Avito demand prediction challenge

#Luciana Sá Brito

INTRODUÇÃO

O AVITO[1] é o maior site de classificados de toda a Rússia. Apresenta em seus anúncios um subconjunto das seguintes características, observadas em seu site: Identificação do produto pelo nome, preço, cidade de origem do produto, data de carregamento do anúncio, horário de carregamento do anúncio, nome da empresa.

Figura 1: Conteúdo típico de um anúncio Avito. Fonte: https://goo.gl/YwQndW. Disponível em 20/05/2018. As expressões “A empresa” e “Ontem” aparecem traduzidas para o português pelo navegador web utilizado.

A empresa disponibilizou pela 4ª vez o seu dataset para um desafio da Kaggle[2], desta vez chamado “Avito Demand Prediction Challenge - Predict demand for an online classified ad”.

As descrições das colunas podem ser encontradas no Anexo 1 a este relatório. O desafio consiste em prever a demanda por um anúncio on-line com base em sua descrição completa (título, descrição, imagens etc.), seu contexto (geograficamente onde foi postado, anúncios similares já publicados) e demanda histórica para anúncios semelhantes em contextos semelhantes.

Para isso pretende-se trabalhar com foco na otimização de resultados na utilização dos algoritmos, ou seja, queremos produzir um modelo com o melhor desempenho possível, que nos forneça, então, a melhor predição da saída desejada.

[1] <https://www.avito.ru/rossiya>. Acesso em 20/05/2018.
[2] Plataforma crowdsourcing de Ciência de Dados adquirida pelo grupo Alphabeto em 2017.

Neste estudo, a saída desejada é formada pela matriz (user_id, deal_probability[3]) em que as submissões são pontuadas através do cálculo do desvio médio quadrático, definido como na equação a seguir, onde *y* chapéu é o valor predito e *y* é o valor original.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2}$$

REFERENCIAL TEÓRICO

O *aprendizado de máquina* se refere ao uso de modelos que são aprendidos a partir dos dados [também] pode ser chamado de *modelo preditivo* ou *mineração de dados*. Os modelos preditivos se dividem em sua grande parte em *supervisionados*, *não supervisionados*, *semi supervisionados* e *online* [1].

Para a condução do trabalho foi feito um levantamento de informações sobre práticas de Data Science que têm sido utilizadas por outros competidores. Nesta pesquisa, destacaram-se os trabalhos de Stevens, C [2][3], Moreira, G [4] e Lathwal[4].

Stevens, C destaca-se pela publicação do seu procedimento experimental para o desafio *Avito Demand Prediction* na revista eletrônica *Medium*, onde ensina que "ter uma boa compreensão de como os usuários estão criando os dados que [...] usará é imprescindível para o design do seu algoritmo [sendo também] importante baixar o dataset para o seu PC a fim de garantir maior interatividade com os dados [E mais:] iniciar a pesquisa com um análise de dados exploratória pode trazer importantes insights na hora de escolher qual modelo implementar".

Lathwal, em seu notebook desafio[5] destaca também a importância das análises estatísticas nas etapas iniciais da da Extração de Conhecimento em Datasets (KDD) com a finalidade de entender o comportamento das *features*, utilizando bibliotecas de visualização de dados para mostrar o resultado de suas análises.

Stevens, C ainda comenta em [2], acerca da importância de fazer uma engenharia de feature para que a partir dos dataset, se possa perceber quais são as features que melhor se encaixariam no problema que está sendo proposto.

---

[3] Traduzida como "probabilidade de venda".
[4] <https://www.kaggle.com/codename007>. Disponível em 10/06/2018.
[5] <https://www.kaggle.com/codename007/avito-eda-fe-time-series-dt-visualization>. Disponível em 10/06/2018.
.

Moreira, G, em [4] aborda as etapas do KDD, destacando as etapas de pré-processamento, engenharia de features, aplicação do modelo, apresentação dos resultados e *tuning* de parametrização como etapas que devem ser observadas por qualquer cientista de dados.

PERGUNTA DE PESQUISA

Considerando os dados de um anúncio do site Avito, qual a probabilidade dele ser efetivo e ter sua negociação concretizada?

