Faculdade de Tecnologia
Universidade Estadual de Campinas

CET0621 – Aprendizado de Máquina na Análise de Dados
Lista de Exercícios 3

Nome do Aluno 1: Gustavo Ferreira Lima **RA: 2023611300**

Nome do Aluno 2: Mateus de Almeida Frigo **RA: 2023611431**

Instruções

- Esta lista de exercícios deve ser desenvolvida em **trios**;
- As respostas devem estar em um arquivo PDF, juntamente com o **nome e RA dos membros do trio**;
- **Apenas um dos membros** deve postar o arquivo no Moodle;
- Esta lista contém questões teóricas e práticas.

Questões:

1) (3,0pt) A base de dados apresentada na Tabela 1 contém 10 amostras associadas a um problema de estimação de dados, onde A é o atributo de entrada (variável independente) e B é a saída (variável dependente). Neste contexto, pede-se:

   a) Com as amostras da Tabela 1, construa um gráfico de dispersão dos dados, colocando o atributo A no eixo x e o atributo B no eixo y.

   b) A partir da análise do gráfico obtido no item anterior, indique se seria possível utilizar um **método paramétrico** para identificar a função que relaciona A e B (justifique sua resposta). Caso sua resposta seja afirmativa, indique qual abordagem você seguiria.

   c) Supondo que um cientista de dados aplicou Regressão Linear aos dados da Tabela 1 e obteve a equação de reta $B'' = 0{,}9697 \times A + 0{,}667$, aplique tal equação aos valores de A e obtenha os valores estimados B''.

   d) A partir dos valores de B'' obtidos no item anterior, calcule o *Erro Quadrático Médio* (EQM) e o *Erro Quadrático Relativo* (EQR) desta estimação.

2) (2,0pt) A base de dados apresentada na Tabela 2 contém a saída apresentada por dois estimadores, sendo um deles uma MLP e outro um Regressor Polinomial, para 10 amostras associadas a um problema de estimação de dados. A saída esperada (correta) para cada uma das 10 amostras também é dada na Tabela 2. Neste contexto, pede-se:

   a) Apresente a saída que um *ensemble*, formado pela MLP e pelo Regressor Polinomial fornecidos, teria para cada uma das 10 amostras dos dados. Este ensemble deve realizar a **combinação das saídas dos componentes por média simples**.

   b) Calcule e apresente o *Erro Quadrático Médio* (EQM) das estimações feitas pela MLP, pelo Regressor Polinomial e pelo *Ensemble*.

Tabela 1 - Base de dados para um problema de estimação de dados.

ID	A (entrada)	B (saída esperada)	B'' (saída do estimador)
1	1	1	
2	2	4	
3	3	9	
4	4	16	
5	5	22	
6	6	40	
7	7	50	
8	8	70	
9	9	80	
10	10	105	

Tabela 2 – Saídas de dois estimadores para uma base de dados.

Valor Esperado	MLP	Regressão Polinomial	Ensemble
0,75	0,5	0,9	
2,0	2,1	1,9	
3,8	3,8	3,8	
4,2	4,0	4,5	
5,0	4,8	5,2	
6,4	6,7	6,1	
7,7	8,0	7,4	
8,0	8,0	8,0	
9,1	9,5	8,7	
10,8	10,7	10,9	

3) (5,0pt) Utilizando alguma **ferramenta computacional ou biblioteca de sua preferência** (como Weka, scikit-learn e Orange), realize um estudo comparativo entre o desempenho dos algoritmos *MLP* e Regressor Linear, quando aplicados ao conjunto de dados conhecido como *Boston House-price* (http://lib.stat.cmu.edu/datasets/boston), disponível na biblioteca de bases de dados *Statlib*[1].

Observações:

- Se você for utilizar o Weka, tanto a MLP quanto o Regressor Linear ("*LinearRegression*") estão na categoria "FUNCTIONS" de classificadores.
- O atributo MEDV (definido como *class* no arquivo ARFF) corresponde à saída esperada para o *dataset*.
- O atributo CHAS está definido como *binary* no arquivo ARFF. Converta-o para *real* (linha 56 do arquivo ARFF).

