
Autores: Prof. João Nogueira e Prof. Dino Magri

Contato: joao.nogueira@fisica.ufc.br e professor.dinomagri@gmail.com

Licença deste notebook:



[Clique aqui para saber mais sobre a licença CC BY v4.0](#)

Projeto - Parte 2 - Aprendizagem Supervisionada - Classificação - Exercícios

NOTA MÁXIMA: 25 pontos

NOME COMPLETO: MARCIO FERNANDES CRUZ, EDUARDO BOLYHOS, ALISSON SALES



▼ Definição do Problema

O departamento de CRM/Marketing do iFood fez uma solicitação para o time de Ciência de Dados (que estão atrelados ao time de dados dentro da área de TI) para criar um modelo de Machine Learning para prever os clientes que darão Churn.

O objetivo deles com o modelo é atuar sobre os clientes com maiores chances de darem churn no próximo mês.

- A ação será realizada 1 vez por mês, todo dia 01.
- Como a frequência de compras dos clientes do iFood é relativamente alta, então ficou-se decidido na reunião com a área cliente que o modelo seria construído usando features construídas em um período fechado de 1 mês para prever se um dado cliente irá deixar de comprar (churn) no próximo mês.

A métrica principal de avaliação do modelo é a AUC, dado que o score gerado pelo modelo será usado para ordenar a base.

```
1 from google.colab import drive  
2 drive.mount('/content/drive')
```

Mounted at /content/drive

```
1 import os  
2 import numpy as np  
3 import pandas as pd
```

▼ Lendos os Dados

```
1 import pandas as pd  
2 import os  
3  
4 WORK_DIR = '/content/drive/MyDrive/Colab Notebooks/Data Set Data Science'  
5 DATA_DIR = os.path.join(WORK_DIR, 'ifood')  
6 df_orders = pd.read_csv(os.path.join(DATA_DIR, 'df_orders.csv'))  
  
1 df_orders.head(4)
```

	order_id	order_number	order_timestamp_local	order_shift	last_status_dat
0	630e2af0-b456-4b3a-b964-4d66ce5cc5df	1290139943	2019-12-11T13:22:06.497Z	weekday lunch	11T15:22
1	66f8163d-f081-4fbda-adb8-cfbff9213f7	1141271215	2019-11-06T17:57:17.967Z	weekday dinner	06T18:18

▼ Análise Exploratória (1 ponto)

C144eabade3

(0.25 ponto) q1 Quantas linhas e colunas existem na base df_orders ?

```
3          0    679245732      0ET10.1G.10.1117      dinner      0ET21.17
1 total_linha = df_orders.shape[0]
2 total_colunas = df_orders.shape[1]
3
4 print(f'Resposta: O total de linhas é {total_linha} e o de colunas é {total_colunas}')
```

Resposta: O total de linhas é 492138 e o de colunas é 43.

(0.25 ponto) q2 Verifique se todos os pedidos são únicos

```
1 if len(df_orders['order_id'].unique())==df_orders.shape[0]:  
2     print('Resposta: Os pedidos são únicos')  
3 else:  
4     print('Resposta: Os pedidos não são únicos')  
5
```

Resposta: Os pedidos são únicos

(0.25 ponto) Q3 Qual o pedido mais antigo e o mais recente da base? Utilize a variável `order_timestamp_local`.

```
1 df_orders['order_timestamp_local'] = pd.to_datetime(df_orders['order_timestamp_local'])
2 df_velho = (
3     df_orders.groupby('order_number')
4     .agg(velho = ('order_timestamp_local','min')).nsmallest(1, 'velho'))
5 df_velho.reset_index(inplace=True)
6 data_pedido_mais_antigo = df_velho.loc[0]["velho"]
7 numero_pedido_mais_antigo = df_velho.loc[0]["order_number"]
8
9 df_novo = (
10    df_orders.groupby('order_number')
11    .agg(novo = ('order_timestamp_local','max')).nlargest(1, 'novo'))
12 df_novo.reset_index(inplace=True)
13 data_pedido_mais_novo = df_novo.loc[0]["novo"]
14 numero_pedido_mais_novo = df_novo.loc[0]["order_number"]
15
16 print(f'Reposta: O pedido mais antigo é {numero_pedido_mais_antigo} com data de {data_pedido_mais_antigo}')
17 print(f'Reposta: O pedido mais novo é {numero_pedido_mais_novo} com data de {data_pedido_mais_novo}')
18
```

Resposta: O pedido mais antigo é 667742584 com data de 2019-06-01 00:00:15.369000+00:00
Resposta: O pedido mais novo é 1373960993 com data de 2019-12-31 23:31:59.972000+00:00

(0,25 ponto) q4 Quantos usuários únicos existem na base?

```
1 print('Resposta: O número de usuários únicos na base é ', len(df_orders['customer_id'].unique()))
2
3 Resposta: O número de usuários únicos na base é  30079
```

▼ Criacão da ABT (10 pontos)

Será que variáveis como Recência, Frequência e Valor podem nos ajudar a prever se um dado cliente irá realizar um novo pedido no próximo mês?

Vamos criar variáveis que descrevem o comportamento do cliente com base em 1 mês e usar essas variáveis para prever se o cliente irá comprar no próximo mês (caso em que acontecerá um churn e talvez seja interessante oferecermos um cupom de desconto através de notificações no app).

