Nanodegree Engenheiro de Machine Learning

Relatório de Projeto Final

Rafael Novello
21 de março de 2018

Definição

Visão Geral do Projeto

Este documento compõe o relatório do projeto final do curso Nanodegree Engenheiro de Machine learning. O projeto foi desenvolvido usando de um desafio da plataforma Kaggle que consiste na classificação de lances realizados por pessoas e robôs em um leilão digital.

O leilão é uma forma de negociação, venda ou compra, de bens ou serviços muito utilizada. A dinâmica consiste, de forma muito resumida, na oferta de um item ou lote ao qual é estabelecido um preço mínimo inicial. Os interessados no item fazem suas ofertas (lances) por um determinado período e adquire o item aquele que fizer o maior lance até o fim do período do leilão.

Uma modalidade de leilão muito popular no Brasil é o Pregão Eletrônico que é realizado pelo governo em processos de licitação (aquisição de produtos ou serviços pelo governo). O pregão tem uma dinâmica inversa ao leilão. Neste, o governo anuncia publicamente a necessidade de compra ou contratação e os interessados competem entre si para alcançar o menor preço.

Segundo dados no Ministério do Planejamento, em 2013 o pregão eletrônico proporcionou uma economia de 9,1 bilhões de reais nas compra públicas do governo federal [1].

Descrição do Problema

Como descrito na seção anterior, o objetivo do leilão, assim como, e principalmente, do pregão eletrônico é permitir a competição igualitária entre os interessados. Porém, o uso de robôs (softwares que automatizam tarefas, neste caso o envio do lance ao leilão ou pregão) vem prejudicando sites de leilão e o pregão eletrônico do governo brasileiro, como podemos ver em várias matérias jornalísticas [2], [3], [4]

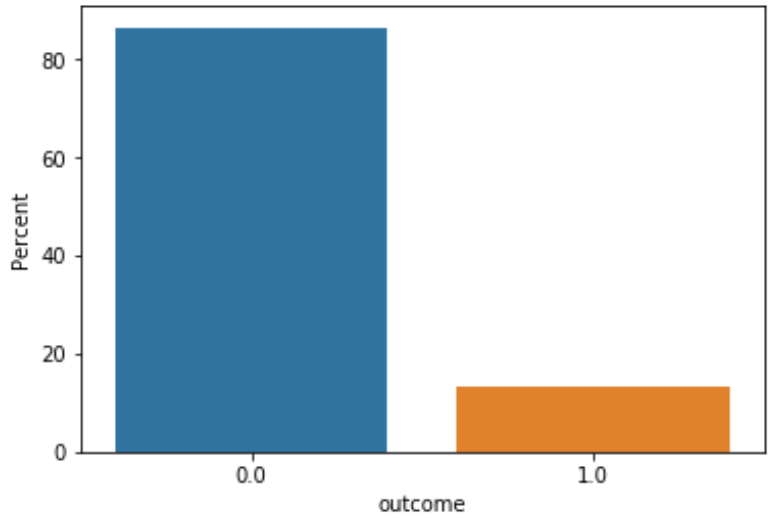
Identificar lances feitos por robôs é um grande desafio para plataformas de leilão, que precisam garantir a igualdade entre os “competidores”. Apenas assim as plataformas serão capazes de evitar que todos os ganhadores serão sistemas automatizados.

Para a identificação dos lances feitos por robôs e por operadores humanos vamos usar uma base de lances históricos feitos em uma plataforma de leilão online. Nesta base os operadores foram categorizados em robôs e humanos e desta forma podemos identificar todos os lances feitos por cada categoria de operador. Esta será nossa entrada.

O resultado esperado é a identificação dos lances feitos por cada uma das categorias de operadores através de uma variável binária, onde o valor 1 representa um lance feito por robô e o valor 0 representa um lance feito por humano. Esta será nossa saída.

