Aprendizagem 2023/24

Homework III – Group 001

(ist1102082, ist1103252)

I. Pen-and-paper

1)

1. a)

Observations: $\left\{ \begin{pmatrix} 0.7 \\ -0.3 \end{pmatrix}_{x_1}, \begin{pmatrix} 0.4 \\ 0.5 \end{pmatrix}_{x_2}, \begin{pmatrix} -0.2 \\ 0.8 \end{pmatrix}_{x_3}, \begin{pmatrix} -0.4 \\ 0.3 \end{pmatrix}_{x_4} \right\}$

Targets: $(0.8_{z_1}, 0.6_{z_2}, 0.3_{z_3}, 0.3_{z_4})$

Data points: $\left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}_{c_1}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}_{c_2}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}_{c_3} \right\}$

$$\phi_j(x) = \exp\left(-\frac{\|x - c_j\|^2}{2}\right)$$

$\lambda = 0.1$ (regularization term)

Using an auxiliar python script, we can calculate the $\phi_j(x)$ for all observations:

- $\phi_1(x_1) = \exp\left(-\frac{\left\| \begin{pmatrix} 0.7 \\ -0.3 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right\|^2}{2}\right) = 0.74826$; $\phi_2(x_1) = 0.74826$; $\phi_3(x_1) = 0.10127$
- $\phi_1(x_2) = 0.81465$; $\phi_2(x_2) = 0.27717$; $\phi_3(x_2) = 0.33121$
- $\phi_1(x_3) = 0.71177$; $\phi_2(x_3) = 0.09633$; $\phi_3(x_3) = 0.71177$
- $\phi_1(x_4) = 0.88250$; $\phi_2(x_4) = 0.16122$; $\phi_3(x_4) = 0.65377$

Aprendizagem 2023/24

Homework III – Group 001

(ist1102082, ist1103252)

$$\Phi = \begin{bmatrix} 1 & 0.74826 & 0.74826 & 0.10127 \\ 1 & 0.81465 & 0.27117 & 0.33121 \\ 1 & 0.71177 & 0.09633 & 0.71177 \\ 1 & 0.88250 & 0.16122 & 0.65377 \end{bmatrix}$$

Closed solution Ridge regression:

$$\rightarrow W = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T z$$

$$\Phi^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.74826 & 0.81465 & 0.71177 & 0.88250 \\ 0.74826 & 0.27117 & 0.09633 & 0.16122 \\ 0.10127 & 0.33121 & 0.71177 & 0.65377 \end{bmatrix}$$

$$\Phi^T\Phi = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.74826 & 0.81465 & 0.71177 & 0.88250 \\ 0.74826 & 0.27117 & 0.09633 & 0.16122 \\ 0.10127 & 0.33121 & 0.71177 & 0.65377 \end{bmatrix} \times \begin{bmatrix} 1 & 0.74826 & 0.74826 & 0.10127 \\ 1 & 0.81465 & 0.27117 & 0.33121 \\ 1 & 0.71177 & 0.09633 & 0.71177 \\ 1 & 0.88250 & 0.16122 & 0.65377 \end{bmatrix} =$$

$$= \begin{bmatrix} 4 & 3.15718 & 1.27699 & 1.79802 \\ 3.15718 & 2.50897 & 0.99165 & 1.42916 \\ 1.27698 & 0.99165 & 0.66870 & 0.33955 \\ 1.79802 & 1.42916 & 0.33955 & 1.06399 \end{bmatrix}$$

$$\lambda I = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$$

Aprendizagem 2023/24

Homework III – Group 001

(ist1102082, ist1103252)

$$\Phi^T\Phi + \lambda I = \begin{bmatrix} 4.1 & 3.15718 & 1.27699 & 1.79802 \\ 3.15718 & 2.60897 & 0.99165 & 1.42916 \\ 1.27698 & 0.99165 & 0.76870 & 0.33955 \\ 1.79802 & 1.42916 & 0.33955 & 1.16399 \end{bmatrix}$$

$$(\Phi^T\Phi + \lambda I)^{-1} \approx \begin{bmatrix} 4.54826 & -3.77682 & -1.86117 & -1.86155 \\ -3.77681 & 5.99285 & -0.88543 & -1.26432 \\ -1.86117 & -0.88543 & 4.33276 & 2.72156 \\ -1.86155 & -1.26432 & 2.72156 & 4.53204 \end{bmatrix}$$

pseudo-inverse Matrix (Moore-Penrose Matrix)

$$z = \begin{bmatrix} 0.8 \\ 0.6 \\ 0.3 \\ 0.3 \end{bmatrix}$$

$$\Phi^T z = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.74826 & 0.81465 & 0.71177 & 0.88250 \\ 0.74826 & 0.27117 & 0.09633 & 0.16122 \\ 0.10127 & 0.33121 & 0.71177 & 0.65377 \end{bmatrix} \begin{bmatrix} 0.8 \\ 0.6 \\ 0.3 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 2.00000 \\ 1.56568 \\ 0.83898 \\ 0.68940 \end{bmatrix}$$

$$W = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T z \approx$$

$$= \begin{bmatrix} 4.54826 & -3.77682 & -1.86117 & -1.86155 \\ -3.77681 & 5.99285 & -0.88543 & -1.26432 \\ -1.86117 & -0.88543 & 4.33276 & 2.72156 \\ -1.86155 & -1.26432 & 2.72156 & 4.53204 \end{bmatrix} \begin{bmatrix} 2.00000 \\ 1.56568 \\ 0.83898 \\ 0.68940 \end{bmatrix} = \begin{bmatrix} 0.33914 \\ 0.19945 \\ 0.40096 \\ -0.29600 \end{bmatrix}$$

