HOMEWORK #3

APRENDIZAGEM
LEIC-A 2023/2024

Grupo #24
- Gonçalo Alves (Nº 103540)
- Daniel Nunes (Nº 103095)

I. Pen and Paper

1)

	y_1	y_2	z
x_1	0.7	-0.3	0.8
x_2	0.4	0.5	0.6
x_3	-0.2	0.8	0.3
x_4	-0.4	0.3	0.3

a) Tendo em conta que é feita uma transformação nos dados originais com base na função de base radial, podemos então calcular a matriz X (design matrix) destes valores transformados.

$$\phi_j(x) = e^{-\frac{\|x - c_j\|^2}{2}}$$

$$\phi_1(x_1) = e^{-\frac{\left\|\begin{pmatrix}0.7\\-0.3\end{pmatrix} - \begin{pmatrix}0\\0\end{pmatrix}\right\|^2}{2}} = e^{-\frac{0.7^2 + (-0.3)^2}{2}} = 0.74826$$

$$\phi_1(x_2) = e^{-\frac{\left\|\begin{pmatrix}0.4\\0.5\end{pmatrix} - \begin{pmatrix}0\\0\end{pmatrix}\right\|^2}{2}} = e^{-\frac{0.4^2 + 0.5^2}{2}} = 0.81465$$

$$\phi_1(x_3) = e^{-\frac{\left\|\begin{pmatrix}-0.2\\0.8\end{pmatrix} - \begin{pmatrix}0\\0\end{pmatrix}\right\|^2}{2}} = e^{-\frac{(-0.2)^2 + 0.8^2}{2}} = 0.71177$$

$$\phi_1(x_4) = e^{-\frac{\left\|\begin{pmatrix}-0.4\\0.3\end{pmatrix} - \begin{pmatrix}0\\0\end{pmatrix}\right\|^2}{2}} = e^{-\frac{(-0.4)^2 + 0.3^2}{2}} = 0.88250$$

$$\phi_2(x_1) = e^{-\frac{\left\|\binom{0.7}{-0.3}-\binom{1}{-1}\right\|}{2}} = e^{-\frac{(-0.3)^2+0.7^2}{2}} = 0.74826$$

$$\phi_2(x_2) = e^{-\frac{\left\|\binom{0.4}{0.5}-\binom{1}{-1}\right\|^2}{2}} = e^{-\frac{(-0.6)^2+1.5^2}{2}} = 0.27117$$

$$\phi_2(x_3) = e^{-\frac{\left\|\binom{-0.2}{0.8}-\binom{1}{-1}\right\|^2}{2}} = e^{-\frac{(-1.2)^2+1.8^2}{2}} = 0.09633$$

$$\phi_2(x_4) = e^{-\frac{\left\|\binom{-0.4}{0.3}-\binom{1}{-1}\right\|^2}{2}} = e^{-\frac{(-1.4)^2+1.3^2}{2}} = 0.16122$$

$$\phi_3(x_1) = e^{-\frac{\left\|\binom{0.7}{-0.3}-\binom{-1}{1}\right\|}{2}} = e^{-\frac{1.7^2+(-1.3)^2}{2}} = 0.10127$$

$$\phi_3(x_2) = e^{-\frac{\left\|\binom{0.4}{0.5}-\binom{-1}{1}\right\|^2}{2}} = e^{-\frac{1.4^2+(-0.5)^2}{2}} = 0.33121$$

$$\phi_3(x_3) = e^{-\frac{\left\|\binom{-0.2}{0.8}-\binom{-1}{1}\right\|^2}{2}} = e^{-\frac{0.8^2+(-0.2)^2}{2}} = 0.71177$$

$$\phi_3(x_4) = e^{-\frac{\left\|\binom{-0.4}{0.3}-\binom{-1}{1}\right\|^2}{2}} = e^{-\frac{0.6^2+(-0.7)^2}{2}} = 0.65376$$

