Projeto de Conclusão de Curso

Engenheira de Machine Learning – Nanodegree

Cibele Castelo Nogueira
cibele_cast@hotmail.com.br

SUMÁRIO

1. DEFINIÇÃO

1.1. VISÃO GERAL DO PROJETO

O projeto apresenta o problema e solução para a competição Porto Seguro's Safe Driver Prediction que está disponível no site Kaggle[1], que é uma plataforma para modelagens preditivas e competições analíticas. Estas competições são colocadas por empresas e pesquisadores com o objetivo de produzir os melhores modelos para prever os dados.

A competição escolhida é hospedada pela Porto Seguro: uma das melhores e maiores empresas de seguros no Brasil. Esta empresa oferece seguros para ramos diversos que incluem seguros para veículos, viagens, vida, aluguel, capitalização, eventos, etc. Além disso, possui mais de 14 mil funcionários e mais de 15 milhões de clientes em todo o brasil.

Nada arruína a emoção de comprar um carro novinho mais rapidamente do que ver a sua nova conta do seguro. A picada é ainda mais dolorosa quando você sabe que você é um bom motorista. Não parece ser justo que você tenha que pagar tanto se você tem sido cauteloso na estrada por anos.

A Porto Seguro, uma das maiores empresas de seguro de carro e casa, completamente concorda. As imprecisões nas previsões de sinistros da companhia de seguros de automóveis aumentam o custo do seguro para bons condutores e reduzem o preço para maus condutores.

Nesta competição, você é desafiado a construir um modelo que prevê a probabilidade de um motorista acionar o seguro de veículo no próximo ano. A Porto Seguro tem utilizado Machine Learning nos últimos 20 anos, e eles estão olhando para a comunidade de Machine Learning da Kaggle para explorar novos métodos mais poderosos. Uma previsão mais precisa irá permitir que eles sigam adaptando seus preços com a esperança que a cobertura do seguro automóvel seja mais acessível para mais motoristas.

Segue abaixo os dados da duração da competição:

Inicio: 29 de Setembro de 2017
Término: 29 de Novembro de 2017
Duração: 2 meses

O conjunto de dados é compostos pelos seguintes arquivos:

- **train.csv:** é o conjunto de dados de treinamento. Cada linha do conjunto de dados corresponde a um titular do seguro e a coluna "target" significa se um sinistro foi preenchido ou não.
- **test.csv:** é o conjunto de teste.
- **sample_submission.csv:** é o arquivo de submissão a ser enviado para a competição da kaggle.

[1] https://www.kaggle.com/c/porto-seguro-safe-driver-prediction
[2] https://www.kaggle.com
[3] https://www.portoseguro.com.br/

1.2. DECLARAÇÃO DO PROBLEMA

Cada linha do conjunto de dados corresponde a um titular do seguro e a coluna “target” significa se um sinistro foi preenchido ou não. Com o intuito de proteção de seus dados, as features do dataset foram renomeadas, não possibilitando assim saber o significado real das variáveis. Elas foram colocadas em grupos similares: ind, reg, car e calc e de acordo com o “dono da competição” o significado dessas abreviaturas são:

- Ind está relacionado a individuo ou motorista;
- Reg está relacionada com região;
- Car está relacionado com carro mesmo;
- Calc é uma variável de cálculo.

Este problema de identificar clientes que acionaram o sinistro ou não, é um tipo de problema de aprendizagem de máquina supervisionada, mais especificadamente de classificação, pois a variável alvo (‘target’) apresenta valores discretos e não contínuos.

As seguintes tarefas vão ser realizadas:

1. Definição do problema
2. Coleta de Dados – Baixar os conjuntos de dados disponibilizados
3. Análise de dados
   3.1 Visualização do conjunto de treinamento.
   3.2 Verificação de Duplicação, Valores Faltantes e Outliers.
4. Processamento de dados
5. Construção do Modelo Preditivo
   5.1 Divisão do conjunto de treinamento em conjunto de dados de treino e teste.
   5.2 Definição e implementação dos algoritmos e técnicas utilizados.
6. Definição do melhor modelo.
7. Aperfeiçoamento do modelo
   7.1 Alteração de parâmetros
   7.2 Remoção de Outliers
   7.3 Utilização de Grid-Search para escolha de parâmetros melhores.
8. Avaliação do modelo
   8.1 Uso de métricas para avaliação
   8.2 Comparação dos Benchmarks
9. Utilização do test.csv e submissão do sample_submission.csv para a competição na plataforma Kaggle.

Assim, o modelo preditivo final terá como objetivo prevê de forma precisa, para dados não vistos antes, se o seguro foi acionado ou não.

1.3. MÉTRICAS

As submissões serão avaliadas na competição por meio da utilização do coeficiente Gini. O coeficiente Gini é uma medida de desigualdade desenvolvida pelo estatístico italiano Corrado Gini e é um parâmetro internacional usado para medir a desigualdade de distribuição de renda entre os países, porém pode ser utilizado também para qualquer distribuição. Ele possui um intervalo de avaliação que vai de 0 até 0.5.

Quanto mais próximo de 0.5 melhor é o *score.* O gini pode ser calculado conforme a fórmula abaixo.

$$gini = 2 \times AUC - 1$$

A métrica AUC é uma abreviação para *area under the curve .* Ela é utilizada em análise de classificação com o objetivo de determinar quais dos modelos utilizados preveem melhores as classes. Ela varia de 0 a 1. Quanto mais próximo de 1 melhor é a pontuação.

Tabela 1 : Classificação e intervalos da métrica AUC

AUC	Resultado
0.9-1.0	Excelente
0.8-0.9	Muito Bom
0.7-0.8	Bom
0.6-0.7	Suficiente
0.5-0.6	Ruim
<0.5	Test não é útil

A métrica AUC pode ser interpretada como a avaliação do *rank* de amostras positivas. As amostras positivas, neste caso, são aquelas com “target=1” (*pessoas que acionaram o sinistro). É equivalente à probabilidade de um ponto escolhido aleatoriamente da classe positiva ter um *score* mais alto de acordo com o classificador do que um ponto escolhido aleatoriamente da classe negativa. Então, um AUC perfeito = 1 significa que todos os pontos positivos tiveram um score alto que todos os pontos negativos. Para problemas de classificação com classes não balanceadas, como é o caso desta competição, a utilização da métrica AUC para seleção do modelo é muito mais significativa do que *accuracy*.

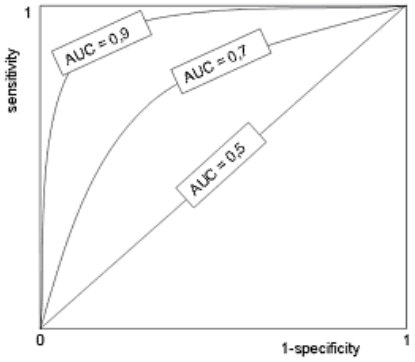


Figura 1: Curva AUC

Fonte: Simundic (2009)

2.ANÁLISE

2.1 EXPLORAÇÃO DOS DADOS

A competição possui dois conjunto de dados, sendo um conjunto de dados de treinamento contendo 595212 linhas e 59 colunas e o conjunto de teste contendo 892816 linhas e 58 colunas. O conjunto de dados de treinamento vai ser dividido em dois conjuntos: treinamento e teste, o primeiro será utilizado para a modelagem preditiva, utilizando-se assim da variável “target”, e o segundo para verificar se o modelo possui bom desempenho para dados não vistos. Após a construção de alguns modelos preditivos, será utilizado o conjunto de teste (test.csv) para prever os valores do “target”, e as submissões “sample_submission” serão enviadas para a competição no site da Kaggle.

O conjunto de dados é formado por variáveis binárias, nominais, intervalares e ordinais, que possuem diferentes formatos, níveis e grupos. Por possuirem essas diferenças se faz necessário a criação de metadados que contenham informações sobre elas, permitindo dessa maneira, análise e visualização exploratória que irão possibilitar a análise das features e identificação de padrões.

Dessa forma as variáveis foram separadas em:

- 2 formatos: int64 ou float64;
- 3 níveis: id, input e target;
- 4 tipos: binaria, nominal, intervalar e ordinal;
- 5 grupos: ind, reg, car, calc e nogroup.