(1.0 ponto) q5 Crie um DataFrame chamado `df_features_train` que deverá conter os dados do período 2019-06-01 (inclusivo) até 2019-07-01 (exclusivo), ou seja deve conter somente os dados referente ao mês 6.

```
1 filtro = (df_orders["order_timestamp_local"]>="2019-06-01") & (df_orders["order_timestamp_local"]<"2019-07-01")
2 df_features_train = df_orders[filtro]
3
4 print(f'Análise: Em Junho/2019 há {df_features_train.shape[0]} observações.')
5
```

Análise: Em Junho/2019 há 88451 observações.

(1.0 ponto) q6 Crie um DataFrame chamado `df_target_train` que deverá conter os dados do período 2019-07-01 (inclusivo) até 2019-08-01 (exclusivo), ou seja deve conter somente os dados referente ao mês 7.

Depois selecione apenas a coluna `customer_id` e remova os duplicados.

```
1 filtro = (df_orders["order_timestamp_local"]>="2019-07-01") & (df_orders["order_timestamp_local"]<"2019-08-01")
2 df_target_train = df_orders[filtro]
3
4 df_target_train = df_target_train[['customer_id']]
5 df_target_train.drop_duplicates(inplace=True)
6
7 print(f'Existem {df_target_train.shape[0]} clientes que fizeram compra em Julho/2019')
```

Existem 18441 clientes que fizeram compra em Julho/2019

(2.0 ponto) q7 Faça o agrupamento no DataFrame `df_features_train` pela feature `customer_id`. Faça as agregações abaixo, renomeando o resultado para as seguintes colunas `receita_1m`, `qtd_pedidos_1m` e `data_ultima_venda`, respectivamente.

- `'paid_amount'` - `'sum'`
- `'order_id'` - `'nunique'`
- `'order_timestamp_local'` - `'max'`

Lembre-se de utilizar o conceito que vimos em sala para renomear o resultado da agregação.

Também utilize o método `reset_index` para reiniciar os indices.

Salve na variável `df_abt_train`.

```
1 # Treino sobre os dados de Junho/2019
2
3 df_abt_train = (
4     df_features_train
5     .groupby("customer_id")
6     .agg(receita_1m = ('paid_amount', 'sum'),
7           qtd_pedidos_1m = ('order_id', 'nunique'),
8           data_ultima_venda = ('order_timestamp_local', 'max')
9     )
10    .reset_index()
11 )
```

(3.0 pontos) q8 Utilize o método `.assign` do DataFrame para fazer as seguintes alterações:

- Altere o tipo da coluna `data_ultima_venda` para o formato de data utilizando o comando `pd.to_datetime` com o parâmetro `utc=True`.
- Crie uma nova coluna chamada `data_ref` com o valor de `2019-07-01` e faça a converção utilizando o comando `pd.to_datetime` com o parâmetro `utc=True`
- Crie uma nova coluna chamada `recencia` e salve os dias referentes a subtração da coluna `data_ref` e `data_ultima_venda`. Lembre-se de extrair os dias (`.dt.days`)
- Por fim, selecione apenas as seguintes colunas: `['data_ref', 'customer_id', 'receita_1m', 'qtd_pedidos_1m', 'recencia']`

O resultado dessas alterações devem ser salvas na variável `df_abt_train`

```
1 df_abt_train["data_ultima_venda"] = pd.to_datetime(df_abt_train["data_ultima_venda"], utc=True)
2 df_abt_train["data_ref"] = "2019-07-01"
3 df_abt_train["data_ref"] = pd.to_datetime(df_abt_train["data_ref"], utc=True)
4
5 df_abt_train = df_abt_train.assign(recencia = lambda df: (df['data_ref'] - df['data_ultima_venda']).dt.days)
6
7 df_abt_train = df_abt_train[['data_ref', 'customer_id', 'receita_1m', 'qtd_pedidos_1m', 'recencia']]
```

(1.0 pontos) q9 Faça o `merge` com o DataFrame `df_abt_train` com o `df_target_train` com os seguintes parametros:

- `how='left'`

- on='customer_id'
- indicator=True

Salve o merge na variável df_abt_train.

```
1 df_abt_train = df_abt_train.merge(df_target_train, how='left', on="customer_id", indicator=True)
2 df_abt_train.shape
```

```
(30079, 6)
```

(1.0 pontos) q10 Utilize o método .assign para criar a coluna churn_next_month.

Utilize a função np.where para fazer a criação. Compare as colunas _merge com left_only, se ambas forem iguais, o resultado deve ser 1, caso contrário deve ser 0.

- Obs: não esquecer de dropar a coluna _merge depois de ter criado a variável target.

```
1 df_abt_train = df_abt_train.assign(churn_next_month = np.where(df_abt_train['_merge']=='left_only','1','0'))
2 df_abt_train.drop("_merge", axis=1, inplace=True)
3
```

(1.0 pontos) q11 Verifique a distribuição da coluna churn.

```
1 nao_churn = df_abt_train["churn_next_month"].value_counts(normalize=True)[0]
2 churn = df_abt_train["churn_next_month"].value_counts(normalize=True)[1]
3 total_linhas = df_abt_train.shape[0]
4
5 print(f'Reposta: A distribuição é {nao_churn} para não-churn e {churn} para churn em um total de {total_linhas} observações')
6
```

```
Resposta: A distribuição é 0.6130855414076266 para não-churn e 0.3869144585923734 para churn em um total de 30079 observações
```

CHECKPOINT - O DataFrame df_abt_train deverá conter exatamente (30079, 6). E a distribuição da variável target deve ser 0.613086 para não-churn (0) e 0.386914 para churn (1).