Para obter o resultado esperado será usado um modelo de aprendizagem supervisionada, um classificador binário, que será melhor detalhado a frente.

Métricas



Como apresentado na imagem acima, a quantidade de amostras pertencentes a categoria 0 (humanos) é muito maior que a categoria 1 (robôs). A proporção das categorias no dataset é de 87% lances válidos e 13% de lances feitos por robôs.

Devido a natureza desbalanceada dos dados e a importância de não descartar erroneamente um lance feito por um usuário real da plataforma, as métricas de avaliação usadas foram Precision, Recall e, para auxiliar, foi usando ROC (Receiver Operating Characteristic ou Características operacionais do receptor em tradução livre).

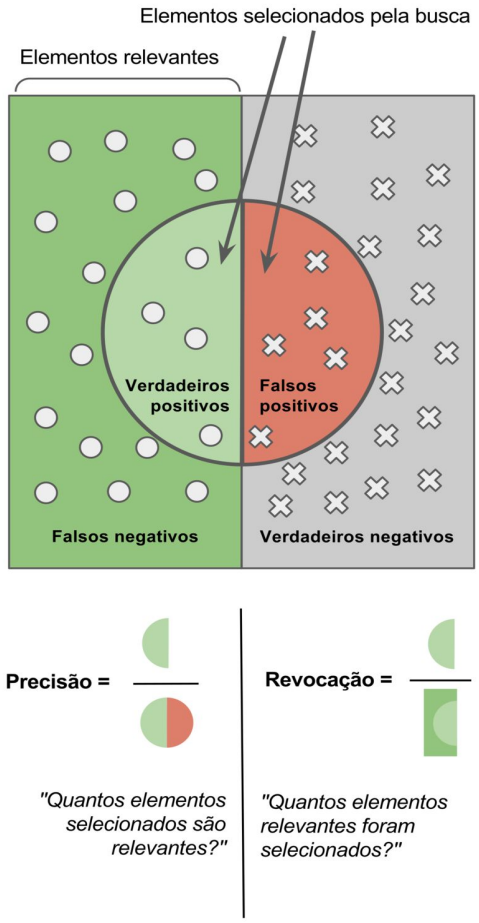
Precision e Recall foram usados por lidarem bem com conjuntos desbalanceados e por permitirem comparar objetivamente os resultados de cada modelo testado. O índice ROC foi usado para avaliar a performance do modelo criado em relação ao desafio na plataforma Kaggle.

As métricas de Precision e Recall, Precisão e Revocação em tradução livre, são métricas relacionadas a relevância dos resultados obtidos. Do ponto de vista probabilístico, Precisão mede as chances de um resultado obtido ser relevante enquanto a Revocação mede as chances de um resultado relevante ser obtido. Em outras palavras, Precisão nos diz quantos elementos selecionados são relevantes e Revocação dos diz quantos elementos relevantes foram selecionados conforme a imagem abaixo.

Como o dataset está organizado com o label 1 para os lances feitos por robôs será preciso definir para as métricas Precision e Recall a label positiva como 0 (zero), assim estaremos “procurando” pelos lances feitos por usuários reais e não o contrário.

Com este entendimento e as definições de Precision e Recall, podemos concluir que para os objetivos deste projeto é melhor termos um Recall mais alto do que o Precision, isso porque entendemos

que descartar um lance feito por um usuário real devido a um falso negativo é mais danoso do que considerar real um lance feito por um robô (falso positivo).



Análise

Exploração dos dados

Os dados disponibilizados na plataforma Kaggle para este desafio [5] estão organizados da seguinte forma:

Para o conjunto de dados do licitante (participante do leilão)

- bidder_id - Identificador exclusivo de um licitante.
- payment_account - conta de pagamento associado a um lance. Estes são obscuros para proteger a privacidade.
- endereço - endereço de correspondência de um licitante. Estes são obscuros para proteger a privacidade.
- Resultado - Etiqueta de um licitante que indica se é ou não um robô. O valor 1.0 indica um robô, onde o valor 0.0 indica humano.