De acordo com o organizador da competição, as features e linhas não são dependentes entre si, como por exemplo "ps_ind_01" e "ps_ind_02" são somente rótulos para esconder os nomes verdadeiro e infelizmente o real valor das variáveis não pode ser compartilhado.

Primeiramente, foi calculado alguns parâmetros de estatistica básica, entre eles: a quantidade, média, desvio padrão, min, 25%, 50%, 75% e max para os diferentes tipos de variáveis. Abaixo, os resultados da estátistica básica para as variáveis categóricas:

- Os valores variam de 0 até 104, sendo a coluna “ps_car_11_cat” a responsável pelo maior valor.
- As outras colunas apresentam uma “distância” pequena entre o min e o max exibidos.
- As colunas ps_ind_04_cat, ps_car_02_cat, ps_car_03_cat, ps_car_05_cat, ps_car_07_cat, ps_car_08_cat apresentam 0 ou 1 como resultado.

Para as variáveis númericas ordinais, obteve-se min = 1 e max = 25, verificou-se que somente a váriavel ps_ind_14 apresenta o min, 25%, 50% e 75% = 0.

Observou-se que a coluna “id” representa a identificação dos clientes por conter a soma de valores diferentes igual a 595212 que é a mesma quantidade de linhas presente no dataset.

O dataset já está com algumas alterações no que diz respeito a transformação dos dados. A verificação mostrou que não há duplicidade nos dados,e inicialmente verificou-se também, não haver valores faltantes, mas após análise no contéudo dos dados, foi possível concluir que em algumas colunas existia o valor -1 atribuido a várias linhas. Dessa forma, considera-se que colunas que possuem linhas com valores -1, na verdade possuem valores nulos. Para detectar a nulidade das variáveis e realizar a análise de dados, foi necessário substituir o valor – 1 por NaN. Importante ressaltar também, que o dataset é composto somente por números, não apresentando assim colunas do tipo ‘string’, por exemplo.

Como o dataset é muito grande, a visualização exploratória para os diferentes tipos de variáveis irá ser realizada a partir de uma amostra com fração de 0.5 do conjunto de dados das variáveis a serem analisadas.

2.2 VISUALIZAÇÃO EXPLORATÓRIA

A visualização exploratória será realizada com o objetivo de verificar: a distribuição dos valores binários do target, a quantidade de valores faltantes para cada variável, possíveis outliers, e por último será realizado a geração de diferentes tipos de gráficos para os diferentes tipos de variáveis.

2.2.1 VISUALIZAÇÂO DO TARGET

A visualização da distribuição dos valores -1 e 0 na váriavel target, demostram uma grande desproporção entre estes valores. A quantidade de pessoas que não acionaram o sinistro, valor 0, é bem maior da quantidade de pessoas que acionaram.

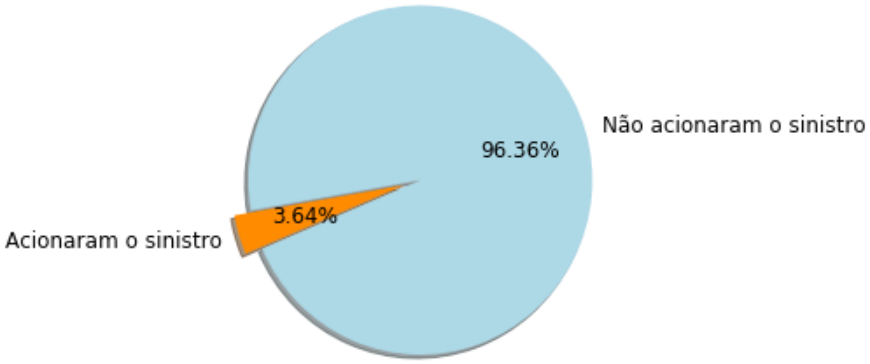


Figura 2: Porcentagem de clientes que acionaram ou não o sinistro

2.2.2 VISUALIZAÇÂO DOS VALORES FALTANTES

Colunas que possuem traços brancos são aquelas que possuem valores faltantes. A figura 3 permite inferir que existe 3 colunas que possuem uma quantidade maior de valores faltantes em relação as outras, elas são: ps_car_03_cat,ps_car_05_cat e ps_reg_03.

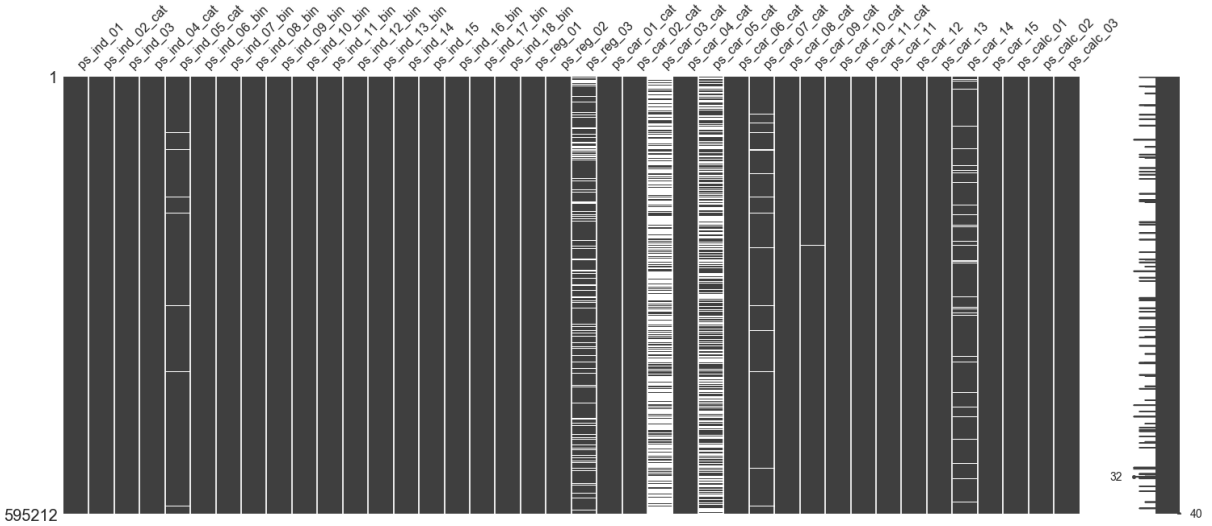


Figura 3: Visualização gráfica dos valores faltantes obtidos em cada coluna

Realizou-se uma filtragem das colunas que apresentaram porcentagem de nulidade acima de 0%. Dessa forma, o gráfico abaixo mostra somente as colunas que mais apresentam valores faltantes dentro do dataset. Estas colunas podem impactar no resultado da validação do modelo, por isso, elas devem revistas visando a necessidade de retirada de dados sujos que possam atrapalhar na modelagem dos dados preditivos.

O gráfico abaixo permite analisar que:

- ps_car_03_cat apresenta um valor gritante de quase 70% de valores faltantes;
- ps_car_05_cat apresenta 45% de valores faltantes;
- ps_reg_03 apresenta 18% de valores faltantes.

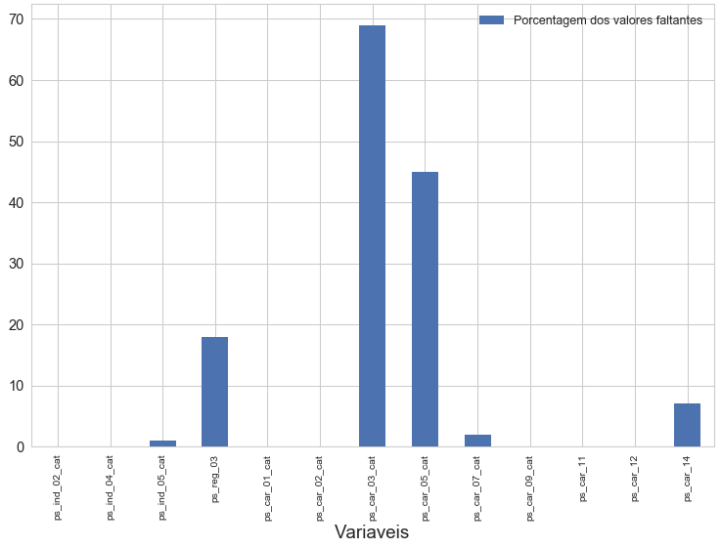


Figura 4: Visualização gráfica das variáveis que mais apresentam valores faltantes

2.2.3 CATEGÓRICAS NOMINAIS

Primeiramente, será utilizado o gráfico stripplot da biblioteca Seaborn, que desenha um scatterplot para cada variavel categórica.