MÉTODO

A) O dataset

O dataset é composto por atributos numéricos, atributos categóricos, atributos temporais e de atributos de texto-livre. O número de linhas nas tabelas disponibilizadas são 1.503.523 para a tabela de treinamento (train.csv) e 508.547 para a tabela de teste (test.csv) contra 18 colunas de dados estruturados, que incluem as características (features) que poderão ser utilizadas para a resolução do desafio[6].

Havia também tabelas de treinamento e teste para os dados de imagem dos produtos. Outras informações sobre o dataset encontram-se no ANEXO 1. Os dados de imagem não foram utilizados nesta primeira abordagem do problema. Os dados disponibilizados são referentes a anúncios da AVITO para o intervalo entre os dias 15/03/2017 e 28/03/2017.

B) Knowledge discovery in databases (KDD)

B.1) Ferramentas utilizadas

Foram utilizados o notebook Jupyter no kernel da competição e a linguagem Python, com auxílio das bibliotecas Math, Pandas, Numpy, Matplotlib e Sklearn.

---

[6] Os arquivos disponibilizados, junto com a descrição das colunas da tabela de dados, pode ser encontrada em <https://www.kaggle.com/c/avito-demand-prediction/data>. Acesso em 23/05/2018.

B.2) Etapas do KDD

B.2.1) Pré processamento/Data munging

ANÁLISE EXPLORATÓRIA DO DATASET

O primeiro passo para exploração do dataset foi baixar o arquivo e visualizá-lo fazendo uso de um leitor de arquivos no formato csv. Com isso, informações importantes sobre as colunas do dataset puderam ser levantadas:

i. A coluna item_id parece conter valores únicos e deve se tratar de um identificador único de cada uma das linhas. Isso indica que esta coluna provavelmente não poderá ser usada nos modelos preditivos.
ii. A coluna user_id possui um comportamento similar ao item_id, porém ao analisá-la juntamente com a coluna item_seq_number, supõe-se que podem haver valores iguais de user_id, uma vez que um usuário pode ter mais de um anúncio publicado no site. Ao que tudo indica, essa coluna também não será útil para o modelo.
iii. item_seq_number parece representar o número ordinal do anúncio publicado por um usuário. Esta coluna pode ser útil para sinalizar comportamentos do tipo "o primeiro anúncio que um usuário publica tem maior chance de ser vendido que os seus anúncios posteriores".
iv. price diz o preço de venda dos anúncios. Pode ser uma coluna chave para a construção do modelo.
v. category_name é a categoria específica do anúncio. Por ser discreta, ela pode ser de grande ajuda na performance do modelo.
vi. parent_category_name é a uma categoria mais genérica do anúncio. Seus valores parecem variar menos que a category_name.
vii. A coluna city descreve a cidade de cada um dos anúncios. Como o número de cidade de um país é finito e pode ser enumerado, esta também pode ser uma coluna importante para a construção de um modelo eficiente.
viii. A coluna region informa a região da Rússia onde o anúncio foi publicado. Parece ter um número bem pequeno de opções.
ix. user_type indica a natureza do usuário que está publicando o anúncio. A coluna tem três valores possíveis: Private, Shop e Company.
x. title define o título do anúncio, que é a fonte primária de informação textual do anúncio.

xi. Description contém informações detalhadas a respeito do anúncio.
xii. A data em que o anúncio foi criado é armazenada na coluna activation_date, no formato americano yyyy-mm-dd. Por não se tratar de um dado essencialmente numérico, será necessário tratar essa coluna antes de utilizá-la.
xiii. A coluna deal_probability representa a chance de um anúncio se transformar em uma negociação concretizada. Contudo, não ficou tão claro o critério usado pela Avito para definir o valor desta coluna.
xiv. A coluna image armazena o código da imagem do anúncio. Nem todos os anúncios possuem imagens, o que pode ser um bom indicador de probabilidade de venda.
xv. image_top_1 armazena um código de classificação da imagem atribuído pela própria AVITO.

ENGENHARIA DE FEATURE

i. Separou-se os campos de ano, mês e dia para obter-se números. Foi feita a transformação da data para dados numéricos e criação de novas colunas no dataframe.
ii. Preenchimento dos valores vazios nas colunas com zero para evitar erros do tipo NAN.
iii. Converteu-se as colunas de classes discretas para valores inteiros, para que pudessem ser utilizados em quaisquer algoritmos.
iv. Transformou-se os valores da coluna image, atribuindo-se zero para linhas sem código de imagem e 1 para colunas que possuem código de imagem.
v. Separou-se os grupos de treinamento e de teste, na proporção 70%/30% para utilização na etapa de treinamento do algoritmo.

