Modelo Regressão - GITAGE

Utilizando lazypredict

```
# 1) Instalando pacote
!pip install lazypredict
```

Retorno 👇

```
Requirement already satisfied: lazypredict in /usr/local/lib/python3.10/dist-packages (0.2.12)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from lazypredict) (8.1.7)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from lazypredict) (1.2.2
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from lazypredict) (1.5.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from lazypredict) (4.66.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from lazypredict) (1.3.2)
Requirement already satisfied: lightgbm in /usr/local/lib/python3.10/dist-packages (from lazypredict) (4.1.0)
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (from lazypredict) (2.0.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from lightgbm->lazypredict) (1.
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from lightgbm->lazypredict) (1.
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->l
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->lazypredict
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-lear
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1-
```

```
# 2) Importando bibliotecas
import pandas as pd
import numpy as np
import lazypredict
from lazypredict.Supervised import LazyClassifier
from lazypredict.Supervised import LazyRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingRegressor
```

```
# 3) Ignorando avisos
import warnings
warnings.filterwarnings("ignore")
```

```
# 4) Configurando quantidade de colunas para aparecer em um Datraframe
pd.set_option('display.max_columns',100)
```

```
# 5) Usando o método read_csv() para carregar o arquivo CSV
df = pd.read_csv('/content/dbGITAGE.csv')
```

```
# 6) Mostrando a base de dados
df.head()
```

Retorno 👇

```

Nota em Matemática  Nota em Leitura Nota em escrita Sexo Fem_0  Sexo Fem_1  Sexo Masc_0 Sexo Masc_1 Indígenas_0 I
0   -0.58   -0.03   0.59    0   1   1   0   1   0   1   0   1   0   0   1   1   0   1   0   1   0   1   0   0   1
1   1.85    1.60    1.19    1   0   0   1   1   0   1   0   1   0   0   1   1   0   1   0   1   0   0   1   1   0
2   -0.71   0.40    0.52    0   1   1   0   1   0   1   0   1   0   0   1   1   0   1   0   1   0   1   0   0   1
3   0.14    -0.03   -0.41   1   0   0   1   1   0   0   1   1   0   1   0   1   0   1   0   1   0   1   0   0   1
4   1.00    1.04    1.12    0   1   1   0   1   0   1   0   1   0   0   1   1   0   1   0   1   0   0   1   1   0
```

Dicionário de dados:

- a_mulher: aluna - sexo feminino
- a_homem: aluno - sexo masculino

- a_groupA: Indígenas
- a_groupB: Afrodescendentes
- a_groupC: Europeus
- a_groupD: Pardos
- a_groupE: Asiáticos

- p_ensino_fundamental: pai aluno - nível ensino fundamental
- p_ensino_medio: pai aluno - nível ensino medio
- p_tecnologo: pai aluno - nível ensino tecnologo
- p_alguma_faculdade: pai aluno - nível alguma faculdade
- p_bacharelado: pai aluno - nível bacharelado
- p_mestrado: pai aluno - nível mestrado

- a_curso_de_preparacao: aluno - nível curso preparatório
- a_score_em_matematica: aluno - nota em matemática
- a_score_em_leitura: aluno - nota em leitura
- a_score_em_escrita: aluno - nota em escrita

Origem da base de dados:

- https://www.kaggle.com/code/budhadityadutta/student-performance-analysis-prediction

```
# 7) Renomeando colunas

df.rename(columns={'a_mulher':'Sexo Fem',
                   'a_homem':'Sexo Masc',
                   'a_groupA':'Indígenas',
                   'a_groupB':'Afrodescendentes',
                   'a_groupC':'Europeus',
                   'a_groupD':'Pardos',
                   'a_groupE':'Asiáticos',
                   'p_ensino_fundamental':'Fundamental pai',
                   'p_ensino_medio':'Ensino médio pai',
                   'p_tecnologo':'Tecnólogo pai',
                   'p_alguma_faculdade':'Alguma faculdade pai',
                   'p_bacharelado':'Bacharelado pai',
                   'p_mestrado':'Mestrado pai',
                   'a_curso_de_preparacao':'Curso preparatório',
                   'a_score_em_matematica':'Nota em Matemática',
                   'a_score_em_leitura':'Nota em Leitura',
                   'a_score_em_escrita':'Nota em escrita'},inplace=True)