Figura 5: Visualização gráfica da quantidade de diferentes valores que cada variável categórica possui

Na figura 5 verifica-se que:

- ps_cat_11_cat possui a maior quantidade de valores diferentes;
- As outras categorias não ultrapassam do valor entre 0 e 20.

O gráfico de barra, permite visualizar a frequência: número de vezes que uma categoria aparece no dataset.

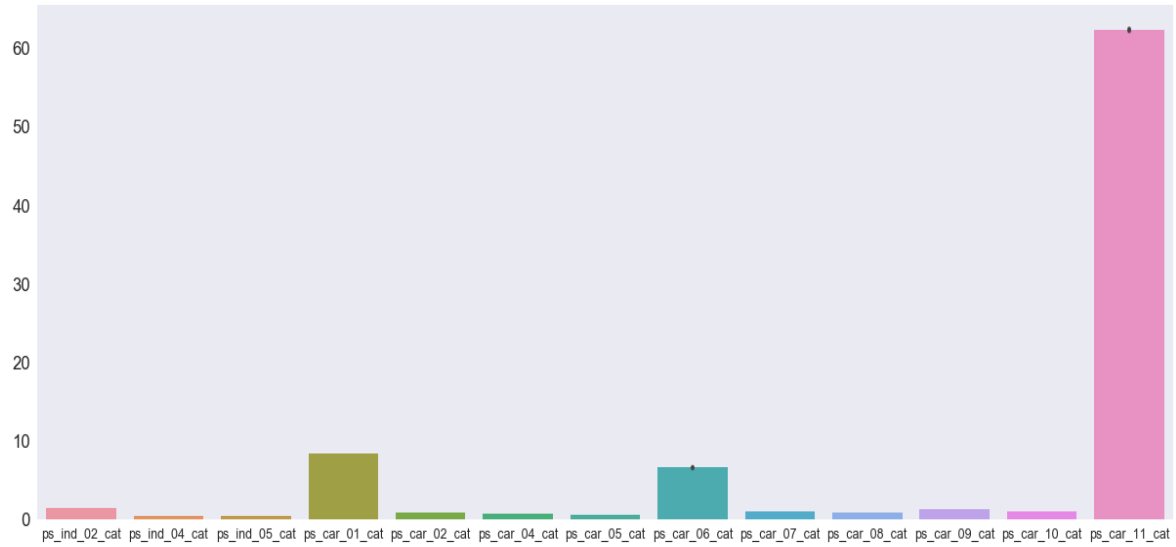


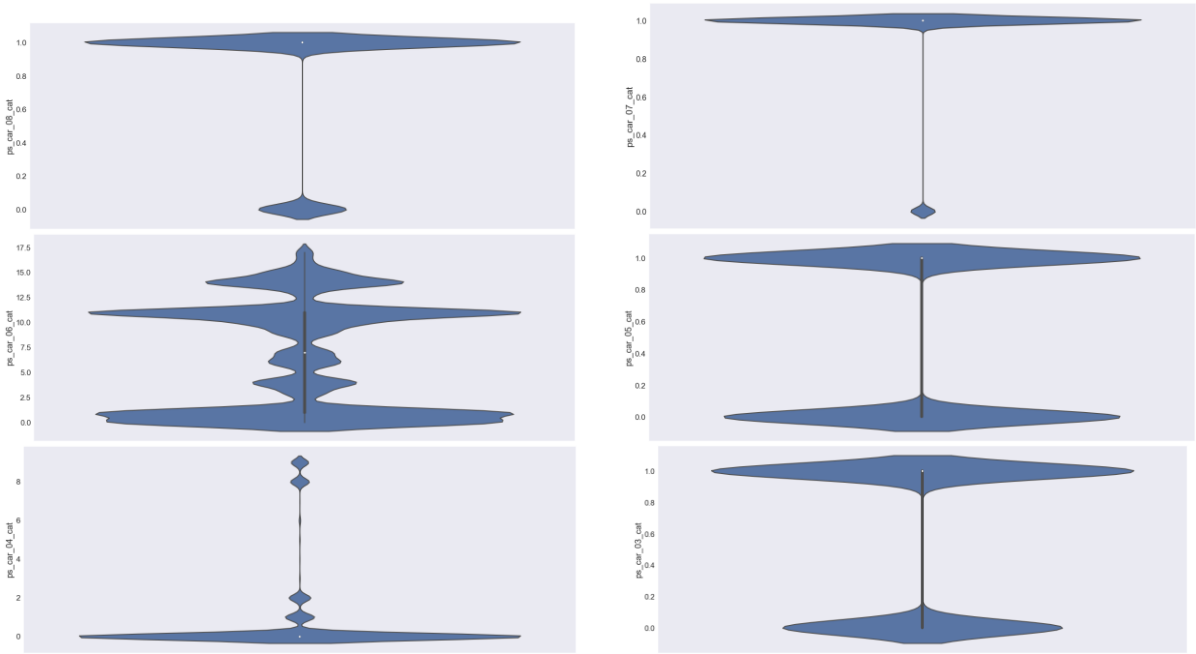
Figura 6: Visualização gráfica da frequência para cada categoria

.

A figura 6 permite verificar que:

- ps_car_11_cat possui a maior frequência;
- ps_car_01_cat e ps_car_06_cat possuem frequências notáveis;
- O conjunto de dados deste tipo é formado por muitas váriaveis que apresentam frequências baixas.

O gráfico Violin Plot será utilizado para mostrar a distribuição de dados quantitativos ao longo de vários níveis de cada varíavel categorica nominal.