Para o conjunto de dados de licitação

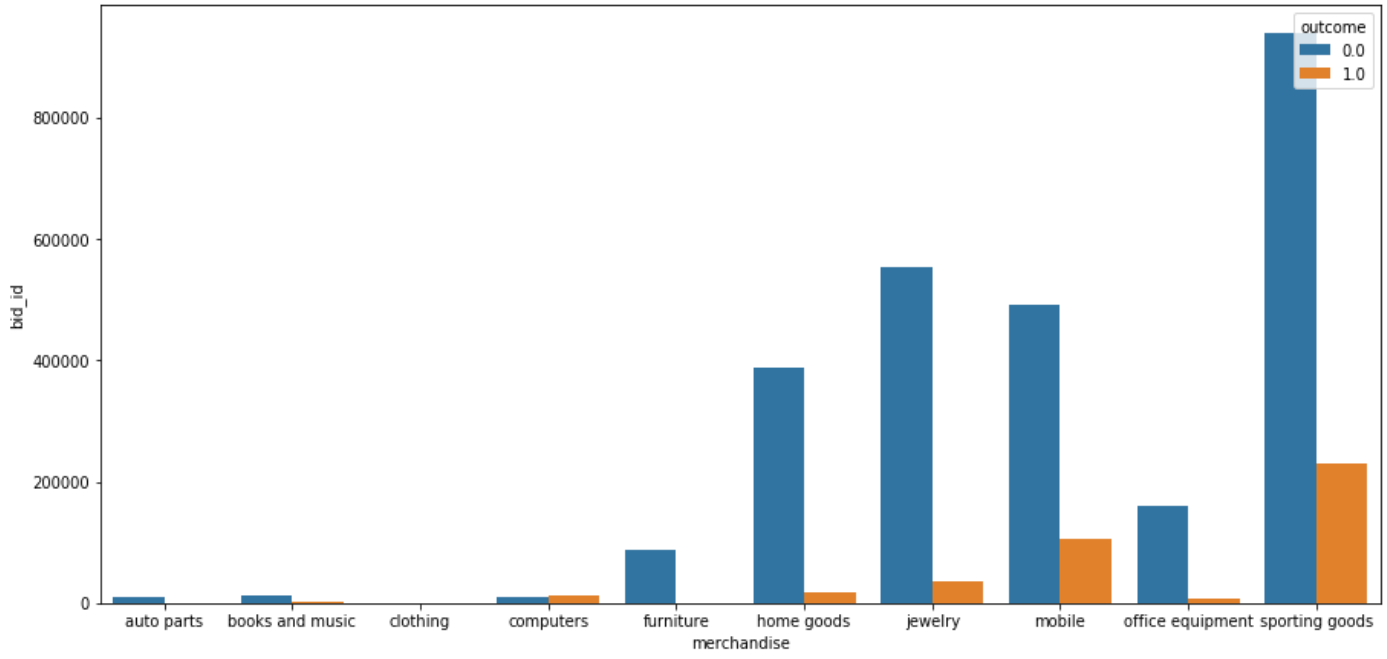
- bid_id - ID exclusivo para este lance
- bidder_id - Identificador exclusivo de um licitante (o mesmo que o bidder_id usado no train.csv e test.csv)
- Leilão - Identificador exclusivo de um leilão

- mercadoria - A categoria da campanha do site de leilões, o que significa que o licitante pode chegar a este site por meio da busca de "bens domésticos", mas acabou por licitar "bens esportivos" - e isso leva a que esse campo seja "bens domésticos". Este campo categórico pode ser um termo de pesquisa ou propaganda online.
- dispositivo - modelo de telefone de um visitante
- Hora - Tempo em que a oferta é feita (transformada para proteger a privacidade).
- país - O país ao qual o IP pertence
- IP - endereço IP de um licitante (ofuscado para proteger a privacidade).
- url - url do qual o licitante foi encaminhado (ofuscado para proteger a privacidade).