B.2.2) Escolha do modelo de aprendizagem

Utilizou-se um modelo de Regressão Linear com o objetivo de predizer qual a probabilidade de, considerando os dados de um anúncio no site Avito, esse anúncio ser efetivo e ter sua negociação concretizada.

O problema parece se encaixar no modelo de regressão à medida que apresenta variáveis de entrada (colunas do dataset) selecionadas com o objetivo de fornecer como variável de saída um um valor que represente o melhor ajuste percentual de uma reta para determinados pontos dados. Nesta competição, o melhor ajuste é representado pelo resultado que minimiza o erro quadrático.

O valor que representa o melhor ajuste percentual é o deal_probability. Então, quanto melhor for a performance do algoritmo utilizado, mais próximos os valores de saída estarão do deal_probability. Os valores de saída, neste estudo são calculados através do script que se encontram no ANEXO 2.

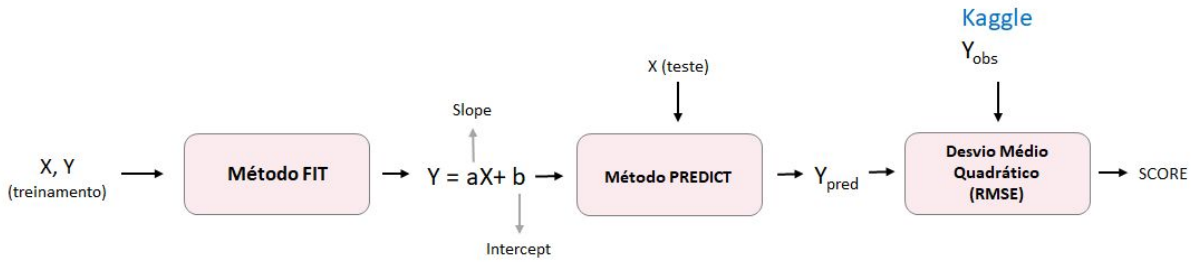

Figura 2: Etapas do modelo de aprendizagem com cálculo do deal_probability e do desvio médio quadrático.

A Figura 2 mostra as etapas do modelo que foi implementado neste estudo. As variáveis *X* e *Y* do dataset de treinamento representam, respectivamente, as features escolhidas e os valores de deal probability do conjunto de treinamento. Aplica-se à *X* e *Y* o método FIT e obtendo-se como saída os valores de *a* e *b*, intercept e slope, respectivamente. Esta é a etapa de aprendizagem do algoritmo de regressão. A partir da equação da curva que melhor descreve o conjunto de pontos dados na etapa de treinamento, o modelo, a partir da entrada da variável *X* do dataset de teste, é capaz de predizer os valores de *Y* de teste associados (deal probability do conjunto de teste). Os valores preditos são, então, submetidos na plataforma do desafio, que retorna o valor do desvio quadrático médio entre eles e os valores de deal probability observados pelo cliente (AVITO). A plataforma retorna também a posição do participante no desafio, com relação aos outros concorrentes.

B.2.3) Implementação do modelo de Regressão Linear

- Escolheu-se as features que seriam utilizadas na análise, a partir das features originárias do dataset e as obtidas na etapa de pré-processamento. As features escolhidas foram: price, item_seq_number, image_top_1, category_name, parent_category_name, city, region, image, length_description, length_title, words_description, words_title, percent_capital_description e percent_capital_title.

As colunas que não foram utilizadas foram as: item_id, user_id, param_1, param_2, param_3, activation_date e user_type.

- Atribuiu-se as features que foram escolhidas à variáveis de treinamento e teste.
- Atribuiu-se os targets (deal_probability) para o treinamento do modelo.
- Instanciou-se o modelo de regressão linear.
- Treinou-se o modelo.
- Fez-se predições a partir do modelo para os dados contidos no arquivo test.csv.
- Predições com valores abaixo de zero (raríssimas exceções) foram transformadas em zero.

B.2.4) Submissão dos resultados

A submissão dos resultados se deu através da transformação do dataframe em arquivo csv e submissão do resultado da regressão.