Logo, a design matrix é igual a:

$$X = \begin{bmatrix} 1 & \phi_1(x_1) & \phi_2(x_1) & \phi_3(x_1) \\ 1 & \phi_1(x_2) & \phi_2(x_2) & \phi_3(x_2) \\ 1 & \phi_1(x_3) & \phi_2(x_3) & \phi_3(x_3) \\ 1 & \phi_1(x_4) & \phi_2(x_4) & \phi_3(x_4) \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0.74826 & 0.74826 & 0.10127 \\ 1 & 0.81465 & 0.27117 & 0.33121 \\ 1 & 0.71177 & 0.09633 & 0.71177 \\ 1 & 0.88250 & 0.16122 & 0.65376 \end{bmatrix}$$

Podemos então calcular os coeficientes da regressão usando a fórmula do cálculo da Regressão de Ridge.

$$w = (X^T \cdot X + \lambda \cdot I)^{-1} \cdot X^T \cdot z =$$

$$= \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.74826 & 0.81465 & 0.71177 & 0.88250 \\ 0.74826 & 0.27117 & 0.09633 & 0.16122 \\ 0.10127 & 0.33121 & 0.71177 & 0.65376 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0.74826 & 0.74826 & 0.10127 \\ 1 & 0.81465 & 0.27117 & 0.33121 \\ 1 & 0.71177 & 0.09633 & 0.71177 \\ 1 & 0.88250 & 0.16122 & 0.65376 \end{bmatrix} + \right.$$

$$\left. + \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.74826 & 0.81465 & 0.71177 & 0.88250 \\ 0.74826 & 0.27117 & 0.09633 & 0.16122 \\ 0.10127 & 0.33121 & 0.71177 & 0.65376 \end{bmatrix} \cdot \begin{bmatrix} 0.8 \\ 0.6 \\ 0.3 \\ 0.3 \end{bmatrix} =$$

$$= \begin{bmatrix} -0.17800 & -0.18666 & 1.81971 & -0.25829 \\ -0.18666 & 2.28561 & -1.90060 & -0.81733 \\ 1.81971 & -1.90060 & -0.89615 & 0.89889 \\ -0.25829 & -0.81733 & 0.89889 & 2.20302 \end{bmatrix} \cdot$$

$$\cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.74826 & 0.81465 & 0.71177 & 0.88250 \\ 0.74826 & 0.27117 & 0.09633 & 0.16122 \\ 0.10127 & 0.33121 & 0.71177 & 0.65376 \end{bmatrix} \cdot \begin{bmatrix} 0.8 \\ 0.6 \\ 0.3 \\ 0.3 \end{bmatrix} =$$

$$= \begin{bmatrix} 0.33914 \\ 0.19945 \\ 0.40096 \\ -0.29600 \end{bmatrix}$$

Assim, a regressão de Ridge que foi computada através da transformação dos valores originais tem a seguinte expressão:

$$\hat{z}(x) = 0.33914 + 0.19945\,\phi_1(x) + 0.40096\,\phi_2(x) - 0.29600\,\phi_3(x).$$

(equivalente a $\hat{z} = X \cdot w$)

① b) Antes de calcularmos o valor da RMSE, temos de calcular os valores estimados de cada observação de treino:

$$\hat{z} = \begin{bmatrix} 1 & 0.74826 & 0.74826 & 0.10127 \\ 1 & 0.81465 & 0.27117 & 0.33121 \\ 1 & 0.71177 & 0.09633 & 0.71177 \\ 1 & 0.88250 & 0.16122 & 0.65376 \end{bmatrix} \begin{bmatrix} 0.33914 \\ 0.19945 \\ 0.40096 \\ -0.29600 \end{bmatrix} =$$

$$= \begin{bmatrix} 0.75844 \\ 0.51232 \\ 0.30905 \\ 0.38628 \end{bmatrix}$$

A Root Mean Square Error (RMSE) é dada pela seguinte expressão:

$$RMSE(\hat{z}, z) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (z_i - \hat{z}_i)^2}$$