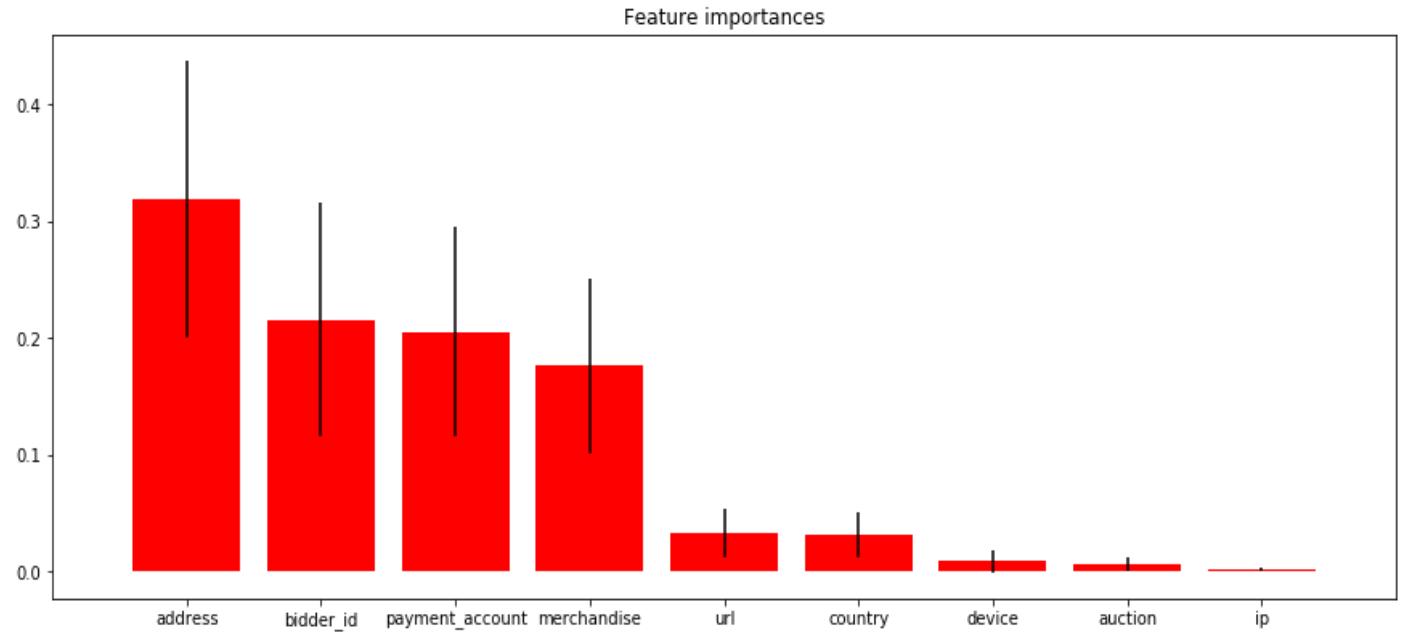
A base de lances contém 7.656.334 registros que são divididos em 3.071.224 no conjunto de treinamento (que deve ser dividido em treinamento e validação) e 4.585.110 no conjunto de testes. O conjunto de testes não contém a variável alvo, que deve ser inserida pelo resultado do modelo treinado para submissão na plataforma Kaggle e avaliação.

Na base de treinamento a proporção de lances feitos por humanos é de 87% e 13% feitos por robôs, o que mostra que o dataset está desbalanceado e por isso deve-se tomar algumas precauções.

Para o melhor entendimento dos dados, as features foram apresentadas separando por lances feitos por humanos e por robôs o que gerou a visualização a seguir:



Esta visualização mostra a diferença de concentração dos lances feitos por robôs por cada feature, o que levou a uma avaliação da importância de cada feature para a distinção entre as categorias.