```

```
# 8) Verificando a base de dados com tipo
df.info()
```

Retorno 👇

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 31 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Nota em Matemática     1000 non-null   float64
 1   Nota em Leitura        1000 non-null   float64
 2   Nota em escrita        1000 non-null   float64
 3   Sexo Fem_0             1000 non-null   int64
 4   Sexo Fem_1             1000 non-null   int64
 5   Sexo Masc_0            1000 non-null   int64
 6   Sexo Masc_1            1000 non-null   int64
 7   Indígenas_0            1000 non-null   int64
```

```
 8   Indígenas_1             1000 non-null   int64
 9   Afrodescendentes_0      1000 non-null   int64
 10  Afrodescendentes_1      1000 non-null   int64
 11  Europeus_0              1000 non-null   int64
 12  Europeus_1              1000 non-null   int64
 13  Pardos_0                1000 non-null   int64
 14  Pardos_1                1000 non-null   int64
 15  Asiáticos_0             1000 non-null   int64
 16  Asiáticos_1             1000 non-null   int64
 17  Fundamental pai_0       1000 non-null   int64
 18  Fundamental pai_1       1000 non-null   int64
 19  Ensino médio pai_0      1000 non-null   int64
 20  Ensino médio pai_1      1000 non-null   int64
 21  Tecnólogo pai_0         1000 non-null   int64
 22  Tecnólogo pai_1         1000 non-null   int64
 23  Alguma faculdade pai_0  1000 non-null   int64
 24  Alguma faculdade pai_1  1000 non-null   int64
 25  Bacharelado pai_0       1000 non-null   int64
 26  Bacharelado pai_1       1000 non-null   int64
 27  Mestrado pai_0          1000 non-null   int64
 28  Mestrado pai_1          1000 non-null   int64
 29  Curso preparatório_0    1000 non-null   int64
 30  Curso preparatório_1    1000 non-null   int64
dtypes: float64(3), int64(28)
memory usage: 242.3 KB
```

```python
# 9) Verificando a base de dados com cabeçalho
df.head()
```

Retorno 👇

	Nota em Matemática	Nota em Leitura	Nota em escrita	Sexo Fem_0	Sexo Fem_1	Sexo Masc_0	Sexo Masc_1	Indígenas																	
0	-0.58	-0.03	0.59	0	1	1	0	1	0	1	0	1	0	0	1	1	0	1	0	1	0	1	0	0	1
1	1.85	1.60	1.19	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0	0	1	1	0
2	-0.71	0.40	0.52	0	1	1	0	1	0	1	0	1	0	0	1	1	0	1	0	1	0	1	0	0	1
3	0.14	-0.03	-0.41	1	0	0	1	1	0	0	1	1	0	1	0	1	0	1	0	1	0	1	0	0	1
4	1.00	1.04	1.12	0	1	1	0	1	0	1	0	1	0	0	1	1	0	1	0	1	0	0	1	1	0

```python
# 10) Mudando o tipo de dado

col_cat = list(df.select_dtypes(include='object').columns)
df[col_cat] = df[col_cat].astype('category')
```

```python
# 11) Verificando todas as categorias dos atributos
categorico_val = []
continuo_val = []

# 12) Verificando os dados únicos em cada coluna e separando em dados categóricos e continuos
for column in df.columns:
    print('==============================')
    print(f"{column} : {df[column].unique()}")

    if len(df[column].unique()) <= 10:
        categorico_val.append(column)
    else:
        continuo_val.append(column)
```

Retorno 👇