B.2.5) Observação do resultado no ranking do Kaggle e tunning de parametrização.

Esta etapa se inicia com a observação dos resultados e comparação destes com os resultados obtidos em possíveis submissões anteriores. Objetiva aprimorar a técnica utilizada através da revisão de todas as etapas do KDD à busca de novos insights para a resolução do problema proposto no desafio.

CONSIDERAÇÕES FINAIS

Ao final desta pesquisa percebeu-se a importância de valorizar a análise exploratória dos dados no KDD para municiar a pesquisadora de informações para uma *engenharia de features* eficiente para a escolha do modelo de *machine learning* a utilizar.

Diante do resultado obtido, ficou muito claro que as informações relevantes para os tomadores de decisão estão contidas nas features que foram escolhidas para a aplicação do modelo, que foram as mesmas que fizeram o desvio médio quadrático se aproximar cada vez mais do seu valor ideal, que seria zero.

As colunas param_1, param_2 e param_3 parecem ser dados opcionais dos anúncios. Pela natureza dos valores destas colunas, os dados nelas contidos devem variar

completamente de anúncio para anúncio, o que dificulta a utilização destas informações no modelo.

O resultado final obtido, desvio médio quadrático = 0.2595, e o consequente lugar no ranking que indicava a posição da submissão dos resultados (1481º lugar de 1612º lugar até 01:35 de 09/06/2018 no horário GMT-3), deu pistas de que há muito ainda que se fazer para melhorar o valor do deal_probability, talvez utilizando técnicas de ensemble aliadas ao processamento de dados em linguagem natural que não puderam ser explorados devido à dificuldade de utilizar esses dados dentro do modelo de regressão implementado.

O processamento das colunas de imagem não foi feito, mas também contribuiria para a melhorias do modelo. Para próximos trabalhos a possibilidade de utilização de processamento de imagens e textos poderia contribuir para a criação de um ensemble, que seria constituído pela integração de diversos modelos a fim de criar um modelo preditivo de maior performance.

REFERÊNCIAS

[1] GRUS, Joel. **Data Science do Zero:** PRIMEIRAS REGRAS COM PYTHON. Rio de Janeiro: Alta Books Editora, 2016. 315 p. (9788576089988). Tradução de: Data Science from Scratch: First Preinciples with Python.

[2] STEVENS, Chris. **Approaching a competition on Kaggle: Avito Demand Prediction Challenge (Part 1).** 2018. Disponível em: <https://medium.com/@chrisstevens1/approaching-a-competition-on-kaggle-avito-demand-prediction-challenge-part-1-9bc4e6299ba4>. Acesso em: 09 jun. 2018.

[3] STEVENS, Chris. **Approaching a competition on Kaggle: Avito Demand Prediction Challenge (Part 2).** 2018. Disponível em: <https://medium.com/@chrisstevens1/approaching-a-competition-on-kaggle-avito-demand-prediction-challenge-part-2-a1afbe063bb8>. Acesso em: 09 jun. 2018.

[4] MOREIRA, Gabriel. **Feature Engineering:** Getting the most of data for predictive models. 2017. QCon São Paulo, Palestra. Disponível em: <https://www.infoq.com/br/presentations/extraindo-o-potencial-maximo-dos-dados-para-modelos-preditivos>. Acesso em: 09 jun. 2018.

ANEXO 1

Descrição das colunas do dataset:

- train.csv - Train data.
    - item_id - Ad id.
    - user_id - User id.
    - region - Ad region.
    - city - Ad city.
    - parent_category_name - Top level ad category as classified by Avito's ad model.
    - category_name - Fine grain ad category as classified by Avito's ad model.
    - param_1 - Optional parameter from Avito's ad model.
    - param_2 - Optional parameter from Avito's ad model.
    - param_3 - Optional parameter from Avito's ad model.
    - title - Ad title.
    - description - Ad description.
    - price - Ad price.
    - item_seq_number - Ad sequential number for user.
    - activation_date- Date ad was placed.
    - user_type - User type.
    - image - Id code of image. Ties to a jpg file in train_jpg. Not every ad has an image.
    - image_top_1 - Avito's classification code for the image.
    - deal_probability - The target variable. This is the likelihood that an ad actually sold something. It's not possible to verify every transaction with certainty, so this column's value can be any float from zero to one.
- test.csv - Test data. Same schema as the train data, minus deal_probability.
- train_active.csv - Supplemental data from ads that were displayed during the same period as train.csv. Same schema as the train data minus deal_probability, image, and image_top_1.
- test_active.csv - Supplemental data from ads that were displayed during the same period as test.csv. Same schema as the train data minus deal_probability, image, and image_top_1.
- periods_train.csv - Supplemental data showing the dates when the ads from train_active.csv were activated and when they where displayed.