Esta visualização mostra que apenas 4 das 9 features realmente contribuem para a distinção entre as categorias, o que indica a necessidade de aplicação de técnicas de feature engineering para agregar mais informação ao dataset. Tais técnicas de feature engineering podem ser muito avançadas e fogem do escopo do projeto mas são uma sugestão de próximos passos.

Algoritmos e técnicas

Por se tratar de um problema de classificação entre duas categorias onde existe uma base para treinamento com a variável alvo preenchida, foi usada uma abordagem de aprendizado supervisionado, onde o(s) algoritmo(s) são previamente treinados com esta base classificada e depois avaliado(s) com uma amostra sem a variável alvo.

Em relação aos algoritmos usados, foram testados com sucesso os seguintes algoritmos:

- **Nearest Neighbors**: Os métodos baseados em “vizinhos mais próximos” (tradução livre) podem ser usados em tarefas de regressão e classificação e consistem em encontrar um número pré-definido de amostras de treinamento mais próximas da distância do novo ponto e prever a variável target a partir destes. Quando usado para classificação, como neste projeto, o novo ponto é classificado pela categoria da maioria de seus vizinhos.
- **Random Forest**: Baseado em árvores de decisão, este método usa um conjunto de árvores de decisão que são construídas durante o treinamento do modelo, treinadas com diferentes amostras dos dados de treinamento e, quando usado para classificação, o resultado da predição do modelo é a moda das predições das árvores que foram criadas.
- **Neural Net**: Inspiradas no cérebro animal/humano, as redes neurais usam elementos chamados perceptrons (ou apenas neurônios) para formar uma estrutura em camadas onde, normalmente, cada neurônio de uma camada se conecta a todos os outros neurônios da camada seguinte. A cada uma destas conexões é atribuído um peso que passa a ser ajustado em um processo

chamado backpropagation. Cada neurônio uma uma função não linear para calcular sua saída, que passa a ser a entrada para os neurônios da camada seguinte com a ajuda de seu peso.

- **AdaBoost**: O princípio usado neste algoritmo é treinar um conjunto de modelos mais simples, como pequenas árvores de decisão, e o resultado do modelo é a combinação entre os resultados destes pequenos modelos através de soma ou voto majoritário ponderado. Assim como Random Forest o modelo de AdaBoost se enquadra nos algoritmos de ensemble.
- **Naive Bayes**: Pertencente a família de classificadores probabilísticos que se baseiam na aplicação do teorema de Bayes pressupondo a independência das features. Em resumo, um classificador de Naive Bayes assume que a presença de uma característica (feature) em uma categoria não está relacionada a qualquer outra característica. Mesmo que estas características dependam uma das outras todas elas contribuem independentemente para a probabilidade da classificação.
- **Quadratic Discriminant Analysis**: Também pertencente a família de classificadores probabilísticos, o QDA (acronimo) assume que as medidas de cada classe são distribuídas de forma normal. O QDA não supõe que exista uma covariância de cada classe seja idêntica.

O objetivo em treinar uma variedade de algoritmos distintos era observar aquele que apresentaria a melhor performance na classificação dos dados de treinamento, para então otimizar os parâmetros daquele que apresentasse os melhores resultados.

Benchmark

Por se tratar de um desafio na plataforma Kaggle, é provável que a melhor referência para comparação seja o ranking associado ao desafio. Neste ranking [6] o melhor modelo teve uma taxa de acertos de 94,3% e as melhores 50 submissões ficaram acima de 92,3% de acerto. A plataforma Kaggle usa a métrica ROC (Receiver Operating Characteristic) como avaliação dos resultados apresentados pelos competidores.

Para apoiar durante o processo de treinamento e escolha do melhor modelo, foi utilizar o algoritmo Dummy Classifier da biblioteca SciKit-Learn [7] como baseline de comparação.