```
==============================
Nota em Matemática : [-0.57798657  1.84942581 -0.70919805  0.14367657  0.99655119  0.01246509
  0.93094545 -1.43086119  0.79973397  0.40609953 -0.97162101  0.53731101
 -0.84040953 -0.77480379 -2.15252433  1.25897415 -0.18435213 -0.11874639
 -0.05314065  1.39018563  2.04624304 -0.64359231  1.45579137  0.07807083
 -2.34934155  1.19336841 -1.03722675 -0.31556361 -0.38116935  0.73412823
  0.20928231 -0.44677509  0.47170527  0.34049379 -0.24995787 -2.93979322
 -1.75888989  1.06215693  1.12776267  1.32457989  1.52139711 -1.10283249
  2.11184878 -1.29964971  1.98063729  0.27488805 -1.62767841 -1.16843823
 -1.95570711  0.86533971 -1.36525545 -1.23404397 -1.56207267 -0.51238083
  1.65260859  0.60291675 -3.13661044  0.66852249  1.71821433 -1.89010137
 -1.69328415 -0.90601527 -2.02131285 -1.49646693 -2.67737026  1.58700285
  1.78382007 -2.21813007  1.91503155 -2.41494729 -1.82449563 -2.28373581
 -3.0710047  -2.80858174 -2.61176452 -3.46463914 -2.87418748 -2.48055303
 -2.08691859]
==============================
Nota em Leitura : [-0.02709151  1.60407283  0.39842962  1.03671132 -0.94905396  0.89487094
 -0.66537321  0.32750943 -0.0980117  -1.44549529  0.25658924  0.18566905
 -1.16181453 -1.72917604 -1.65825585  1.53315264  0.82395075  0.04382868
 -0.31077227  1.24947188 -1.30365491  0.61119019  1.1785517   0.75303056
  0.54027    -2.50929812 -1.23273472 -1.01997415  0.46934981  0.11474887
 -0.23985208 -0.45261264  1.10763151 -0.16893189 -2.5802183   1.46223245
 -0.59445302 -2.65113849 -0.87813378 -1.51641547  0.68211037  1.32039207
  1.88775358 -0.38169246 -1.3745751   1.81683339  0.96579113  1.39131226
  1.7459132  -1.09089434 -0.7362934   2.02959396 -1.80009623 -0.52353283
  1.95867377 -2.15469717  1.67499302 -0.80721359 -1.58733566 -3.2185
 -2.22561736 -1.94193661 -1.87101642 -3.07665962  2.10051415 -2.01285679
 -2.86389906 -2.72205868 -2.29653755 -2.08377698 -2.36745774]
==============================
Nota em escrita : [ 0.58994292  1.18920774  0.52335794 -0.40883178  1.12262276 -1.0080966
  0.72311288 -0.74175668  0.25701802 -0.0093219  -1.3410215   0.45677296
  0.05726308 -1.14126656 -1.74053138 -1.87370134  0.78969786  1.05603778
  0.323603   -0.47541676  0.12384806  1.25579272  0.19043304 -1.20785154
 -0.6751717   0.6565279   0.39018798 -2.14004126  0.85628284 -1.40760648
 -0.14249186 -0.27566182  1.38896268 -0.07590688 -0.87492664  0.9894528
 -0.54200174  1.58871762 -0.20907684 -2.20662624 -0.80834166 -2.40638118
 -1.54077644  1.3223777   1.72188758 -1.80711636  1.45554766 -0.60858672
 -2.07345628  0.92286782  1.85505754 -0.3422468  -1.07468158  2.05481248
 -0.94151162 -1.47419146 -1.27443652  1.6553026  -1.6739464   1.9882275
  1.52213264 -3.60491082 -1.94028632  1.92164252 -3.3385709   1.78847256
 -2.27321122 -2.47296616 -2.87247604 -2.73930608 -1.60736142 -2.80589106
 -2.60613612 -3.07223098]
==============================
Sexo Fem_0 : [0 1]
==============================
Sexo Fem_1 : [1 0]
==============================
Sexo Masc_0 : [1 0]
==============================
Sexo Masc_1 : [0 1]
==============================
Indígenas_0 : [1 0]
==============================
Indígenas_1 : [0 1]
==============================
Afrodescendentes_0 : [1 0]
==============================
Afrodescendentes_1 : [0 1]
```