Para este estudo, pede-se:

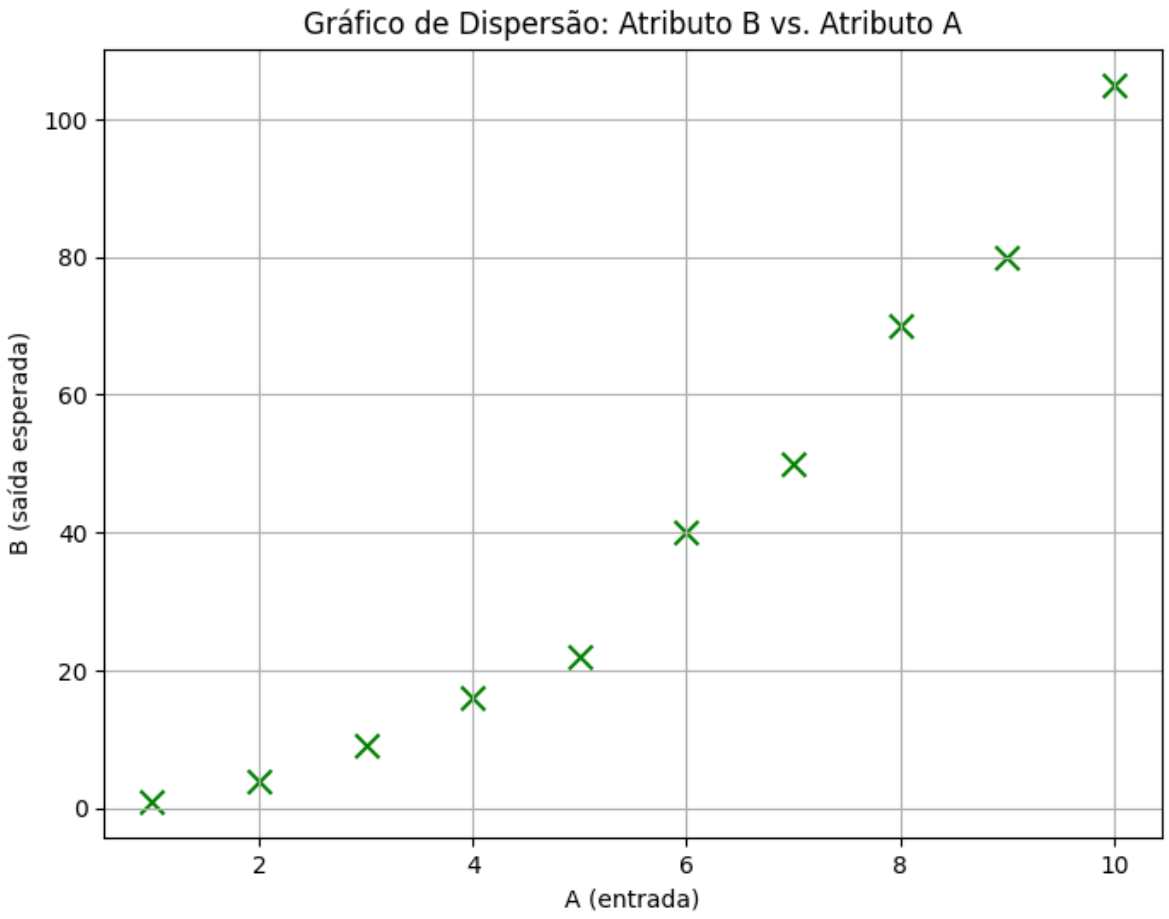
- Para avaliar cada algoritmo, adote a estratégia de validação cruzada com 10 pastas;
- Descreva **detalhadamente** a metodologia experimental empregada. Apresente as etapas de pré-processamento utilizadas e, para cada algoritmo, liste os valores dos parâmetros utilizados.
- Avalie se diferentes estratégias de pré-processamento levam a resultados diferentes.
- Apresente o Erro Absoluto Médio para cada estimador treinado. Para o melhor Regressor Linear, apresente também o modelo (equação) obtido.
- Discuta os resultados obtidos.

