Mini Projeto MLP

Grupo

- Camila Vieira
- Dayane Lira
- José Vinicius

Objetivo

Verificar como a alteração de parâmtros (número de camadas, número de unidades, taxa de aprendizagem, funções de ativação, dropout, regularização, etc) interfere no resultado do experimento (os testes serão realizados com o Conjunto de Dados de Sintomas de Lombalgia disponível no Kaggle), utilizando sklearn.neural_network.MLPClassifier. Importante avaliar os dados, ver normalização e eliminar atributos identificadores. Medir acurácia, matriz de confusão, precision, recall.

Importações e Downloads

```
!pip install scikit-optimize
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split, ShuffleSplit
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from skopt.optimizer import gbrt_minimize
from skopt.space.space import Categorical, Integer, Real
from skopt.utils import use_named_args
from imblearn.over_sampling import SMOTE
```

Preparando os dados

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

```
Mounted at /content/drive
```

```
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/backpain/Dataset_spine.csv")
df.drop([df.columns[-1]],axis=1,inplace=True)
df.head()
```

	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col
0	63.027817	22.552586	39.609117	40.475232	98.672917	-0.254400	0.744503	12.566
1	39.056951	10.060991	25.015378	28.995960	114.405425	4.564259	0.415186	12.887
2	68.832021	22.218482	50.092194	46.613539	105.985135	-3.530317	0.474889	26.834
3	69.297008	24.652878	44.311238	44.644130	101.868495	11.211523	0.369345	23.560
4	49.712859	9.652075	28.317406	40.060784	108.168725	7.918501	0.543360	35.494

Pelo dicionário de dados, sabemos que:

- Col1 - pelvic_incidence
- Col2 - pelvic tilt
- Col3 - lumbar_lordosis_angle
- Col4 - sacral_slope
- Col5 - pelvic_radius
- Col6 - degree_spondylolisthesis
- Col7 - pelvic_slope
- Col8 - direct_tilt
- Col9 - thoracic_slope

- Col10 - cervical_tilt
- Col11 - sacrum_angle
- Col12 - scoliosis_slope
- Class_att - Abnormal, Normal (Normality)

Por meio dos doze primeiros atributos, iremos prever o último.

```
#Renomeando colunas
new_columns = ['pelvic_incidence','pelvic_tilt','lumbar_lordosis_angle',
        'sacral_slope','pelvic_radius','degree_spondylolisthesis',
        'pelvic_slope','direct_tilt','thoracic_slope','cervical_tilt',
        'sacrum_angle','scoliosis_slope','normality']

df.columns = new_columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 310 entries, 0 to 309
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   pelvic_incidence          310 non-null    float64
 1   pelvic_tilt               310 non-null    float64
 2   lumbar_lordosis_angle     310 non-null    float64
 3   sacral_slope              310 non-null    float64
 4   pelvic_radius             310 non-null    float64
 5   degree_spondylolisthesis  310 non-null    float64
 6   pelvic_slope              310 non-null    float64
 7   direct_tilt               310 non-null    float64
 8   thoracic_slope            310 non-null    float64
 9   cervical_tilt             310 non-null    float64
 10  sacrum_angle              310 non-null    float64
 11  scoliosis_slope           310 non-null    float64
 12  normality                 310 non-null    object 
dtypes: float64(12), object(1)
memory usage: 31.6+ KB
```

Os dados estão completos, sem campos NaN. Os atributos são todos do tipo númerico, no entanto o target é do tipo object.

```
df["normality"].value_counts()
```

```
Abnormal    210
Normal      100
Name: normality, dtype: int64
```

```
#Transformar o último campo com encoder para 0 (Abnormal) e 1 (Normal)
label_encoder = LabelEncoder()
df["normality"] = label_encoder.fit_transform(df["normality"])
df["normality"].value_counts()
```

```
0    210
1    100
Name: normality, dtype: int64
```

```
#Separar atributos e label
x = df.iloc[:, :-1]
y = df["normality"]
```

Análise Exploratória dos Dados e Pré-Processamento

Verificar eficácia sem pré-processamento

```
def estimar_loss(x, y):
  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15,random_state=42, shuffle=True)
  #Modelo com valores usuais
  mlp_classifier = MLPClassifier(hidden_layer_sizes=(100,), activation='relu', solver='adam', alpha=0.0001,
                                 batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5,
                                 max_iter=200, shuffle=True, random_state=42, tol=0.0001, verbose=False,
                                 warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False,
                                 validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10,
                                 max_fun=15000)
  #Treinando
  mlp_classifier.fit(x_train, y_train)
  print("Loss do MLP Classifier:", np.mean(mlp_classifier.loss_))