```
==============================
Europeus_0 : [1 0]
==============================
Europeus_1 : [0 1]
==============================
Pardos_0 : [0 1]
==============================
Pardos_1 : [1 0]
==============================
Asiáticos_0 : [1 0]
==============================
Asiáticos_1 : [0 1]
==============================
Fundamental pai_0 : [1 0]
==============================
Fundamental pai_1 : [0 1]
==============================
Ensino médio pai_0 : [1 0]
==============================
Ensino médio pai_1 : [0 1]
==============================
Tecnólogo pai_0 : [1 0]
==============================
Tecnólogo pai_1 : [0 1]
==============================
Alguma faculdade pai_0 : [0 1]
==============================
Alguma faculdade pai_1 : [1 0]
==============================
Bacharelado pai_0 : [1 0]
==============================
Bacharelado pai_1 : [0 1]
==============================
Mestrado pai_0 : [1 0]
==============================
Mestrado pai_1 : [0 1]
==============================
Curso preparatório_0 : [0 1]
==============================
Curso preparatório_1 : [1 0]
```

```
# 13) Checando as variáveis categóricas
categorico_val
```

Retorno 👇

```
['Sexo Fem_0',
 'Sexo Fem_1',
 'Sexo Masc_0',
 'Sexo Masc_1',
 'Indígenas_0',
 'Indígenas_1',
 'Afrodescendentes_0',
 'Afrodescendentes_1',
 'Europeus_0',
 'Europeus_1',
 'Pardos_0',
 'Pardos_1',
```

```
 'Asiáticos_0',
 'Asiáticos_1',
 'Fundamental pai_0',
 'Fundamental pai_1',
 'Ensino médio pai_0',
 'Ensino médio pai_1',
 'Tecnólogo pai_0',
 'Tecnólogo pai_1',
 'Alguma faculdade pai_0',
 'Alguma faculdade pai_1',
 'Bacharelado pai_0',
 'Bacharelado pai_1',
 'Mestrado pai_0',
 'Mestrado pai_1',
 'Curso preparatório_0',
 'Curso preparatório_1']
```

```
# 14) Checando as variáveis continuas
continuo_val
```

Retorno 👇

```
['Nota em Matemática', 'Nota em Leitura', 'Nota em escrita']
```

```
# 15) Preparando os dados categóricos
dataset = pd.get_dummies(df, columns = categorico_val)

# 16) Criando objeto scaler
scaler = StandardScaler()

# 17) Padronizando dados continuos
col_to_scale = ['Nota em Matemática','Nota em Leitura','Nota em escrita']

# 18) Adequando a base de dados para machine learning
dataset[col_to_scale] = scaler.fit_transform(dataset[col_to_scale])
```

```
# 19) Visualizando Dataset padronizando
dataset.head(10)
```

Retorno 👇

	Nota em Matemática	Nota em Leitura	Nota em escrita	Sexo Fem_0_0		Sexo Fem_0_1		Sexo Fem_1_0		Sexo Fem	
0	-0.58	-0.03	0.59	1	0	0	1	0	1	1	0
1	1.85	1.60	1.19	0	1	1	0	1	0	0	1
2	-0.71	0.40	0.52	1	0	0	1	0	1	1	0
3	0.14	-0.03	-0.41	0	1	1	0	1	0	0	1
4	1.00	1.04	1.12	1	0	0	1	0	1	1	0
5	0.01	-0.95	-1.01	0	1	1	0	1	0	0	1
6	0.93	0.89	0.72	1	0	0	1	0	1	1	0
7	-1.43	-0.67	-0.74	1	0	0	1	0	1	1	0
8	0.80	0.33	0.26	0	1	1	0	1	0	0	1
9	-0.71	-0.10	0.52	1	0	0	1	0	1	1	0