Respostas:

1. a)

Utilizaremos os valores de 'A' no eixo X e 'B' no eixo Y, geramos o grafico utilizando codigo Python:

- **Valores de A (eixo X):** 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
- **Valores de B (eixo Y):** 1, 4, 9, 16, 22, 40, 50, 70, 80, 105



O gráfico de dispersão desses dados revela uma relação não linear.

Observa-se que, à medida que 'A' aumenta, 'B' também aumenta, mas a taxa de crescimento de 'B' se acelera, sugerindo uma relação quadrática ou exponencial, e não linear.

Codigo utilizado:

```
import matplotlib.pyplot as plt

# Dados da Tabela 1, Questão 1
A = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] # Atributo A (entrada)
B = [1, 4, 9, 16, 22, 40, 50, 70, 80, 105] # Atributo B (saída esperada)

# Criação do gráfico de dispersão
plt.figure(figsize=(8, 6))
plt.scatter(A, B, color='green', marker='x', s=100)
plt.title('Gráfico de Dispersão: Atributo B vs. Atributo A')
plt.xlabel('A (entrada)') # Rótulo do eixo X
plt.ylabel('B (saída esperada)') # Rótulo do eixo Y
plt.grid(True)
plt.show()
```

b) A partir da análise do gráfico de dispersão, observamos que os pontos se distribuem em uma curva que não é uma linha reta, mas que apresenta um padrão definido e crescente. A taxa de crescimento da variável 'B' aumenta à medida que 'A' cresce. Sobre método paramétrico, é possível que utilizar para identificar a função que relaciona 'A' e 'B'.

Isso por que os métodos paramétricos assumem que a relação entre as variáveis pode ser descrita por uma função matemática com um número fixo de parâmetros. Embora a relação visível no gráfico não seja linear, ela não é aleatória ou caótica. Já os pontos seguem um padrão claro e suave, isso indica que uma função matemática específica, com seus respectivos parâmetros, pode ser ajustada aos dados para modelar essa relação.

Considerando o formato da dispersão dos pontos, a abordagem mais adequada seria a Regressão Polinomial.

A Regressão Polinomial é um tipo de regressão paramétrica que permite modelar relações não lineares entre a variável independente (A) e a variável dependente (B) através da inclusão de termos polinomiais. O grau do polinômio poderia ser ajustado e validado com base em métricas de desempenho para garantir o melhor ajuste sem overfitting.

C) A partir dos valores de B'' obtidos no item anterior (1c) e os valores de B (saída esperada) da Tabela 1, podemos calcular as métricas:

Resultados:

Valores de A (entrada): [1 2 3 4 5 6 7 8 9 10]

Valores de B (saída esperada): [1 4 9 16 22 40 50 70 80 105]

Valores de B'' (saída do estimador calculados): [1.6367 2.6064 3.5761 4.5458 5.5155 6.4852 7.4549 8.4246 9.3943 10.364]

Codigo python utilizado:

```
import numpy as np

# Valores de A (entrada) e B (saída esperada) da Tabela 1
A = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]) # Atributo A (entrada)
B_esperado = np.array([1, 4, 9, 16, 22, 40, 50, 70, 80, 105]) # Atributo B (saída esperada)

# Coeficientes da equação de Regressão Linear
# B = 0,9697 * A + 0,667
coef_angular = 0.9697
coef_linear = 0.667

# Cálculo dos valores de B (saída do estimador) aplicando a equação a cada valor de A
B_estimado = (coef_angular * A) + coef_linear

print("Valores de A (entrada):", A)
print("Valores de B (saída esperada):", B_esperado)
print("Valores de B'' (saída do estimador calculados):", B_estimado)
```