```

```
14 #Avaliação Iterativa: A loss será avaliada em cada estágio do pré-processamento, para entender como cada etapa afeta o desempenho do
```

```
1 estimar_loss(x,y)
```

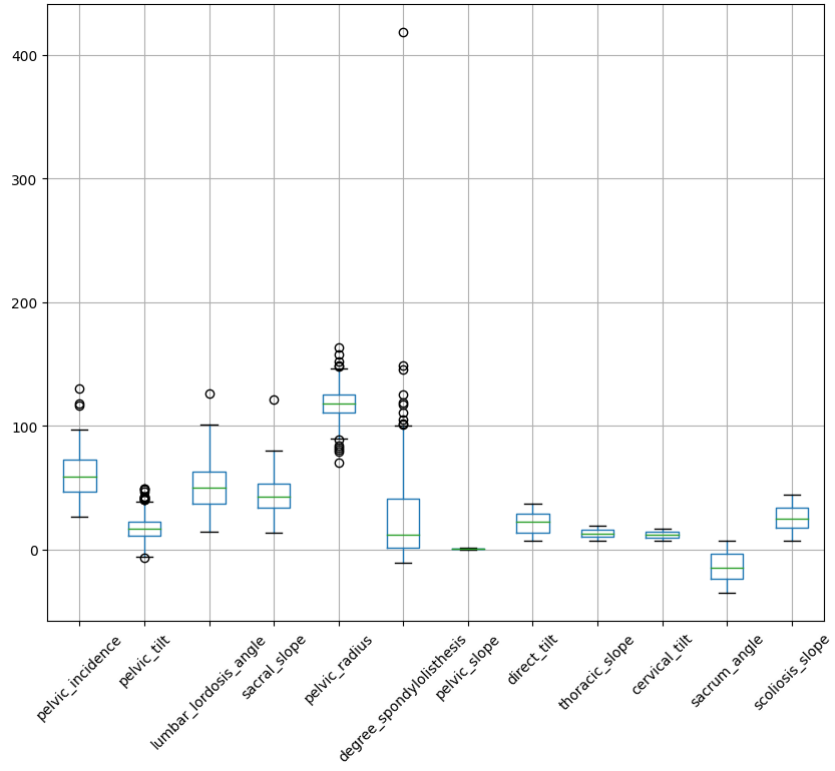
```
Loss do MLP Classifier: 0.27808027948957015
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
```

Retirando Outliers

Box Plot

Os box plots são úteis para visualizar a distribuição de valores em cada característica, mostrando a mediana, quartis e outliers. Isso ajuda a identificar discrepâncias nos dados que podem afetar a modelagem.

```
1 #Box plot
2 plt.figure(figsize=(10, 8))
3 x.boxplot()
4 plt.xticks(rotation=45)
5 plt.show()
```