```
# 20) Visualizando o Dataset padronizado e categorias preparadas
dataset.info()
```

Retorno 👇

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 59 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Nota em Matemática        1000 non-null   float64
 1   Nota em Leitura           1000 non-null   float64
 2   Nota em escrita           1000 non-null   float64
 3   Sexo Fem_0_0              1000 non-null   uint8
 4   Sexo Fem_0_1              1000 non-null   uint8
 5   Sexo Fem_1_0              1000 non-null   uint8
 6   Sexo Fem_1_1              1000 non-null   uint8
 7   Sexo Masc_0_0             1000 non-null   uint8
 8   Sexo Masc_0_1             1000 non-null   uint8
 9   Sexo Masc_1_0             1000 non-null   uint8
 10  Sexo Masc_1_1             1000 non-null   uint8
 11  Indígenas_0_0             1000 non-null   uint8
 12  Indígenas_0_1             1000 non-null   uint8
 13  Indígenas_1_0             1000 non-null   uint8
 14  Indígenas_1_1             1000 non-null   uint8
 15  Afrodescendentes_0_0      1000 non-null   uint8
 16  Afrodescendentes_0_1      1000 non-null   uint8
 17  Afrodescendentes_1_0      1000 non-null   uint8
 18  Afrodescendentes_1_1      1000 non-null   uint8
 19  Europeus_0_0              1000 non-null   uint8
 20  Europeus_0_1              1000 non-null   uint8
 21  Europeus_1_0              1000 non-null   uint8
 22  Europeus_1_1              1000 non-null   uint8
 23  Pardos_0_0                1000 non-null   uint8
 24  Pardos_0_1                1000 non-null   uint8
 25  Pardos_1_0                1000 non-null   uint8
 26  Pardos_1_1                1000 non-null   uint8
 27  Asiáticos_0_0             1000 non-null   uint8
 28  Asiáticos_0_1             1000 non-null   uint8
 29  Asiáticos_1_0             1000 non-null   uint8
 30  Asiáticos_1_1             1000 non-null   uint8
 31  Fundamental pai_0_0       1000 non-null   uint8
 32  Fundamental pai_0_1       1000 non-null   uint8
 33  Fundamental pai_1_0       1000 non-null   uint8
 34  Fundamental pai_1_1       1000 non-null   uint8
 35  Ensino médio pai_0_0      1000 non-null   uint8
 36  Ensino médio pai_0_1      1000 non-null   uint8
 37  Ensino médio pai_1_0      1000 non-null   uint8
 38  Ensino médio pai_1_1      1000 non-null   uint8
 39  Tecnólogo pai_0_0         1000 non-null   uint8
 40  Tecnólogo pai_0_1         1000 non-null   uint8
 41  Tecnólogo pai_1_0         1000 non-null   uint8
 42  Tecnólogo pai_1_1         1000 non-null   uint8
 43  Alguma faculdade pai_0_0  1000 non-null   uint8
 44  Alguma faculdade pai_0_1  1000 non-null   uint8
 45  Alguma faculdade pai_1_0  1000 non-null   uint8
 46  Alguma faculdade pai_1_1  1000 non-null   uint8
 47  Bacharelado pai_0_0       1000 non-null   uint8
 48  Bacharelado pai_0_1       1000 non-null   uint8
 49  Bacharelado pai_1_0       1000 non-null   uint8
 50  Bacharelado pai_1_1       1000 non-null   uint8
 51  Mestrado pai_0_0          1000 non-null   uint8
 52  Mestrado pai_0_1          1000 non-null   uint8
```

```
 53  Mestrado pai_1_0          1000 non-null   uint8
 54  Mestrado pai_1_1          1000 non-null   uint8
 55  Curso preparatório_0_0    1000 non-null   uint8
 56  Curso preparatório_0_1    1000 non-null   uint8
 57  Curso preparatório_1_0    1000 non-null   uint8
 58  Curso preparatório_1_1    1000 non-null   uint8
dtypes: float64(3), uint8(56)
memory usage: 78.2 KB
```

```
# 21) Exportando no Google Drive.
dataset.to_csv('dbGITAGE.csv', index=False)
```

```
# 22) Identificando Recursos (X): Todas as colunas, exceto "Nota em Matemática"
X = dataset.drop('Nota em Matemática', axis=1)

# 23) Identificando Variável Alvo (Y): A coluna "Nota em Matemática"
y = dataset['Nota em Matemática']
```

```
# 24) Separando a base em Treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.2,
                                                    random_state=1)
```

```
# 25) Criando um objeto do pacote Lazy
clf = LazyRegressor(verbose=0,
                    ignore_warnings=True,
                    custom_metric=None)
```

```
# 26) Ajustando os dados de treinamento e teste
models, predictions = clf.fit(X_train, X_test, y_train, y_test)
models
```

