Classificação de Florestas

Este trabalho tem como objetivo realizar melhorias para a submissão realizada no 1º semestre, prever o tipo de floresta com base em variáveis numéricas, utilizando algoritmos de aprendizagem supervisionada.

Objetivos

Dos principais objectivos as melhorias foram:

2. Testar e avaliar diferentes algoritmos e estratégias de validação;
3. Identificar o(s) modelo(s) com melhor desempenho;
4. Submeter os modelos na plataforma Kaggle;
5. Realizar uma análise aprofundada dos resultados obtidos.

Melhorias do trabalho

Normalização dos dados com escalonamento simples;

Acrescentado um conjunto de validação para avaliar o modelo.

Utilização recorrente da técnica de GridSearch;

Avaliação do desempenho com a média de erros absolutos;

Utilização da técnica BaggingClassifier;

Apresentação de todos os modelos submetidos e seus desempenhos publicos e privados;

train_test_split(stratify=data_train_y) argumento que garante que as classes estejam divididas em ambos os conjuntos: treino e teste;

Tratamento e Análise de dados

In [ ]:

```
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

from datetime import datetime
start_time = datetime.now()
```

Após as importações realizadas ao longo da implementação, procedemos ao carregamento dos dados, à remoção e separação de colunas relevantes, como a variável **'id'** e a classe **'floresta'**, além de realizar uma verificação minuciosa dos dados.

In [ ]:
```
### Carregamento de Dados
data = pd.read_csv('sample_data/train.csv') #
test_final = pd.read_csv('sample_data/test.csv')

### Remoção da Coluna 'id'
data = data.drop('id',axis=1) # 'id' é o nome da coluna no ficheiro csv

### Separação de Variáveis Descritivas da de Classe
data_train_X = data.drop('floresta',axis=1) # 'floresta' é o nome da coluna no f
data_train_y = data.floresta

### Separação da Coluna 'id' das Variáveis Descritivas
data_test_X = test_final.drop('id',axis=1)
data_test_id = test_final.id

### Verificação Inicial dos Dados
print("data_train_X.head")
print()
print(data_train_X.head())
print()
print()

print("target.head")
print()
print(data_train_y.head())
print()
print()

print("data.head()")
print()
print(data.head())
print()
print()

print("data_test_X.head")
print()
print(data_test_X.head())
print()
print()

print("data_test_id.head")
print()
print(data_test_id.head())
print()
print()
```

```
data_train_X.head

   elevacao  aspeto  inclinacao  dh_agua  dv_agua  dh_estrada  sombra_9  \
0      2596      51           3      258        0         510       221
1      2785     155          18      242      118        3090       238
2      2579     132           6      300      -15          67       230
3      2606      45           7      270        5         633       222
4      2605      49           4      234        7         573       222

   sombra_12  sombra_15  dh_Incendio  area  solo
0        232        148         6279     4     5
1        238        122         6211     4     5
2        237        140         6031     4     5
3        225        138         6256     4     5
4        230        144         6228     4     5
```

```
target.head

0    6
1    3
2    3
3    6
4    6
Name: floresta, dtype: int64
```

```
data.head()

   elevacao  aspeto  inclinacao  dh_agua  dv_agua  dh_estrada  sombra_9  \
0      2596      51           3      258        0         510       221
1      2785     155          18      242      118        3090       238
2      2579     132           6      300      -15          67       230
3      2606      45           7      270        5         633       222
4      2605      49           4      234        7         573       222

   sombra_12  sombra_15  dh_Incendio  area  solo  floresta
0        232        148         6279     4     5         6
1        238        122         6211     4     5         3
2        237        140         6031     4     5         3
3        225        138         6256     4     5         6
4        230        144         6228     4     5         6
```

```
data_test_X.head

   elevacao  aspeto  inclinacao  dh_agua  dv_agua  dh_estrada  sombra_9  \
0      2703     330          27       30       17        3141       146
1      2524      94           7      212       -4         684       232
2      2536      99           6      234        0         659       230
3      2489      11           4      175       13         840       216
4      2493      63          10      127       20         840       229

   sombra_12  sombra_15  dh_Incendio  area  solo
0        197        184         6186     4     7
1        229        130         5474     4    18
2        232        136         5475     4    18
3        232        153         5254     4    18
4        221        124         5197     4    18
```

```
data_test_id.head

0    10621
1    10622
2    10623
3    10624
4    10625
Name: id, dtype: int64
```

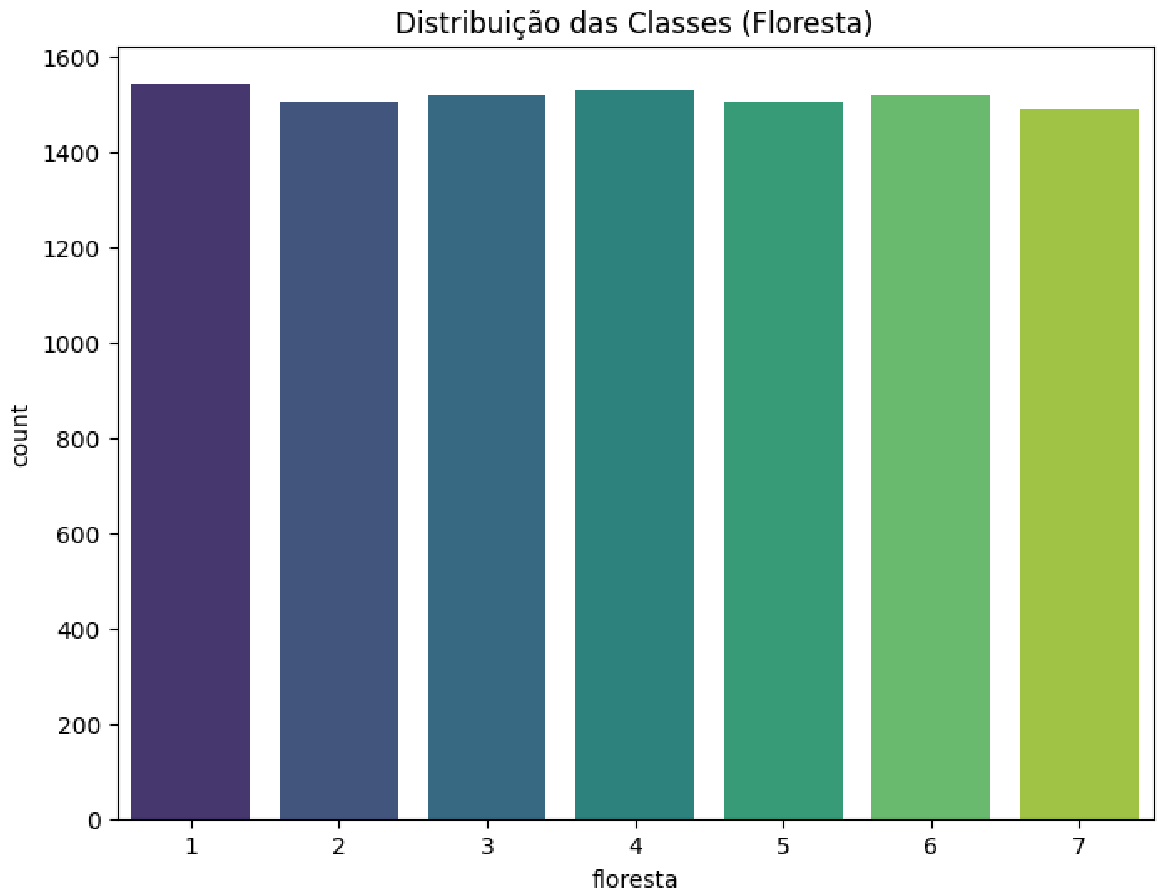
In [ ]:

```python
### Informações Gerais e Estatísticas Descritivas
print("Info:\n")
print(data.info)
print()
print()

print("Descrição:\n")
print(data.describe())
print()
print()

print("Valores nulos:\n")
print(data.isnull().sum())
print()
print()

### Visualização da Distribuição das Etiquetas da Classe 'floresta'
plt.figure(figsize = (8, 6))
sns.countplot(x='floresta', data=data, palette='viridis')
plt.title("Distribuição das Classes (Floresta)")
plt.show()
print()
```

```
Info:

<bound method DataFrame.info of        elevacao  aspeto  inclinacao  dh_agua  dv_
agua  dh_estrada  sombra_9  \
0          2596      51           3      258        0         510       221
1          2785     155          18      242      118        3090       238
2          2579     132           6      300      -15          67       230
3          2606      45           7      270        5         633       222
4          2605      49           4      234        7         573       222
...         ...     ...         ...      ...      ...         ...       ...
10615      2617      45           9      240       56         666       223
10616      2503     157           4       67        4         674       224
10617      2610     259           1      120       -1         607       216
10618      2570     346           2        0        0         331       215
10619      2533      71           9      150       -3         577       230

       sombra_12  sombra_15  dh_Incendio  area  solo  floresta
0            232        148         6279     4     5         6
1            238        122         6211     4     5         3
2            237        140         6031     4     5         3
3            225        138         6256     4     5         6
4            230        144         6228     4     5         6
...          ...        ...          ...   ...   ...       ...
10615        221        133         6244     4     5         6
10616        240        151         5600     4    18         6
10617        239        161         6096     4     5         6
10618        235        158         5745     4     5         3
10619        223        126         5552     4    18         6

[10620 rows x 13 columns]>


Descrição:

           elevacao        aspeto    inclinacao       dh_agua       dv_agua  \
count  10620.000000  10620.000000  10620.000000  10620.00000  10620.000000
mean    2752.124200    156.575047     16.578437    228.42580     51.808945
std      417.881891    110.020251      8.481794    209.45953     61.291132
min     1879.000000      0.000000      0.000000      0.00000   -134.000000
25%     2378.000000     64.000000     10.000000     67.00000      5.000000
50%     2755.000000    125.000000     15.000000    180.00000     33.000000
75%     3109.000000    260.000000     22.000000    330.00000     80.000000
max     3849.000000    360.000000     52.000000   1343.00000    554.000000

         dh_estrada      sombra_9     sombra_12     sombra_15   dh_Incendio  \
count  10620.000000  10620.000000  10620.000000  10620.000000  10620.000000
mean    1723.080226    212.710264    218.830414    134.864407   1516.787571
std     1329.501289     30.615163     22.963430     46.221620   1111.750922
min        0.000000      0.000000     99.000000      0.000000      0.000000
25%      768.000000    196.000000    207.000000    106.000000    726.000000
50%     1318.000000    220.000000    222.000000    138.000000   1260.000000
75%     2278.250000    235.000000    235.000000    167.000000   1994.000000
max     6890.000000    254.000000    254.000000    247.000000   6853.000000

               area          solo      floresta
count  10620.000000  10620.000000  10620.000000
mean       2.198964      9.698776      3.985782
std        1.119837      6.038451      1.999785
min        1.000000      1.000000      1.000000
25%        1.000000      4.000000      2.000000
```

```
50%         2.000000     11.000000      4.000000
75%         3.000000     13.000000      6.000000
max         4.000000     21.000000      7.000000


Valores nulos:

elevacao       0
aspeto         0
inclinacao     0
dh_agua        0
dv_agua        0
dh_estrada     0
sombra_9       0
sombra_12      0
sombra_15      0
dh_Incendio    0
area           0
solo           0
floresta       0
dtype: int64
```

```
/tmp/ipython-input-6-2301825852.py:19: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.countplot(x='floresta', data=data, palette='viridis')
```