Logo, ao usarmos os 4 dados de treino:

$$RMSE(\hat{z}, z) = \sqrt{\frac{1}{4} \times \left((0.8-0.75844)^2 + (0.6-0.51232)^2 + (0.3-0.30905)^2 + (0.3-0.38628)^2 \right)} =$$

$$= \underline{\underline{0.06508}}$$

② $W^1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$, $b^1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, $W^2 = \begin{bmatrix} 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, $b^2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$,

$W^3 = \begin{bmatrix} 1 & 1 \\ 3 & 1 \\ 1 & 1 \end{bmatrix}$, $b^3 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

$$f(x) = \frac{e^{0.5x-2} - e^{-0.5x+2}}{e^{0.5x-2} + e^{-0.5x+2}} = \tanh(0.5x-2), \quad loss: \frac{1}{2}\|z - \hat{z}\|_2^2$$

- Perform one batch gradient descent update, learning rate $\eta = 0.1$

for training obs.: $X_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$ $X_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix}$

B A

- Forward propagation

X1

$z^1 = W^1 \cdot X_1 + b^1 = \begin{bmatrix} 5 \\ 6 \\ 5 \end{bmatrix}$

$z^2 = W^2 \cdot f(z^1) + b^2 = \begin{bmatrix} 4.971 \\ 2.686 \end{bmatrix}$

$\begin{bmatrix} 0.462 \\ 0.762 \\ 0.462 \end{bmatrix}$

$z^3 = W^3 \cdot f(z^2) + b^3 = \begin{bmatrix} 0.874 \\ 1.775 \\ 0.874 \end{bmatrix}$

$\begin{bmatrix} 0.450 \\ 0.576 \end{bmatrix}$

X2

$z^1 = W^1 \cdot X_2 + b^1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

$z^2 = W^2 \cdot f(z^1) + b^2 = \begin{bmatrix} -4.431 \\ -1.715 \end{bmatrix}$

$\begin{bmatrix} -0.905 \\ -0.905 \\ -0.905 \end{bmatrix}$

$z^3 = W^3 \cdot f(z^2) + b^3 = \begin{bmatrix} -0.993 \\ -2.992 \\ -0.993 \end{bmatrix}$

$\begin{bmatrix} -0.999 \\ -0.993 \end{bmatrix}$

- $$\frac{\partial(f(x))}{\partial x} = \frac{\partial(\tanh(0.5x-2))}{\partial x} = \operatorname{sech}^2(0.5x-2) \cdot 0.5$$

- $\operatorname{sech}^2(0.5 z_1^1 - 2) \cdot 0.5 = \begin{bmatrix} 0.393 \\ 0.210 \\ 0.393 \end{bmatrix}$
- $\operatorname{sech}^2(0.5 z_2^1 - 2) \cdot 0.5 = \begin{bmatrix} 0.090 \\ 0.090 \\ 0.090 \end{bmatrix}$

- $\operatorname{sech}^2(0.5 z_1^2 - 2) \cdot 0.5 = \begin{bmatrix} 0.399 \\ 0.334 \end{bmatrix}$
- $\operatorname{sech}^2(0.5 z_2^2 - 2) \cdot 0.5 = \begin{bmatrix} 0.0004 \\ 0.007 \end{bmatrix}$

- $\operatorname{sech}^2(0.5 z_1^3 - 2) \cdot 0.5 = \begin{bmatrix} 0.081 \\ 0.176 \\ 0.081 \end{bmatrix}$
- $\operatorname{sech}^2(0.5 z_2^3 - 2) \cdot 0.5 = \begin{bmatrix} 0.134 \\ 0.002 \\ 0.013 \end{bmatrix}$