D) Codigo python para avaliar os resultados:

```
import numpy as np

# Dados da Tabela 1
A = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]) # Atributo A (entrada)
B_esperado = np.array([1, 4, 9, 16, 22, 40, 50, 70, 80, 105]) # Atributo B (saída esperada)
```

```
# Coeficientes da equação de Regressão Linear
# B = 0,9697 * A + 0,667
coef_angular = 0.9697
coef_linear = 0.667

# Cálculo dos valores de B (saída do estimador) aplicando a equação a cada valor de A
B_estimado = (coef_angular * A) + coef_linear

# Cálculo do Erro Quadrático Médio (EQM)
eqm = np.mean((B_esperado - B_estimado)**2)

# Cálculo do Erro Quadrático Relativo (EQR)
eqr = np.sum((B_esperado - B_estimado)**2) / np.sum(B_esperado**2)

print(f"Erro Quadrático Médio (EQM): {eqm:.4f}")
print(f"Erro Quadrático Relativo (EQR): {eqr:.4f}")
```

Resultados:

Erro Quadrático Médio (EQM): 2110.0698

Erro Quadrático Relativo (EQR): 0.7740

Os valores calculados para EQM e EQR são notavelmente altos.

Um **EQM de 2122,09312** indica que a média das diferenças quadráticas entre os valores reais e os previstos pela Regressão Linear é muito grande, sugerindo que o modelo está cometendo erros significativos em suas estimativas.

Já um **EQR de aproximadamente 0,7784** (ou 77,84%) é um valor elevado. Isso significa que a soma dos erros quadrados do modelo representa quase 78% da variabilidade total dos próprios dados esperados. O modelo de Regressão Linear está longe de capturar a relação subjacente nos dados, confirmando a análise visual do gráfico de dispersão do item 1a, que já apontava para uma relação não linear e que a Regressão Linear seria inadequada para este problema.

2. A) O ensemble é formado pela média simples das saídas da MLP e do Regressor Polinomial para cada amostra, com base na fórmula: (MLP + Regressor Polinominal) / 2

Amostra 1: (0.5 + 0.9) / 2 = 0.7
Amostra 2: (2.1 + 1.9) / 2 = 2.0
Amostra 3: (3.8 + 3.8) / 2 = 9.8
Amostra 4: (4.0 + 4.5) / 2 = 4.25
Amostra 5: (4.8 + 5.2) / 2 = 5.0

Amostra 6: (6.7 + 6.1) / 2 = 6.4
Amostra 7: (8.0 + 7.4) / 2 = 7.7
Amostra 8: (8.0 + 8.0) / 2 = 8.0
Amostra 9: (9.5 + 8.7) / 2 = 9.1
Amostra 10: (10.7 + 10.9) / 2 = 10.8

B) **Erro Quadrático Médio da MLP**

(0.75 - 0.5)^2 = 0.25^2 = 0.0625
(2.0 - 2.1)^2 = (-0.1)^2 = 0.01
(3.8 - 3.8)^2 = 0^2 = 0
(4.2 - 4.0)^2 = 0.2^2 = 0.04
(5.0 - 4.8)^2 = 0.2^2 = 0.04
(6.4 - 6.7)^2 = (-0.3)^2 = 0.09
(7.7 - 8.0)^2 = (-0.3)^2 = 0.09
(8.0 - 8.0)^2 = 0^2 = 0
(9.2 - 9.5)^2 = (-0.3)^2 = 0.09
(10.8 - 10.7)^2 = 0.1^2 = 0.01

Soma dos erros Quadráticos:

0.0625 + 0.01 + 0 + 0.04 + 0.04 + 0.09 + 0.09 + 0 + 0.09 + 0.01 = 0.435

EQM da MLP = 0.435 / 10 = 0.0435

EQM do Regressor Polinomial

(0.75 - 0.9)^2 = (-0.15)^2 = 0.0225
(2.0 - 1.9)^2 = 0.1^2 = 0.01
(3.8 - 3.8)^2 = 0^2 = 0
(4.2 - 4.5)^2 = (-0.3)^2 = 0.09
(5.0 - 5.2)^2 = (-0.2)^2 = 0.04
(6.4 - 6.1)^2 = 0.3^2 = 0.09
(7.7 - 7.4)^2 = 0.3^2 = 0.09
(8.0 - 8.0)^2 = 0^2 = 0
(9.2 - 8.7)^2 = 0.5^2 = 0.25
(10.8 - 10.9)^2 = (-0.1)^2 = 0.01

