)*

Refinamento

De modo geral o aperfeiçoamento do resultado foi realizado com:

- Grid Search (testando-se mais valores nos hiperparâmetros)
- Exclusão de mais variáveis com a função de baixa variância

Primeiramente foi realizado um *Gradient Boosting* e o *LightGBM* com os valores dos hiperparâmetros fixados para que, de forma rápida, fosse obtida uma solução inicial para a comparabilidade de resultados com outras técnicas utilizando o *grid search.*

Em seguida, foram aplicadas as outras técnicas citadas na seção anterior. Nesse momento novamente o *Gradient Boosting* mostrou os melhores resultados.

No processo de refinamento foram excluídas mais variáveis categóricas com variância próxima a zero e testando novamente os dois algoritmos mais relevantes: *Gradient Boosting e LightGBM*.

Os resultados do Gini para os modelos ficaram próximos, sendo que o melhor na validação cruzada foi o *Gradient Boosting* com Gini = 0,28, sendo portanto, o modelo final.

	Modelos	Hiperparâmetros	Gini na validação cruzada	Gini na base teste (30%)
Soluções iniciais - hiperparâmetros fixados	LightGBM	num_leaves , feature_fraction, bagging_fraction, learning_rate, max_depth	0,2661	0,2625
	Gradient Boosting	n_estimators, learning_rate, max_depth	0,259	0,247
Técnicas aplicadas	Naive Bayes	-	0,226	0,220
	Gradient Boosting	n_estimators : **200** learning_rate : [**0,08** ; 1] max_depth : [**3** ; 5 ; 7]	0,280	0,270
	Random Forest	n_estimators : **350** min_samples_split : [**0,02** ; 0,06 ; 0,08] max_depth : [3,5,**8**]	0,262	0,249
	Extreme Gradient Boosting	min_child_weight : [1, **5**, 10], max_depth: [3, 5, **8**], gamma : [**1**, 10]	0,257	0,2485

Refinamento	**Modelo Final** Gradient Boosting	n_estimators : **200** learning_rate : [1, **0,07**] max_depth : [**4** ; 7]	0,281	0,269
	LightGBM		0,276	0,259

Tabela 1: Resultados de Gini nos modelos

IV. RESULTADOS

Modelo de avaliação e validação

O modelo do *Gradient Boosting* foi escolhido devido à melhor performance encontrada durante o processo de *grid search* e validação cruzada, no qual os parâmetros foram otimizados considerando diferentes conjuntos de dados para treino. Com o resultado na base teste próximo ao da validação cruzada há mais uma garantia que o modelo está com boa performance para um conjunto de dados que não foi utilizado para o treino do modelo, confirmando a confiabilidade do modelo.