- $T1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$

- $T2 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$

- $f(z_1^3) = \begin{bmatrix} -0.916 \\ -0.805 \\ -0.916 \end{bmatrix}$
- $f(z_2^3) = \begin{bmatrix} -0.987 \\ -0.998 \\ -0.987 \end{bmatrix}$

- $f(z_1^3) - T_1 = \begin{bmatrix} 0.084 \\ -1.805 \\ 0.084 \end{bmatrix}$
- $f(z_2^3) - T_2 = \begin{bmatrix} -1.987 \\ 0.002 \\ 0.013 \end{bmatrix}$

$$\bullet \text{Delta3}_{x_1}: \left(f(z^3_1) - T1\right) \cdot \left(\operatorname{sech}^2(0.5\, z^3_1 - 2) \cdot 0.5\right) = \begin{bmatrix} 0.0067 \\ -0.3177 \\ 0.0067 \end{bmatrix}$$

$$\bullet \text{Delta3}_{x_2}: \left(f(z^3_2) - T2\right) \cdot \left(\operatorname{sech}^2(0.5\, z^3_2 - 2) \cdot 0.5\right) = \begin{bmatrix} -0.0266 \\ 3.369e^{-6} \\ 1.804e^{-4} \end{bmatrix}$$

$$\bullet \text{Delta2}_{x_1}: \Big(\underbrace{W^{3T} \cdot \text{Delta3}_{x_1}}_{\begin{bmatrix} -0.937 \\ -0.304 \end{bmatrix}} \Big) \cdot \left(\operatorname{sech}^2(0.5\, z^2_1 - 2) \cdot 0.5\right) = \begin{bmatrix} -0.374 \\ -0.102 \end{bmatrix}$$

$$\bullet \text{Delta2}_{x_2}: \Big(\underbrace{W^{3T} \cdot \text{Delta3}_{x_2}}_{\begin{bmatrix} -0.02640 \\ -0.02641 \end{bmatrix}} \Big) \cdot \left(\operatorname{sech}^2(0.5\, z^2_2 - 2) \cdot 0.5\right) = \begin{bmatrix} -1.1509e^{-5} \\ -1.7289e^{-4} \end{bmatrix}$$

$$\bullet \text{Delta1}_{x_1}: \Big(\underbrace{W^{2T} \cdot \text{Delta2}_{x_1}}_{\begin{bmatrix} -0.476 \\ -1.599 \\ -0.476 \end{bmatrix}} \Big) \cdot \left(\operatorname{sech}^2(0.5\, z^1_1 - 2) \cdot 0.5\right) = \begin{bmatrix} -0.187 \\ -0.336 \\ -0.187 \end{bmatrix}$$

$$\bullet \text{Delta1}_{x_2}: \Big(\underbrace{W^{2T} \cdot \text{Delta2}_{x_2}}_{\begin{bmatrix} -0.00018 \\ -0.00022 \\ -0.00018 \end{bmatrix}} \Big) \cdot \left(\operatorname{sech}^2(0.5\, z^1_2 - 2) \cdot 0.5\right) = \begin{bmatrix} -1.666e^{-5} \\ -1.978e^{-5} \\ -1.666e^{-5} \end{bmatrix}$$

• Update weights

$$\overset{(1)}{\text{Delta1}_{x_1}} \cdot X_1^T = \begin{bmatrix} -0.187 & -0.187 & -0.187 & -0.187 \\ -0.336 & -0.336 & -0.336 & -0.336 \\ -0.187 & -0.187 & -0.187 & -0.187 \end{bmatrix}$$

$$\overset{(2)}{\text{Delta1}_{x_2}} \cdot x_2^T = \begin{bmatrix} -1.666e^{-5} & 0 & 0 & 1.666e^{-5} \\ -1.978e^{-5} & 0 & 0 & 1.978e^{-5} \\ -1.666e^{-5} & 0 & 0 & 1.666e^{-5} \end{bmatrix}$$