Metodologia

Pré-processamento de dados

Dada a existência de variáveis categóricas no dataset de treinamento, foi necessário aplicar um processo de encoding nas mesmas. Este processo foi feito com a ajuda da classe LabelEncoder da biblioteca SciKit-Learn [8].

As features *bid_id*, *time, ip, auction* e *device* foram removidas do dataset de treinamento que ficou com as seguintes features *bidder_id, merchandise, country, url, payment_account* e *address*. Este processo foi realizado com a ajuda de um algoritmo de árvore de decisão que explicitou quais features mais relevantes para a classificação, como apresentado da seção de Análise.

Implementação

O projeto iniciou com a análise dos dados de treinamento, verificando se existe ausência de dados para alguma feature e fazendo algumas visualizações como para a distribuição entre os lances feitos por robôs e por humanos. O objetivo era saber qual a proporção do desbalanceamento do dataset entre os lances feitos por humanos e por robôs.

Em seguida foi verificada a correlação entre as features do modelo e a variável alvo. Neste processo foi utilizado um algoritmo de árvore de decisão para avaliar as features mais relevantes para a classificação.

Na sequência foi aplicado o pré-processamento descrito na seção anterior, transformando features categóricas com um processo de encoding. Este processo é necessário para a maioria dos modelos de machine learning.

Após o pré-processamento foi possível testar os modelos descritos na seção “Algoritmos e técnicas” com um subset dos dados de treinamento. Para esta fase foram utilizados os parâmetros padrão de cada modelo, pois o objetivo é descobrir apenas qual deles apresenta melhor taxa de acertos nas classificações.

Ao descobrir o melhor modelo, a última etapa foi a otimização do mesmo. Neste processo foi usada a classe RandomizedSearchCV da biblioteca Scikit-Learn para encontrar o melhor conjunto de parâmetros para o modelo. Este processo consiste em treinar várias versões do modelo escolhido com diferentes combinações de hiperparâmetros com o objetivo de selecionar aqueles que compõem a melhor versão do modelo escolhido.

Resultados

Modelo de avaliação e validação

Os modelos citados na seção “Algoritmos e técnicas” foram treinados usando os parâmetros padrão e foi obtido dos mesmos as métricas ROC score, Precison e Recall. Como é possível ver na tabela abaixo, os modelos usando os algoritmos Nearest Neighbors e AdaBoost apresentaram os melhores resultados usando os parâmetros padrão.

name	ROC score	precision	recall
QDA	0.825217	0.915222	0.985079
AdaBoost	0.975941	0.950072	0.992688
Nearest Neighbors	0.984086	0.990570	0.994437
Naive Bayes	0.763995	0.889778	0.999812
Random Forest	0.975583	0.935616	0.999927
Neural Net	0.501161	0.865418	1.000000

O modelo escolhido foi o Nearest Neighbors pois o mesmo apresentou o melhor conjunto Precision/Recall entre os modelos testados, mesmo ficando um pouco atrás do Neural Net especificamente em Recall . Esta escolha foi tomada pois tanto o modelo Nearest Neighbors quanto o Random Forest tiveram ótimos resultados nas métricas ROC e Precision mas o modelo escolhido apresentou melhor Recall e, como abordado na seção Métricas, um modelo com melhor Recall atende melhor às expectativas apresentadas.

Justificativa

O modelo obtido foi comparado com um modelo de baseline criado com a ajuda do algoritmo Dummy Classifier da biblioteca SciKit-Learn e com o scores apresentados no desafio Kaggle. Abaixo seguem os resultados das comparações:

Métrica / Modelo	Dummy Classifier	Nearest Neighbors Classifier
Precision	0.865	0.990
Recall	0.865	0.994
ROC Score	0.499	0.984

Quando comparado com os modelos submetidos no desafio Kaggle, o modelo Nearest Neighbors performou bem já que as melhores 50 submissões ficaram acima de 92,3% de ROC Score.