Retorno 👇

```
100%|██████████| 42/42 [00:11<00:00,  3.54it/s]
```

	Adjusted R-Squared	R-Squared	RMSE	Time Taken
Model				
OrthogonalMatchingPursuit	0.77	0.84	0.36	0.02
OrthogonalMatchingPursuitCV	0.77	0.84	0.36	0.03
LassoCV	0.77	0.84	0.36	0.14
ElasticNetCV	0.77	0.84	0.36	0.26
LassoLarsIC	0.77	0.84	0.36	0.05
LassoLarsCV	0.77	0.84	0.36	0.09
HuberRegressor	0.76	0.83	0.37	0.04

BayesianRidge	0.76	0.83	0.37	0.02
KernelRidge	0.76	0.83	0.37	0.04
RidgeCV	0.76	0.83	0.37	0.12
Ridge	0.76	0.83	0.37	0.08
TransformedTargetRegressor	0.76	0.83	0.37	0.10
LinearRegression	0.76	0.83	0.37	0.05
LinearSVR	0.76	0.83	0.37	0.15
RANSACRegressor	0.76	0.83	0.37	1.55
SGDRegressor	0.75	0.82	0.38	0.11
GradientBoostingRegressor	0.74	0.81	0.39	0.25
SVR	0.73	0.81	0.39	0.32
LarsCV	0.73	0.81	0.39	0.14
NuSVR	0.73	0.81	0.40	0.12
RandomForestRegressor	0.71	0.80	0.41	3.43
AdaBoostRegressor	0.71	0.79	0.41	0.18
MLPRegressor	0.70	0.79	0.41	0.92
ExtraTreesRegressor	0.69	0.78	0.42	0.94
LGBMRegressor	0.69	0.78	0.42	0.18
BaggingRegressor	0.67	0.77	0.43	0.08
HistGradientBoostingRegressor	0.67	0.77	0.43	0.81
TweedieRegressor	0.66	0.76	0.44	0.29
XGBRegressor	0.66	0.76	0.44	0.82
ExtraTreeRegressor	0.45	0.61	0.56	0.05
DecisionTreeRegressor	0.37	0.56	0.60	0.03
KNeighborsRegressor	0.34	0.53	0.61	0.02
ElasticNet	0.12	0.37	0.71	0.02
PassiveAggressiveRegressor	0.09	0.36	0.72	0.02
Lasso	-0.41	-0.00	0.90	0.02
DummyRegressor	-0.41	-0.00	0.90	0.02