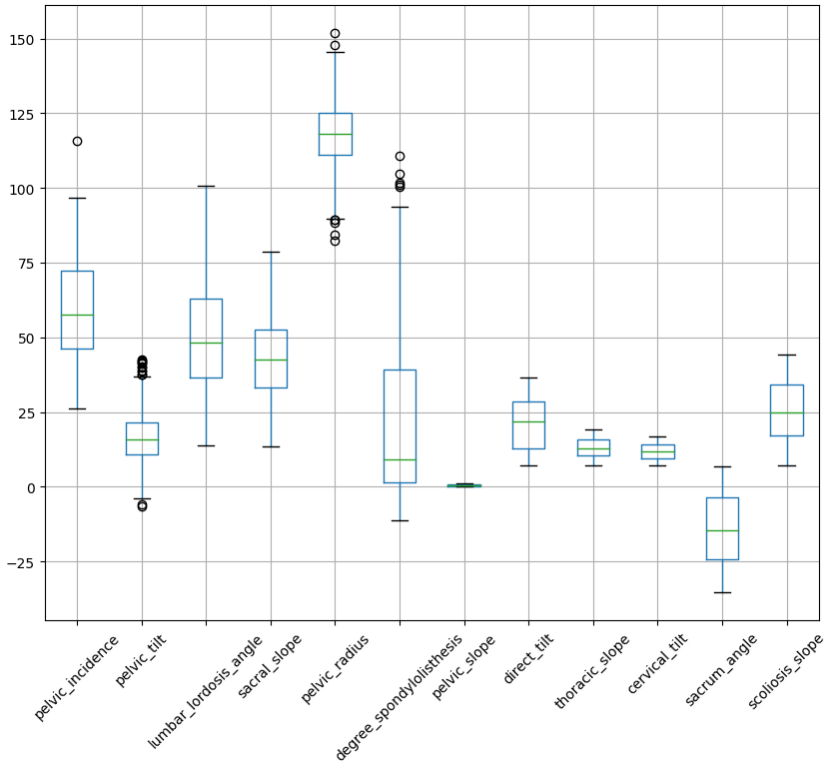
Em um conjunto de dados relativamente pequeno, observamos pelo box plot que a maioria dos outliers está localizada nas proximidades das margens do Intervalo Interquartil (IQR). A abordagem padrão para identificar outliers é a regra dos 1,5 vezes o IQR, mas optamos por aumentar o fator multiplicador para 2, devido ao seu tamanho limitado e à distribuição observada dos outliers.

```
1 #Retirada de outliers
2 df_clean = df.copy()
3 for col in x:
4   lower_bound = df_clean[col].quantile(0.25) - 2 * (df_clean[col].quantile(0.75) - df_clean[col].quantile(0.25))
5   upper_bound = df_clean[col].quantile(0.75) + 2 * (df_clean[col].quantile(0.75) - df_clean[col].quantile(0.25))
6   df_clean = df_clean[(df_clean[col] >= lower_bound) & (df_clean[col] <= upper_bound)]
7 x_clean = df_clean.iloc[:, :-1]
```

```
8 y_clean = df_clean["normality"]
```

```
Loss do MLP Classifier: 0.30581135424572015
```

```
1 #Box plot
2 plt.figure(figsize=(10, 8))
3 x_clean.boxplot()
4 plt.xticks(rotation=45)
5 plt.show()
```



Balanceamento de classes

Análise dos Atributos Binários

Gráfico de contagem adequado para atributos binários, mostra a distribuição da variável. Relevante para entender o equilíbrio da classe.

```
1 #Análise dos atributos binários
2 plt.figure(figsize=(6, 4))
3 sns.countplot(x="normality", data=df_clean)
4 plt.xlabel("Atributo Binário")
5 plt.ylabel("Contagem")
6 plt.show()
```

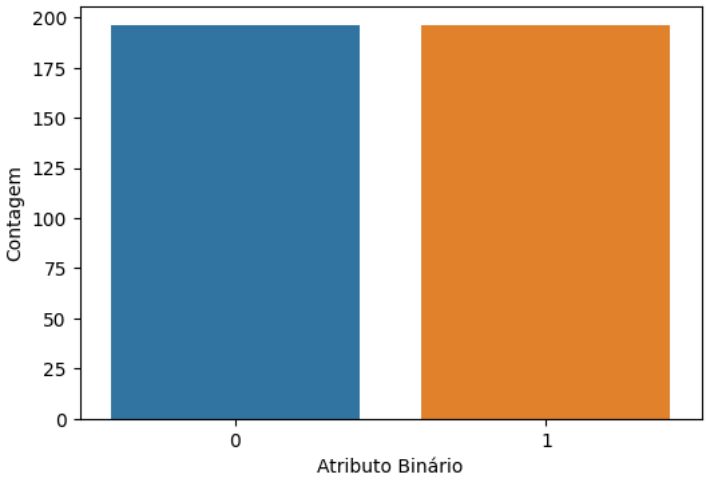

```
print(df_clean["normality"].value_counts())
```

```
0    196
1    100
Name: normality, dtype: int64
```

```
#Balanceando dados
smote = SMOTE(random_state=42)
x_resampled, y_resampled = smote.fit_resample(x_clean, y_clean)
balanced_df = pd.concat([pd.DataFrame(x_resampled, columns=x_clean.columns), pd.Series(y_resampled, name="normality")], axis=1)
balanced_x = balanced_df.iloc[:, :-1]
balanced_y = balanced_df["normality"]
estimar_loss(balanced_x, balanced_y)
print(balanced_df["normality"].value_counts())
```

```
Loss do MLP Classifier: 0.23469823051296654
0    196
1    196
Name: normality, dtype: int64
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
```

```
#Análise dos atributos binários
plt.figure(figsize=(6, 4))
sns.countplot(x="normality", data=balanced_df)
plt.xlabel("Atributo Binário")
plt.ylabel("Contagem")
plt.show()
```