Soma dos erros quadráticos:
0.0225 + 0.01 + 0 + 0.09 + 0.04 + 0.09 + 0.09 + 0 + 0.25 + 0.01 = 0.6025

EQM do Regressor Polinomial = 0.6025 / 10 = 0.06025

EQM do Ensemble

(0.75 - 0.7)^2 = 0.05^2 = 0.0025
(2.0 - 2.0)^2 = 0^2 = 0
(3.8 - 3.8)^2 = 0^2 = 0
(4.2 - 4.25)^2 = (-0.05)^2 = 0.0025
(5.0 - 5.0)^2 = 0^2 = 0
(6.4 - 6.4)^2 = 0^2 = 0
(7.7 - 7.7)^2 = 0^2 = 0
(8.0 - 8.0)^2 = 0^2 = 0
(9.2 - 9.1)^2 = 0.1^2 = 0.01
(10.8 - 10.8)^2 = 0^2 = 0

Soma dos erros quadráticos:
0.0025 + 0 + 0 + 0.0025 + 0 + 0 + 0 + 0 + 0.01 + 0 = 0.015

EQM do Ensemble: 0.015 / 10 = 0.0015

Resposta Final:

MLP: 0.0435
Regressor Polinomial: 0.06025
Ensemble: 0.0015

3) Estudo Comparativo entre MLP e Regressor Linear (Dataset Boston House-price)

Etapas do Código:

3.1 Carregamento do Dataset:

a. O código inicia carregando o dataset "Boston House-price" diretamente de uma URL externa (`http://lib.stat.cmu.edu/datasets/boston/boston_housing.data`) usando `pandas.read_csv`. Isso contorna a remoção do `load_boston()` em versões mais recentes do scikit-learn. As colunas são nomeadas e a variável alvo (MEDV) é identificada.

3.2 Pré-processamento:

b. Foi aplicada a **Padronização (StandardScaler)** aos atributos de entrada (X). Essa etapa é crucial para a MLP (Redes Neurais) para garantir que todos os atributos contribuam igualmente para o aprendizado e que a otimização por gradiente seja mais eficiente. Embora não estritamente necessária para a `LinearRegression`, é uma boa prática geral. O atributo CHAS já é lido como numérico, então não foi necessária uma conversão explícita.

3.3 Configuração dos Modelos:

c. **Regressor Linear (`LinearRegression`):** Um modelo simples que busca a melhor reta para ajustar os dados.
d. **MLP Regressor (`MLPRegressor`):** Uma Rede Neural Multi-layer Perceptron. Foi configurada com duas camadas ocultas (100 e 50 neurônios), função de ativação ReLU e otimizador Adam, com um limite de 1000 iterações e *early stopping* para evitar overfitting.

3.4 Avaliação (Validação Cruzada 10-Fold):

e. O desempenho de ambos os modelos foi avaliado usando **Validação Cruzada K-Fold com 10 pastas**. Isso significa que o dataset foi dividido em 10 partes; o modelo foi treinado 10 vezes, a cada vez usando 9 partes para treino e 1 para teste. Isso fornece uma estimativa mais robusta e menos enviesada do desempenho do modelo em dados não vistos.
f. A métrica de avaliação utilizada foi o **Erro Absoluto Médio (MAE)**, que mede a média das diferenças absolutas entre os valores preditos e os valores reais, nas mesmas unidades da variável alvo (preço das casas).

Conclusão:

Com base nos resultados da validação cruzada que obtivemos:

- **Regressor Linear: MAE Médio = 3.3767 (+/- 0.2944)**
- **MLP Regressor: MAE Médio = 3.0097 (+/- 0.3820)**

Observamos que o MLP Regressor obteve um Erro Absoluto Médio (MAE) ligeiramente menor (3.0097) em comparação com o Regressor Linear (3.3767). Isso indica que, para o dataset Boston House-price e com as configurações utilizadas, a MLP foi capaz de fazer previsões, em média, mais próximas dos valores reais.