Normalização dos dados

- Escalonamento Simples (Simple Feature Scaling): esta tecnica consiste em dividir cada valor pelo valor máximo da coluna:

In [ ]:
```
# Estes atributos, por serem categoricos são excluidos do escalonamento ['area',

features = ['elevacao', 'aspeto', 'inclinacao', 'dh_agua', 'dv_agua', 'dh_estrad
            'sombra_9', 'sombra_12', 'sombra_15', 'dh_Incendio']
for feature in features:
    data_train_X[feature] = data_train_X[feature] / data_train_X[feature].max()
    data_test_X[feature] = data_test_X[feature] / data_test_X[feature].max()

print(data_train_X.head())
print()
print(data_test_X.head())
print()
```

```
   elevacao    aspeto  inclinacao   dh_agua   dv_agua  dh_estrada  sombra_9  \
0  0.674461  0.141667    0.057692  0.192107  0.000000    0.074020  0.870079
1  0.723565  0.430556    0.346154  0.180194  0.212996    0.448476  0.937008
2  0.670044  0.366667    0.115385  0.223380 -0.027076    0.009724  0.905512
3  0.677059  0.125000    0.134615  0.201042  0.009025    0.091872  0.874016
4  0.676799  0.136111    0.076923  0.174237  0.012635    0.083164  0.874016

   sombra_12  sombra_15  dh_Incendio  area  solo
0   0.913386   0.599190     0.916241     4     5
1   0.937008   0.493927     0.906318     4     5
2   0.933071   0.566802     0.880053     4     5
3   0.885827   0.558704     0.912885     4     5
4   0.905512   0.582996     0.908799     4     5

   elevacao    aspeto  inclinacao   dh_agua   dv_agua  dh_estrada  sombra_9  \
0  0.702260  0.919220        0.54  0.023184  0.042184    0.464233  0.574803
1  0.655755  0.261838        0.14  0.163833 -0.009926    0.101094  0.913386
2  0.658872  0.275766        0.12  0.180835  0.000000    0.097399  0.905512
3  0.646661  0.030641        0.08  0.135240  0.032258    0.124150  0.850394
4  0.647701  0.175487        0.20  0.098145  0.049628    0.124150  0.901575

   sombra_12  sombra_15  dh_Incendio  area  solo
0   0.775591   0.741935     0.884599     4     7
1   0.901575   0.524194     0.782783     4    18
2   0.913386   0.548387     0.782926     4    18
3   0.913386   0.616935     0.751323     4    18
4   0.870079   0.500000     0.743172     4    18
```

Metodo de avaliação de desempenho Divisão Treino-Teste

In [ ]:
```
### train_size, test_size e random_state val's
train_size_val = 0.75
valid_size_val = 0.25
random_state_val = 1364
```

In [ ]:
```
### Divisão dos Dados em Treino e Teste

print()
```

```
print("train_size_val ->", train_size_val)
print("valid_size_val -> ", valid_size_val)
print("random_state_val -> ", random_state_val)

res = train_test_split(data_train_X, data_train_y,
                       train_size= train_size_val,
                       test_size=valid_size_val,
                       random_state=random_state_val, stratify=data_train_y)
X_train, X_valid, y_train, y_valid = res
```

```
train_size_val -> 0.75
valid_size_val ->  0.25
random_state_val ->  1364
```

Após concluirmos o tratamento dos dados e realizarmos uma análise com mínimo de detalhe para compreender o tipo de dados que estavamos a lidar, as nossas primeiras abordagens foram explorar algoritmos previamente estudados no âmbito da disciplina, testando-os com o conjunto de dados em análise.

Experiências para melhores modelos

- **selecionar os valores para os melhores parâmetros do modelo**
- **Pesquisar o valor dos parametros para o melhor modelo com GridSearch**
- **submeter parâmetros obtidos**
- **Filtrar valores para os parâmetros**
- **selecionar valor de cada um dos parâmetros com menor erro absoluto médio**
- **submeter com melhores parâmetros**

DecisionTreeClassifier

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_absolute_error


tree_model = DecisionTreeClassifier(random_state=1)
tree_model.fit(X_train, y_train)
tree_model.predict(X_valid)

print(tree_model.score(X_train, y_train))
print(tree_model.score(X_valid, y_valid))

val_mean_error = mean_absolute_error(tree_model.predict(X_valid), y_valid)
print(val_mean_error)

tree_model = DecisionTreeClassifier(random_state=1)
tree_model.fit(data_train_X, data_train_y)
tree_model.predict(data_test_X)

print(tree_model.score(data_train_X, data_train_y))
```

```
# convert to file submision.csv
```

```
1.0
0.7574387947269303
0.47570621468926555
1.0
```

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score

### Validação Cruzada

tree_model = DecisionTreeClassifier(random_state=1)
scores = cross_val_score(tree_model, data_train_X, data_train_y, cv=5) # 5-fold
print(scores)
```

```
[0.73116761 0.7212806  0.7113936  0.76082863 0.75329567]
```

Grid Search para melhores atributos

In [ ]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_absolute_error

# Grid_Search and Cross validation

# Define the parameter grid
param_grid = {
    'random_state': [1, 12, 1364],
    'criterion': ['gini', 'entropy'],
    'max_depth': [None],
    'min_samples_split': [2, 5, 10],
    'splitter': ['best'],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2'],
    'max_leaf_nodes': [900,910,890]
    }

# Create the grid search object
tree_model = DecisionTreeClassifier()
grid_search = GridSearchCV(estimator=tree_model, param_grid=param_grid, cv=5, sc

# Fit the model
grid_search.fit(X_train, y_train)

# Get the best parameters
print(grid_search.best_params_)
print(grid_search.best_score_)
print(grid_search.best_estimator_)
```

```
{'criterion': 'entropy', 'max_depth': None, 'max_features': 'sqrt', 'max_leaf_nod
es': 900, 'min_samples_leaf': 1, 'min_samples_split': 5, 'random_state': 1364, 's
plitter': 'best'}
0.7329566854990583
DecisionTreeClassifier(criterion='entropy', max_features='sqrt',
                       max_leaf_nodes=900, min_samples_split=5,
                       random_state=1364)
```

Implementação com Cross validation

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error

tree_model = DecisionTreeClassifier(random_state=1364,max_leaf_nodes= 890)
scores = cross_val_score(tree_model, X_train, y_train, cv=5) # 5-fold cross-vali
print(scores)
print(scores.mean())

print()
tree_model.fit(X_train, y_train)
tree_model.predict(X_valid)

print(tree_model.score(X_train, y_train))
print(tree_model.score(X_valid, y_valid))

val_mean_error = mean_absolute_error(tree_model.predict(X_valid), y_valid)
print(val_mean_error)
```

```
[0.76710609 0.74952919 0.74450722 0.76773384 0.75768989]
0.7573132454488387

0.9658505963590709
0.7548022598870057
0.4806026365348399
```

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_absolute_error

tree_model = DecisionTreeClassifier(random_state=1364, criterion='gini', max_dep
tree_model.fit(X_train, y_train)
tree_model.predict(X_valid)

print(tree_model.score(X_train, y_train))
print(tree_model.score(X_valid, y_valid))

val_mean_error = mean_absolute_error(tree_model.predict(X_valid), y_valid)
print(f'\n{val_mean_error}')
```

```
0.9256748273697426
0.7412429378531074

0.5220338983050847
```

Apurar melhor valor para número máximo de nós folha da arvore (max_leaf_nodes) pelo erro médio absoluto

In [ ]:
```
from sklearn.metrics import mean_absolute_error
from sklearn.tree import DecisionTreeClassifier

def get_mae(max_leaf_nodes, train_X, val_X, train_y, val_y):
    model = DecisionTreeClassifier(max_leaf_nodes=max_leaf_nodes, random_state=1
    model.fit(train_X, train_y)
    preds_val = model.predict(val_X)
    mae = mean_absolute_error(val_y, preds_val)
    return(mae)

def get_best_max_leaf_nodes(max_leaf_nodes, X_train, X_valid, y_train, y_valid):
```

```
  my_mae={max_leaf_nodes: get_mae(max_leaf_nodes, X_train, X_valid, y_train, y_v
  print(my_mae)

  return min(my_mae, key=my_mae.get)

max_leaf_nodes= [500,800,870,880,890,900,910,920,980,990,1000]
best_max_leaf_nodes = get_best_max_leaf_nodes(max_leaf_nodes, X_train, X_valid,
print(f"\nMelhor valor para max_leaf_nodes: {best_max_leaf_nodes}")
```

```
{500: 0.4768361581920904, 800: 0.47796610169491527, 870: 0.4806026365348399, 880:
0.4817325800376648, 890: 0.4806026365348399, 900: 0.47909604519774013, 910: 0.479
09604519774013, 920: 0.47909604519774013, 980: 0.4768361581920904, 990: 0.4813559
3220338985, 1000: 0.47947269303201506}

Melhor valor para max_leaf_nodes: 500
```

In [ ]:

```
from sklearn.metrics import mean_absolute_error

tree_model = DecisionTreeClassifier(random_state=1364,max_leaf_nodes= 890)
tree_model.fit(data_train_X, data_train_y)
y_pred = tree_model.predict(data_test_X)

print(tree_model.score(data_train_X, data_train_y))

val_mean_error = mean_absolute_error(tree_model.predict(X_valid), y_valid)
print(val_mean_error)
```

```
0.9475517890772128
0.10056497175141244
```

Com objectivo de gerar o modelo para submissão, temos abaixo, o código:

In [ ]:

```
tree_model.fit(data_train_X, data_train_y)
y_pred = tree_model.predict(data_test_X)

print("Exatidão do conjunto de treino")
print(f"{tree_model.score(data_train_X, data_train_y)}\n")

print("id", "floresta")
for i in range(10):
  print(data_test_id[i], y_pred[i])

submission = pd.DataFrame({
  "id" : data_test_id,
  "Floresta": y_pred
  })

submission.to_csv("submission.csv", index=False)
print("\nGerado Arquivo submission")

# O modelo
print()
print(tree_model)
print()
print()
end_time = datetime.now()
print('Tempo para correr esta experiência : {}'.format(end_time - start_time))
```