$$W^1 = W^1 - \left(\eta \cdot \left((1) + (2)\right)\right) = \begin{bmatrix} 1.01872 & 1.018719 & 1.018719 & 1.018717 \\ 1.033589 & 1.033587 & 2.033587 & 1.033585 \\ 1.01872 & 1.018719 & 1.018719 & 1.018717 \end{bmatrix}$$

$$\overset{(1)}{\text{Delta2}_{x_1}} \cdot f(z^1_1)^T = \begin{bmatrix} -0.1773 & -0.2852 & -0.17305 \\ -0.0469 & -0.0773 & -0.0469 \end{bmatrix}$$

$$\overset{(2)}{\text{Delta2}_{x_2}} \cdot f(z^1_2)^T = \begin{bmatrix} 1.0418e^{-5} & 1.0418e^{-5} & 1.0418e^{-5} \\ 1.565e^{-4} & 1.565e^{-4} & 1.565e^{-4} \end{bmatrix}$$

$$W^2 = W^2 - (\eta \cdot (① + ②)) = \begin{bmatrix} 1.0173 & 4.0285 & 1.0173 \\ 1.0047 & 1.0077 & 1.0047 \end{bmatrix}$$

$$\overset{①}{Delta3_{x_1}} \cdot f(Z_1^2)^T = \begin{bmatrix} 0.00305 & -0.0039405 \\ -0.14313 & 0.18314 \\ 0.00305 & -0.0039055 \end{bmatrix}$$

$$\overset{②}{Delta3_{x_2}} \cdot f(Z_2^2)^T = \begin{bmatrix} 2.658e^{-2} & 2.6421e^{-2} \\ -3.368e^{-6} & -3.3475e^{-6} \\ -1.8038e^{-4} & -1.7928e^{-4} \end{bmatrix}$$

$$W^3 = W^3 - (\eta \cdot (① + ②)) = \begin{bmatrix} 0.99703 & 0.99774 \\ 3.01431 & 0.98169 \\ 0.99971 & 1.0004 \end{bmatrix}$$

• Update biases

$$b1 = b1 - (\eta \cdot (delta1_{x_1} + delta1_{x_2})) = \begin{bmatrix} 1.0187 \\ 1.0336 \\ 1.0187 \end{bmatrix}$$

$$b2 = b2 - (\eta \cdot (delta2_{x_1} + delta2_{x_2})) = \begin{bmatrix} 1.0374 \\ 1.01017 \end{bmatrix}$$

$$b3 = b3 - (\eta \cdot (delta3_{x_1} + delta3_{x_2})) = \begin{bmatrix} 1.00198 \\ 1.03177 \\ 0.9993 \end{bmatrix}$$

Homework 3 (Part II)

Aprendizagem 2023/2024 - LEIC @ IST

Group #24

- Daniel Nunes (Nº 103095)
- Gonçalo Alves (Nº 103540)

In [ ]:
```
import warnings
from sklearn.exceptions import ConvergenceWarning

# This code snippet ommits all ConvergenceWarnings related to # MLP regressors.
# To show these warnings, comment the line below, restart the jupyter kernel
# and re-run all code snippets from this notebook.
warnings.filterwarnings("ignore", category=ConvergenceWarning)
```