▼ Modelagem (14 pontos)

(2 pontos) q12 Crie uma ABT Out of Time avançando 1 mês no código anterior. Utilize o mês 08 para criar essa ABT. Utilize os nomes df_features_oot, df_target_oot e df_abt_oot.

```
1 # features
2 filtro = (df_orders["order_timestamp_local"]>="2019-07-01") & (df_orders["order_timestamp_local"]<"2019-08-01")
3 df_features_oot = df_orders[filtro]
4
5 # target
6 filtro = (df_orders["order_timestamp_local"]>="2019-08-01") & (df_orders["order_timestamp_local"]<"2019-09-01")
7 df_target_oot = df_orders[filtro]
8 df_target_oot = df_target_oot[["customer_id"]]
9 df_target_oot.drop_duplicates(inplace=True)
10
11 #abt
12 df_abt_oot = (
13     df_features_oot
14     .groupby("customer_id")
15     .agg(receita_1m = ('paid_amount', 'sum'),
16          qtd_pedidos_1m = ('order_id', 'nunique'),
17          data_ultima_venda = ('order_timestamp_local', 'max')
18     )
19     .reset_index()
20 )
21
22
23 df_abt_oot["data_ultima_venda"] = pd.to_datetime(df_abt_oot["data_ultima_venda"], utc=True)
24 df_abt_oot["data_ref"] = "2019-08-01"
25 df_abt_oot["data_ref"] = pd.to_datetime(df_abt_oot["data_ref"], utc=True)
26 df_abt_oot = df_abt_oot.assign(recencia = lambda df: (df['data_ref'] - df['data_ultima_venda']).dt.days)
27
28 df_abt_oot = df_abt_oot[["data_ref", "customer_id", "receita_1m", "qtd_pedidos_1m", "recencia"]]
29
30 # merge
31 df_abt_oot = df_abt_oot.merge(df_target_oot, how='left', on="customer_id", indicator=True)
32
33 # churn
```

```

34 df_abt_oot = df_abt_oot.assign(churn_next_month = np.where(df_abt_oot['_merge']=='left_only','1','0'))
35 df_abt_oot.drop("_merge", axis=1, inplace=True)
36
37 nao_churn = df_abt_oot["churn_next_month"].value_counts(normalize=True)[0]
38 churn = df_abt_oot["churn_next_month"].value_counts(normalize=True)[1]
39 total_linhas = df_abt_oot.shape[0]
40 total_colunas = df_abt_oot.shape[1]
41
42 print(f'A distribuição é {nao_churn} para não-churn e {churn} para churn em um total de {total_linhas} observações com {total_colunas}')
A distribuição é 0.7595575077273466 para não-churn e 0.24044249227265332 para churn em um total de 18441 observações com 6 colunas

```

Porque devemos criar uma ABT Out of Time?

Quando em nossa base possuímos uma coluna separatriz por data, podemos usar uma ABT Out Of Time, onde deixamos um período posterior como validação, simulando algo que ainda "não aconteceu", tentando validar uma situação mais próximo do modelo em produção, por fim, aumentando a possibilidade de sucesso pós-implantação do trabalho.

CHECKPOINT - O DataFrame `df_abt_oot` deverá conter exatamente (18441, 6). E a distribuição da variável `target` deve ser 0.759558 para não-churn (0) e 0.240442 para churn (1).

(1 ponto) q13 Faça um estudo de valores faltantes tanto na `df_abt_train` e `df_abt_oot`.

```

1 df_abt_train.info();
2 df_abt_oot.info();
3
4 print('\nAnálise: Tanto a ABT de treino como a ABT de Out of Time não possui valores "missing"\n' \
5      'É normal esta situação pelo fato destas bases ter sido tradas anteriormente com funções de agregação.'
6 )

<class 'pandas.core.frame.DataFrame'>
Int64Index: 30079 entries, 0 to 30078
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   data_ref         30079 non-null   datetime64[ns, UTC] 
 1   customer_id     30079 non-null   object  
 2   receita_1m       30079 non-null   float64 
 3   qtd_pedidos_1m  30079 non-null   int64  
 4   recencia        30079 non-null   int64  
 5   churn_next_month 30079 non-null   object  
dtypes: datetime64[ns, UTC](1), float64(1), int64(2), object(2)
memory usage: 1.6+ MB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18441 entries, 0 to 18440
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   data_ref         18441 non-null   datetime64[ns, UTC] 
 1   customer_id     18441 non-null   object  
 2   receita_1m       18441 non-null   float64 
 3   qtd_pedidos_1m  18441 non-null   int64  
 4   recencia        18441 non-null   int64  
 5   churn_next_month 18441 non-null   object  
dtypes: datetime64[ns, UTC](1), float64(1), int64(2), object(2)
memory usage: 1008.5+ KB

```

Análise: Tanto a ABT de treino como a ABT de Out of Time não possui valores "missing"
É normal esta situação pelo fato destas bases ter sido tradas anteriormente com funções de agregação.

Análise: Tanto a ABT de treino como a ABT de Out of Time não possui valores "missing" É normal esta situação pelo fato destas bases ter sido tradas anteriormente com funções de agregação.

(1 ponto) q14 Treine uma árvore de decisão na base de treino usando Stratified K-Fold e avalie utilizando a métrica **AUC** na base Out of Time.

Utilize o parâmetro `random_state=42`.

Lembre-se de instalar as bibliotecas necessárias (scikit-learn, feature-engine, etc).

Utilize o método `cross_val_score` para avaliar os resultados da validação cruzada no conjunto de treino.