The Ridge regression is: $y(x) = 0.33914 + 0.19945x_1 + 0.40096x_2 - 0.29600x_3$ //

Aprendizagem 2023/24

Homework III – Group 001

(ist1102082, ist1103252)

b)

$$RMSE(\hat{z}, z) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(z_i - \hat{z}_i)^2} \quad , \text{with } z = \begin{bmatrix} 0.8 \\ 0.6 \\ 0.3 \\ 0.3 \end{bmatrix}$$

$$\hat{z} = \phi w =$$

$$= \begin{bmatrix} 1 & 0.74826 & 0.74826 & 0.10127 \\ 1 & 0.81465 & 0.27117 & 0.33721 \\ 1 & 0.71177 & 0.09633 & 0.71177 \\ 1 & 0.88250 & 0.16122 & 0.65377 \end{bmatrix} \begin{bmatrix} 0.33914 \\ 0.19945 \\ 0.40096 \\ -0.29600 \end{bmatrix} = \begin{bmatrix} 0.75844 \\ 0.51232 \\ 0.30905 \\ 0.38629 \end{bmatrix}$$

$$RMSE(\hat{z}, z) = \sqrt{\frac{1}{4}\sum_{i=1}^{4}(z_i - \hat{z}_i)^2} = 0.06508$$

2)

2.

- MLP classifier of three outcomes: A, B, C
- $W^{[1]} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$, $b^{[1]} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, $W^{[2]} = \begin{bmatrix} 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, $b^{[2]} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $W^{[3]} = \begin{bmatrix} 1 & 1 \\ 3 & 1 \\ 1 & 1 \end{bmatrix}$, $b^{[3]} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$
- Activation function: $f(x) = \frac{e^{0.5x-2} - e^{-0.5x+2}}{e^{0.5x-2} + e^{-0.5x+2}} = \tanh(0.5x-2)$

$$f'(x) = \frac{0.5}{\cosh^2(2-(0.5x))}$$

- Squared error loss: $\frac{1}{2}\|z - \hat{z}\|^2$
- $\eta = 0.1$
- $t_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$
- Observations $x_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$; $x_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix}$
- $t_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

TÉCNICO LISBOA

Aprendizagem 2023/24

Homework III – Group 001

(ist1102082, ist1103252)

Starting with x_1: (We will be using programming functions to help us with heavier calculations)

$\rightarrow a^{[1](1)} = W^{[1]} \cdot x_1 + b^{[1]} = \begin{bmatrix} 5 \\ 6 \\ 5 \end{bmatrix}$; $V^{[1](1)} = f(a^{[1](1)}) = \begin{bmatrix} 0.46212 \\ 0.76159 \\ 0.46212 \end{bmatrix}$

$\rightarrow a^{[2](1)} = W^{[2]} \cdot x_1 + b^{[2]} = \begin{bmatrix} 4.97061 \\ 2.63583 \end{bmatrix}$; $V^{[2](1)} = f(a^{[2](1)}) = \begin{bmatrix} 0.45048 \\ -0.57642 \end{bmatrix}$

$\rightarrow a^{[3](1)} = W^{[3]} \cdot x_1 + b^{[3]} = \begin{bmatrix} 0.87406 \\ 1.77503 \\ 0.87406 \end{bmatrix}$; $V^{[3](1)} = f(a^{[3](1)}) = \begin{bmatrix} -0.91590 \\ -0.30494 \\ -0.91590 \end{bmatrix}$

And now, the same calculations but using x_2 (also using programming functions as a calculation resource)

$\rightarrow a^{[1](2)} = W^{[1]} \cdot x_2 + b^{[1]} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$; $V^{[1](2)} = f(a^{[1](2)}) = \begin{bmatrix} -0.90515 \\ -0.90515 \\ -0.90515 \end{bmatrix}$

$\rightarrow a^{[2](2)} = W^{[2]} \cdot x_2 + b^{[2]} = \begin{bmatrix} -4.43089 \\ -1.91544 \end{bmatrix}$; $V^{[2](2)} = f(a^{[2](2)}) = \begin{bmatrix} -0.99956 \\ -0.99343 \end{bmatrix}$