Normalização e Padronização

Sumário Estatístico

O resumo estatístico fornece estatísticas descritivas básicas para cada característica no df. Isso inclui a contagem de exemplos, média, desvio padrão, valor mínimo, quartis e valor máximo. Isso ajuda a ter uma visão geral das características e de suas escalas. Pode ser útil para identificar características com escalas muito diferentes e visualizar a necessidade de normalização.

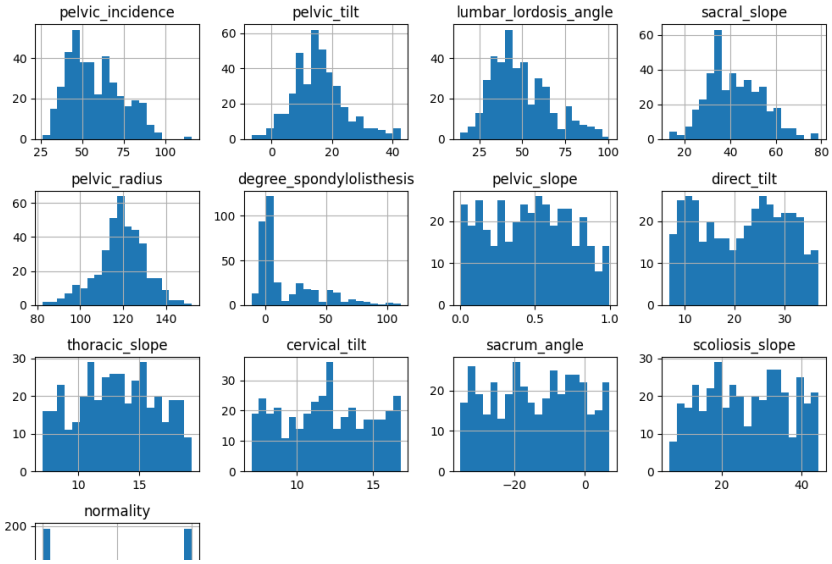
```
#Sumário Estatístico
balanced_df.describe()
```

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_
count	392.000000	392.000000	392.000000	392.000000	392.
mean	57.810532	15.906806	49.330108	41.903726	119.
std	15.978789	8.756700	17.283902	11.835850	11

Distribuição de valores

Os histogramas e os gráficos KDE mostram a distribuição dos valores em cada característica. Isso ajuda a entender a forma da distribuição, verificar se as características seguem uma distribuição normal e observar como os valores estão distribuídos. Isso pode influenciar a decisão de normalizar ou padronizar as características.

```
#Distribuição de valores
balanced_df.hist(bins=20, figsize=(10, 8))
plt.tight_layout()
plt.show()

sns.set(style="whitegrid")
plt.figure(figsize=(10, 8))
for col in x:
    sns.kdeplot(balanced_df[col], label=col)
plt.legend()
plt.show()
```




```
#Normalização
columns_to_normalize = ['pelvic_incidence','pelvic_tilt','lumbar_lordosis_angle','pelvic_radius'] #Distribuição semelhante a normal
scaler = MinMaxScaler()
normalized_df = balanced_df.copy()
normalized_df[columns_to_normalize] = scaler.fit_transform(balanced_df[columns_to_normalize])
normalized_x = normalized_df.iloc[:, :-1]
normalized_y = normalized_df["normality"]
estimar_loss(normalized_x, normalized_y)
```

```
Loss do MLP Classifier: 0.20813731852034187
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
```

```
#Padronização
columns_to_standardize = ['pelvic_tilt','sacral_slope','pelvic_radius','pelvic_slope','direct_tilt','thoracic_slope',
                          'cervical_tilt','sacrum_angle','scoliosis_slope'] #Desvio padrão pequeno
scaler = StandardScaler()
standardize_df = normalized_df.copy()
standardize_df[columns_to_standardize] = scaler.fit_transform(normalized_df[columns_to_standardize])
standardize_x = standardize_df.iloc[:, :-1]
standardize_y = standardize_df["normality"]
estimar_loss(standardize_x, standardize_y)
```