Os seguintes passos devem estar contemplados:

- Importar as bibliotecas necessárias
- Criar as variáveis `target`, `key_vars`, `num_vars`, `cat_vars`, `features`
- Criar `X_train` e `y_train`
- Criar o pipeline (mesmo que utilize apenas 1 step)
- Utilizar `StratifiedKFold` e salve na variável `skf`

- Utilizar o cross_val_score
- Calcular a média da AUC para todos os folds calculados no passo anterior
- Criar X_oot e y_oot
- "Fit" o pipeline criado
- Calcular e predizer a probabilidade da classe 1 (y_proba) na OOT
- Calcular a AUC score na OOT

ATENÇÃO: Parte dessas etapas se repetem para os próximos exercícios

```

1 # Importação de Bibliotecas
2 !pip install feature-engine
3 !pip install feature-engine lightgbm xgboost catboost==0.25.1
4
5 from sklearn.model_selection import cross_validate
6 from sklearn.model_selection import StratifiedKFold
7 from sklearn.tree import DecisionTreeClassifier
8 from feature_engine.imputation import ArbitraryNumberImputer, MeanMedianImputer
9 from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
10 from xgboost import XGBClassifier
11 from lightgbm import LGBMClassifier
12 from catboost import CatBoostClassifier
13 from sklearn.pipeline import Pipeline
14 from sklearn.metrics import roc_auc_score
15 from sklearn.model_selection import cross_val_score
16
17 #LightGDM
18 !pip install lightgbm
19 from lightgbm import LGBMClassifier
20 from sklearn.model_selection import GridSearchCV
21

[?] Collecting feature-engine
  Downloading feature_engine-1.3.0-py2.py3-none-any.whl (260 kB)
    |██████████| 260 kB 15.6 MB/s
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from feature-engine) (1.0.2)
Requirement already satisfied: numpy>=1.18.2 in /usr/local/lib/python3.7/dist-packages (from feature-engine) (1.21.6)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.7/dist-packages (from feature-engine) (1.4.1)
Requirement already satisfied: pandas>=1.0.3 in /usr/local/lib/python3.7/dist-packages (from feature-engine) (1.3.5)
Collecting statsmodels>=0.11.1
  Downloading statsmodels-0.13.2-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (9.8 MB)
    |██████████| 9.8 MB 45.3 MB/s
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=1.0.3->feature-engine)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=1.0.3->feature-engine) (2022.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas>=1.0.3->feature-engine)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=1.0.0->feature-engine)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=1.0.0->feature-engine)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.7/dist-packages (from statsmodels>=0.11.1->feature-engine)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.7/dist-packages (from statsmodels>=0.11.1->feature-engine)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging>=21.3->statsmodels)
Installing collected packages: statsmodels, feature-engine
  Attempting uninstall: statsmodels
    Found existing installation: statsmodels 0.10.2
    Uninstalling statsmodels-0.10.2:
      Successfully uninstalled statsmodels-0.10.2
Successfully installed feature-engine-1.3.0 statsmodels-0.13.2
Requirement already satisfied: feature-engine in /usr/local/lib/python3.7/dist-packages (1.3.0)
Requirement already satisfied: lightgbm in /usr/local/lib/python3.7/dist-packages (2.2.3)
Requirement already satisfied: xgboost in /usr/local/lib/python3.7/dist-packages (0.90)
Collecting catboost==0.25.1
  Downloading catboost-0.25.1-cp37-none-manylinux1_x86_64.whl (67.3 MB)
    |██████████| 67.3 MB 15 kB/s
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from catboost==0.25.1) (3.2.2)
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (from catboost==0.25.1) (5.5.0)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.7/dist-packages (from catboost==0.25.1) (1.21.6)
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-packages (from catboost==0.25.1) (0.10.1)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from catboost==0.25.1) (1.15.0)
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.7/dist-packages (from catboost==0.25.1) (1.3.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from catboost==0.25.1) (1.4.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24.0->catboost==0.25.1)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24.0->catboost==0.25.1) (2022.1)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from feature-engine) (1.0.2)
Requirement already satisfied: statsmodels>=0.11.1 in /usr/local/lib/python3.7/dist-packages (from feature-engine) (0.13.2)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=1.0.0->feature-engine)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=1.0.0->feature-engine)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.7/dist-packages (from statsmodels>=0.11.1->feature-engine)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.7/dist-packages (from statsmodels>=0.11.1->feature-engine)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging>=21.3->statsmodels)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>catboost==0.25.1) (1.4.4)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>catboost==0.25.1) (0.11.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib>catboost==0.25.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-packages (from plotly>catboost==0.25.1) (8.0.1)
Installing collected packages: catboost
  Successfully installed catboost-0.25.1
Requirement already satisfied: lightgbm in /usr/local/lib/python3.7/dist-packages (2.2.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from lightgbm) (1.0.2)