```
Exatidão do conjunto de treino
0.9475517890772128

id floresta
10621 2
10622 6
10623 6
10624 6
10625 6
10626 3
10627 3
10628 3
10629 3
10630 2

Gerado Arquivo submission

DecisionTreeClassifier(max_leaf_nodes=890, random_state=1364)


Tempo para correr esta experiência : 0:00:58.616632
```

ExtraTreeClassifier

Implementação do GridSearch para encontrar o melhor random_state no intervalo de 1 a 1000 com dois tipos de avaliação: accuracy e neg_mean_absolute_error.

In [ ]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import ExtraTreeClassifier

param_grid = {
    'random_state': [i for i in range(1,1000)]
}

extra_tree_model = ExtraTreeClassifier()
grid_search = GridSearchCV(estimator=extra_tree_model, param_grid=param_grid, cv

grid_search.fit(X_train, y_train)

print(grid_search.best_params_)
print(grid_search.best_score_)
print(grid_search.best_estimator_)
```

```
{'random_state': 126}
0.7196484620213435
ExtraTreeClassifier(random_state=126)
```

In [ ]:
```
from sklearn.tree import ExtraTreeClassifier
from sklearn.metrics import mean_absolute_error

extra_tree_model = ExtraTreeClassifier(random_state=126)
extra_tree_model.fit(X_train, y_train)
y_prediction = extra_tree_model.predict(X_valid)

print(extra_tree_model.score(X_train, y_train))
print(extra_tree_model.score(X_valid, y_valid))
```

```
print()
print(mean_absolute_error(y_prediction, y_valid))
```

```
1.0
0.7141242937853107

0.56045197740113
```

In [ ]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import ExtraTreeClassifier

param_grid = {
    'random_state': [i for i in range(1,1000)]
}

extra_tree_model = ExtraTreeClassifier()
grid_search = GridSearchCV(estimator=extra_tree_model, param_grid=param_grid, cv

grid_search.fit(X_train, y_train)

print(grid_search.best_params_)
print(grid_search.best_score_)
print(grid_search.best_estimator_)
```

```
{'random_state': 503}
-0.5605775266792217
ExtraTreeClassifier(random_state=503)
```

In [ ]:
```
from sklearn.tree import ExtraTreeClassifier
from sklearn.metrics import mean_absolute_error


extra_tree_model = ExtraTreeClassifier(random_state=503)
extra_tree_model.fit(X_train, y_train)
y_prediction = extra_tree_model.predict(X_valid)

print(extra_tree_model.score(X_train, y_train))
print(extra_tree_model.score(X_valid, y_valid))

val_mean_error = mean_absolute_error(y_prediction, y_valid)
print()
print(val_mean_error)
```

```
1.0
0.6945386064030132

0.5992467043314501
```

Implementação do GridSearch com outros parâmetros do algoritmo ExtraTreeClassifier com avaliação 'neg_mean_absolute_error'

In [ ]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import ExtraTreeClassifier

param_grid = {
    'random_state': [126],
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [None],
    'min_samples_split': [2],
    'min_samples_leaf': [1],
```

```
    'max_features': [None],
    'max_leaf_nodes': [314,1886]
    }


extra_tree_model = ExtraTreeClassifier()
grid_search = GridSearchCV(estimator=extra_tree_model, param_grid=param_grid, cv

grid_search.fit(X_train, y_train)

print(grid_search.best_params_)
print(grid_search.best_score_)
print(grid_search.best_estimator_)
grid_search.predict(X_valid)

print(grid_search.score(X_train, y_train))
print(grid_search.score(X_valid, y_valid))
```

```
{'criterion': 'gini', 'max_depth': None, 'max_features': None, 'max_leaf_nodes':
314, 'min_samples_leaf': 1, 'min_samples_split': 2, 'random_state': 126, 'splitte
r': 'best'}
-0.4718141870684243
ExtraTreeClassifier(max_features=None, max_leaf_nodes=314, random_state=126,
                    splitter='best')
-0.24846202134337728
-0.47570621468926555
```

In [ ]:

```
from sklearn.tree import ExtraTreeClassifier
extra_tree_model = ExtraTreeClassifier(max_features=None, max_leaf_nodes=314, ra
                    splitter='best')
extra_tree_model.fit(X_train, y_train)
extra_tree_model.predict(X_valid)

print(extra_tree_model.score(X_train, y_train))
print(extra_tree_model.score(X_valid, y_valid))
```

```
0.8725674827369743
0.7578154425612053
```

Apurar melhor valor para número máximo de nós folha da arvore (max_leaf_nodes) e profundidade máxima da arvore (max_depth) pelo menor erro médio absoluto

Número máximo de nós folha da arvore (max_leaf_nodes)

In [ ]:

```
from sklearn.metrics import mean_absolute_error
from sklearn.tree import ExtraTreeClassifier

def get_mae(max_leaf_nodes, train_X, val_X, train_y, val_y):
    model = ExtraTreeClassifier(max_leaf_nodes=max_leaf_nodes, random_state=126)
    model.fit(train_X, train_y)
    preds_val = model.predict(val_X)
    mae = mean_absolute_error(val_y, preds_val)
    return(mae)

def get_best(max_leaf_nodes, X_train, X_valid, y_train, y_valid):
  my_mae={max_leaf_nodes: get_mae(max_leaf_nodes, X_train, X_valid, y_train, y_v
  print(my_mae)
```

```
  return min(my_mae, key=my_mae.get)

max_leaf_nodes= [314,[i for i in range(990, 1990)]]
best = get_best(max_leaf_nodes, X_train, X_valid, y_train, y_valid)
print(f"\nMelhor valor para max_leaf_nodes: {best}")
```

{990: 0.6557438794726931, 991: 0.6557438794726931, 992: 0.655367231638418, 993: 0.6538606403013183, 994: 0.6538606403013183, 995: 0.6542372881355932, 996: 0.6538606403013183, 997: 0.6531073446327683, 998: 0.6527306967984934, 999: 0.6527306967984934, 1000: 0.6531073446327683, 1001: 0.6531073446327683, 1002: 0.6531073446327683, 1003: 0.6531073446327683, 1004: 0.6531073446327683, 1005: 0.6531073446327683, 1006: 0.6531073446327683, 1007: 0.655367231638418, 1008: 0.655367231638418, 1009: 0.656497175141243, 1010: 0.656497175141243, 1011: 0.655367231638418, 1012: 0.6557438794726931, 1013: 0.6557438794726931, 1014: 0.6557438794726931, 1015: 0.6557438794726931, 1016: 0.655367231638418, 1017: 0.655367231638418, 1018: 0.6549905838041431, 1019: 0.6549905838041431, 1020: 0.6549905838041431, 1021: 0.655367231638418, 1022: 0.6549905838041431, 1023: 0.6549905838041431, 1024: 0.6542372881355932, 1025: 0.6542372881355932, 1026: 0.6531073446327683, 1027: 0.6531073446327683, 1028: 0.6531073446327683, 1029: 0.6531073446327683, 1030: 0.6527306967984934, 1031: 0.6527306967984934, 1032: 0.6527306967984934, 1033: 0.6527306967984934, 1034: 0.6527306967984934, 1035: 0.6527306967984934, 1036: 0.6527306967984934, 1037: 0.6549905838041431, 1038: 0.6538606403013183, 1039: 0.6538606403013183, 1040: 0.6523540489642184, 1041: 0.6523540489642184, 1042: 0.6523540489642184, 1043: 0.6523540489642184, 1044: 0.6519774011299435, 1045: 0.6508474576271186, 1046: 0.6508474576271186, 1047: 0.6497175141242938, 1048: 0.6497175141242938, 1049: 0.6497175141242938, 1050: 0.6504708097928437, 1051: 0.6512241054613936, 1052: 0.6508474576271186, 1053: 0.6512241054613936, 1054: 0.6523540489642184, 1055: 0.6500941619585687, 1056: 0.6497175141242938, 1057: 0.6497175141242938, 1058: 0.6493408662900189, 1059: 0.6497175141242938, 1060: 0.6497175141242938, 1061: 0.6497175141242938, 1062: 0.6497175141242938, 1063: 0.6497175141242938, 1064: 0.6497175141242938, 1065: 0.6497175141242938, 1066: 0.6497175141242938, 1067: 0.6497175141242938, 1068: 0.6497175141242938, 1069: 0.6497175141242938, 1070: 0.6497175141242938, 1071: 0.6497175141242938, 1072: 0.6508474576271186, 1073: 0.6497175141242938, 1074: 0.6497175141242938, 1075: 0.6497175141242938, 1076: 0.6497175141242938, 1077: 0.6497175141242938, 1078: 0.6500941619585687, 1079: 0.6493408662900189, 1080: 0.6493408662900189, 1081: 0.6493408662900189, 1082: 0.6493408662900189, 1083: 0.6493408662900189, 1084: 0.6485875706214689, 1085: 0.6485875706214689, 1086: 0.6482109227871939, 1087: 0.6482109227871939, 1088: 0.6482109227871939, 1089: 0.6482109227871939, 1090: 0.6482109227871939, 1091: 0.6482109227871939, 1092: 0.6482109227871939, 1093: 0.6482109227871939, 1094: 0.6482109227871939, 1095: 0.6482109227871939, 1096: 0.6482109227871939, 1097: 0.6482109227871939, 1098: 0.6516007532956686, 1099: 0.6459510357815442, 1100: 0.6448210922787194, 1101: 0.6448210922787194, 1102: 0.6448210922787194, 1103: 0.6448210922787194, 1104: 0.64030131826742, 1105: 0.64030131826742, 1106: 0.6410546139359699, 1107: 0.6414312617702448, 1108: 0.6406779661016949, 1109: 0.6406779661016949, 1110: 0.6418079096045197, 1111: 0.6429378531073446, 1112: 0.6418079096045197, 1113: 0.6418079096045197, 1114: 0.6418079096045197, 1115: 0.6418079096045197, 1116: 0.6418079096045197, 1117: 0.6429378531073446, 1118: 0.6429378531073446, 1119: 0.6418079096045197, 1120: 0.6418079096045197, 1121: 0.6421845574387948, 1122: 0.6418079096045197, 1123: 0.6418079096045197, 1124: 0.6410546139359699, 1125: 0.6410546139359699, 1126: 0.6410546139359699, 1127: 0.6410546139359699, 1128: 0.6410546139359699, 1129: 0.6418079096045197, 1130: 0.6418079096045197, 1131: 0.6418079096045197, 1132: 0.6418079096045197, 1133: 0.6418079096045197, 1134: 0.6418079096045197, 1135: 0.6418079096045197, 1136: 0.6414312617702448, 1137: 0.6418079096045197, 1138: 0.6418079096045197, 1139: 0.6418079096045197, 1140: 0.6414312617702448, 1141: 0.6418079096045197, 1142: 0.6418079096045197, 1143: 0.6418079096045197, 1144: 0.6410546139359699, 1145: 0.64030131826742, 1146: 0.64030131826742, 1147: 0.639924670433145, 1148: 0.639924670433145, 1149: 0.6406779661016949, 1150: 0.6391713747645951, 1151: 0.6391713747645951, 1152: 0.6391713747645951, 1153: 0.6391713747645951, 1154: 0.6369114877589453, 1155: 0.639924670433145, 1156: 0.639924670433145, 1157: 0.639924670433145, 1158: 0.639924670433145, 1159: 0.639924670433145, 1160: 0.6380414312617703, 1161: 0.6380414312617703, 1162: 0.6380414312617703, 1163: 0.6380414312617703, 1164: 0.63954802259887, 1165: 0.639924670433145, 1166: 0.6406779661016949, 1167: 0.64030131826742, 1168: 0.63954802259887, 1169: 0.63954802259887, 1170: 0.63954802259887, 1171: 0.63954802259887, 1172: 0.63954802259887, 1173: 0.63954802259887, 1174: 0.6384180790960452, 1175: 0.6384180790960452, 1176: 0.6391713747645951, 1177: 0.6376647834274953, 1178: 0.63766478