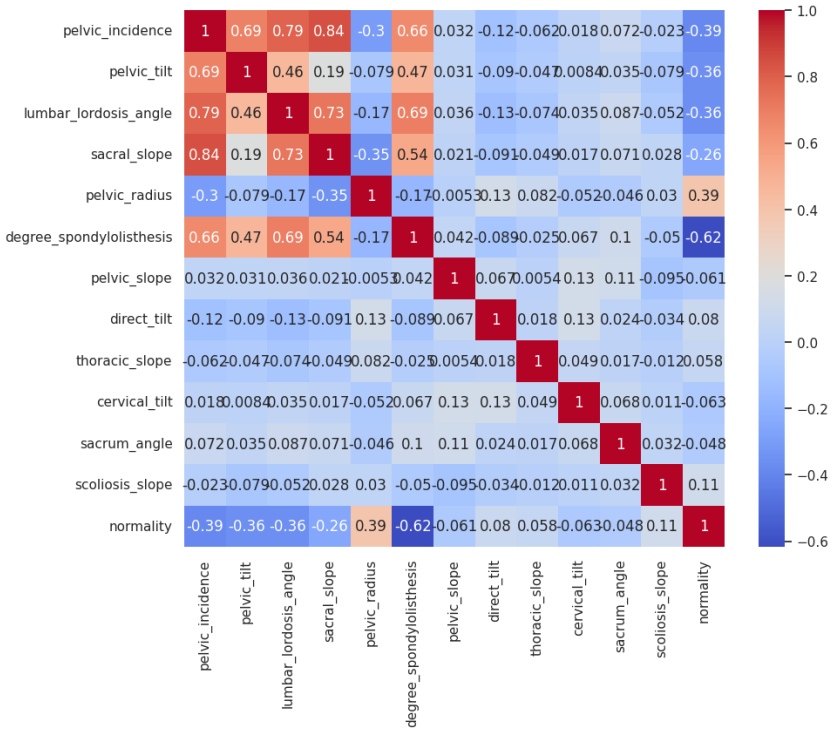
```
Loss do MLP Classifier: 0.15922471077707612
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
```

Análise de Atributos

Análise de Correlações

A matriz de correlação e o mapa de calor ajudam a entender as relações lineares entre as características. Isso pode ser útil para identificar pares que estão altamente correlacionadas, o que pode indicar redundância. Em alguns casos, pode ser útil para a seleção de atributos, removendo características altamente correlacionadas.

```
#Análise de Correlações
correlation_matrix = standardize_df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.show()
```




```
correlation_matrix = standardize_df.corr()
df_filtered = standardize_df.copy()
corr_threshold = 0.8
#Caso nenhuma coluna tenha correlação de 80% ou mais, ficará o mesmo df
highly_correlated = (correlation_matrix.abs() > corr_threshold) & (correlation_matrix != 1)
features_to_remove = set()
for col in highly_correlated.columns:
    correlated_cols = highly_correlated.index[highly_correlated[col]].tolist()
    if len(correlated_cols) > 1:
      features_to_remove.add(correlated_cols[1])
df_filtered = df_filtered.drop(columns=features_to_remove)
x_filtered = df_filtered.iloc[:, :-1]
y_filtered = df_filtered["normality"]
estimar_loss(x_filtered,y_filtered)
```

```
Loss do MLP Classifier: 0.15922471077707612
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
```

Treinamento e Teste (Valores Usuais)

```
#Divisão conjunto treino e test
x_train, x_test, y_train, y_test = train_test_split(x_filtered, y_filtered, test_size=0.15,random_state=42, shuffle=True)
print(f'x_train: {x_train.shape}')
print(f'x_test: {x_test.shape}')
```