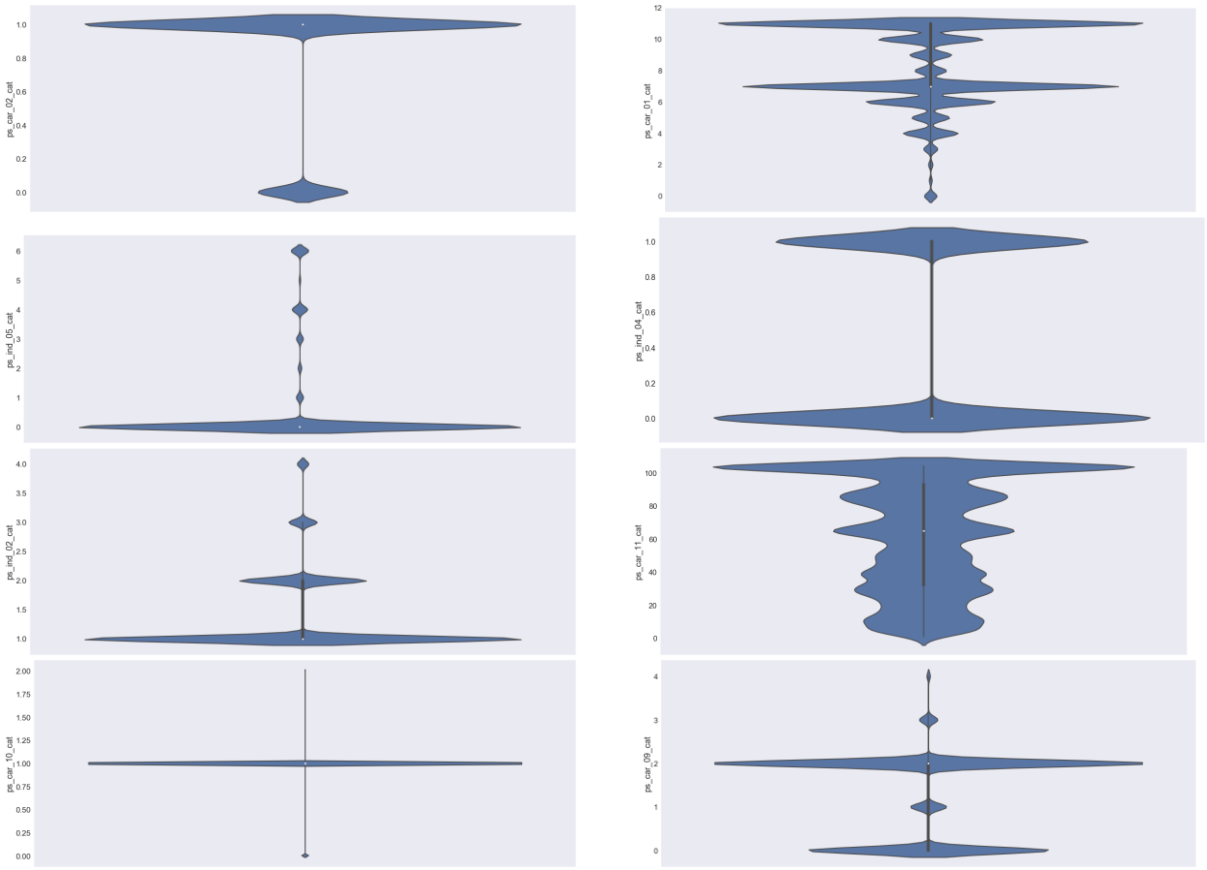


Figura 7: Distribuições das variáveis categóricas

A figura 7, permite verificar que:

- ps_car_10_cat apresenta praticamente quase toda sua distribuição no valor 1;
- ps_car_11 apresenta uma boa distribuição entre todos os seus valores, sendo a concentração em valores maiores que 100;
- ps_car_04_cat apresenta praticamente quase toda a sua distribuição no valor 0.

2.2.4 VARIÁVEIS INTERVALARES

Para as variáveis intervalares será utilizado um gráfico que mostra o pearson – coeficiente de correlação linear entre cada par de variáveis intervalares. Dessa forma é possível visualizar quais variáveis possuem correlação linear positiva ou negativa, uma com as outras. Depois será utilizado o gráfico de dispersão com todos os pares de variáveis que obtiveram valor de Pearson significativo. Conforme o gráfico abaixo, pode-se visualizar que as váriaveis que obtiveram valores significativos foram:

- ps_reg_02 and ps_reg_03 (0.74)
- ps_car_12 and ps_car13 (0.67)
- ps_car_12 and ps_car14 (0.60)
- ps_car_13 and ps_car15 (0.53)

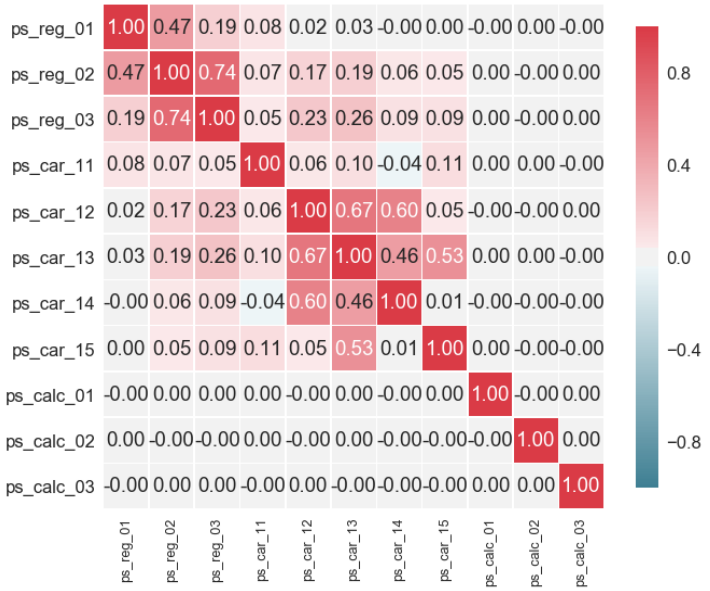
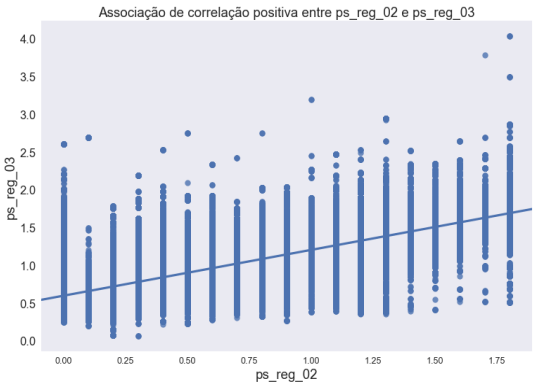
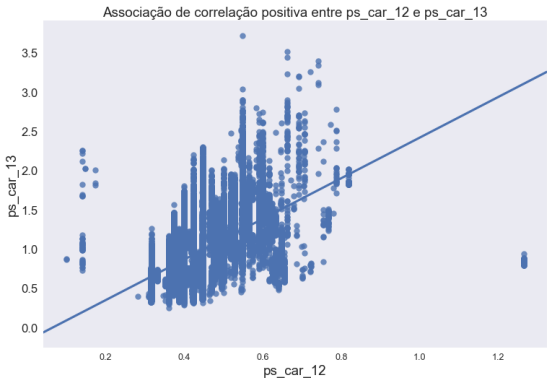


Figura 8: Correlação Linear entre cada par de variáveis intervalares

Os gráficos de dispersão permitem visualizar a tendência, a força e a forma de cada ponto ou conjunto de pontos distribuídos no gráfico. Cada ponto representa uma observação, e a localização desse ponto depende do valor das duas variáveis x e y. Este gráfico permite compreender como cada variável independente afeta o aumento ou a diminuição da variável dependente, e possibilita também verificar anomalias, outliers e conjuntos de pontos que não seguem o padrão. Para a criação dos gráficos foi utilizada a biblioteca Seaborn.

- ps_reg_02 and ps_reg_03 (0.74)
- ps_car_12 and ps_car13 (0.67)
- ps_car_12 and ps_car14 (0.60)
- ps_car_13 and ps_car15 (0.53)





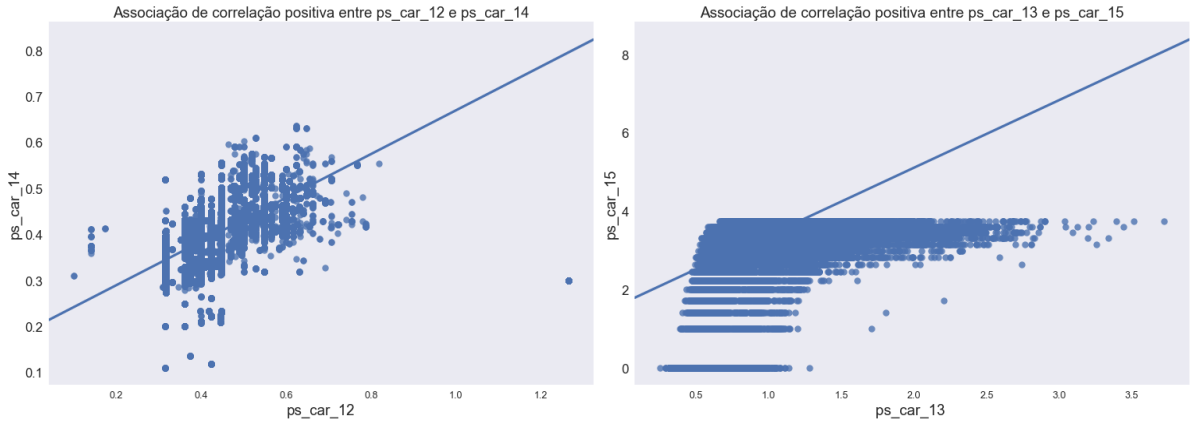


Figura 9: Gráficos de Dispersão para os pares de variáveis categóricas que mais apresentaram correlação linear

2.2.5 VARIÁVEIS ORDINAIS

Diferentemente das váriaveis intervalares, as váriaveis ordinais não apresentaram nenhum tipo de correlação linear significativa.