Justificativa

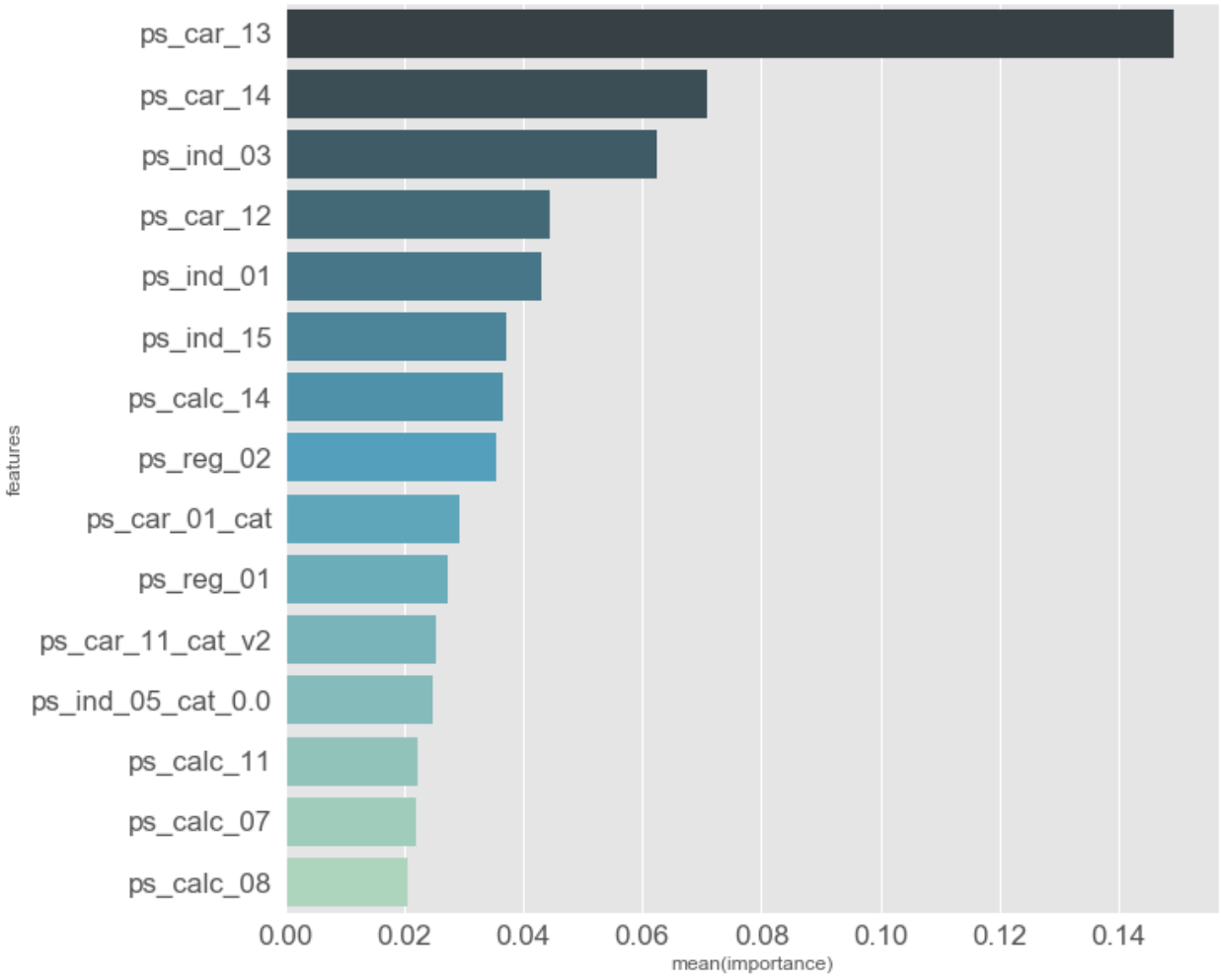
O resultado do benchmark apresenta 0,29 na base do *Kaggle*, o modelo aqui apresentado tem na base de validação cruzada 0,28 e 0,273 na própria base teste do *Kaggle,* que estão muito próximos do benchmark. O resultado também é ótimo ao considerarmos o nível de complexidade do modelo vencedor da competição e este apresentado no projeto, no qual o problema é resolvido com a aplicação de somente um algoritmo.

V. CONCLUSÃO

Forma de livre de visualização

O gráfico abaixo (figura 8) apresenta as 15 variáveis mais relevantes no modelo para responder a ocorrência de sinistro. Principais observações:

- As informações mais relevantes para responder o sinistro são ps_car_13 e ps_car_14 cuja importância já havia sido observada na seção de visualização dos dados;
- *ps_car_11_cat* reagrupada de acordo com a taxa da resposta ficou entre as mais importantes;
- Predominam como mais importantes as variáveis ordinais e contínuas.