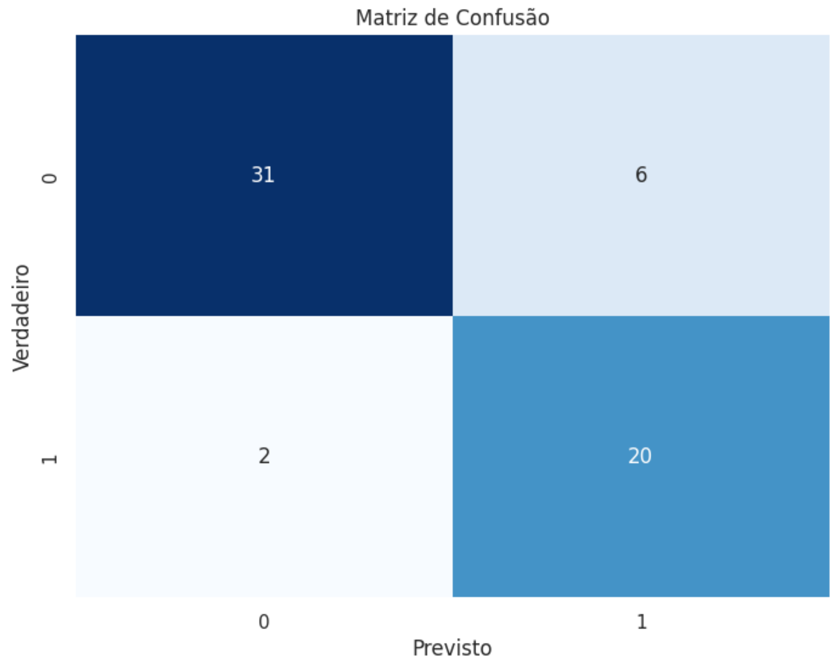
```
x_train: (333, 12)
x_test: (59, 12)
```

```
# Criando MLP
losses = []
for i in range(10):
  mlp_classifier = MLPClassifier(hidden_layer_sizes=(100,), activation='relu', solver='adam', alpha=0.0001,
                                 batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5,
                                 max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False,
                                 warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False,
                                 validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10,
                                 max_fun=15000)

# Treinando com os parâmetros usuais
  mlp_classifier.fit(x_train, y_train)
  losses.append(mlp_classifier.loss_)
print("Loss do MLP Classifier:", np.mean(losses))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
Loss do MLP Classifier: 0.15478382217131986
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimi
  warnings.warn(
```

```
y_pred = mlp_classifier.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Acurácia do modelo: {accuracy:.2f}")
print("Classification Report")
print(classification_report(y_test, y_pred))

confusion = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(confusion, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel('Previsto')
plt.ylabel('Verdadeiro')
plt.title('Matriz de Confusão')
plt.show()
```

```
Acurácia do modelo: 0.86
Classification Report
              precision    recall  f1-score   support

           0       0.94      0.84      0.89        37
           1       0.77      0.91      0.83        22

    accuracy                           0.86        59
   macro avg       0.85      0.87      0.86        59
weighted avg       0.88      0.86      0.87        59
```