```

```
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from lightgbm) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from lightgbm) (1.21.6)
```

```

1 # Criação das variáveis
2 key_vars = ['data_ref', 'customer_id']
3 num_vars = ['receita_1m', 'qtd_pedidos_1m', 'recencia']
4 cat_vars = [] # Não tem variáveis categórias
5 target = 'churn_next_month'
6 features = cat_vars + num_vars
7
8 # sobre os dados de Junho/2019
9 X_train = df_abt_train[features]
10 y_train = df_abt_train[target]
11
12 # sobre os dados de Agosto/2019
13 X_oot = df_abt_oot[features]
14 y_oot = df_abt_oot[target]
15
16 random_state = 42
17
18 steps_modelos_arvores = [
19     ('numeric_imputer', MeanMedianImputer(variables=num_vars, imputation_method='median'))
20 ]
21
22 # Utilizar StratifiedKFold e salve na variável skf
23 skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=random_state)
24
25 lista_modelo = [
26     ('decision_tree', DecisionTreeClassifier(random_state=random_state))
27 ]
28
29 # "Fitar" Modelo
30 pipeline = Pipeline(steps=steps_modelos_arvores + [lista_modelo[-1]])
31 pipeline.fit(X_train, y_train)
32
33 # Utilizar o cross_val_score
34 cv_results = cross_val_score(pipeline, X_train, y_train, scoring='roc_auc', cv=skf, n_jobs=-1)
35 print(f'decision_tree: A médias dos Folds é {cv_results.mean()}')
36
37 # Base Out of Time
38 y_proba_oot = pipeline.predict_proba(X_oot)[:, 1]
39 print(f'decision_tree: Performance do modelo na Out of Time com o uso da AUC: {roc_auc_score(y_oot, y_proba_oot)}')
40
41 decision_tree: A médias dos Folds é 0.6147471340887323
42 decision tree: Performance do modelo na Out of Time com o uso da AUC: 0.5942238945608709

```

(1 ponto) Q15 Treine uma Random Forest na base de treino usando Stratified K-Fold e avalie utilizando a métrica AUC na base Out of Time.

Utilize o parametro random_state=42

ATENÇÃO: Utilize as variáveis criadas anteriormente (`X_train`, `y_train`, `X_oot`, `y_oot`, `skf`)

```

1 # Utilizar StratifiedKFold e salve na variável skf
2 skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=random_state)
3
4 lista_modelo = [
5     ('random_forest', RandomForestClassifier(random_state=random_state)),
6 ]
7
8 # "Fitar" Modelo
9 pipeline = Pipeline(steps=steps_modelos_arvores + [lista_modelo[-1]])
10 pipeline.fit(X_train, y_train)
11
12 # Utilizar o cross_val_score
13 # cv_results = cross_val_score(pipeline, X_train, y_train, scoring='roc_auc', cv=skf, n_jobs=-1)
14 # print(f'random_forest: A médias dos Folds é {cv_results.mean()}')
15
16 # Base Out of Time
17 y_proba_oot = pipeline.predict_proba(X_oot)[:, 1]
18 print(f'random_forest: Performance do modelo na Out of Time com o uso da AUC: {roc_auc_score(y_oot, y_proba_oot)}')

random_forest: Performance do modelo na Out of Time com o uso da AUC: 0.663481166176368

```

(1 ponto) Q16 Treine um LGBM na base de treino usando Stratified K-Fold e avalie utilizando a métrica AUC na base Out of Time.

Utilize o parâmetro `random_state=42`

ATENÇÃO: Utilize as variáveis criadas anteriormente (`X_train`, `y_train`, `X_oot`, `y_oot`, `skf`)

```

1 # Utilizar StratifiedKFold e salve na variável skf
2 skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=random_state)
3
4 lista_modelo = [
5     ('lgbm', LGBMClassifier(random_state=random_state)),
6 ]
7
8 # "Fitar" Modelo
9 pipeline_lgbm = Pipeline(steps=steps_modelos_arvores + [lista_modelo[-1]])
10 pipeline_lgbm.fit(X_train, y_train)
11
12 # Utilizar o cross_val_score
13 # cv_results = cross_val_score(pipeline, X_train, y_train, scoring='roc_auc', cv=skf, n_jobs=-1)
14 # print(f'lgbm: A médias dos Folds é {cv_results.mean()}')
15
16 # Base Out of Time
17 y_proba_oot = pipeline_lgbm.predict_proba(X_oot)[:, 1]
18 print(f'lgbm: Performance do modelo na Out of Time com o uso da AUC: {roc_auc_score(y_oot, y_proba_oot)}')

lgbm: Performance do modelo na Out of Time com o uso da AUC: 0.7330607764614375

```

(1 ponto) q17 Para o melhor algoritmo dos 3 treinados anteriormente (lightgbm), faça um GridSearchCV para encontrar um conjunto de hiperparâmetros mais otimizado (lembre de otimizar a **AUC**).

Utilize os seguintes hiperparâmetros:

```
'lgbm__max_depth': [3, 5, 7, 9],
'lgbm__learning_rate': [0.01, 0.015, 0.025, 0.05, 0.1],
'lgbm__n_estimators': [100, 300, 500]
```

ATENÇÃO: Utilize as variáveis criadas anteriormente (X_train, y_train, X_oot, y_oot, skf)

ATENÇÃO: Utilize o pipeline já criado para o algoritmo LightGBM

```

1 # O melhor algoritmo foi o LGBM, logo iremos fazer um grid search para ele
2 parametros = {
3     'lgbm__max_depth': [3, 5, 7, 9],
4     'lgbm__learning_rate': [0.01, 0.015, 0.025, 0.05, 0.1],
5     'lgbm__n_estimators': [100, 300, 500]
6 }
7
8 grid_search = GridSearchCV(pipeline_lgbm, parametros, scoring='roc_auc', cv=skf, n_jobs=-1, verbose=1)
9 grid_search.fit(X_train, y_train)

Fitting 5 folds for each of 60 candidates, totalling 300 fits
GridSearchCV(cv=StratifiedKFold(n_splits=5, random_state=42, shuffle=True),
             estimator=Pipeline(steps=[('numeric_imputer',
                                         MeanMedianImputer(variables=['receita_1m',
                                                                        'qtd_pedidos_1m',
                                                                        'recencia'])),
                                         ('lgbm',
                                         LGBMClassifier(random_state=42))]),
             n_jobs=-1,
             param_grid={'lgbm__learning_rate': [0.01, 0.015, 0.025, 0.05, 0.1],
                         'lgbm__max_depth': [3, 5, 7, 9],
                         'lgbm__n_estimators': [100, 300, 500]},
             scoring='roc_auc', verbose=1)

1 print(f'O melhor Score é {grid_search.best_score_}')

0 melhor Score é 0.7569120124999488

1 # Recupere os hiperparâmetros
2 print(f'Os melhores hiperparâmetros são: {grid_search.best_params_}')
3

Os melhores hiperparâmetros são: {'lgbm__learning_rate': 0.05, 'lgbm__max_depth': 3, 'lgbm__n_estimators': 100}

1 # Avalei o desempenho do melhor modelo no grid search na base OOT
2 melhor_modelo = grid_search.best_estimator_
3
4 y_proba_oot = melhor_modelo.predict_proba(X_oot)[:, 1]
5 roc_auc_score_teste1 = roc_auc_score(y_oot, y_proba_oot)
6
7 print(f'lgbm: Performance do modelo na Out of Time com o uso da AUC: {roc_auc_score_teste1}')

lgbm: Performance do modelo na Out of Time com o uso da AUC: 0.7372138323518183