34274953, 1179: 0.6376647834274953, 1180: 0.6376647834274953, 1181: 0.63314500941
61959, 1182: 0.6346516007532956, 1183: 0.6346516007532956, 1184: 0.63427495291902
07, 1185: 0.6342749529190207, 1186: 0.6342749529190207, 1187: 0.6342749529190207,
1188: 0.6342749529190207, 1189: 0.6342749529190207, 1190: 0.632768361581921, 119
1: 0.632015065913371, 1192: 0.632015065913371, 1193: 0.632015065913371, 1194: 0.6
32015065913371, 1195: 0.632015065913371, 1196: 0.632015065913371, 1197: 0.6312617
702448211, 1198: 0.6312617702448211, 1199: 0.6312617702448211, 1200: 0.6323917137
476459, 1201: 0.6350282485875707, 1202: 0.6350282485875707, 1203: 0.6323917137476
459, 1204: 0.6323917137476459, 1205: 0.6312617702448211, 1206: 0.631261770244821
1, 1207: 0.6312617702448211, 1208: 0.6312617702448211, 1209: 0.6312617702448211,
1210: 0.6308851224105462, 1211: 0.6308851224105462, 1212: 0.6286252354048965, 121
3: 0.6286252354048965, 1214: 0.6286252354048965, 1215: 0.6286252354048965, 1216:
0.6286252354048965, 1217: 0.6286252354048965, 1218: 0.6286252354048965, 1219: 0.6
286252354048965, 1220: 0.6286252354048965, 1221: 0.6286252354048965, 1222: 0.6244
82109227872, 1223: 0.6214689265536724, 1224: 0.6222222222222222, 1225: 0.62222222
22222222, 1226: 0.6222222222222222, 1227: 0.6222222222222222, 1228: 0.61996233521
65725, 1229: 0.6199623352165725, 1230: 0.6199623352165725, 1231: 0.61883239171374
76, 1232: 0.6188323917137476, 1233: 0.6188323917137476, 1234: 0.6177024482109228,
1235: 0.6177024482109228, 1236: 0.6184557438794727, 1237: 0.6192090395480226, 123
8: 0.6192090395480226, 1239: 0.6180790960451977, 1240: 0.6180790960451977, 1241:
0.6180790960451977, 1242: 0.6180790960451977, 1243: 0.6180790960451977, 1244: 0.6
180790960451977, 1245: 0.6173258003766479, 1246: 0.6173258003766479, 1247: 0.6180
790960451977, 1248: 0.6169491525423729, 1249: 0.6169491525423729, 1250: 0.6169491
525423729, 1251: 0.616195856873823, 1252: 0.6169491525423729, 1253: 0.61619585687
3823, 1254: 0.6225988700564972, 1255: 0.6180790960451977, 1256: 0.618079096045197
7, 1257: 0.6180790960451977, 1258: 0.6180790960451977, 1259: 0.6180790960451977,
1260: 0.6180790960451977, 1261: 0.6180790960451977, 1262: 0.6180790960451977, 126
3: 0.6180790960451977, 1264: 0.6180790960451977, 1265: 0.6180790960451977, 1266:
0.6180790960451977, 1267: 0.6165725047080979, 1268: 0.6165725047080979, 1269: 0.6
165725047080979, 1270: 0.6165725047080979, 1271: 0.6165725047080979, 1272: 0.6165
725047080979, 1273: 0.6154425612052731, 1274: 0.6154425612052731, 1275: 0.6154425
612052731, 1276: 0.6154425612052731, 1277: 0.6154425612052731, 1278: 0.6131826741
996234, 1279: 0.6131826741996234, 1280: 0.6128060263653484, 1281: 0.6128060263653
484, 1282: 0.6128060263653484, 1283: 0.6128060263653484, 1284: 0.612806026365348
4, 1285: 0.6128060263653484, 1286: 0.6128060263653484, 1287: 0.6128060263653484,
1288: 0.6128060263653484, 1289: 0.6128060263653484, 1290: 0.6116760828625235, 129
1: 0.6116760828625235, 1292: 0.6116760828625235, 1293: 0.6116760828625235, 1294:
0.6116760828625235, 1295: 0.6116760828625235, 1296: 0.6112994350282486, 1297: 0.6
105461393596987, 1298: 0.6105461393596987, 1299: 0.6109227871939736, 1300: 0.6109
227871939736, 1301: 0.6112994350282486, 1302: 0.6112994350282486, 1303: 0.6112994
350282486, 1304: 0.6112994350282486, 1305: 0.6112994350282486, 1306: 0.6112994350
282486, 1307: 0.6109227871939736, 1308: 0.6109227871939736, 1309: 0.6109227871939
736, 1310: 0.6109227871939736, 1311: 0.6109227871939736, 1312: 0.610922787193973
6, 1313: 0.6109227871939736, 1314: 0.6109227871939736, 1315: 0.6097928436911487,
1316: 0.6097928436911487, 1317: 0.6097928436911487, 1318: 0.6101694915254238, 131
9: 0.6101694915254238, 1320: 0.6101694915254238, 1321: 0.6101694915254238, 1322:
0.6090395480225989, 1323: 0.6090395480225989, 1324: 0.6090395480225989, 1325: 0.6
094161958568738, 1326: 0.6090395480225989, 1327: 0.6090395480225989, 1328: 0.6090
395480225989, 1329: 0.6090395480225989, 1330: 0.6090395480225989, 1331: 0.6090395
480225989, 1332: 0.6094161958568738, 1333: 0.6090395480225989, 1334: 0.6090395480
225989, 1335: 0.6090395480225989, 1336: 0.6090395480225989, 1337: 0.6090395480225
989, 1338: 0.6090395480225989, 1339: 0.6090395480225989, 1340: 0.609039548022598
9, 1341: 0.6090395480225989, 1342: 0.6090395480225989, 1343: 0.6090395480225989,
1344: 0.6086629001883239, 1345: 0.6086629001883239, 1346: 0.6086629001883239, 134
7: 0.608286252354049, 1348: 0.6090395480225989, 1349: 0.608286252354049, 1350: 0.
6086629001883239, 1351: 0.607532956685499, 1352: 0.608286252354049, 1353: 0.60828
6252354049, 1354: 0.608286252354049, 1355: 0.6105461393596987, 1356: 0.6105461393
596987, 1357: 0.6105461393596987, 1358: 0.6105461393596987, 1359: 0.6105461393596
987, 1360: 0.6105461393596987, 1361: 0.6105461393596987, 1362: 0.610546139359698
7, 1363: 0.6105461393596987, 1364: 0.6094161958568738, 1365: 0.6086629001883239,