Otimização de Hiperparâmetros

```
parameters = [Integer(1, 2, name='num_layers'),
              Integer(1, 128, name='nn_fst_layer'),
              Integer(1, 128, name='nn_snd_layer'),
              Categorical(['identity', 'logistic', 'tanh', 'relu'], name='activation'),
              Categorical(['adam', 'sgd', 'lbfgs'], name='solver'),
              Real(1e-6, 1e-2, prior='log-uniform', name='alpha'),
              Integer(1, 100, name='batch_size'),
              Categorical(['constant', 'invscaling', 'adaptive'], name='learning_rate'),
              Real(1e-6, 1e-3, prior='log-uniform', name='learning_rate_init'),
              Real(1e-6, 1e-1, prior='log-uniform', name='power_t'),
              Integer(500, 1000, name='max_iter'),
              Real(1e-6, 1e-2, prior='log-uniform', name='tol'),
              Real(0.1, 0.9, name='momentum'),
              Categorical([True, False], name='nesterovs_momentum'),
              Real(0.01, 0.5, name='validation_fraction'),
              Real(0.1, 0.9, name='beta_1'),
              Real(0.001, 0.999, name='beta_2'),
              Real(1e-10, 1e-6, prior='log-uniform', name='epsilon'),
              Integer(1, 100, name='n_iter_no_change'),
              Integer(1, 30000, name='max_fun')]

@use_named_args(parameters)
def objective(**params):
  print(params)
  split = ShuffleSplit(n_splits=2, test_size=0.15)
  indices = [train for  (train, test) in split.split(x_train.to_numpy())]
  data_x, data_y = (x_train.to_numpy())[indices[0]], (y_train.to_numpy())[indices[0]]
  split = ShuffleSplit(n_splits=5, test_size=0.2)
  accuracy = []
  for train, test in split.split(data_x):
    if(params["num_layers"] == 1):
        hidden_layer=(params["nn_fst_layer"],)
    if(params["num_layers"] == 2):
        hidden_layer=(params["nn_fst_layer"],params["nn_snd_layer"])

    mlp_classifier = MLPClassifier(hidden_layer_sizes=hidden_layer, activation=params['activation'], solver=params['solver'],
                                   alpha=params['alpha'],batch_size=params['batch_size'], learning_rate=params['learning_rate'],
                                   learning_rate_init=params['learning_rate_init'], power_t=params['power_t'],
                                   max_iter=params['max_iter'], shuffle=True, random_state=42,
                                   tol=params['tol'], verbose=False, warm_start=False, momentum=params['momentum'],
                                   nesterovs_momentum=params['nesterovs_momentum'], early_stopping=False,
                                   validation_fraction=params['validation_fraction'], beta_1=params['beta_1'],
                                   beta_2=params['beta_2'], epsilon=params['epsilon'], n_iter_no_change=params['n_iter_no_change'],
                                   max_fun=params['max_fun'])
    mlp_classifier.fit(data_x[train], data_y[train])
    y_pred = mlp_classifier.predict(x_test)
    accuracy.append(accuracy_score(y_test, y_pred))
  return -np.array(accuracy).mean()

result = gbrt_minimize(func=objective, dimensions=parameters, n_calls=50, acq_func='EI', n_jobs=-1)
```

```
{'num_layers': 2, 'nn_1st_layer': 80, 'nn_2nd_layer': 1, 'activation': 'logistic', 'solver': 'lbfgs', 'alpha': 0.004523168826748
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but MLPClassifier was fitted witho
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but MLPClassifier was fitted witho
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
result.x
```

```
[2,
 42,
 71,
 'logistic',
 'lbfgs',
 0.0016797304312511862,
 52,
 'invscaling',
 0.0005208012653703742,
 1.3499894600491117e-06,
 801,
 2.437733590381186e-06,
 0.805771513687481,
 False,
 0.4688985210894677,
 0.2334915810999844,
 0.9522742442962157,
 3.351998611086149e-07,
 53,
 23205]
```

```
if(result.x[0] == 1):
  hidden_layer=(result.x[1],)
if(result.x[0] == 2):
  hidden_layer=(result.x[1],result.x[2])

losses = []
for i in range(10):
  mlp_classifier = MLPClassifier(hidden_layer_sizes=hidden_layer, activation=result.x[3], solver=result.x[4], alpha=result.x[5],
                              batch_size=result.x[6], learning_rate=result.x[7], learning_rate_init=result.x[8], power_t=result.x[9
                              max_iter=result.x[10], shuffle=True, random_state=None, tol=result.x[11], verbose=False,
                              warm_start=False, momentum=result.x[12], nesterovs_momentum=result.x[13], early_stopping=False,
                              validation_fraction=result.x[14], beta_1=result.x[15], beta_2=result.x[16], epsilon=result.x[17],
                              n_iter_no_change=result.x[18], max_fun=result.x[19])

  mlp_classifier.fit(x_train, y_train)
  losses.append(mlp_classifier.loss_)
print("Loss do MLP Classifier:", np.mean(losses))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to c
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to c
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to c
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to c
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to c
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to c
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to c
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
Loss do MLP Classifier: 0.0013173242808889325
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failed to c
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```python
y_pred = mlp_classifier.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Acurácia do modelo: {accuracy:.2f}")
print("Classification Report")
print(classification_report(y_test, y_pred))

confusion = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(confusion, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel('Previsto')
plt.ylabel('Verdadeiro')
plt.title('Matriz de Confusão')
plt.show()
```

```
Acurácia do modelo: 0.90
```