```

```

1 # Crie a variável best_model e salve o melhor estimador do grid_search nessa variável
2 best_model = grid_search.best_estimator_

1 # Treine o melhor estimador (best_model)
2 best_model.fit(X_train, y_train)

Pipeline(steps=[('numeric_imputer',
                 MeanMedianImputer(variables=['receita_1m', 'qtd_pedidos_1m',
                                                'recencia'])),
                ('lgbm',
                 LGBMClassifier(learning_rate=0.05, max_depth=3,
                                random_state=42))])

1 # Calcule a probabilidade da classe 1 para a base OOT
2 y_proba_oot = best_model.predict_proba(X_oot)[:, 1]

1 # Imprima a AUC
2 print(f'lgbm: Performance do modelo na Out of Time com o uso da AUC: {roc_auc_score(y_oot, y_proba_oot)}')

lgbm: Performance do modelo na Out of Time com o uso da AUC: 0.7372138323518183

```

Melhoramos a nossa performance com um grid search bem simples! Muito bom!

Talvez ainda haja espaço para melhorias no desempenho do modelo. Lembrando que temos apenas 3 variáveis: `receita_1m`, `qtd_pedidos_1m` e `recencia`.

Vamos utilizar a variável `order_shift` para criar mais 10 features para verificar se isso irá ajudar o modelo de Machine Learning a predizer qual cliente irá dar **churn**.

(1 ponto) q18 Utilize o método `pd.crosstab` com o parâmetro `normalize='index'` nas colunas `customer_id` e `order_shift`. Lembre-se de resetar o índice.

Salve o resultado no DataFrame `df_customer_order_shift_train`.

Imprima os 5 primeiros elementos desse DataFrame.

```

1 df_customer_order_shift_train = (
2     pd.crosstab(index=df_orders["customer_id"], columns=df_orders["order_shift"], normalize='index')
3     .reset_index()
4 )
5
6 df_customer_order_shift_train.head(5)

```

order_shift	customer_id	weekday_breakfast	weekday_dawn
0	0001a8e61d8b08ad436e8e6f4adeb399b88df962c72d9d...	0.0	0.0
1	0001a9f97d01d2696cf70c7657ee2d039388d691720ff9...	0.0	0.0
2	0004720dc16aed1f98fd79f59736170e0d686199cd9ae5...	0.0	0.0
3	0006a32816a3af172048de7db87c97c4c8c7ad7e6385fa...	0.0	0.0

(1 ponto) q19 Faça o merge com a `df_abt_train` com os seguintes parâmetros:

- `how='left'`
- `on='customer_id'`

Salve o DataFrame resultante na mesma variável `df_abt_train`.

Imprima as 5 primeiras linhas.

```

1 df_abt_train = df_customer_order_shift_train.merge(df_abt_train, how='left', on="customer_id")
2 # df_abt_train.drop("_merge", axis=1, inplace=True)
3 df_abt_train.head(5)

```

	customer_id	weekday breakfast	weekday dawn	weekday dinner
0	0001a8e61d8b08ad436e8e6f4adeb399b88df962c72d9d...	0.0	0.0	0.166667
1	0001a9f97d01d2696cf70c7657ee2d039388d691720ff9...	0.0	0.0	0.333333

(1 ponto) q20 Replique essas duas etapas para a base de OOT.

- Crie o DataFrame `df_customer_order_shift_oot` utilizando o comando `pd.crosstab` com o parâmetro `normalize='index'`. Lembre-se de resetar o índice.
- Faça o merge com a `df_abt_oot` com os parâmetros `how='left'` e `on='customer_id'`.
- Imprima as 5 primeiras linhas do `df_abt_oot`.

```

1 df_customer_order_shift_oot = (
2     pd.crosstab(index=df_features_oot["customer_id"], columns=df_features_oot["order_shift"], normalize='index')
3     .reset_index()
4 )
5
6 df_abt_oot = df_customer_order_shift_oot.merge(df_abt_oot, how='left', on="customer_id")
7 df_abt_oot.head(5)
8

```

	customer_id	weekday breakfast	weekday dawn	weekday dinner
0	000a1fac4f7a67cc3f2e7667167597cb2c9a1b9edafe18...	0.0	0.0	0.666667
1	0014b7013c66a05d0b5ce0687d614ac220d3ae1af398d2...	0.0	0.0	0.400000
2	00150a9d8edc32b5ac2c1a8089e5615dbd297cea2c6cba...	0.0	0.0	0.000000
3	00197c67cab97917c3e147877fc36ae9e9208f4bd578ca...	0.0	0.0	0.800000
4	001b8d424e1aa9b762831ccce74a6b4c648a83f2d8a239...	0.0	0.0	0.250000

Criamos mais 10 variáveis tanto na ABT de treino quanto na de teste (OOT). Essas variáveis indicam o percentual de pedidos realizados por cada usuário em um horário específico do dia.