O desvio padrão do MAE (por exemplo, +/- 0.3820 para MLP) mostra a variabilidade do desempenho do modelo entre as diferentes pastas da validação cruzada. Embora a MLP tenha um MAE médio menor, seu desvio padrão é um pouco maior que o do Regressor Linear, sugerindo uma consistência ligeiramente menor entre as diferentes divisões dos dados. No entanto, a diferença no MAE médio é relevante e o MAE menor da MLP a torna a vencedora

nesta comparação específica.

A superioridade da MLP sugere que a relação entre os atributos de entrada e o preço das casas no dataset Boston House-price possui certa complexidade não linear que o Regressor Linear (um modelo puramente linear) não consegue capturar tão bem. A capacidade das camadas ocultas da rede neural em aprender padrões complexos e não lineares permite que ela se ajuste melhor aos dados.

Em resumo, o estudo demonstra que, para este problema de regressão, a **MLP é o modelo de melhor desempenho** na previsão dos preços das casas, superando o Regressor Linear.

Codigo utilizado em Python:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_absolute_error

# --- 1. Carregamento do Dataset Boston House-price
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
target = raw_df.values[1::2, 2]

X = data
y = target
feature_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
'PTRATIO', 'B', 'LSTAT']

df = pd.DataFrame(X, columns=feature_names)
df['MEDV'] = y

print("Dataset Boston House-price carregado com sucesso a partir de URL externa.")
print(f"\nShape do Dataset: {df.shape}")
print("Primeiras 5 linhas do Dataset:")
print(df.head())

# --- 2. Pré-processamento

# Estratégia de pré-processamento: Padronização (StandardScaler)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
print("\nEtapa de Pré-processamento: Padronização (StandardScaler) aplicada aos atributos de entrada.")

# --- 3. Configuração dos Modelos

# Regressor Linear
linear_reg_model = LinearRegression()

# MLP Regressor (Multi-layer Perceptron)
mlp_reg_model = MLPRegressor(hidden_layer_sizes=(100, 50), activation='relu', solver='adam',
                             max_iter=1000, random_state=42, early_stopping=True, n_iter_no_change=10)

print("\nModelos configurados: Regressor Linear e MLP Regressor.")
print(f"Parâmetros do MLPRegressor: {mlp_reg_model.get_params()}")

# --- 4. Avaliação com Validação Cruzada (10-fold Cross-Validation)

n_splits = 10
kf = KFold(n_splits=n_splits, shuffle=True, random_state=42) # Shuffle para embaralhar os dados

mae_linear_reg = []
mae_mlp_reg = []

print(f"\nIniciando Validação Cruzada ({n_splits}-fold)...")

for fold, (train_index, test_index) in enumerate(kf.split(X_scaled)):
    X_train, X_test = X_scaled[train_index], X_scaled[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Treinamento e Avaliação do Regressor Linear
    linear_reg_model.fit(X_train, y_train)
    y_pred_linear = linear_reg_model.predict(X_test)
    mae_linear_reg.append(mean_absolute_error(y_test, y_pred_linear))

    # Treinamento e Avaliação do MLP Regressor
    mlp_reg_model.fit(X_train, y_train)
    y_pred_mlp = mlp_reg_model.predict(X_test)
    mae_mlp_reg.append(mean_absolute_error(y_test, y_pred_mlp))

    print(f"Fold {fold+1}/{n_splits} - MAE Linear Reg: {mae_linear_reg[-1]:.4f}, MAE MLP Reg: {mae_mlp_reg[-1]:.4f}")
```

```
# Calcular a média dos MAEs para cada modelo
mean_mae_linear = np.mean(mae_linear_reg)
std_mae_linear = np.std(mae_linear_reg)

mean_mae_mlp = np.mean(mae_mlp_reg)
std_mae_mlp = np.std(mae_mlp_reg)

print("\n--- Resultados Finais da Validação Cruzada ---")
print(f"Regressor Linear: MAE Médio = {mean_mae_linear:.4f} (+/- {std_mae_linear:.4f})")
print(f"MLP Regressor: MAE Médio = {mean_mae_mlp:.4f} (+/- {std_mae_mlp:.4f})")