Salvar como PDF

```
              0       0.94      0.89      0.92        37
```

```
1 !apt-get install texlive texlive-xetex texlive-latex-extra pandoc
2 !pip install pypandoc
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
pandoc is already the newest version (2.9.2.1-3ubuntu2).
pandoc set to manually installed.
The following additional packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono
  fonts-texgyre fonts-urw-base35 libapache-pom-java libcommons-logging-java
  libcommons-parent-java libfontbox-java libfontenc1 libgs9 libgs9-common
  libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1
  libruby3.0 libsynctex2 libteckit0 libtexlua53 libtexluajit2 libwoff1
  libzzip-0-13 lmodern poppler-data preview-latex-style rake ruby
  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0
  rubygems-integration t1utils teckit tex-common tex-gyre texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-recommended texlive-pictures texlive-plain-generic tipa
  xfonts-encodings xfonts-utils
Suggested packages:
  fonts-noto fonts-freefont-otf | fonts-freefont-ttf libavalon-framework-java
  libcommons-logging-java-doc libexcalibur-logkit-java liblog4j1.2-java
  poppler-utils ghostscript fonts-japanese-mincho | fonts-ipafont-mincho
  fonts-japanese-gothic | fonts-ipafont-gothic fonts-arphic-ukai
  fonts-arphic-uming fonts-nanum ri ruby-dev bundler debhelper gv
  | postscript-viewer perl-tk xpdf | pdf-viewer xzdec
  texlive-fonts-recommended-doc texlive-latex-base-doc python3-pygments
  icc-profiles libfile-which-perl libspreadsheet-parseexcel-perl
  texlive-latex-extra-doc texlive-latex-recommended-doc texlive-luatex
  texlive-pstricks dot2tex prerex texlive-pictures-doc vprerex
  default-jre-headless tipa-doc
The following NEW packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono
  fonts-texgyre fonts-urw-base35 libapache-pom-java libcommons-logging-java
  libcommons-parent-java libfontbox-java libfontenc1 libgs9 libgs9-common
  libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1
  libruby3.0 libsynctex2 libteckit0 libtexlua53 libtexluajit2 libwoff1
  libzzip-0-13 lmodern poppler-data preview-latex-style rake ruby
  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0
  rubygems-integration t1utils teckit tex-common tex-gyre texlive texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-extra texlive-latex-recommended texlive-pictures
  texlive-plain-generic texlive-xetex tipa xfonts-encodings xfonts-utils
0 upgraded, 55 newly installed, 0 to remove and 16 not upgraded.
Need to get 182 MB of archives.
After this operation, 572 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-droid-fallback all 1:6.0.1r16-1.1build1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-lato all 2.0-2.1 [2,696 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 poppler-data all 0.4.11-1 [2,171 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-common all 6.17 [33.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-urw-base35 all 20200910-1 [6,367 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9-common all 9.55.0~dfsg1-0ubuntu5.4 [752 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libidn12 amd64 1.38-4ubuntu1 [60.0 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/main amd64 libijs-0.35 amd64 0.35-15build2 [16.5 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/main amd64 libjbig2dec0 amd64 0.19-3build2 [64.7 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9 amd64 9.55.0~dfsg1-0ubuntu5.4 [5,032 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libkpathsea6 amd64 2021.20210626.59705-1ubuntu0.1 [60.3 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/main amd64 libwoff1 amd64 1.0.2-1build4 [45.2 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy/universe amd64 dvisvgm amd64 2.13.1-1 [1,221 kB]
```

```
1 !jupyter nbconvert --to PDF "/content/drive/MyDrive/Colab Notebooks/Mini-projeto-MLP.ipynb"
```

```
[NbConvertApp] Converting notebook /content/drive/MyDrive/Colab Notebooks/Mini-projeto-MLP.ipynb to PDF
[NbConvertApp] Support files will be in Mini-projeto-MLP_files/
[NbConvertApp] Making directory ./Mini-projeto-MLP_files
[NbConvertApp] Making directory ./Mini-projeto-MLP_files
[NbConvertApp] Making directory ./Mini-projeto-MLP_files
[NbConvertApp] Making directory ./Mini-projeto-MLP_files
[NbConvertApp] Making directory ./Mini-projeto-MLP_files
[NbConvertApp] Making directory ./Mini-projeto-MLP_files
[NbConvertApp] Making directory ./Mini-projeto-MLP_files
[NbConvertApp] Making directory ./Mini-projeto-MLP_files
[NbConvertApp] Making directory ./Mini-projeto-MLP_files
[NbConvertApp] Writing 214309 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 583171 bytes to /content/drive/MyDrive/Colab Notebooks/Mini-projeto-MLP.pdf
```

✓ 19s completed at 8:36 AM