- weekday breakfast
- weekday dawn
- weekday dinner
- weekday lunch
- weekday snack
- weekend breakfast
- weekend dawn
- weekend dinner
- weekend lunch
- weekend snack

```

1 # Verifique as dimensões de df_abt_train e df_abt_oot
2 print(f'a base df_abt_train possui {df_abt_train.shape[0]} linhas e {df_abt_train.shape[1]} colunas')
3 print(f'a base df_abt_oot possui {df_abt_oot.shape[0]} linhas e {df_abt_oot.shape[1]} colunas')

a base df_abt_train possui 30079 linhas e 16 colunas
a base df_abt_oot possui 18441 linhas e 16 colunas

```

 **CHECKPOINT** - O DataFrame `df_abt_train` deverá conter exatamente (30079, 16) e DataFrame `df_abt_oot` deverá conter (18441, 16).

(1 ponto) q21 Com as features criadas, vamos utilizar o melhor algoritmo testado até o momento (LighGBM) para avaliar se as novas features impactam o modelo a ser gerado. Avalie utilizando a métrica **AUC**.

- Necessário recriar o `X_train`, `y_train`, `X_oot` e `y_oot` incluindo as novas features.
- Recuperar os hiperparâmetros do melhor modelo (`best_model`) para serem utilizados

- Crie um novo pipeline adicionando os hiperparâmetro recuperados.
- "Fit" o pipeline criado
- Calcular e predizer a probabilidade da classe 1 (y_proba) na OOT
- Calcular a AUC score na OOT

```

1 # Utilize as seguintes variáveis:
2 target = 'churn_next_month'
3 num_vars = ['receita_1m', 'qtd_pedidos_1m', 'recencia',
4             'weekday breakfast', 'weekday dawn', 'weekday dinner', 'weekday lunch', 'weekday snack',
5             'weekend breakfast', 'weekend dawn', 'weekend dinner', 'weekend lunch', 'weekend snack']
6 cat_vars = []
7 features = num_vars + cat_vars
8
9 # Adicione seu código aqui
10 # sobre os dados de Junho/2019
11 X_train = df_abt_train[features]
12
13 # target sobre os dados de Julho/2019
14 y_train = df_abt_train[target]
15
16 # sobre os dados de Julho/2019
17 X_oot = df_abt_oot[features]
18
19 # target sobre os dados de Agosto/2019
20 y_oot = df_abt_oot[target]
21
22 # Treine o melhor estimador (best_model)
23 grid_search.fit(X_train, y_train)
24 print(f'O melhor Score é {grid_search.best_score_} e, os melhores hiperparâmetros são: {grid_search.best_params_}')
25
26 # Melhor Modelo
27 best_model_teste2 = grid_search.best_estimator_
28 best_model_teste2.fit(X_train, y_train)
29
30 y_proba_oot = best_model_teste2.predict_proba(X_oot)[:, 1]
31 roc_auc_score_teste2 = roc_auc_score(y_oot, y_proba_oot)
32
33 print(f'lgbm: Performance do modelo na Out of Time com o uso da AUC: {roc_auc_score_teste2}')
34

Fitting 5 folds for each of 60 candidates, totalling 300 fits
O melhor Score é 0.8609328088268894 e, os melhores hiperparâmetros são: {'lgbm_learning_rate': 0.05, 'lgbm_max_depth': 5, 'lgbm_lgbm: Performance do modelo na Out of Time com o uso da AUC: 0.723730642250239

```

(1 ponto) q22 Houve melhora em relação ao modelo anterior?

```

1 desempenho =((roc_auc_score_teste1 / roc_auc_score_teste2)-1)
2
3 print('Resultado:')
4 print(f'O primeiro teste sem variáveis de horário do pedido apresentou um AUC de {roc_auc_score_teste1} ')
5 print(f'O segundo teste com as variáveis de horário do pedido apresentou um AUC de {roc_auc_score_teste2} ')
6 print(f'O desempenho do primeiro modelo foi de {desempenho*100}% melhor')

Resultado:
O primeiro teste sem variáveis de horário do pedido apresentou um AUC de 0.7372138323518183
O segundo teste com as variáveis de horário do pedido apresentou um AUC de 0.723730642250239
O desempenho do primeiro modelo foi de 1.8630121918918663% melhor

```

[RESPOSTA]: Não houve melhora comparado ao modelo anterior.

(1 ponto) q23 Qual foi o melhor modelo encontrado e quais as features utilizadas?

```

1 best_model.get_params('variables')

{'lgbm': LGBMClassifier(learning_rate=0.05, max_depth=3, random_state=42),
 'lgbm_boosting_type': 'gbdt',
 'lgbm_class_weight': None,
 'lgbm_colsample_bytree': 1.0,
 'lgbm_importance_type': 'split',
 'lgbm_learning_rate': 0.05,
 'lgbm_max_depth': 3,
 'lgbm_min_child_samples': 20,
 'lgbm_min_child_weight': 0.001,
 'lgbm_min_split_gain': 0.0,
 'lgbm_n_estimators': 100,
 'lgbm_n_jobs': -1,
 'lgbm_num_leaves': 31,