Data importing and preparation

In [ ]:
```
import pandas as pd

data = pd.read_csv("winequality-red.csv", delimiter=';')

X = data.drop("quality", axis=1)
y = data["quality"]

display(data)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0

1599 rows × 12 columns

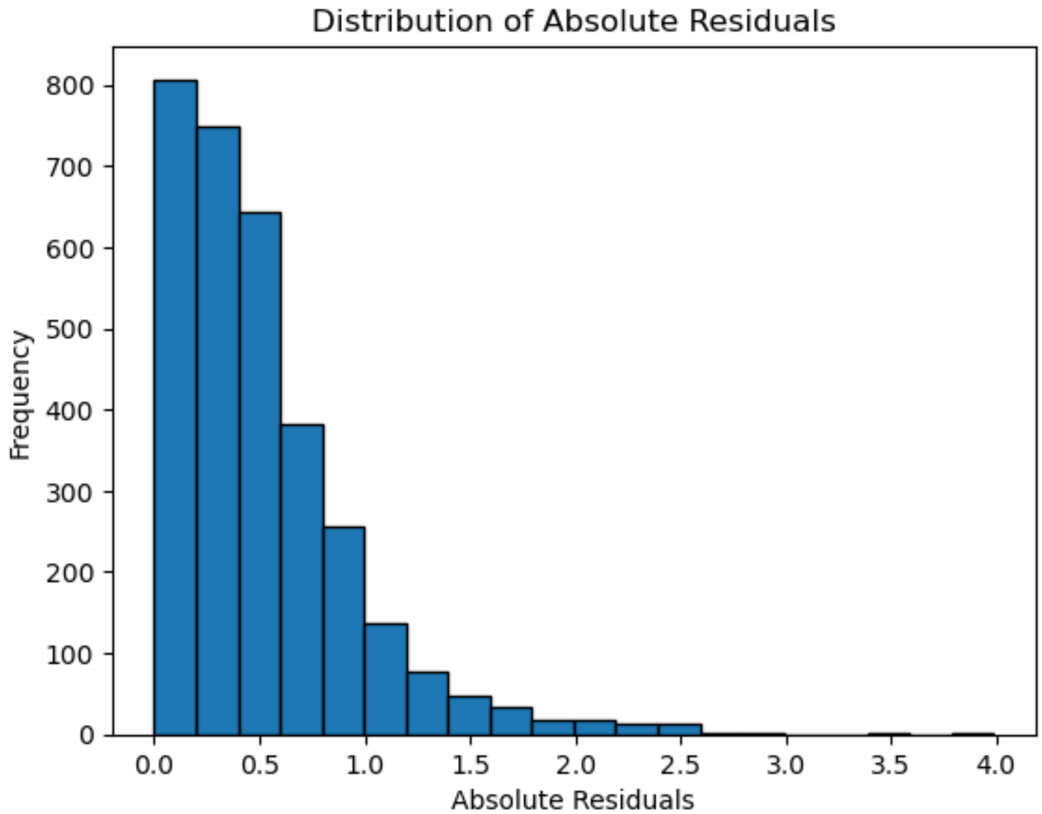
- Use a 80-20 training-test split with a fixed seed (random_state=0)
- Average the performance of each MLP from 10 runs (for reproducibility consider seeding the MLPs with random_state $\in$ {1..10}).

In [ ]:
```python
from sklearn.model_selection import train_test_split

random_seeds = range(1, 11)
X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size=0.2, random_state=0)
```

Exercise 1

Learn a MLP regressor with 2 hidden layers of size 10, rectifier linear unit activation on all nodes, and early stopping with 20% of training data set aside for validation. All remaining parameters (e.g., loss, batch size, regularization term, solver) should be set as default. Plot the distribution of the residues (in absolute value) using a histogram.

In [ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPRegressor

# This array will store the predictions of the MLP regressors of this exercise
# and will be used later on
y_seed_predictions = []

absolute_residuals = []
for random_seed in random_seeds:
    # Define MLPRegressor with 2 hidden layers of size 10, ReLU activation,
    # and early stopping
    mlp = MLPRegressor(
        hidden_layer_sizes=(10, 10),
        activation='relu',
        random_state=random_seed,
        early_stopping=True,
        validation_fraction=0.2
    )
    mlp.fit(X_train, y_train)

    # Create predictions for this set of observations
    y_pred = mlp.predict(X_test)
    y_seed_predictions.append(y_pred)

    # Calculate the absolute residuals for this run
    residuals = np.abs(y_test - y_pred)
    absolute_residuals.extend(residuals)


# Plot the distribution of absolute residuals using a histogram
plt.hist(absolute_residuals, bins=20, edgecolor='k')
plt.xlabel('Absolute Residuals')
plt.ylabel('Frequency')
plt.title('Distribution of Absolute Residuals')
plt.show()
```