1366: 0.6086629001883239, 1367: 0.6086629001883239, 1368: 0.6086629001883239, 136
9: 0.6094161958568738, 1370: 0.6109227871939736, 1371: 0.6109227871939736, 1372:
0.616195856873823, 1373: 0.6116760828625235, 1374: 0.6116760828625235, 1375: 0.60
71563088512241, 1376: 0.6071563088512241, 1377: 0.6071563088512241, 1378: 0.60941
61958568738, 1379: 0.6094161958568738, 1380: 0.6094161958568738, 1381: 0.60941619
58568738, 1382: 0.6094161958568738, 1383: 0.6094161958568738, 1384: 0.60941619585
68738, 1385: 0.6094161958568738, 1386: 0.6094161958568738, 1387: 0.60941619585687
38, 1388: 0.6097928436911487, 1389: 0.6097928436911487, 1390: 0.6097928436911487,
1391: 0.6097928436911487, 1392: 0.6097928436911487, 1393: 0.6097928436911487, 139
4: 0.6097928436911487, 1395: 0.6097928436911487, 1396: 0.6097928436911487, 1397:
0.6090395480225989, 1398: 0.6086629001883239, 1399: 0.607532956685499, 1400: 0.60
7532956685499, 1401: 0.6060263653483993, 1402: 0.6060263653483993, 1403: 0.606026
3653483993, 1404: 0.6060263653483993, 1405: 0.6052730696798494, 1406: 0.605273069
6798494, 1407: 0.6056497175141243, 1408: 0.6056497175141243, 1409: 0.605649717514
1243, 1410: 0.6056497175141243, 1411: 0.6056497175141243, 1412: 0.605649717514124
3, 1413: 0.6056497175141243, 1414: 0.6056497175141243, 1415: 0.6056497175141243,
1416: 0.6056497175141243, 1417: 0.6056497175141243, 1418: 0.6056497175141243, 141
9: 0.6056497175141243, 1420: 0.6056497175141243, 1421: 0.6056497175141243, 1422:
0.6056497175141243, 1423: 0.6033898305084746, 1424: 0.6033898305084746, 1425: 0.6
030131826741997, 1426: 0.6030131826741997, 1427: 0.6030131826741997, 1428: 0.6030
131826741997, 1429: 0.6030131826741997, 1430: 0.6030131826741997, 1431: 0.6030131
826741997, 1432: 0.6030131826741997, 1433: 0.6033898305084746, 1434: 0.6037664783
427495, 1435: 0.6033898305084746, 1436: 0.6033898305084746, 1437: 0.6033898305084
746, 1438: 0.6033898305084746, 1439: 0.6033898305084746, 1440: 0.603389830508474
6, 1441: 0.6033898305084746, 1442: 0.6033898305084746, 1443: 0.6033898305084746,
1444: 0.6037664783427495, 1445: 0.6037664783427495, 1446: 0.6033898305084746, 144
7: 0.6033898305084746, 1448: 0.6033898305084746, 1449: 0.6037664783427495, 1450:
0.6037664783427495, 1451: 0.6037664783427495, 1452: 0.6037664783427495, 1453: 0.6
041431261770245, 1454: 0.6037664783427495, 1455: 0.6037664783427495, 1456: 0.6037
664783427495, 1457: 0.6037664783427495, 1458: 0.6037664783427495, 1459: 0.6033898
305084746, 1460: 0.6033898305084746, 1461: 0.6033898305084746, 1462: 0.6033898305
084746, 1463: 0.6033898305084746, 1464: 0.6030131826741997, 1465: 0.6030131826741
997, 1466: 0.6030131826741997, 1467: 0.6030131826741997, 1468: 0.603013182674199
7, 1469: 0.6026365348399246, 1470: 0.6030131826741997, 1471: 0.6018832391713748,
1472: 0.6018832391713748, 1473: 0.6018832391713748, 1474: 0.6018832391713748, 147
5: 0.6018832391713748, 1476: 0.6018832391713748, 1477: 0.5984934086629002, 1478:
0.599623352165725, 1479: 0.5984934086629002, 1480: 0.5984934086629002, 1481: 0.59
84934086629002, 1482: 0.5984934086629002, 1483: 0.5984934086629002, 1484: 0.59849
34086629002, 1485: 0.5984934086629002, 1486: 0.5984934086629002, 1487: 0.59811676
08286252, 1488: 0.5958568738229755, 1489: 0.5958568738229755, 1490: 0.59585687382
29755, 1491: 0.5947269303201507, 1492: 0.5943502824858757, 1493: 0.59435028248587
57, 1494: 0.5951035781544256, 1495: 0.5951035781544256, 1496: 0.5951035781544256,
1497: 0.5954802259887005, 1498: 0.5954802259887005, 1499: 0.5954802259887005, 150
0: 0.5951035781544256, 1501: 0.5954802259887005, 1502: 0.5954802259887005, 1503:
0.5951035781544256, 1504: 0.5951035781544256, 1505: 0.5943502824858757, 1506: 0.5
943502824858757, 1507: 0.5943502824858757, 1508: 0.5958568738229755, 1509: 0.5958
568738229755, 1510: 0.5962335216572505, 1511: 0.5962335216572505, 1512: 0.5973634
651600753, 1513: 0.5973634651600753, 1514: 0.5973634651600753, 1515: 0.5996233521
65725, 1516: 0.5973634651600753, 1517: 0.5962335216572505, 1518: 0.59623352165725
05, 1519: 0.5962335216572505, 1520: 0.5951035781544256, 1521: 0.5928436911487759,
1522: 0.5939736346516008, 1523: 0.5939736346516008, 1524: 0.5939736346516008, 152
5: 0.5939736346516008, 1526: 0.5951035781544256, 1527: 0.5951035781544256, 1528:
0.5951035781544256, 1529: 0.5962335216572505, 1530: 0.5962335216572505, 1531: 0.5
962335216572505, 1532: 0.5984934086629002, 1533: 0.5984934086629002, 1534: 0.5939
736346516008, 1535: 0.5939736346516008, 1536: 0.5939736346516008, 1537: 0.5939736
346516008, 1538: 0.5939736346516008, 1539: 0.5939736346516008, 1540: 0.5924670433
14501, 1541: 0.5890772128060263, 1542: 0.5894538606403014, 1543: 0.58870056497175
14, 1544: 0.5887005649717514, 1545: 0.592090395480226, 1546: 0.5909604519774011,
1547: 0.5909604519774011, 1548: 0.5905838041431262, 1549: 0.5909604519774011, 155
0: 0.5902071563088512, 1551: 0.5902071563088512, 1552: 0.5902071563088512, 1553:

0.5902071563088512, 1554: 0.5902071563088512, 1555: 0.5902071563088512, 1556: 0.5
902071563088512, 1557: 0.5902071563088512, 1558: 0.5902071563088512, 1559: 0.5913
37099811676, 1560: 0.591337099811676, 1561: 0.591337099811676, 1562: 0.5913370998
11676, 1563: 0.5890772128060263, 1564: 0.591337099811676, 1565: 0.59133709981167
6, 1566: 0.591337099811676, 1567: 0.591337099811676, 1568: 0.591337099811676, 156
9: 0.591337099811676, 1570: 0.591337099811676, 1571: 0.591337099811676, 1572: 0.5
91337099811676, 1573: 0.591337099811676, 1574: 0.591337099811676, 1575: 0.5947269
303201507, 1576: 0.5935969868173258, 1577: 0.5935969868173258, 1578: 0.5935969868
173258, 1579: 0.5935969868173258, 1580: 0.5935969868173258, 1581: 0.5935969868173
258, 1582: 0.5935969868173258, 1583: 0.5935969868173258, 1584: 0.593220338983050
8, 1585: 0.5932203389830508, 1586: 0.5932203389830508, 1587: 0.5932203389830508,
1588: 0.5932203389830508, 1589: 0.5935969868173258, 1590: 0.5932203389830508, 159
1: 0.5947269303201507, 1592: 0.5943502824858757, 1593: 0.5943502824858757, 1594:
0.5943502824858757, 1595: 0.5943502824858757, 1596: 0.5947269303201507, 1597: 0.5
947269303201507, 1598: 0.5951035781544256, 1599: 0.5939736346516008, 1600: 0.5939
736346516008, 1601: 0.5939736346516008, 1602: 0.5939736346516008, 1603: 0.5939736
346516008, 1604: 0.5939736346516008, 1605: 0.5947269303201507, 1606: 0.5958568738
229755, 1607: 0.5962335216572505, 1608: 0.5962335216572505, 1609: 0.5966101694915
255, 1610: 0.5966101694915255, 1611: 0.5943502824858757, 1612: 0.594350282485875
7, 1613: 0.5943502824858757, 1614: 0.5943502824858757, 1615: 0.5943502824858757,
1616: 0.5943502824858757, 1617: 0.592090395480226, 1618: 0.592090395480226, 1619:
0.592090395480226, 1620: 0.5898305084745763, 1621: 0.5898305084745763, 1622: 0.59
09604519774011, 1623: 0.5909604519774011, 1624: 0.5909604519774011, 1625: 0.59133
7099811676, 1626: 0.591337099811676, 1627: 0.591337099811676, 1628: 0.59133709981
1676, 1629: 0.5902071563088512, 1630: 0.5902071563088512, 1631: 0.590207156308851
2, 1632: 0.5902071563088512, 1633: 0.5902071563088512, 1634: 0.5898305084745763,
1635: 0.5905838041431262, 1636: 0.5898305084745763, 1637: 0.591337099811676, 163
8: 0.591337099811676, 1639: 0.591337099811676, 1640: 0.591337099811676, 1641: 0.5
91337099811676, 1642: 0.591337099811676, 1643: 0.5917137476459511, 1644: 0.591713
7476459511, 1645: 0.5917137476459511, 1646: 0.5917137476459511, 1647: 0.591713747
6459511, 1648: 0.5917137476459511, 1649: 0.5917137476459511, 1650: 0.591713747645
9511, 1651: 0.5917137476459511, 1652: 0.592090395480226, 1653: 0.592090395480226,
1654: 0.592467043314501, 1655: 0.592467043314501, 1656: 0.5917137476459511, 1657:
0.592467043314501, 1658: 0.5928436911487759, 1659: 0.5928436911487759, 1660: 0.59
2090395480226, 1661: 0.592090395480226, 1662: 0.592090395480226, 1663: 0.59209039
5480226, 1664: 0.592090395480226, 1665: 0.592090395480226, 1666: 0.59322033898305
08, 1667: 0.5932203389830508, 1668: 0.5932203389830508, 1669: 0.5932203389830508,
1670: 0.5932203389830508, 1671: 0.5932203389830508, 1672: 0.5943502824858757, 167
3: 0.5935969868173258, 1674: 0.5935969868173258, 1675: 0.5935969868173258, 1676:
0.5943502824858757, 1677: 0.5939736346516008, 1678: 0.5943502824858757, 1679: 0.5
939736346516008, 1680: 0.5939736346516008, 1681: 0.5935969868173258, 1682: 0.5935
969868173258, 1683: 0.592090395480226, 1684: 0.592090395480226, 1685: 0.592090395
480226, 1686: 0.592090395480226, 1687: 0.592090395480226, 1688: 0.59209039548022
6, 1689: 0.5932203389830508, 1690: 0.592467043314501, 1691: 0.592467043314501, 16
92: 0.592467043314501, 1693: 0.592467043314501, 1694: 0.592467043314501, 1695: 0.
592467043314501, 1696: 0.5902071563088512, 1697: 0.5902071563088512, 1698: 0.5902
071563088512, 1699: 0.5905838041431262, 1700: 0.5905838041431262, 1701: 0.5909604
519774011, 1702: 0.5909604519774011, 1703: 0.5909604519774011, 1704: 0.5909604519
774011, 1705: 0.5909604519774011, 1706: 0.5909604519774011, 1707: 0.5909604519774
011, 1708: 0.5917137476459511, 1709: 0.5917137476459511, 1710: 0.591713747645951
1, 1711: 0.5917137476459511, 1712: 0.5917137476459511, 1713: 0.591337099811676, 1
714: 0.591337099811676, 1715: 0.591337099811676, 1716: 0.591337099811676, 1717:
0.591337099811676, 1718: 0.591337099811676, 1719: 0.591337099811676, 1720: 0.5913
37099811676, 1721: 0.591337099811676, 1722: 0.591337099811676, 1723: 0.5913370998
11676, 1724: 0.591337099811676, 1725: 0.5935969868173258, 1726: 0.593596986817325
8, 1727: 0.5935969868173258, 1728: 0.5932203389830508, 1729: 0.5935969868173258,
1730: 0.5935969868173258, 1731: 0.5935969868173258, 1732: 0.5935969868173258, 173
3: 0.5935969868173258, 1734: 0.5935969868173258, 1735: 0.5935969868173258, 1736:
0.5935969868173258, 1737: 0.5935969868173258, 1738: 0.5935969868173258, 1739: 0.5
935969868173258, 1740: 0.5935969868173258, 1741: 0.5935969868173258, 1742: 0.5935