    - item_id - Ad id. Maps to an id in train_active.csv. IDs may show up multiple times in this file if the ad was renewed.
    - activation_date - Date the ad was placed.
    - date_from - First day the ad was displayed.
    - date_to - Last day the ad was displayed.
- periods_test.csv - Supplemental data showing the dates when the ads from test_active.csv were activated and when they where displayed. Same schema as periods_train.csv, except that the item ids map to an ad in test_active.csv.
- train_jpg.zip - Images from the ads in train.csv.
- test_jpg.zip - Images from the ads in test.csv.
- sample_submission.csv - A sample submission in the correct format.
- train_jpg_{0, 1, 2, 3, 4}.zip - These are the exact same images as you'll find in train_jpg.zip but split into smaller zip archives so the data are easier to download. If you already have train_jpg.zip you do NOT need to download these. We have not made these zips available in kernels as they would only increase the kernel creation time.

ANEXO 2

Código Python de implementação

```
#importações
import math
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn import preprocessing

#Lendo os dados de treinamento e teste e criando os dataframes (estrutura de dados) train e test
train = pd.read_csv("../input/train.csv")
test = pd.read_csv("../input/test.csv")

#Preenchendo os valores vazios com 0 para evitar nan
train = train.fillna(0)
test = test.fillna(0)

#Convertendo as colunas de classes discretas para valores inteiros, para obtermos colunas numéricas
label_encoders = { } #cria lista dict de chave/valor para
features_to_encode = ['category_name', 'parent_category_name', 'city', 'region', 'user_type']
for feature in features_to_encode:
        label_encoders[feature] = preprocessing.LabelEncoder() #instanciando o objeto label_encoders da classe LabelEnconder
        label_encoders[feature].fit(pd.concat([train[feature], test[feature]])) #gera tabela de correspondência entre dados de uma feature e um número
    train[feature] = label_encoders[feature].transform(train[feature])
    test[feature] = label_encoders[feature].transform(test[feature])
```

```
#Transformando a coluna image do dataset. Como funciona: se a coluna está preenchida com valor significa que o anúncio
#possui um código para a imagem, e então o valor do seu atributo image é setado para 1. Caso contrário, seu valor é setado para 0.
#Primeiro trato as images do dataset de treinamento
images = []
for image in train['image']:
    if len(str(image)) <= 1:
        images.append(0)
    else:
        images.append(1)
train['image'] = images

#Depois trato as images do dataset de teste
images = []
for image in test['image']:
    if len(str(image)) <= 1:
        images.append(0)
    else:
        images.append(1)
test['image'] = images

#Como ainda foi possível utilizar nenhum modelo baseado em linguagem natural, analisou-se o comprimento
#e o número de palavras contidas no título e na descrição do anúncio, partindo da hipótese de que um anúncio mais bem detalhado
#pode gerar maior interesse de compra. Isso melhorou um pouco a perfórmance do modelo.

#computando o comprimento e o número de palavras do título e da descrição do conjunto de treinamento
length_title = [] #número de caracteres da string que corresponde ao título do anúncio
```

```
length_description = [] #número de caracteres da string que corresponde ao comprimento da descrição do anúncio
words_title = [] #quantidade de palavras no título
words_description = [] #quantidade de palavras na descrição
percent_capital_title = [] #porcentagem de capital letters no título
percent_capital_description = [] #porcentagem de capital letters na descrição

#começando a computar o comprimento, o número de palavras e a porcentagem de capital letters no título p/ a base de treinamento
for title in train['title']:
    length_title.append(len(str(title)))
    words_title.append(len(str(title).split(' ')))
    count_capital = sum(1 for letter in str(title) if letter.isupper())
    if count_capital > 0:
                    percent_capital_title.append(int(len(str(title)) / count_capital)) #adiciona no percent_capital_title a (qtdade de caracteres título / qtdade de letras maiúsculas)
    else:
        percent_capital_title.append(0)

#começando a computar o comprimento,o número de palavras e a porcentagem de capital letters na descrição p/ a base de treinamento
for description in train['description']:
    length_description.append(len(str(description)))
    words_description.append(len(str(description).split(' ')))
    count_capital = sum(1 for letter in str(description) if letter.isupper())
    if count_capital > 0:
        percent_capital_description.append(int(len(str(description)) / count_capital))
    else:
        percent_capital_description.append(0)

#criando as novas colunas de comprimento (título e descrição), número de palavras (título e descrição) e porcentagem de capital letters (título e descrição) para o conjunto de treino
```

```
train['length_title'] = length_title
train['length_description'] = length_description
train['words_title'] = words_title
train['words_description'] = words_description
train['percent_capital_title'] = percent_capital_title
train['percent_capital_description'] = percent_capital_description

#computando o comprimento e o número de palavras do título e da descrição do conjunto de teste
length_title = []
length_description = []
words_title = []
words_description = []
percent_capital_title = []
percent_capital_description = []

#começando a computar o comprimento,o número de palavras e a porcentagem de capital letters no título p/ a base de teste
for title in test['title']:
    length_title.append(len(str(title)))
    words_title.append(len(str(title).split(' ')))
    count_capital = sum(1 for letter in str(title) if letter.isupper())
    if count_capital > 0:
        percent_capital_title.append(int(len(str(title)) / count_capital))
    else:
        percent_capital_title.append(0)

#começando a computar o comprimento,o número de palavras e a porcentagem de capital letters na descrição p/ a base de teste
for description in test['description']:
    length_description.append(len(str(description)))
    words_description.append(len(str(description).split(' ')))
```

```
    count_capital = sum(1 for letter in str(description) if letter.isupper())
    if count_capital > 0:
        percent_capital_description.append(int(len(str(description)) / count_capital))
    else:
        percent_capital_description.append(0)

#criando as novas colunas de comprimento (título e descrição), número de palavras (título e descrição) e porcentagem de capital letters (título e descrição) para o conjunto de teste
test['length_title'] = length_title
test['length_description'] = length_description
test['words_title'] = words_title
test['words_description'] = words_description
test['percent_capital_title'] = percent_capital_title
test['percent_capital_description'] = percent_capital_description

#Escolhendo quais features serão usadas para treinar o modelo (aqui somente as features numéricas podem ser utilizadas,
#porque se trata de uma regressão linear).
#Features possíveis: price, item_seq_number, image_top_1, category_name, parent_category_name, city, region, user_type,
#length_title, length_description, words_title, words_description, percent_capital_title, percent_capital_description.
#O atributo user_type parece prejudicar o algoritmo, por isso não foi utilizado.
#Atributos que parecem não fazer diferença no resultado e por isso foram removidos (bloco supracomentado) do modelo: activation_year, month e day.

features = ['price', 'item_seq_number', 'image_top_1', 'category_name', 'parent_category_name', 'city', 'region', 'image',
                        'length_description', 'length_title', 'words_title', 'words_description', 'percent_capital_title', 'percent_capital_description']

x_train = train[features]
```

```
x_test = test[features]

#Guardando os targets para treinar o modelo
y_train = train.deal_probability

#Regra:
#x_train ->>> y_train (deal_probability do conjunto de treinamento. É um dado do problema)
#x_test  ->>> y_test = pred = predições = resposta do desafio = deal_probability

#Instanciando nosso modelo de regressão linear
linear_regression = linear_model.LinearRegression()
#Treinando o modelo
linear_regression.fit(x_train, y_train)
#Fazendo as predições
pred = linear_regression.predict(x_test)

#transforma para zero o pequeno número de casos (exceção) em que a predição tem valor negativo
i = 0
for x in pred:
    if x < 0:
        pred[i] = 0
    i += 1

#Criando um dataframe do pandas com as colunas pedidas pelo kaggle: item_id e deal_probability
my_submission = pd.DataFrame({'item_id': test['item_id'], 'deal_probability': pred})
#Salvando o dataframe do resultado em CSV para que possa ser submetido ao desafio
my_submission.to_csv('submission.csv', index=False)
```