- Figura 8: Importância das variáveis do modelo final

Dado que o evento é raro, uma forma de aplicação pela empresa a partir da modelagem obtida seria uma ação focada no público que possui maior probabilidade de sinistro, podendo essa ser uma ação relacionada ao preço, por exemplo. O gráfico abaixo (figura 9) contém a informação de *recall* e *precision*, para diferentes tamanhos de público onde poderiam ser aplicados algum tipo de ação:

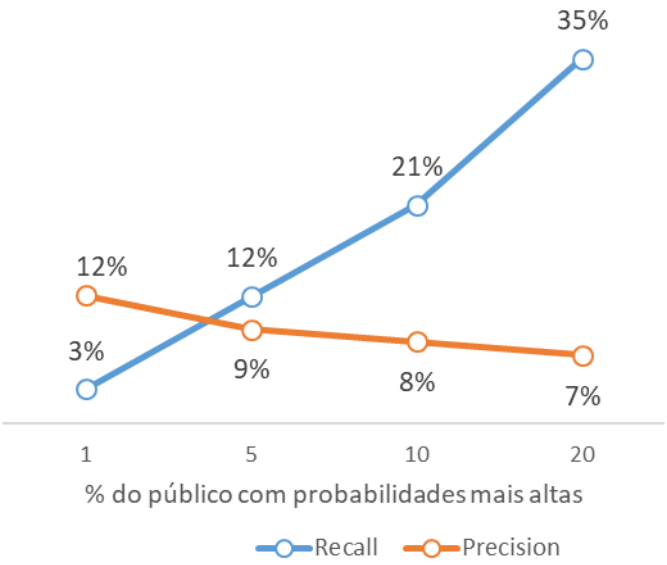


Figura 9: *Recall* e *Precision* para diferentes tamanhos de público

Por exemplo: na captura de 10% do público em média seriam capturados 21% dos casos de sinistro, o que é muito significativo tratando-se de um evento raro e fazendo com que esse público seja um ótimo alvo para uma possível ação.

Reflexão

O projeto aqui realizado foi um desafio da Porto Seguro da plataforma Kaggle. Consistiu em utilizar algoritmos de *Machine Learning* para obtenção da probabilidade de sinistro de indivíduos com um seguro.

O processo utilizado para resolução do problema de forma geral residiu em:

- Análise exploratória da base: % da variável *target,* análise *de missings,* visualização das variáveis categóricas e contínuas;
- Tratamento da base para modelagem: quais variáveis seriam transformadas para indicadores *dummy*, tratamento dos dados faltantes, criação de variáveis;
- Modelagem com *grid search* e validação cruzada com diversos algoritmos;
- Avaliação dos melhores resultados.

Na área de aprendizado de máquina é difícil afirmar que exista uma solução ótima dado a dificuldade de mensurá-la (sempre dependeria de uma base específica para confiar essa conclusão) e as inúmeras possibilidades de metodologia em diversas etapas: tratamento e seleção de *features*, processo de modelagem (como é feita a validação e o *grid search*), complexidade dos algoritmos, etc.

Os resultados do projeto e dos competidores da plataforma do *Kaggle* em geral são muito próximos, confirmando que a solução final obtida com o algoritmo de *Gradient Boosting* está aderente dentro da proposta realizada pela Porto Seguro.

Melhorias

Processos que poderiam ser considerados na resolução do projeto:

- Aplicação de métodos para seleção de variáveis: há bibliotecas no python que realizam esse tipo de análise, ou com *scores* mais simples, (qui-quadrado por exemplo) ou com outros modelos no processo de seleção de *features*;
- Aplicação de vários modelos em conjunto: nesse processo a performance tende a melhorar pois cada algoritmo terá falhas em alguns aspectos específicos e com isso os algoritmos se complementam.
- Aplicação de técnicas de *oversampling / undersampling* para tratar o desbalanceamento da base.

Referências

James, G. et al. An Introduction to Statistical Learning. 1 ed. New York: Springer, 2013.
[1] http://xgboost.readthedocs.io/en/latest/model.html
[2] https://github.com/Microsoft/LightGBM
[3] http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/
[4] https://github.com/dmlc/xgboost