Otimização

Tendo escolhido o modelo, o próximo passo é a otimização do mesmo que é feita com a ajuda do pacote SciKit-Learn.

Conforme documentado na seção Implementação, o modelo escolhido é otimizado testando-se várias combinações de hiperparâmetros com o objetivo de encontrar aqueles que configuram a melhor versão do modelo para a tarefa em específico.

Os parâmetros e valores usados são os descritos abaixo:

```
params = {
    'n_neighbors': [3, 7],
    'weights': ['uniform', 'distance'],
    'algorithm': ['ball_tree', 'kd_tree'],
    'p':[1, 2]
}
```

- **n_neighbors:** Número de vizinhos usados por padrão para consultas de kneighbors;
- **weights:** Função de peso usada na previsão. Os valores são:
    - uniform: Todos os pontos em cada região tem o mesmo peso;
    - distance: Pontos de peso são o inverso de sua distância, quanto mais próximo maior o peso.

- **algorithm:** Algoritmo usado para calcular os vizinhos mais próximos.
- **p:** Parâmetro de potência para a métrica Minkowski.

Após a otimização, os valores selecionados pela classe RandomizedSearchCV como sendo os mais otimizados para este modelo foram:

- **n_neighbors:** 3
- **weights:** distance
- **algorithm:** ball_tree
- **p:** 1

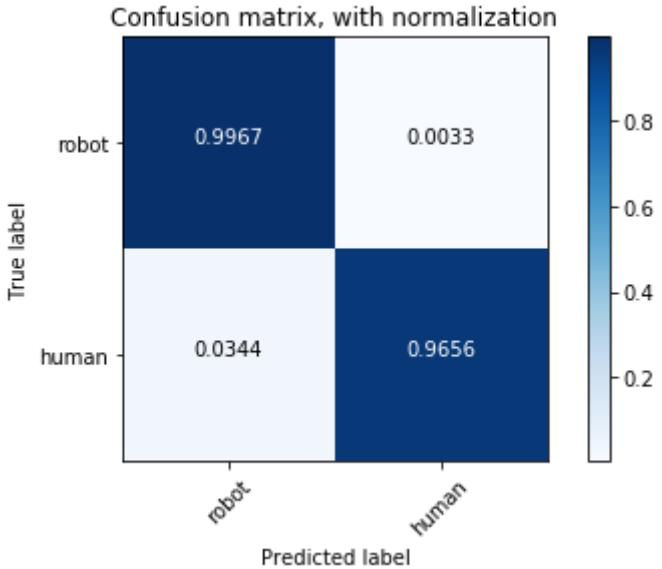
Após a otimização do modelo, houve uma pequena melhora em todas as métricas de avaliação escolhidas, que pode ser vista abaixo, o que comprova o sucesso da operação. A diferença diminuta antes e após a otimização pode ser explicada pelos altos valores iniciais que deixaram uma pequena margem de melhora.

Métricas	Antes	Depois	Diferença em %
Precision	0.990	0.995	0.5%
Recall	0.994	0.997	0.3%
ROC score	0.984	0.989	0.5%

Conclusão

Reflexão

Após a otimização do modelo podemos visualizar uma matriz de confusão com o objetivo de verificar quantos falsos negativos temos na solução final.



Vemos que o modelo otimizado teve apenas 0.03 (3%) de falsos negativos, que são lances de usuários reais que foram classificados como lances de robôs, e apenas 0.003 (0.3%) de falsos positivos, lances de robôs classificados como de humanos. Para os objetivos deste projeto e conforme explicado anteriormente, um menor número de falsos negativos seria ideal, apesar de 3% ser um valor que poderia ser tratado como aceitável.

Tendo em vista o problema inicial, de distinção de lances feitos por robôs em um leilão eletrônico, a solução obtida neste projeto poderia ser aplicada em um ambiente “de produção” desde que a taxa de 3% de falsos negativos seja aceitável para o negócio, o que pode ser possível no caso de uma plataforma de leilões privada de baixo impacto e prioridade. No caso de um pregão eletrônico como o do governo federal, citado como exemplo na seção “Definição”, uma solução mais precisa e robusta seria mais adequada, visto o grande impacto do mesmo na sociedade como um todo.