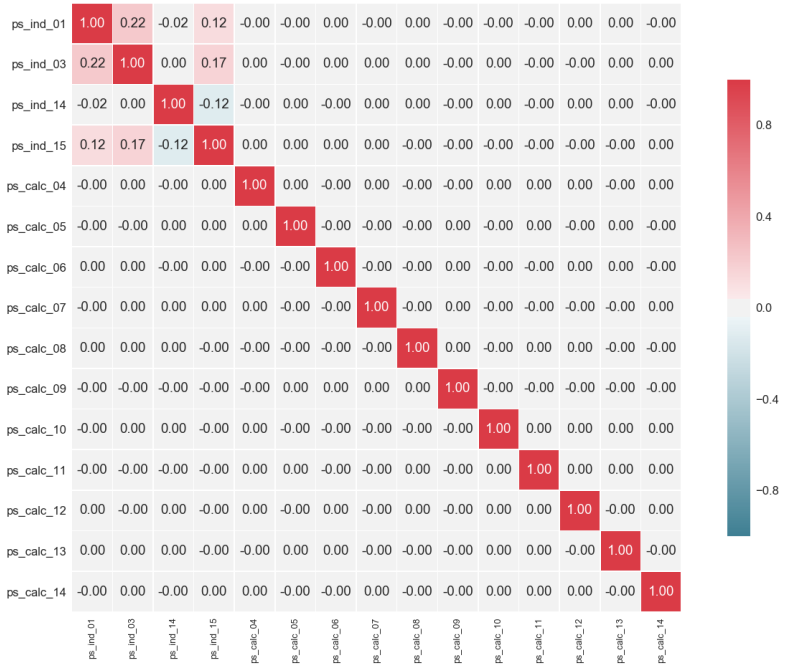


Figura 10: Distribuições das variáveis ordinais

2.3 ALGORITMOS E TÉCNICAS

A modelagem dos dados será feita com a utilização do Scikit-learn, biblioteca de Machine Learning. Serão construídos dois modelos de classificação preditivos para avaliação, o primeiro será construído a partir do algoritmo Decision Tree e o segundo com o algoritmo Random Forest. A partir da avaliação dos resultados, somente um modelo vai ser escolhido para ser aperfeiçoado.

Como o dataset apresenta um desbalançeamento entre as classes da variável alvo “target” e o problema é definido como uma classificação binária, será realizado

primeiramente uma técnica chamada *stratified k-folds* que vai possibilitar que os dados sejam rearranjados para garantir que cada *fold* represente bem o todo. Dessa forma, o conjunto de dados de treinamento irá ser dividido em: conjunto de dados de treino para construir o modelo e conjunto de dados de teste para avaliar se o modelo terá um bom desempenho com novos dados (*que não foram vistos antes). Ressalta-se que para o modelo de Decision Trees será necessário fazer a normalização dos dados, por meio da técnica StardandScaler.

DECISION TREES

Árvores de decisão são amplamente utilizadas para modelos de classificação e regressão. Elas aprendem a hierarquia de questões if/else, levando a uma decisão. Dessa forma, o algoritmo é designado a fazer escolhas. Cada uma das questões a serem respondidas corresponde a um nó na árvore de decisão, e cada nó tem ramos para uma possível resposta. Eventualmente, o algoritmo alcança o nó da folha que contém a correta classificação para a entrada.

VANTAGENS

- Árvores de decisão são simples de entender e interpretar, podem ser visualizadas e requerem pouca preparação de dados;
- Tem desempenho melhor quando configura o mínimo de amostras requeridas; no nó do node ou o max_depth da árvore para evitar o problema de overfitting.

DESVANTAGENS

- A partir do momento que a árvore vai cada vez mais crescendo ela tende a ter problema de sobre ajuste;
- Não manipulam bem dados não numéricos;
- Podem ser muito grandes.

RANDOM FOREST

O algoritmo Random Forest faz parte dos métodos *ensemble,* estes métodos partem da observação que múltiplos modelos dão uma performance melhor do que um único modelo, se o modelo é de alguma forma independente um do outro. Dessa forma, a chave desses modelos é desenvolver uma abordagem de algoritmo que gere vários modelos independentes que serão então combinados em um conjunto(*ensemble).* Random Forests é um modelo generalizo robusto e é uma modificação particular de árvores de decisão binária. Nesse modelo, um subconjunto de observações e um subconjunto de variáveis são escolhidos randomicamente para construir múltiplos modelos independentes baseados em árvore. As árvores são mais “não correlacionadas”, pois apenas um subconjunto de variáveis é usado durante a divisão da árvore, em vez de escolher avidamente o melhor ponto de divisão na construção da árvore.

VANTAGENS:

- É um dos algoritmos de aprendizagem mais precisos atualmente;

- Executa eficientemente em conjunto de dados grandes;
- Pode lidar com a entrada de muitas variáveis sem a necessidade de exclui-las.

DESVANTAGENS:

- Pode ocorrer *overfit* para alguns *datasets* com ruído para tarefas de classificação/regressão;
- Random Forests requer mais memória e é mais demorado para treinar e prever.

Para a construção de ambos algoritmos será alterado alguns parâmetros com o objetivo de melhorar o modelo, pois os valores do default não condizem com a necessidade do modelo. O modelo Random Forest, que é um modelo mais robusto do que o modelo de árvore de decisão, será o modelo final utilizado para resolução do problema.

2.4 BENCHMARK

Esta competição apresenta mais de 4.000 competidores, cada competidor que enviou sua submissão apresentou seu resultado como benchmark, e este resultado é avaliado por meio do coeficiente Gini.

- Maior Benchmark da competição: 0.291
- Menor Benchmark da competição: -0.245

Foram implementados dois classificadores de Decision Tree e outros de Random Forest.

Modelos Preditivos	Pontuação Gini - Kaggle
tree.DecisionTreeClassifier – clf1	0.035
tree.DecisionTreeClassifier – clf2	0.042
RandomForestClassifier (model_rf1)	0.066
RandomForestClassifier (model_rf2)	0.245
RandomForestClassifier (model_rf3)	0.248
RandomForestClassifier(model_rf4)	0.249

Tabela 2 : Performances de Benckmack no conjunto de teste (test.csv)

Após a implementação do modelo final, será realizada uma nova previsão da variável alvo “target” com a utilização do conjunto de testes (test.csv). O resultado obtido será enviado para a plataforma Kaggle e ela nos dará uma pontuação com base na métrica Gini.

3. METODOLOGIA

3.1 PROCESSAMENTO DE DADOS

Na sessão Exploração de dados foi necessário realizar um processamento dos tipos de variáveis presentes no dataset, entre elas variáveis binárias, nominais, intervalares e ordinais, que possuem diferentes formatos, níveis e grupos. Por possuirem essas diferenças se faz necessário a criação de metadados que contenham informações

sobre elas, permitindo dessa maneira, análise e visualização exploratória que irão possibilitar a análise das features e identificação de padrões.

Dessa forma as variáveis foram separadas em:

- 2 formatos: int64 ou float64;
- 3 níveis: id, input e target;
- 4 tipos: binaria, nominal, intervalar e ordinal;
- 5 grupos: ind, reg, car, calc e nogroup.

As variáveis categóricas já passaram por um pré-processamento feito pela Porto Seguro, dessa forma, elas não apresentam nenhum tipo “string”, somente números. Não sendo necessário dessa forma, realizar nenhuma transformação nelas. Optou-se também, por não transformar algumas colunas em *dummy variables* pelo motivo do dataset já conter muitas colunas.

Para a construção do modelo preditivo de Decision Tree foi necessário normalizar os dados antes da construção do modelo.

Observou-se que a coluna “id” representa a identificação dos clientes por conter a soma de valores diferentes igual a 595212 que é a mesma quantidade de linhas presente no dataset. Por este motivo, ela também será removida do conjunto de dados de treinamento e do conjunto de teste.