```

```
'lgbm__objective': None,
'lgbm__random_state': 42,
'lgbm__reg_alpha': 0.0,
'lgbm__reg_lambda': 0.0,
'lgbm__silent': True,
'lgbm__subsample': 1.0,
'lgbm__subsample_for_bin': 200000,
'lgbm__subsample_freq': 0,
'memory': None,
'numeric_imputer': MeanMedianImputer(variables=['receita_1m', 'qtd_pedidos_1m', 'recencia']),
'numeric_imputer__imputation_method': 'median',
'numeric_imputer__variables': ['receita_1m', 'qtd_pedidos_1m', 'recencia'],
'steps': [('numeric_imputer',
    MeanMedianImputer(variables=['receita_1m', 'qtd_pedidos_1m', 'recencia']))],
('lgbm', LGBMClassifier(learning_rate=0.05, max_depth=3, random_state=42))),
'verbose': False}
```

[RESPOSTA]: O melhor modelo foi o primeiro, onde foi utilizado as features: 'receita_1m', 'qtd_pedidos_1m', 'recencia'

(1 ponto) Q24 Rode o shap values para entender quais as variáveis mais importantes do modelo e como elas se relacionam com a variável target (código já está disponível, basta apenas executar e interpretar para responder as perguntas abaixo).

Responda:

- Qual a variável mais importante?
- Como ela se relaciona com a variável target?
- Como as outras variáveis se relacionam com a variável target?

```
1 !pip install shap==0.39.0
```

```
Collecting shap==0.39.0
  Downloading shap-0.39.0.tar.gz (356 kB)
    |██████████| 356 kB 26.3 MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from shap==0.39.0) (1.21.6)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from shap==0.39.0) (1.4.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from shap==0.39.0) (1.0.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from shap==0.39.0) (1.3.5)
Requirement already satisfied: tqdm>4.25.0 in /usr/local/lib/python3.7/dist-packages (from shap==0.39.0) (4.64.0)
Collecting slicer==0.0.7
  Downloading slicer-0.0.7-py3-none-any.whl (14 kB)
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-packages (from shap==0.39.0) (0.51.2)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.7/dist-packages (from shap==0.39.0) (1.3.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from numba->shap==0.39.0) (57.4.0)
Requirement already satisfied: llvmlite<0.35,>=0.34.0.dev0 in /usr/local/lib/python3.7/dist-packages (from numba->shap==0.39.0) (0.
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->shap==0.39.0) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->shap==0.39.0) (2022.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas->shap==0.39.
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->shap==0.39.0) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->shap==0.39.0) (3.
Building wheels for collected packages: shap
  Building wheel for shap (setup.py) ... done
  Created wheel for shap: filename=shap-0.39.0-cp37-cp37m-linux_x86_64.whl size=491691 sha256=3aeef1cdf55dc74e3bedff96245196455824b
  Stored in directory: /root/.cache/pip/wheels/ca/25/8f/6ae5df62c32651cd719e972e738a8aaa4a87414c4d2b14c9c0
Successfully built shap
Installing collected packages: slicer, shap
Successfully installed shap-0.39.0 slicer-0.0.7
```

```
1 features = ['receita_1m', 'qtd_pedidos_1m', 'recencia']
2
3 X_train = df_abt_train[features]
4 X_oos = df_abt_oos[features]
```

```
1 best_model
```

```
Pipeline(steps=[('numeric_imputer',
    MeanMedianImputer(variables=['receita_1m', 'qtd_pedidos_1m',
        'recencia'])),
('lgbm',
    LGBMClassifier(learning_rate=0.05, max_depth=3,
        random_state=42))])
```

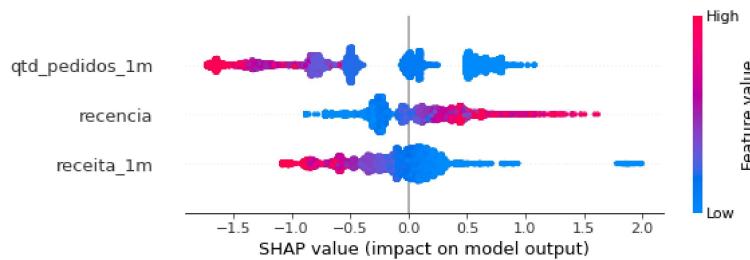
```
1 import shap
2 explainer = shap.TreeExplainer(best_model[-1])
```

```
1 X_oos_transformado = best_model[:-1].transform(X_oos)
2
3 # X_oos_transformado.shape
4 # X_oos_transformado.head(3)
```

```
1 shap_values = explainer.shap_values(X_oot_transformado)

LightGBM binary classifier with TreeExplainer shap values output has changed to a list of ndarray
```

```
1 shap.summary_plot(shap_values[1], X_oot_transformado, plot_type='dot')
```



ADICIONE SUAS RESPOSTAS AQUI

- Qual a variável mais importante?

A variável mais importante é qtd_pedidos_1m

- Como ela se relaciona com a variável target?

Observação:

A classe 0 em nosso modelo significa que não haverá churn no próximo mês

Enquanto que a classe 1 significa que haverá churn no próximo mês

A concentração de pontos "High" da feature "qtde_pedidos_1m" está concentrada no lado esquerdo, que é a classe 0, ou seja, que há tendência de que não haverá churn

Conclusão:

A interpretação é que se o valor desta feature for alto, haverá uma chance alta que não haverá churn no próximo mês

- Como as outras variáveis se relacionam com a variável target?

recencia: Quanto mais alto for a recência, maior a possibilidade de não haver churn

receita_1m: Quanto mais alto for esta feature, maior a possibilidade de não haver churn.