Ao escolher este desafio Kaggle a impressão inicial é que este seria um projeto relativamente simples pois, apesar do grande volume de dados, trata-se de uma classificação binária entre lances feitos por pessoas reais e por robôs autômatos em um leilão online. Porém, ao iniciar a manipulação dos dados surgiu o primeiro desafio, o dataset desbalanceado. Este foi um aspecto interessante e desafiador do projeto que vale a pena ser melhor abordado como sugestões de melhorias futuras.

Outro ponto bastante interessante deste projeto foi a necessidade de engenharia de features. Em consulta no fórum da plataforma Kaggle e na primeira avaliação do mentor da Udacity ficou claro a necessidade de se aprofundar em técnicas de engenharia de features que fogem ao escopo do projeto de capstone mas que podem ser aplicadas futuramente.

Por final, apesar das dificuldades, os resultados obtidos foram satisfatórios em vista das técnicas aplicadas e o projeto permitiu a consolidação e aprofundamento dos conhecimentos adquiridos neste Nanodegree.

Melhorias

Como apontado na seção anterior, os principais pontos de melhoria observados estão relacionados ao desbalanceamento do dataset e a aplicação de técnicas de engenharia de features e estes pontos serão discutidos mais detalhadamente abaixo:

- Sobre o desbalanceamento do dataset: Trabalhar com datasets desbalanceados representa um grande desafio e durante os estudos para a solução deste projeto foram observadas duas principais técnicas para lidar com este cenário que são o Oversampling e undersampling [9]. No primeiro, o objetivo é aumentar a quantidade de registros da categoria com menos amostras enquanto no segundo o objetivo é o oposto e aplicado à categoria com mais amostras. Estas técnicas podem ser aplicadas a este projeto para balancear o dataset e assim obter melhores resultados.
- Sobre a aplicação de engenharia de features: O objetivo das técnicas de engenharia de features é, em resumo, aumentar a quantidade de features utilizando técnicas de combinação e transformação das features existentes. Estas técnicas podem ser aplicadas a este projeto com o objetivo de melhorar a capacidade do modelo em distinguir as classes de lances.

Referências

1. http://agenciabrasil.ebc.com.br/economia/noticia/2014-02/com-pregao-eletronico-governo-economizou-r-91-bilhoes-em-2013
2. https://istoe.com.br/139247_GOLPE+NO+PREGAO+ELETRONICO/
3. https://idag.jusbrasil.com.br/noticias/2611942/pregao-eletronico-robos-ganham-licitacoes
4. http://g1.globo.com/tecnologia/blog/seguranca-digital/post/robos-participam-de-pregoes-compram-ingressos-e-atuam-na-bolsa.html
5. https://www.kaggle.com/c/facebook-recruiting-iv-human-or-bot/data
6. https://www.kaggle.com/c/facebook-recruiting-iv-human-or-bot/leaderboard
7. http://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html
8. http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
9. https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis
10. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
11. http://scikit-learn.org/stable/modules/neighbors.html
12. https://en.wikipedia.org/wiki/Random_forest
13. http://scikit-learn.org/stable/modules/ensemble.html#forest
14. https://pt.wikipedia.org/wiki/Rede_neural_artificial
15. http://scikit-learn.org/stable/modules/neural_networks_supervised.html
16. http://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble
17. https://en.wikipedia.org/wiki/AdaBoost
18. http://scikit-learn.org/stable/modules/naive_bayes.html#gaussian-naive-bayes[
19. https://en.wikipedia.org/wiki/Naive_Bayes_classifier#Gaussian_naive_Bayes
20. https://en.wikipedia.org/wiki/Quadratic_classifier
21.