Na etapa de Exploração de Dados, os valores -1 foram substuídos por valores nulos com o objetivo de realizar a análise de dados. Mas agora nesta parte de implementação do modelo, os valores nulos foram substitúidos por – 1, para que o modelo tenha uma melhor adaptação e desempenho do modelo.

3.2 IMPLEMENTAÇÂO

Será construído dois modelos preditivos. O primeiro será construído com a utilização do algoritmo Decision Tree e o segundo com o Random Forest. Irá ser utilizado a métrica AUC para avaliar e aperfeiçoar os modelos. Ressalta-se que para o modelo Decision Tree foi necessário utilizar a técnica *StandardScaler* para normalizar os dados.

Será realizado, primeiramente, a divisão desse dataset em conjunto de dados de treinamento e conjunto de dados de teste. Como o dataset possui um grande desbalanceamento entre as classes binárias do target (*96,36% de output = 0), a divisão será feita utilizando da técnica *StratifiedKFold* com o objetivo de preservar o percentual de cada classe, possibilitando assim que os dados sejam rearranjados para garantir que cada *fold* represente bem o todo. Dessa forma, o conjunto de dados de treinamento irá ser dividido em: conjunto de dados de treino para construir o modelo e conjunto de dados de teste para avaliar se o modelo terá um bom desempenho com novos dados (*que não foram vistos antes). Os parâmetros dessa técnica foram: n_splits=3(*quantidade de folds igual a 3), random_state=2017 e shuffle=True. Após a divisão do conjuntos de dados, será construído dois modelos preditivos diferentes.
A realização da previsão do conjunto de testes, será feita com a utilização da função predict_proba para obter estimativas de incerteza para o classificador. Como é uma

probabilidade, o output do predict_proba é sempre entre 0 e 1 e a soma das entradas para ambas classes é sempre 1.

MODELO PREDITIVO - DECISION TREE

Primeiramente, foi utilizado o algoritmo de classificação da árvore de decisão: DecisionTreeClassifier com a alteração do parâmetro *default* de random_state=2017. Obteve-se um score de 0.498. Depois foi construído um segundo classificador com os seguintes parâmetros: max_depth = 6, min_samples_split= 60 , min_samples_leaf = 25 e random_state=2017, obteve-se um score de 0.590. O *score* obtido demostra que os testes foram úteis, porém são ruins.

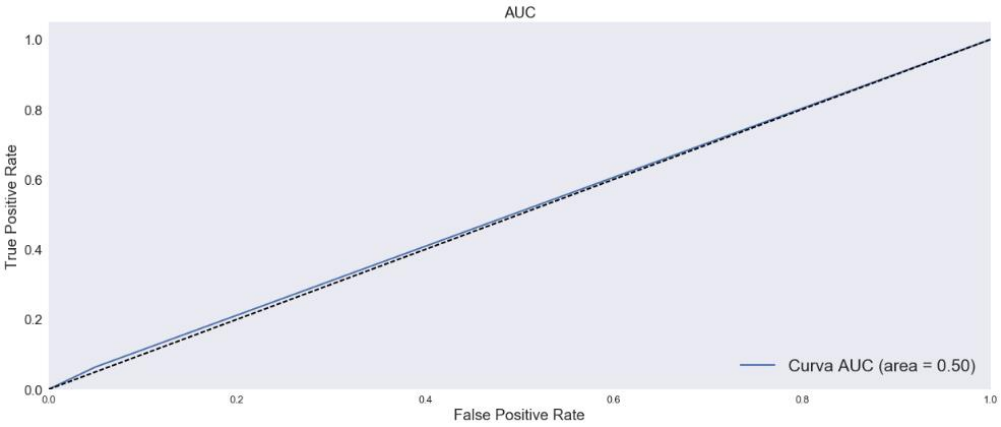


Figura 11: Representação gráfica da curva da métrica AUC para classificador cf1

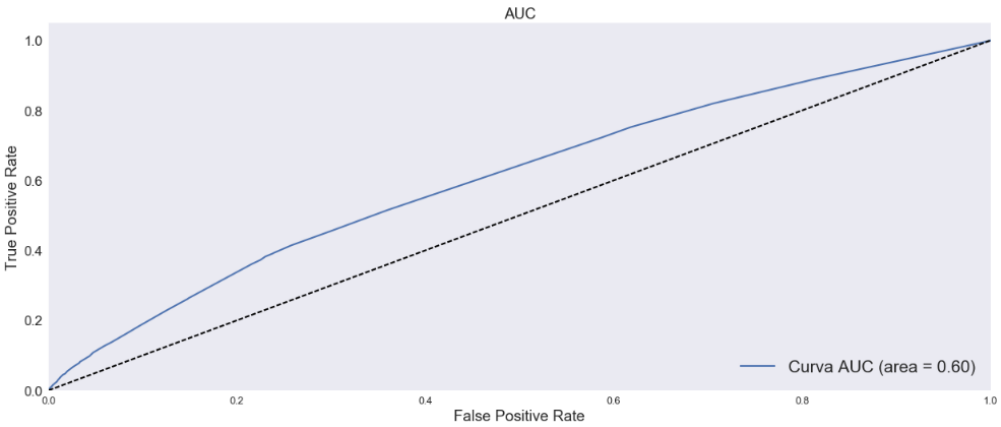


Figura 12: Representação gráfica da curva da métrica AUC para classificador cf2

MODELO PREDITIVO – RANDOM FOREST

Primeiramente, foi utilizado o algoritmo de classificação RandomForestClassifier com a alteração do parâmetro default de random_state=2017. Obteve-se um score de 0.534. Depois foi construído um segundo classificador com os seguintes parâmetros: n_estimators= 100, max_depth = 6, min_samples_split= 60 , min_samples_leaf = 25 e random_state=2017, obteve-se um score de 0.623. Os *scores* obtidos demostram que: para o primeiro classificador o teste foi útil e para o segundo classificador o teste foi suficiente.

Abaixo a representação gráfica da curva da métrica AUC:

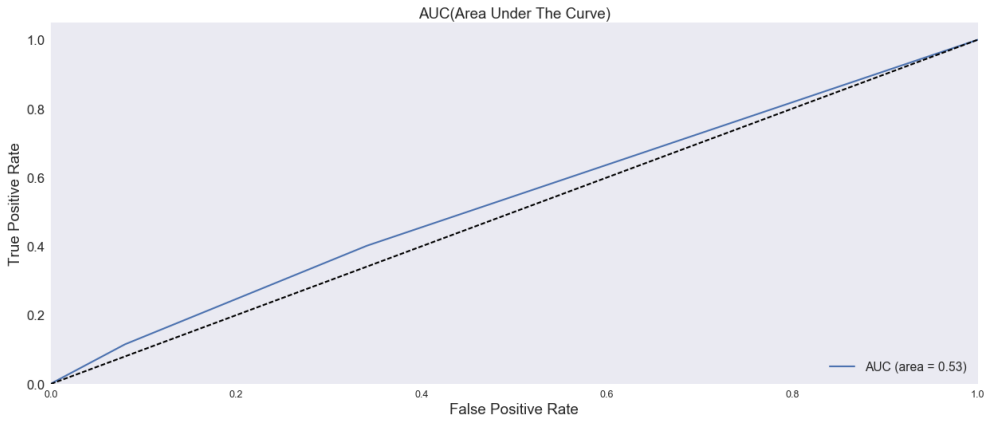


Figura 13: Representação gráfica da curva da métrica AUC para classificador rf1

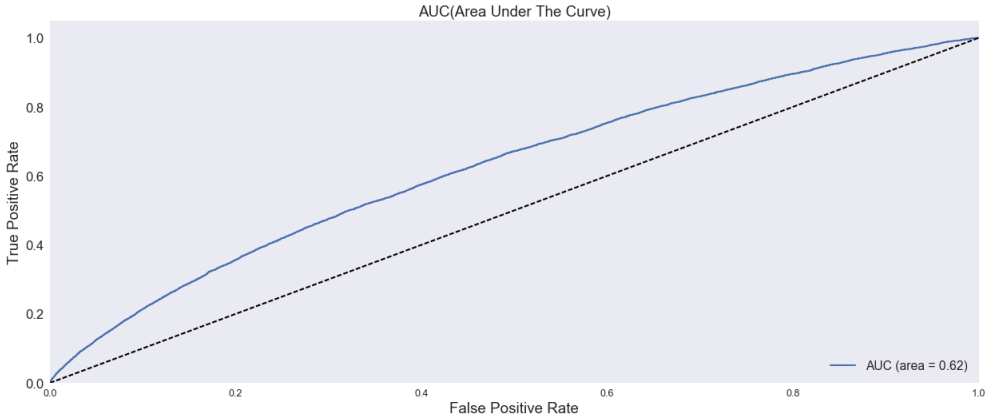


Figura 14: Representação gráfica da curva da métrica AUC para classificador rf2

Como esperado o modelo Random Forest obteve um score maior que o classificador do outro modelo, por este motivo e por ser um modelo mais robusto que oferece mais opções de parâmetros, ele será utilizado para a construção do modelo preditivo.