969868173258, 1743: 0.5935969868173258, 1744: 0.5935969868173258, 1745: 0.5935969
868173258, 1746: 0.5947269303201507, 1747: 0.5947269303201507, 1748: 0.5947269303
201507, 1749: 0.5935969868173258, 1750: 0.5935969868173258, 1751: 0.5935969868173
258, 1752: 0.5939736346516008, 1753: 0.5939736346516008, 1754: 0.593973634651600
8, 1755: 0.5939736346516008, 1756: 0.5939736346516008, 1757: 0.5939736346516008,
1758: 0.5939736346516008, 1759: 0.5939736346516008, 1760: 0.5917137476459511, 176
1: 0.5928436911487759, 1762: 0.5917137476459511, 1763: 0.5932203389830508, 1764:
0.5932203389830508, 1765: 0.5932203389830508, 1766: 0.5932203389830508, 1767: 0.5
932203389830508, 1768: 0.5932203389830508, 1769: 0.5932203389830508, 1770: 0.5932
203389830508, 1771: 0.5943502824858757, 1772: 0.5943502824858757, 1773: 0.5943502
824858757, 1774: 0.5932203389830508, 1775: 0.5932203389830508, 1776: 0.5932203389
830508, 1777: 0.5935969868173258, 1778: 0.5935969868173258, 1779: 0.5943502824858
757, 1780: 0.5939736346516008, 1781: 0.5939736346516008, 1782: 0.593973634651600
8, 1783: 0.5939736346516008, 1784: 0.5939736346516008, 1785: 0.5935969868173258,
1786: 0.5935969868173258, 1787: 0.5935969868173258, 1788: 0.5935969868173258, 178
9: 0.5935969868173258, 1790: 0.5935969868173258, 1791: 0.5947269303201507, 1792:
0.5935969868173258, 1793: 0.5935969868173258, 1794: 0.5935969868173258, 1795: 0.5
935969868173258, 1796: 0.5935969868173258, 1797: 0.5935969868173258, 1798: 0.5935
969868173258, 1799: 0.5935969868173258, 1800: 0.5935969868173258, 1801: 0.5932203
389830508, 1802: 0.5932203389830508, 1803: 0.5932203389830508, 1804: 0.5932203389
830508, 1805: 0.5932203389830508, 1806: 0.5932203389830508, 1807: 0.5932203389830
508, 1808: 0.5932203389830508, 1809: 0.5932203389830508, 1810: 0.593220338983050
8, 1811: 0.5932203389830508, 1812: 0.5932203389830508, 1813: 0.5932203389830508,
1814: 0.5932203389830508, 1815: 0.5932203389830508, 1816: 0.5932203389830508, 181
7: 0.5932203389830508, 1818: 0.592090395480226, 1819: 0.5932203389830508, 1820:
0.5932203389830508, 1821: 0.5932203389830508, 1822: 0.5935969868173258, 1823: 0.5
935969868173258, 1824: 0.5935969868173258, 1825: 0.5932203389830508, 1826: 0.5932
203389830508, 1827: 0.5932203389830508, 1828: 0.5932203389830508, 1829: 0.5928436
911487759, 1830: 0.5928436911487759, 1831: 0.5928436911487759, 1832: 0.5920903954
80226, 1833: 0.592467043314501, 1834: 0.592467043314501, 1835: 0.592467043314501,
1836: 0.592467043314501, 1837: 0.592467043314501, 1838: 0.592467043314501, 1839:
0.592467043314501, 1840: 0.592467043314501, 1841: 0.5928436911487759, 1842: 0.592
8436911487759, 1843: 0.5928436911487759, 1844: 0.5928436911487759, 1845: 0.590583
8041431262, 1846: 0.5905838041431262, 1847: 0.5905838041431262, 1848: 0.590583804
1431262, 1849: 0.5905838041431262, 1850: 0.5902071563088512, 1851: 0.590207156308
8512, 1852: 0.5902071563088512, 1853: 0.5902071563088512, 1854: 0.590207156308851
2, 1855: 0.5902071563088512, 1856: 0.5902071563088512, 1857: 0.5902071563088512,
1858: 0.5902071563088512, 1859: 0.5902071563088512, 1860: 0.5902071563088512, 186
1: 0.5909604519774011, 1862: 0.5909604519774011, 1863: 0.5909604519774011, 1864:
0.5909604519774011, 1865: 0.5909604519774011, 1866: 0.5917137476459511, 1867: 0.5
917137476459511, 1868: 0.5905838041431262, 1869: 0.5905838041431262, 1870: 0.5905
838041431262, 1871: 0.5905838041431262, 1872: 0.5905838041431262, 1873: 0.5905838
041431262, 1874: 0.5905838041431262, 1875: 0.5905838041431262, 1876: 0.5905838041
431262, 1877: 0.5905838041431262, 1878: 0.5905838041431262, 1879: 0.5905838041431
262, 1880: 0.5905838041431262, 1881: 0.5905838041431262, 1882: 0.590583804143126
2, 1883: 0.5905838041431262, 1884: 0.5905838041431262, 1885: 0.5905838041431262,
1886: 0.5883239171374764, 1887: 0.5890772128060263, 1888: 0.5890772128060263, 188
9: 0.5890772128060263, 1890: 0.592467043314501, 1891: 0.591337099811676, 1892: 0.
591337099811676, 1893: 0.591337099811676, 1894: 0.591337099811676, 1895: 0.591337
099811676, 1896: 0.591337099811676, 1897: 0.591337099811676, 1898: 0.591337099811
676, 1899: 0.591337099811676, 1900: 0.591337099811676, 1901: 0.591337099811676, 1
902: 0.591337099811676, 1903: 0.591337099811676, 1904: 0.591337099811676, 1905:
0.591337099811676, 1906: 0.591337099811676, 1907: 0.5917137476459511, 1908: 0.591
7137476459511, 1909: 0.5909604519774011, 1910: 0.5909604519774011, 1911: 0.590960
4519774011, 1912: 0.5902071563088512, 1913: 0.5905838041431262, 1914: 0.590583804
1431262, 1915: 0.5905838041431262, 1916: 0.5905838041431262, 1917: 0.590583804143
1262, 1918: 0.5909604519774011, 1919: 0.5905838041431262, 1920: 0.590583804143126
2, 1921: 0.5905838041431262, 1922: 0.5905838041431262, 1923: 0.5905838041431262,
1924: 0.5905838041431262, 1925: 0.5905838041431262, 1926: 0.5905838041431262, 192
7: 0.5909604519774011, 1928: 0.5909604519774011, 1929: 0.5909604519774011, 1930:

```
0.591337099811676, 1931: 0.591337099811676, 1932: 0.591337099811676, 1933: 0.5902
071563088512, 1934: 0.5902071563088512, 1935: 0.5917137476459511, 1936: 0.5917137
476459511, 1937: 0.5917137476459511, 1938: 0.5917137476459511, 1939: 0.5924670433
14501, 1940: 0.5917137476459511, 1941: 0.5917137476459511, 1942: 0.59322033898305
08, 1943: 0.5943502824858757, 1944: 0.5943502824858757, 1945: 0.5943502824858757,
1946: 0.5943502824858757, 1947: 0.5902071563088512, 1948: 0.5932203389830508, 194
9: 0.5932203389830508, 1950: 0.5932203389830508, 1951: 0.5932203389830508, 1952:
0.5932203389830508, 1953: 0.5932203389830508, 1954: 0.5932203389830508, 1955: 0.5
932203389830508, 1956: 0.5932203389830508, 1957: 0.5932203389830508, 1958: 0.5932
203389830508, 1959: 0.5932203389830508, 1960: 0.5932203389830508, 1961: 0.5932203
389830508, 1962: 0.5932203389830508, 1963: 0.5932203389830508, 1964: 0.5932203389
830508, 1965: 0.592467043314501, 1966: 0.5932203389830508, 1967: 0.59322033898305
08, 1968: 0.5917137476459511, 1969: 0.5917137476459511, 1970: 0.5917137476459511,
1971: 0.5917137476459511, 1972: 0.5917137476459511, 1973: 0.5917137476459511, 197
4: 0.592090395480226, 1975: 0.592090395480226, 1976: 0.592090395480226, 1977: 0.5
92467043314501, 1978: 0.592467043314501, 1979: 0.592467043314501, 1980: 0.5924670
43314501, 1981: 0.5932203389830508, 1982: 0.5932203389830508, 1983: 0.59359698681
73258, 1984: 0.5935969868173258, 1985: 0.5935969868173258, 1986: 0.59284369114877
59, 1987: 0.5928436911487759, 1988: 0.5939736346516008, 1989: 0.5939736346516008}

Melhor valor para max_leaf_nodes: 1886
```

Profundidade máxima da arvore (max_depth)

In [ ]:

```python
from sklearn.metrics import mean_absolute_error
from sklearn.tree import ExtraTreeClassifier

def get_mae(max_depth, train_X, val_X, train_y, val_y):
    model = ExtraTreeClassifier(max_depth=max_depth, random_state=126)
    model.fit(train_X, train_y)
    preds_val = model.predict(val_X)
    mae = mean_absolute_error(val_y, preds_val)
    return(mae)

def get_best(max_depth, X_train, X_valid, y_train, y_valid):
  my_mae={max_depth: get_mae(max_depth, X_train, X_valid, y_train, y_valid) for
  print(my_mae)

  return min(my_mae, key=my_mae.get)

max_depth= [i for i in range(1,500)]
best = get_best(max_depth, X_train, X_valid, y_train, y_valid)
print(f"\nMelhor valor para max_depth: {best}")
```