LassoLars	-0.41	-0.00	0.90	0.04
GaussianProcessRegressor	-88.70	-62.56	7.16	0.14
Lars	-168.91	-119.39	9.85	0.05

```
# 27) Criando o modelo GradientBoostingRegressor
regressor = GradientBoostingRegressor()

# 28) Treinando o modelo
regressor.fit(X_train, y_train)

# 29) Fazendo previsões
y_pred = regressor.predict(X_test)
```

```
# 30) Adaptando função Regressão para o SGDRegressor

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

def print_regression_score(model, X_train, y_train, X_test, y_test, train=True):
    if train:
        y_pred_train = model.predict(X_train)
        mse_train = mean_squared_error(y_train, y_pred_train)
        r2_train = r2_score(y_train, y_pred_train)
        mae_train = mean_absolute_error(y_train, y_pred_train)
        rmse_train = mean_squared_error(y_train, y_pred_train, squared=False)  # Calcula o RMSE
        # Calcula o MAPE
        mape_train = np.mean(np.abs((y_train - y_pred_train) / y_train)) * 100

        print("Resultado do treinamento:\n============================")
        print(f"Mean Absolute Error (MAE) - Treinamento: {mae_train:.2f}")
        print(f"Mean Squared Error (MSE) - Treinamento: {mse_train:.2f}")
        print(f"Root Mean Squared Error (RMSE) - Treinamento: {rmse_train:.2f}")
        print(f"Mean Absolute Percentage Error (MAPE) - Treinamento: {mape_train:.2f}%")
        print(f"R-squared (R2) - Treinamento: {r2_train:.2f}")

    else:
        y_pred_test = model.predict(X_test)
        mse_test = mean_squared_error(y_test, y_pred_test)
        r2_test = r2_score(y_test, y_pred_test)
        mae_test = mean_absolute_error(y_test, y_pred_test)
        rmse_test = mean_squared_error(y_test, y_pred_test, squared=False)  # Calcula o RMSE

        # Calcula o MAPE
        mape_test = np.mean(np.abs((y_test - y_pred_test) / y_test)) * 100

        print("Resultado do teste:\n============================")
        print(f"Mean Absolute Error (MAE) - Teste: {mae_test:.2f}")
        print(f"Mean Squared Error (MSE) - Teste: {mse_test:.2f}")
        print(f"Root Mean Squared Error (RMSE) - Teste: {rmse_test:.2f}")
        print(f"Mean Absolute Percentage Error (MAPE) - Teste: {mape_test:.2f}%")
        print(f"R-squared (R2) - Teste: {r2_test:.2f}")
```

```
# 31) Avaliando o modelo SGDRegressor
print_regression_score(regressor, X_train, y_train, X_test, y_test, train=True)
print_regression_score(regressor, X_train, y_train, X_test, y_test, train=False)
```

Retorno 👇

```
Resultado do treinamento:
============================
Mean Absolute Error (MAE) - Treinamento: 0.26
Mean Squared Error (MSE) - Treinamento: 0.11
Root Mean Squared Error (RMSE) - Treinamento: 0.33
Mean Absolute Percentage Error (MAPE) - Treinamento: 125.17%
R-squared (R2) - Treinamento: 0.89
Resultado do teste:
============================
Mean Absolute Error (MAE) - Teste: 0.32
Mean Squared Error (MSE) - Teste: 0.15
Root Mean Squared Error (RMSE) - Teste: 0.39
Mean Absolute Percentage Error (MAPE) - Teste: 177.45%
R-squared (R2) - Teste: 0.81
```

```python
# 32) Calculando o r-quadrado ajustado
def adjusted_r2(y_test, y_pred,X_train):
    from sklearn.metrics import r2_score
    adj_r2 = (1 - ((1 - r2_score(y_test, y_pred)) * (len(y_test) - 1)) / (len(y_test) - X_train.shape[1] - 1))
    return adj_r2
```

```python
# 33) Imprimindo r-quadrado ajustado
print(round(adjusted_r2(y_test,y_pred,X_train),2))
```

```python
# 34) Salvando o modelo
from joblib import dump

# Salvando o modelo treinado
dump(regressor, 'modelo_score_aluno_reg')
```

Retorno 👇

```
['modelo_score_aluno_reg']
```