Exercise 2

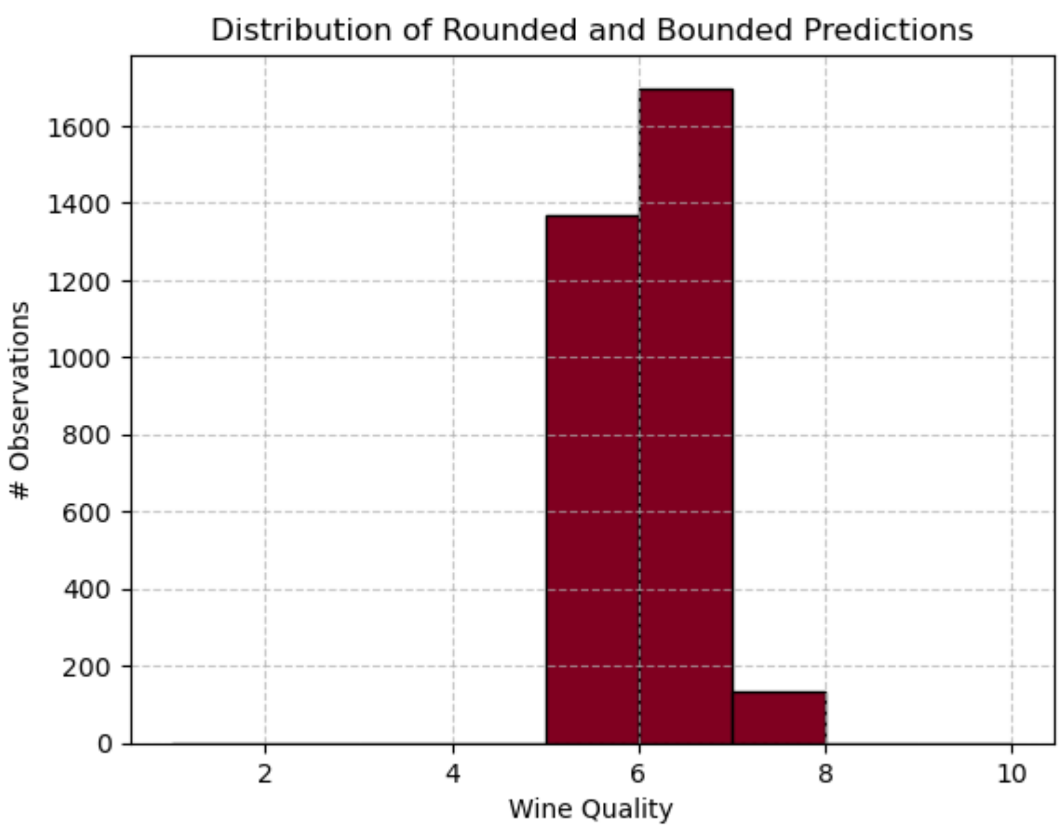
Since we are in the presence of a integer regression task, a recommended trick is to round and bound estimates. Assess the impact of these operations on the MAE of the MLP learnt in previous question.

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

# Function to round and bound estimates
def round_and_bound(predictions, lower_bound, upper_bound):
    return np.clip(np.round(predictions), lower_bound, upper_bound)

lower_bound = 1
upper_bound = 10

original_seed_mae = []
round_bound_seed_mae = []

y_pred_rounded_bounded_all = []

for y_pred in y_seed_predictions:
    # Calculate the original MAE for the unrounded predictions
    original_seed_mae.append(mean_absolute_error(y_test, y_pred))