{1: 2.047080979284369, 2: 1.656497175141243, 3: 1.1781544256120526, 4: 1.34350282
4858757, 5: 1.551412429378531, 6: 1.3148775894538607, 7: 1.3258003766478343, 8:
1.2056497175141243, 9: 0.8753295668549906, 10: 0.8595103578154426, 11: 0.78455743
8794727, 12: 0.6689265536723163, 13: 0.7653483992467043, 14: 0.6674199623352166,
15: 0.7548022598870057, 16: 0.6836158192090396, 17: 0.6617702448210923, 18: 0.584
5574387947269, 19: 0.6120527306967984, 20: 0.6135593220338983, 21: 0.574764595103
5781, 22: 0.5495291902071563, 23: 0.5713747645951036, 24: 0.5713747645951036, 25:
0.5480225988700564, 26: 0.5389830508474577, 27: 0.5514124293785311, 28: 0.5887005
649717514, 29: 0.5728813559322034, 30: 0.5698681732580038, 31: 0.600376647834274
9, 32: 0.6003766478342749, 33: 0.6003766478342749, 34: 0.6003766478342749, 35: 0.
6003766478342749, 36: 0.6003766478342749, 37: 0.6003766478342749, 38: 0.600376647
8342749, 39: 0.6003766478342749, 40: 0.6003766478342749, 41: 0.6003766478342749,
42: 0.6003766478342749, 43: 0.6003766478342749, 44: 0.6003766478342749, 45: 0.600
3766478342749, 46: 0.6003766478342749, 47: 0.6003766478342749, 48: 0.600376647834
2749, 49: 0.6003766478342749, 50: 0.6003766478342749, 51: 0.6003766478342749, 52:
0.6003766478342749, 53: 0.6003766478342749, 54: 0.6003766478342749, 55: 0.6003766
478342749, 56: 0.6003766478342749, 57: 0.6003766478342749, 58: 0.600376647834274
9, 59: 0.6003766478342749, 60: 0.6003766478342749, 61: 0.6003766478342749, 62: 0.
6003766478342749, 63: 0.6003766478342749, 64: 0.6003766478342749, 65: 0.600376647
8342749, 66: 0.6003766478342749, 67: 0.6003766478342749, 68: 0.6003766478342749,
69: 0.6003766478342749, 70: 0.6003766478342749, 71: 0.6003766478342749, 72: 0.600
3766478342749, 73: 0.6003766478342749, 74: 0.6003766478342749, 75: 0.600376647834
2749, 76: 0.6003766478342749, 77: 0.6003766478342749, 78: 0.6003766478342749, 79:
0.6003766478342749, 80: 0.6003766478342749, 81: 0.6003766478342749, 82: 0.6003766
478342749, 83: 0.6003766478342749, 84: 0.6003766478342749, 85: 0.600376647834274
9, 86: 0.6003766478342749, 87: 0.6003766478342749, 88: 0.6003766478342749, 89: 0.
6003766478342749, 90: 0.6003766478342749, 91: 0.6003766478342749, 92: 0.600376647
8342749, 93: 0.6003766478342749, 94: 0.6003766478342749, 95: 0.6003766478342749,
96: 0.6003766478342749, 97: 0.6003766478342749, 98: 0.6003766478342749, 99: 0.600
3766478342749, 100: 0.6003766478342749, 101: 0.6003766478342749, 102: 0.600376647
8342749, 103: 0.6003766478342749, 104: 0.6003766478342749, 105: 0.600376647834274
9, 106: 0.6003766478342749, 107: 0.6003766478342749, 108: 0.6003766478342749, 10
9: 0.6003766478342749, 110: 0.6003766478342749, 111: 0.6003766478342749, 112: 0.6
003766478342749, 113: 0.6003766478342749, 114: 0.6003766478342749, 115: 0.6003766
478342749, 116: 0.6003766478342749, 117: 0.6003766478342749, 118: 0.6003766478342
749, 119: 0.6003766478342749, 120: 0.6003766478342749, 121: 0.6003766478342749, 1
22: 0.6003766478342749, 123: 0.6003766478342749, 124: 0.6003766478342749, 125: 0.
6003766478342749, 126: 0.6003766478342749, 127: 0.6003766478342749, 128: 0.600376
6478342749, 129: 0.6003766478342749, 130: 0.6003766478342749, 131: 0.600376647834
2749, 132: 0.6003766478342749, 133: 0.6003766478342749, 134: 0.6003766478342749,
135: 0.6003766478342749, 136: 0.6003766478342749, 137: 0.6003766478342749, 138:
0.6003766478342749, 139: 0.6003766478342749, 140: 0.6003766478342749, 141: 0.6003
766478342749, 142: 0.6003766478342749, 143: 0.6003766478342749, 144: 0.6003766478
342749, 145: 0.6003766478342749, 146: 0.6003766478342749, 147: 0.600376647834274
9, 148: 0.6003766478342749, 149: 0.6003766478342749, 150: 0.6003766478342749, 15
1: 0.6003766478342749, 152: 0.6003766478342749, 153: 0.6003766478342749, 154: 0.6
003766478342749, 155: 0.6003766478342749, 156: 0.6003766478342749, 157: 0.6003766
478342749, 158: 0.6003766478342749, 159: 0.6003766478342749, 160: 0.6003766478342
749, 161: 0.6003766478342749, 162: 0.6003766478342749, 163: 0.6003766478342749, 1
64: 0.6003766478342749, 165: 0.6003766478342749, 166: 0.6003766478342749, 167: 0.
6003766478342749, 168: 0.6003766478342749, 169: 0.6003766478342749, 170: 0.600376
6478342749, 171: 0.6003766478342749, 172: 0.6003766478342749, 173: 0.600376647834
2749, 174: 0.6003766478342749, 175: 0.6003766478342749, 176: 0.6003766478342749,
177: 0.6003766478342749, 178: 0.6003766478342749, 179: 0.6003766478342749, 180:
0.6003766478342749, 181: 0.6003766478342749, 182: 0.6003766478342749, 183: 0.6003
766478342749, 184: 0.6003766478342749, 185: 0.6003766478342749, 186: 0.6003766478
342749, 187: 0.6003766478342749, 188: 0.6003766478342749, 189: 0.600376647834274
9, 190: 0.6003766478342749, 191: 0.6003766478342749, 192: 0.6003766478342749, 19
3: 0.6003766478342749, 194: 0.6003766478342749, 195: 0.6003766478342749, 196: 0.6
003766478342749, 197: 0.6003766478342749, 198: 0.6003766478342749, 199: 0.6003766

478342749, 200: 0.6003766478342749, 201: 0.6003766478342749, 202: 0.6003766478342
749, 203: 0.6003766478342749, 204: 0.6003766478342749, 205: 0.6003766478342749, 2
06: 0.6003766478342749, 207: 0.6003766478342749, 208: 0.6003766478342749, 209: 0.
6003766478342749, 210: 0.6003766478342749, 211: 0.6003766478342749, 212: 0.600376
6478342749, 213: 0.6003766478342749, 214: 0.6003766478342749, 215: 0.600376647834
2749, 216: 0.6003766478342749, 217: 0.6003766478342749, 218: 0.6003766478342749,
219: 0.6003766478342749, 220: 0.6003766478342749, 221: 0.6003766478342749, 222:
0.6003766478342749, 223: 0.6003766478342749, 224: 0.6003766478342749, 225: 0.6003
766478342749, 226: 0.6003766478342749, 227: 0.6003766478342749, 228: 0.6003766478
342749, 229: 0.6003766478342749, 230: 0.6003766478342749, 231: 0.600376647834274
9, 232: 0.6003766478342749, 233: 0.6003766478342749, 234: 0.6003766478342749, 23
5: 0.6003766478342749, 236: 0.6003766478342749, 237: 0.6003766478342749, 238: 0.6
003766478342749, 239: 0.6003766478342749, 240: 0.6003766478342749, 241: 0.6003766
478342749, 242: 0.6003766478342749, 243: 0.6003766478342749, 244: 0.6003766478342
749, 245: 0.6003766478342749, 246: 0.6003766478342749, 247: 0.6003766478342749, 2
48: 0.6003766478342749, 249: 0.6003766478342749, 250: 0.6003766478342749, 251: 0.
6003766478342749, 252: 0.6003766478342749, 253: 0.6003766478342749, 254: 0.600376
6478342749, 255: 0.6003766478342749, 256: 0.6003766478342749, 257: 0.600376647834
2749, 258: 0.6003766478342749, 259: 0.6003766478342749, 260: 0.6003766478342749,
261: 0.6003766478342749, 262: 0.6003766478342749, 263: 0.6003766478342749, 264:
0.6003766478342749, 265: 0.6003766478342749, 266: 0.6003766478342749, 267: 0.6003
766478342749, 268: 0.6003766478342749, 269: 0.6003766478342749, 270: 0.6003766478
342749, 271: 0.6003766478342749, 272: 0.6003766478342749, 273: 0.600376647834274
9, 274: 0.6003766478342749, 275: 0.6003766478342749, 276: 0.6003766478342749, 27
7: 0.6003766478342749, 278: 0.6003766478342749, 279: 0.6003766478342749, 280: 0.6
003766478342749, 281: 0.6003766478342749, 282: 0.6003766478342749, 283: 0.6003766
478342749, 284: 0.6003766478342749, 285: 0.6003766478342749, 286: 0.6003766478342
749, 287: 0.6003766478342749, 288: 0.6003766478342749, 289: 0.6003766478342749, 2
90: 0.6003766478342749, 291: 0.6003766478342749, 292: 0.6003766478342749, 293: 0.
6003766478342749, 294: 0.6003766478342749, 295: 0.6003766478342749, 296: 0.600376
6478342749, 297: 0.6003766478342749, 298: 0.6003766478342749, 299: 0.600376647834
2749, 300: 0.6003766478342749, 301: 0.6003766478342749, 302: 0.6003766478342749,
303: 0.6003766478342749, 304: 0.6003766478342749, 305: 0.6003766478342749, 306:
0.6003766478342749, 307: 0.6003766478342749, 308: 0.6003766478342749, 309: 0.6003
766478342749, 310: 0.6003766478342749, 311: 0.6003766478342749, 312: 0.6003766478
342749, 313: 0.6003766478342749, 314: 0.6003766478342749, 315: 0.600376647834274
9, 316: 0.6003766478342749, 317: 0.6003766478342749, 318: 0.6003766478342749, 31
9: 0.6003766478342749, 320: 0.6003766478342749, 321: 0.6003766478342749, 322: 0.6
003766478342749, 323: 0.6003766478342749, 324: 0.6003766478342749, 325: 0.6003766
478342749, 326: 0.6003766478342749, 327: 0.6003766478342749, 328: 0.6003766478342
749, 329: 0.6003766478342749, 330: 0.6003766478342749, 331: 0.6003766478342749, 3
32: 0.6003766478342749, 333: 0.6003766478342749, 334: 0.6003766478342749, 335: 0.
6003766478342749, 336: 0.6003766478342749, 337: 0.6003766478342749, 338: 0.600376
6478342749, 339: 0.6003766478342749, 340: 0.6003766478342749, 341: 0.600376647834
2749, 342: 0.6003766478342749, 343: 0.6003766478342749, 344: 0.6003766478342749,
345: 0.6003766478342749, 346: 0.6003766478342749, 347: 0.6003766478342749, 348:
0.6003766478342749, 349: 0.6003766478342749, 350: 0.6003766478342749, 351: 0.6003
766478342749, 352: 0.6003766478342749, 353: 0.6003766478342749, 354: 0.6003766478
342749, 355: 0.6003766478342749, 356: 0.6003766478342749, 357: 0.600376647834274
9, 358: 0.6003766478342749, 359: 0.6003766478342749, 360: 0.6003766478342749, 36
1: 0.6003766478342749, 362: 0.6003766478342749, 363: 0.6003766478342749, 364: 0.6
003766478342749, 365: 0.6003766478342749, 366: 0.6003766478342749, 367: 0.6003766
478342749, 368: 0.6003766478342749, 369: 0.6003766478342749, 370: 0.6003766478342
749, 371: 0.6003766478342749, 372: 0.6003766478342749, 373: 0.6003766478342749, 3
74: 0.6003766478342749, 375: 0.6003766478342749, 376: 0.6003766478342749, 377: 0.
6003766478342749, 378: 0.6003766478342749, 379: 0.6003766478342749, 380: 0.600376
6478342749, 381: 0.6003766478342749, 382: 0.6003766478342749, 383: 0.600376647834
2749, 384: 0.6003766478342749, 385: 0.6003766478342749, 386: 0.6003766478342749,
387: 0.6003766478342749, 388: 0.6003766478342749, 389: 0.6003766478342749, 390:
0.6003766478342749, 391: 0.6003766478342749, 392: 0.6003766478342749, 393: 0.6003

```
766478342749, 394: 0.6003766478342749, 395: 0.6003766478342749, 396: 0.6003766478
342749, 397: 0.6003766478342749, 398: 0.6003766478342749, 399: 0.600376647834274
9, 400: 0.6003766478342749, 401: 0.6003766478342749, 402: 0.6003766478342749, 40
3: 0.6003766478342749, 404: 0.6003766478342749, 405: 0.6003766478342749, 406: 0.6
003766478342749, 407: 0.6003766478342749, 408: 0.6003766478342749, 409: 0.6003766
478342749, 410: 0.6003766478342749, 411: 0.6003766478342749, 412: 0.6003766478342
749, 413: 0.6003766478342749, 414: 0.6003766478342749, 415: 0.6003766478342749, 4
16: 0.6003766478342749, 417: 0.6003766478342749, 418: 0.6003766478342749, 419: 0.
6003766478342749, 420: 0.6003766478342749, 421: 0.6003766478342749, 422: 0.600376
6478342749, 423: 0.6003766478342749, 424: 0.6003766478342749, 425: 0.600376647834
2749, 426: 0.6003766478342749, 427: 0.6003766478342749, 428: 0.6003766478342749,
429: 0.6003766478342749, 430: 0.6003766478342749, 431: 0.6003766478342749, 432:
0.6003766478342749, 433: 0.6003766478342749, 434: 0.6003766478342749, 435: 0.6003
766478342749, 436: 0.6003766478342749, 437: 0.6003766478342749, 438: 0.6003766478
342749, 439: 0.6003766478342749, 440: 0.6003766478342749, 441: 0.600376647834274
9, 442: 0.6003766478342749, 443: 0.6003766478342749, 444: 0.6003766478342749, 44
5: 0.6003766478342749, 446: 0.6003766478342749, 447: 0.6003766478342749, 448: 0.6
003766478342749, 449: 0.6003766478342749, 450: 0.6003766478342749, 451: 0.6003766
478342749, 452: 0.6003766478342749, 453: 0.6003766478342749, 454: 0.6003766478342
749, 455: 0.6003766478342749, 456: 0.6003766478342749, 457: 0.6003766478342749, 4
58: 0.6003766478342749, 459: 0.6003766478342749, 460: 0.6003766478342749, 461: 0.
6003766478342749, 462: 0.6003766478342749, 463: 0.6003766478342749, 464: 0.600376
6478342749, 465: 0.6003766478342749, 466: 0.6003766478342749, 467: 0.600376647834
2749, 468: 0.6003766478342749, 469: 0.6003766478342749, 470: 0.6003766478342749,
471: 0.6003766478342749, 472: 0.6003766478342749, 473: 0.6003766478342749, 474:
0.6003766478342749, 475: 0.6003766478342749, 476: 0.6003766478342749, 477: 0.6003
766478342749, 478: 0.6003766478342749, 479: 0.6003766478342749, 480: 0.6003766478
342749, 481: 0.6003766478342749, 482: 0.6003766478342749, 483: 0.600376647834274
9, 484: 0.6003766478342749, 485: 0.6003766478342749, 486: 0.6003766478342749, 48
7: 0.6003766478342749, 488: 0.6003766478342749, 489: 0.6003766478342749, 490: 0.6
003766478342749, 491: 0.6003766478342749, 492: 0.6003766478342749, 493: 0.6003766
478342749, 494: 0.6003766478342749, 495: 0.6003766478342749, 496: 0.6003766478342
749, 497: 0.6003766478342749, 498: 0.6003766478342749, 499: 0.6003766478342749}

Melhor valor para max_depth: 26
```