3.3 REFINAMENTO

Parâmetros que serão utilizados para aperfeiçoamento do modelo de Random Forest:

- n_estimators (default=10) : O n_estimators é o número de árvores no conjunto. Será necessário alteração do valor inicial para um bem mais alto, para assim conseguir uma performance melhor. Verificou que alterar este parâmetro para um valor maior que 100, possibilitou melhorar o *score* do modelo.

- max_depth (default=None) : O max_depth é a profundidade máxima da árvore. Se este parâmetro for definido como None, a árvore será cultivada até que todos os nós da folha sejam puros ou eles tenham menos exemplos que min_samples_split.

- min_samples_split (default=2): O min_samples_split é a quantidade minima de amostras necessárias para dividir o node. Os nodes não serão divididos se tiverem menos que min_samples_split exemplos. Nodes divididos que são pequenos são um fonte de overfitting.

- min_samples_leaf (default=1): O min_samples_leaf é a quantidade mínima de amostras na folha. Uma divisão não é tomada se a divisão conduzir a nós com menos min_samples_leaf.

O primeiro classificador Random Forest foi treinado com a exclusão das variáveis “target” e “id” e somente com o parâmetro random_state=2017, e foi obtido a pontuação AUC de 0.534 e pontuação Gini de 0.0679477536508. Para aperfeiçoar o modelo vai ser utilizados 3 diferentes valores para a quantidade de árvores, assim n_estimator vai ser igual a 100,200 ou 300.

Com n_estimatos=100, max_depth = 6 ,min_samples_split = 60 e min_samples_leaf = 25, o resultado foi suficiente e a pontuação gini de foi para 0.245464667613. Pontuação melhorou: 0.1775169139622

Aumentando o n_estimators para 300; max_depth = 6 ,min_samples_split = 60 e min_samples_leaf = 25, obteve uma pontuação gini no modelo rf4 de 0.247504078786.

Após isso, foi gerada por meio do modelo “modelo_rf4”, a importância das features dentro do modelo e as 20 principais variáveis foram: 'ps_car_13', 'ps_ind_05_cat', 'ps_reg_03', 'ps_ind_17_bin','ps_car_04_cat', 'ps_reg_02', 'ps_car_07_cat', 'ps_car_12', 'ps_ind_07_bin', 'ps_ind_06_bin', 'ps_ind_03', 'ps_car_03_cat' ,'ps_car_01_cat', 'ps_ind_16_bin', 'ps_ind_15', 'ps_car_15', 'ps_car_14', 'ps_car_02_cat', 'ps_reg_01', 'ps_ind_01'. Inicialmente, verificou a possibilidade de remover a variável “ps_car_03_cat “ que é aquela que apresenta 70% de valores faltantes, porém ela faz parte das 20 features que mais apresentam relevância para o modelo, e foi realizado também um teste com o modelo rf5 e verificou que a pontuação diminuiu. Por estes motivos, ela foi mantida no modelo preditivo.

Verificou-se alguns outliers para algumas das primeiras variáveis que são consideradas as mais relevantes para o modelo “modelo_rf4”. Criou-se uma função verificar_target que vai armazenar somente os outliers que contém target=0, pois se for armazenado outliers com target=1, isso diminuiria ainda mais a porcentagem de pessoas que acionaram o sinistro, por isso somente só será removido do modelo outliers que possuem "target"= 0. Após a remoção de alguns outliers da variável “ps_car_12” e treinamento do “modelo_rf5” foi obtido a pontuação de 0.253239448477. O modelo melhorou: 0.00573536969099997.

Foi realizada também a exclusão das 5 últimas colunas e depois das 10 últimas colunas, a pontuação não melhorou, por isso optou-se por não retirar as colunas menos importantes para o modelo.

Utilização da técnica Grid-Search para verificar se o melhor parâmetro “max_depth” . Por ser uma técnica que exige um grande tempo de processamento, não vai ser utilizado muitos parâmetros. Por isso, os parâmetros escolhidos foram:

- n_estimators = 300

- max_depth = 6 ou 8
- min_samples_split = 60
- min_samples_lef=25

Os melhores parâmetros retornados foram: 300, 8, 60 e 25. Alterando a profundidade do modelo para 8, a pontuação melhorou de 0.253239448477 para 0. 0.255060374577. Melhorou 0.0018209261

4. RESULTADOS

4.1 AVALIAÇÃO E VALIDAÇÃO DO MODELO

A métrica roc_auc foi utilizada para avaliar o modelo, em conjunto com a técnica cross-validation que possibilitou avaliar o quão bem o modelo consegue generalizar para dados não vistos. Como é uma classificação binária, por default, o cross-validation utilizará folds estratificados. Ressalta-se que houve uma diferença de valores significativa entre os scores dos diferentes folds, por este motivo optou-se por aumentar a quantidade padrão de folds para 5. A métrica Gini será calculada da seguinte forma: 2 * media (roc_auc) – 1.

Após estas etapas, o modelo final , “modelo_rf6”, realizou a previsão do target com o conjunto de teste(test.csv), a sample_submission foi criada e submetida para a competição da Kaggle que retornou a pontuação gini de 0.255.

4.2 JUSTIFICATIVA

Após submeter a sample_submission do modelo final para a competição, foi obtido um score de: 0.255. Este resultado pode ser considerado bom, visto que o maior resultado foi de: 0.291. Pela complexidade e falta de conhecimento de todo o problema, pode-se dizer que a solução não é totalmente eficiente para a solução do problema, até mesmo porque de acordo com a métrica gini quanto mais próximo de 0.5 melhor é o modelo preditivo. Nossa pontuação é muito melhor do que a pontuação mais baixa da competição que é de: -0.245 e o valor da diferença entre a pontuação obtida e a maior pontuação é de -0.036.

	Modelo – Benchmark model_rf4	Modelo - Final (model_rf6)
Gini	0.249	0.255

Tabela 3 : Performances de Benckmark no conjunto de teste (test.csv)