# --- 5. Obter o Modelo (Equação) do Melhor Regressor Linear
if mean_mae_linear < mean_mae_mlp:
    print("\n--- Modelo (Equação) do Regressor Linear (Melhor Desempenho) ---")
    linear_reg_model.fit(X_scaled, y)
    coefficients = linear_reg_model.coef_
    intercept = linear_reg_model.intercept_

    equation_parts = [f"{intercept:.4f}"]
    for i, coef in enumerate(coefficients):
        equation_parts.append(f" + ({coef:.4f} * {feature_names[i]})")

    print("Equação: MEDV =", "".join(equation_parts).replace(" + -", " - ")) # Ajusta a
exibição de coeficientes negativos
else:
    print("\nO Regressor Linear não foi o modelo com o melhor desempenho médio nesta
comparação.")
```

Resultado:

```
Dataset Boston House-price carregado com sucesso a partir de URL externa.

Shape do Dataset: (506, 14)
Primeiras 5 linhas do Dataset:
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0

   PTRATIO       B  LSTAT  MEDV
0     15.3  396.90   4.98  24.0
1     17.8  396.90   9.14  21.6
2     17.8  392.83   4.03  34.7
3     18.7  394.63   2.94  33.4
4     18.7  396.90   5.33  36.2

Etapa de Pré-processamento: Padronização (StandardScaler) aplicada aos atributos de entrad

Modelos configurados: Regressor Linear e MLP Regressor.
Parâmetros do MLPRegressor: {'activation': 'relu', 'alpha': 0.0001, 'batch_size': 'auto',

Iniciando Validação Cruzada (10-fold)...
Fold 1/10 - MAE Linear Reg: 2.8342, MAE MLP Reg: 3.1844
Fold 2/10 - MAE Linear Reg: 3.4412, MAE MLP Reg: 2.5048
Fold 3/10 - MAE Linear Reg: 3.1200, MAE MLP Reg: 2.3350
Fold 4/10 - MAE Linear Reg: 3.7109, MAE MLP Reg: 2.9349
Fold 5/10 - MAE Linear Reg: 3.8221, MAE MLP Reg: 3.7633
Fold 6/10 - MAE Linear Reg: 3.2136, MAE MLP Reg: 2.8396
Fold 7/10 - MAE Linear Reg: 3.7226, MAE MLP Reg: 3.0249
Fold 8/10 - MAE Linear Reg: 3.4043, MAE MLP Reg: 3.2054
Fold 9/10 - MAE Linear Reg: 3.1702, MAE MLP Reg: 3.2881
Fold 10/10 - MAE Linear Reg: 3.3279, MAE MLP Reg: 3.0165

--- Resultados Finais da Validação Cruzada ---
Regressor Linear: MAE Médio = 3.3767 (+/- 0.2944)
MLP Regressor: MAE Médio = 3.0097 (+/- 0.3820)

O Regressor Linear não foi o modelo com o melhor desempenho médio nesta comparação.
```

Detalhes das linhas:

Etapa de Pré-processamento: Padronização (StandardScaler) aplicada aos atributos de entrada.
Modelos configurados: Regressor Linear e MLP Regressor.

Parâmetros do MLPRegressor: {'activation': 'relu', 'alpha': 0.0001, 'batch_size': 'auto', 'beta_1': 0.9, 'beta_2': 0.999, 'early_stopping': True, 'epsilon': 1e-08, 'hidden_layer_sizes': (100, 50), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_fun': 15000, 'max_iter': 1000, 'momentum': 0.9, 'n_iter_no_change': 10, 'nesterovs_momentum': True, 'power_t': 0.5, 'random_state': 42, 'shuffle': True, 'solver': 'adam', 'tol': 0.0001, 'validation_fraction': 0.1, 'verbose': False, 'warm_start': False}