    # Apply rounding and bounding to the predictions and calculate the MAE
    y_pred_rounded_bounded = round_and_bound(y_pred, lower_bound, upper_bound)
    y_pred_rounded_bounded_all.extend(y_pred_rounded_bounded)

    round_bound_seed_mae.append(mean_absolute_error(y_test, y_pred_rounded_bounded)

# Calculate the average MAE for both cases and print the results
```

```
print(f"Avg. MAE for Original Predictions: {np.mean(original_seed_mae)}")
print(f"Avg. MAE for Rounded and Bounded Predictions: " + \
      f"{np.mean(round_bound_seed_mae)}")
```

```
Avg. MAE for Original Predictions: 0.5097171955009515
Avg. MAE for Rounded and Bounded Predictions: 0.43875000000000003
```

In [ ]:
```
# Plot the distribution of rounded and bounded predictions
redwine_color = (128/255, 0/255, 32/255)  # RGB values should be between 0 and 1
plt.hist(y_pred_rounded_bounded_all, bins=np.arange(lower_bound, upper_bound+1), \
         edgecolor='k', color=redwine_color)

plt.xlabel('Wine Quality')
plt.ylabel('# Observations')
plt.title('Distribution of Rounded and Bounded Predictions')
plt.grid(True, linestyle='--', alpha=0.7)

plt.show()
```



Our answer

There is a slight decrease on the MAE when the round and bound method is applied to the model's predictions, hence the performance of this model improves when this method is applied.

Exercise 3

Similarly assess the impact on RMSE from replacing early stopping by a well-defined number of iterations in {20,50,100,200} (where one iteration corresponds to a batch).

In [ ]:
```
from sklearn.metrics import mean_squared_error
```

```python
# Calculate the original RSME for the predictions with early-stopping
original_rmse = 0
for y_pred in y_seed_predictions:
    original_rmse += np.sqrt(mean_squared_error(y_test, y_pred))

original_rmse = original_rmse / len(y_seed_predictions);
```

In [ ]:

```python
iterations = [20, 50, 100, 200]

# Train the model with different numbers of iterations
rmse_scores = []
for num_iterations in iterations:
    rmse_iteration_scores = []

    for random_seed in random_seeds:
        # Learn a MLP Regressor with the same characteristics of the exercise 1,
        # except early-stopping gets replaced by a fixed number of iterations
        mlp = MLPRegressor(
            hidden_layer_sizes=(10, 10),
            activation='relu',
            random_state=random_seed,
            max_iter=num_iterations
        )
        mlp.fit(X_train, y_train)
        y_pred = mlp.predict(X_test)

        rmse = np.sqrt(mean_squared_error(y_test, y_pred))
        rmse_iteration_scores.append(rmse)

    # Calculate the average RMSE for this number of iterations
    rmse_scores.append(np.mean(rmse_iteration_scores));

# Print the RMSE for each case
print(f"RMSE for early-stopping (20% validation): {original_rmse}")

for i, num_iterations in enumerate(iterations):
    print(f"RMSE for {num_iterations} iterations: {rmse_scores[i]}")
```

```
RMSE for early-stopping (20% validation): 0.6706527958221328
RMSE for 20 iterations: 1.4039789509925442
RMSE for 50 iterations: 0.7996073631460567
RMSE for 100 iterations: 0.6940361469112144
RMSE for 200 iterations: 0.6554543932216472
```

Exercise 4

Critically comment the results obtained in previous question, hypothesizing at least one reason why early stopping favors and/or worsens performance.

Our answer

We can conclude from this data that, as the number of maximum iterations for training a model increases, its RMSE values decrease, meaning that the model gets more accurate as the amount of training increases.

However, too many iterations may start to cause overfitting on our model, meaning that, while it follows its predictions closer to the training data, it can be more error-prone when evaluating completely new sets of data. Hence, the early-stopping strategy tries to strike a balance between the model's accuracy to the training data and the possibility of overfitting, favouring performance in most cases. This might be the reason why its RMSE value is slightly larger than the one calculated from the model that stops training after 200 iterations.