5. CONCLUSÃO

5.1 VISUALIZAÇÃO FORMA-LIVRE

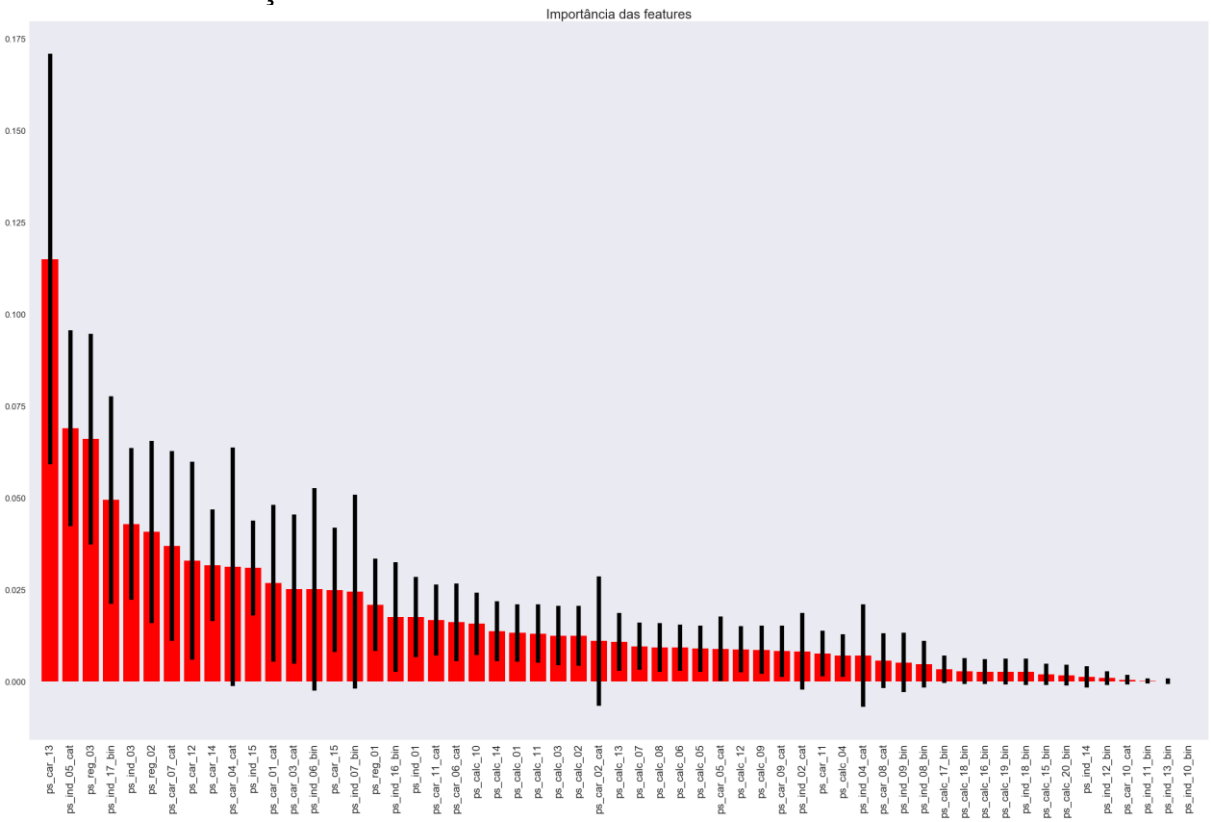


Figura 15 : Importância das Features do Modelo RF6

A importância das features mostra que:

- ps_car_13 foi a feature que teve maior importância no modelo.
- A maioria das features binárias do grupo “calc” e “ind” foram as que menos tiveram relevância. É possivel visualizar que a variável ps_ind_14 que apresenta o min, 25%, 50% e 75% = 0, também não obteve muita relevância.

A análise realizada com as 20 primeiras variáveis do modelo com a utilização dos metadados, possibilitou a obtenção dos seguintes resultados:

- O grupo car e ind foram os grupos que obtiveram a maior concentração de variáveis, apresentaram respectivamente, a quantidade de 9 e 8.
- O grupo reg apresentou somente 3 variáveis e o grupo calc não teve nenhuma variável.
- A quantidade de variáveis intervalares foi a maior com 7, seguida da nominal com 6, da binária com 4 e da ordinal com 3.
- A quantidade de variáveis int64 foi igual a 13 e a de float64 foi é igual a 7.

5.2 REFLEXÃO

Os resultados obtidos na competição comprovaram que é um problema de classificação binária. Os algoritmos utilizados no modelo preditivo, provaram apresentar testes úteis e serviram para resolver o problema, é claro que o algoritmo

Random Forest por pertencer aos tipos de algoritmos Ensembled, apresentou um desempenho bem melhor que um algoritmo mais simples como é o caso do Decision Tree. Os modelos preditivos de Decision Tree rodaram bem mais rápido que os de Random Forest, este modelo demora muito mais para rodar, e se a quantidade de árvores for aumentada e a profundidade também demora mais ainda. Pode-se dizer que não é adequado, com o recurso computacional utilizado, usar a técnica de Grid-Search para verificar o melhor parâmetro neste modelo, pois demora-se muitas horas para verificar os melhores parâmetros dados alguns conjuntos. A maior parte do tempo, diga-se 70%, foi gasto na etapa de exploração e processamento de dados, visto que o dataset possui diferentes tipos de variáveis e estas deveriam ser separadas para realizar a análise de dados.

5.3 MELHORIA

Os resultados obtidos utilizando o algoritmo RandomForest mostrou ser mais eficiente em relação ao modelo mais simples de Decision Tree. O modelo atual já passou por algumas melhorias, entre elas: alteração dos parâmetros, identificação e exclusão de outliers e exclusão de algumas colunas consideradas menos relevantes. Para melhorar ainda mais o modelo preditivo, poderia ser feito algumas melhoras como:

- Testar mais valores de parâmetros, como por exemplo, para n_estimators e a max_depth;
- Realizar o treinamento do modelo somente com as variáveis mais relevantes;
- Realizar a identificação de mais outliers, principalmente das variáveis que tem maior importância para o modelo, e medir se o modelo vai melhorar ou não;
- Realizar a retirada de mais colunas que não apresentam muita importância para o modelo;
- Utilização de *meta-modeling* que combina resultados de diferentes modelos. Mencionou-se a utilização deste tipo de solução para resolução deste problema, vale a pena testar isso em uma solução de um problema futuro.

6.REFERÊNCIAS

SIMUNDIC, Ana Maria. **Diagnostic accuracy – Part 1 Basic concepts: sensitivity and specificity, ROC analysis, STARD statement.** 2009.Disponível em: https://acutecaretesting.org/en/articles /diagnostic-accuracy--part-1brbasic-concepts-sensitivity-and-specificity-roc-analysis-stard-statement.

SUGIYAMA, Masashi. **Introduction to Statistical Machine Learning**, Morgan Kaufmann,pp.524,2016

JAWLIK, Andrew A. **Statistics from A to Z: Confusing Concepts Clarified.** John Wiley & Sons Inc, pp.418, 2016.

GUIDO, S. and Müller, A. **Introduction to Machine Learning with Python**. United States of America, O'Reilly Media, pp.376, 2017.

MCKINNEY, Wes. **Python for Data Analysis**. United States of America, O'Reilly Media, pp.451, 2013.

BOWLES, Michael. **Machine Learning in Python: Essential Techniques for Predictive Analysis**, Wiley,pp.360, 2015.

SWAMYNATHAN, Manohar .**Mastering Machine Learning with Python in Six Steps - 1E (2017).** Apress, pp.258, 2017.

GERON, Aurelien **.Hands-On Machine Learning with Scikit-Lear.** O'Reilly Media, pp.516,2017.

Bert Carremans.Data Preparation & Exploration. <https://www.kaggle.com/ bert carre mans/data-preparation-exploration

https://github.com/felipeeeantunes/udacity_live/blob/master/porto_seguro.ipynb
http://seaborn.pydata.org/examples/many_pairwise_correlations.html
http://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html
https://svds.com/learning-imbalanced-classes/
https://www.kaggle.com/c/porto-seguro-safe-driver-prediction#evaluation
http://amateurdatascientist.blogspot.com.br/2012/01/random-forest-algorithm.html
https://stats.stackexchange.com/questions/132777/what-does-auc-stand-for-and-what-is-it
http://gim.unmc.edu/dxtests/roc3.htm
http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc_crossval.html#sphx-glr-auto-examples-model-selection-plot-roc-crossval-py
http://www.aihorizon.com/essays/generalai/decision_trees.htm
https://svds.com/learning-imbalanced-classes/
http://gim.unmc.edu/dxtests/roc3.htm
http://seaborn.pydata.org/examples/pairgrid_dotplot.html
https://seaborn.pydata.org/generated/seaborn.stripplot.html
http://seaborn.pydata.org/generated/seaborn.pointplot.html
http://seaborn.pydata.org/tutorial/categorical.html
http://seaborn.pydata.org/generated/seaborn.violinplot.html#seaborn.violinplot
https://seaborn.pydata.org/generated/seaborn.boxplot.html#seaborn.boxplot
https://en.wikipedia.org/wiki/Simple_random_sample
http://seaborn.pydata.org/tutorial/categorical.html
http://www.mymarketresearchmethods.com/types-of-data-nominal-ordinal-interval-ratio/
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html
https://stats.stackexchange.com/questions/49540/understanding-stratified-cross-validation