BaggingClassifier aplicada a ExtraTreeClassifier

Tecnica aplicada para classificar varios modelos base cada um em subconjuntos aleatórios do conjunto de dados original e, em seguida, agregar suas previsões individuais (por votação ou por média) para formar uma previsão final.

In [ ]:
```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import ExtraTreeClassifier

extra_tree_model = BaggingClassifier(ExtraTreeClassifier(random_state=1364)
, random_state=1364)
extra_tree_model.fit(X_train, y_train)
print(extra_tree_model.score(X_train, y_train))
print(extra_tree_model.score(X_valid, y_valid))
```

```
0.9944758317639674
0.8033898305084746
```

In [ ]:
```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import ExtraTreeClassifier

extra_tree_model = BaggingClassifier(ExtraTreeClassifier(random_state=126)
, random_state=126)
```

```
extra_tree_model.fit(X_train, y_train)
print(extra_tree_model.score(X_train, y_train))
print(extra_tree_model.score(X_valid, y_valid))
```

```
0.9939736346516007
0.791713747645951
```

In [ ]:
```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import ExtraTreeClassifier

extra_tree_model = BaggingClassifier(ExtraTreeClassifier(random_state=503)
, random_state=503)
extra_tree_model.fit(X_train, y_train)
print(extra_tree_model.score(X_train, y_train))
print(extra_tree_model.score(X_valid, y_valid))
```

```
0.9939736346516007
0.791713747645951
```

In [ ]:
```
extra_tree_model.fit(data_train_X, data_train_y)
y_pred = extra_tree_model.predict(data_test_X)

print(f"{extra_tree_model.score(data_train_X, data_train_y)}\n")

print("id", "floresta")
for i in range(5):
  print(data_test_id[i], y_pred[i])

submission = pd.DataFrame({
  "id" : data_test_id,
  "Floresta": y_pred
  })

submission.to_csv("submission.csv", index=False)
print("\nGerado Arquivo submission")

# O modelo
print()
print(extra_tree_model)
print()
print()
end_time = datetime.now()
print('Tempo para correr esta experiência : {}'.format(end_time - start_time))
```

```
0.8516949152542372

id floresta
10621 2
10622 6
10623 6
10624 6
10625 6

Gerado Arquivo submission

ExtraTreeClassifier(max_features=None, max_leaf_nodes=314, random_state=126,
                    splitter='best')


Tempo para correr esta experiência : 0:08:55.883833
```

Algoritmos e Parâmetros testados

DecisionTreeClassifier

1. com GridSearch

- 'random_state'
- 'criterion'
- 'max_depth'
- 'min_samples_split'
- 'splitter'
- 'min_samples_leaf'
- 'max_features'
- 'max_leaf_nodes'

2. com média de erros absolutos

- 'max_leaf_nodes'

ExtraTreeClassifier

1. com GridSearch
    - 'random_state'
    - 'criterion'
    - 'splitter'
    - 'max_depth'
    - 'min_samples_split'
    - 'min_samples_leaf'
    - 'max_features'
    - 'max_leaf_nodes'
2. com média de erros abosultos
    - 'max_leaf_nodes'
    - 'max_depth'

Estratégia de Avaliação

Nomea-se as estratégias utilizadas:

1. GridSearch e Cross Validation
2. BaggingClassifier
3. mean_absolute_error- media de erros abosolutos
4. Accuracy- media de exatidão

Escolha dos modelos submetidos na plataforma kaggle

Subconjunto dos modelos criados

DecisionTreeClassifier(criterion='entropy', max_features='sqrt', random_state=1)

O subconjunto para GridSearch da DecisionTreeClassifier:

- 'random_state': [1, 12, 123, 246, 692, 1364],
- 'criterion': ['gini', 'entropy'],
- 'max_depth': [None, 10, 20, 30],
- 'min_samples_split': [2, 5, 10],
- 'splitter': ['best', 'random'],
- 'min_samples_leaf': [1, 2, 4],
- 'max_features': ['sqrt', 'log2'],
- 'max_leaf_nodes': [None,5,50,250,500,5000]

Gerou o melhor modelo:

DecisionTreeClassifier(criterion='entropy', max_features='sqrt', random_state=1, max_leaf_nodes= 5000)

Realizei uma nova experiência com novos valores para:

- 'max_leaf_nodes': [500,1000,2500,5000]

Desta vez, max_leaf_nodes=1000 representa o melhor modelo.

DecisionTreeClassifier(criterion='entropy', max_features='sqrt', random_state=1, max_leaf_nodes=1000)

Sabendo que o parametro 'max_leaf_nodes' é de grande relevância no algoritmo DecisionTreeClassifier testei, unica e exclusivamente, os valores do conjunto:

- max_leaf_nodes in [500,800,870,880,890,900,910,920,980,990,1000]

Os valores foram avaliados com base na media de erros abosolutos com a menor media para 890 arvores, gerando desta forma o melhor modelo para o algoritmo em questao:

DecisionTreeClassifier(max_leaf_nodes=890, random_state=1364)

ExtraTreeClassifier

Inicialmente o subconjunto para GridSearch de ExtraTreeClassifier:

- 'random_state': [i for i in range(1,1000)] Com os tipos de avaliação:
    1. 'accuracy'- random_state=126
    2. 'neg_mean_absolute_error'- random_state=503

Em passos seguintes a adicão uma a uma dos subconjuntos:

- 'random_state': [126,503],
- 'criterion': ['gini', 'entropy', 'log_loss'],
- 'splitter': ['best', 'random'],
- 'max_depth': [None,[i for i in range(1,500)]]- None,
- 'min_samples_split': [i for i in range(1,1000)]- 2,
- 'min_samples_leaf': [i for i in range(1,1000)]- 1,
- 'max_features': [None, [i for i in range(1,1000)]]- None,
- 'max_leaf_nodes': [i for i in range(2, 1000)]- 314

Após esta experiência resultou no modelo:

ExtraTreeClassifier(max_features=None, max_leaf_nodes=314, random_state=126, splitter='best')

Caracteristicas dos dois Melhores modelos em publico

DecisionTreeClassifier(max_leaf_nodes=890, random_state=1364)

- Score: 0.78177
- Public score: 0.78314

BaggingClassifier(estimator=ExtraTreeClassifier(random_state=1364), random_state=1364)

- Score: 0.80166
- Public score: 0.81849

Todos os modelos submetidos e seus desempenhos em publico e privado

DecisionTreeClassifier(criterion='entropy', max_features='sqrt', random_state=1)

- Score: 0.75866
- Public score: 0.76512

DecisionTreeClassifier(max_features='sqrt', max_leaf_nodes=870, random_state=1364)

- Score: 0.75096
- Public score: 0.75662

Apresentou o mesmo desempenho para o modelo quando random_state do train_test_split era 12

DecisionTreeClassifier(max_features='sqrt', max_leaf_nodes=870,random_state=1364)

Conclui-se que não faz diferença na submissão final.

DecisionTreeClassifier(max_leaf_nodes=890, random_state=1364)

- Score: 0.78177
- Public score: 0.78314

Apresenta o mesmo desempenho para o modelo usando validação cruzada:

cross_val_score(tree_model, X_train, y_train, cv=5)

DecisionTreeClassifier(max_leaf_nodes=890, random_state=1364)

BaggingClassifier(estimator=ExtraTreeClassifier(random_state=1364), random_state=1364)

- Score: 0.80166
- Public score: 0.81849

ExtraTreeClassifier(max_features=None, max_leaf_nodes=314, random_state=126, splitter='best')

- Score: 0.76508
- Public score: 0.76240