$\rightarrow a^{[3](2)} = W^{[3]} \cdot x_2 + b^{[3]} = \begin{bmatrix} -0.99300 \\ -2.99212 \\ -0.99300 \end{bmatrix}$; $V^{[3](2)} = f(a^{[3](2)}) = \begin{bmatrix} -0.98652 \\ -0.99816 \\ -0.98652 \end{bmatrix}$

Now, we will be doing back propagation

TÉCNICO LISBOA

Aprendizagem 2023/24

Homework III – Group 001

(ist1102082, ist1103252)

Starting with x_1:

$$\rightarrow \delta^{[3](1)} = \left(V^{[3](1)} - t_1\right) \circ \left(f'\left(a^{[3](1)}\right)\right) = \begin{bmatrix} -0.07379 \\ -0.31773 \\ -0.07379 \end{bmatrix}$$

$$\rightarrow \delta^{[2](1)} = \left(\left(W^{[3]}\right)^T \cdot \delta^{[3](1)}\right) \circ \left(f'\left(a^{[2](1)}\right)\right) = \begin{bmatrix} -0.43870 \\ -0.15535 \end{bmatrix}$$

$$\rightarrow \delta^{[1](1)} = \left(\left(W^{[2]}\right)^T \cdot \delta^{[2](1)}\right) \circ \left(f'\left(a^{[1](1)}\right)\right) = \begin{bmatrix} -0.23359 \\ -0.40110 \\ -0.23359 \end{bmatrix}$$

And the same for x_2:

$$\rightarrow \delta^{[3](2)} = \left(V^{[3](2)} - t_2\right) \circ \left(f'\left(a^{[3](2)}\right)\right) = \begin{bmatrix} -0.02660 \\ -0.00983 \\ -0.01321 \end{bmatrix}$$

$$\rightarrow \delta^{[2](2)} = \left(\left(W^{[3]}\right)^T \cdot \delta^{[3](2)}\right) \circ \left(f'\left(a^{[2](2)}\right)\right) = \begin{bmatrix} -1.47439 \\ -2.72552 \end{bmatrix}$$

$$\rightarrow \delta^{[1](2)} = \left(\left(W^{[2]}\right)^T \cdot \delta^{[2](2)}\right) \circ \left(f'\left(a^{[1](2)}\right)\right) = \begin{bmatrix} -2.64099 \\ -3.17617 \\ -2.64099 \end{bmatrix}$$

Now, we will be updating the weights, knowing that: $\boxed{W_{new}^{[k]} = W_{old}^{[k]} + \eta\left(\delta^{[k](1)} \cdot \left(V^{[k-1](1)}\right)^T + \delta^{[k](2)} \cdot \left(V^{[k-1](2)}\right)^T\right)}$

$$\rightarrow W_{new}^{[3]} = W_{old}^{[3]} + \eta\left(\delta^{[3](1)}\left(V^{[2](1)}\right)^T + \delta^{[3](2)}\left(V^{[2](2)}\right)^T\right) = \begin{bmatrix} 1.02336 & 1.02336 & 1.02336 & 1.02336 \\ 1.04011 & 1.04011 & 2.04011 & 1.04011 \\ 1.02336 & 1.02336 & 1.02336 & 1.02336 \end{bmatrix}$$

$$\rightarrow W_{new}^{[2]} = W_{old}^{[2]} + \eta\left(\delta^{[2](1)}\left(V^{[1](1)}\right)^T + \delta^{[2](2)}\left(V^{[1](2)}\right)^T\right) = \begin{bmatrix} 1.02027 & 4.03341 & 1.02027 \\ 1.00715 & 1.01181 & 1.00715 \end{bmatrix}$$

$$\rightarrow W_{new}^{[1]} = W_{old}^{[1]} + \eta\left(\delta^{[1](1)}\left(V^{[0](1)}\right)^T + \delta^{[1](2)}\left(V^{[1](2)}\right)^T\right) = \begin{bmatrix} 1.00067 & 0.99310 \\ 3.01113 & 0.98150 \\ 1.00200 & 0.99443 \end{bmatrix}$$

Finally, we can update the biases, knowing that: $\boxed{b_{new}^{[k]} = b_{old}^{[k]} - \eta\left(\delta^{[k](1)} + \delta^{[k](2)}\right)}$

Homework III – Group 001

(ist1102082, ist1103252)

$$\rightarrow b^{[1]}_{new} = b^{[1]}_{old} - \eta\left(\delta^{[1](1)} + \delta^{[1](2)}\right) = \begin{bmatrix} 1.02336 \\ 1.04011 \\ 1.02336 \end{bmatrix}$$

$$\rightarrow b^{[2]}_{new} = b^{[2]}_{old} - \eta\left(\delta^{[2](1)} + \delta^{[2](2)}\right) = \begin{bmatrix} 1.04387 \\ 1.01556 \end{bmatrix}$$

$$\rightarrow b^{[3]}_{new} = b^{[3]}_{old} - \eta\left(\delta^{[3](1)} + \delta^{[3](2)}\right) = \begin{bmatrix} 1.01003 \\ 1.03160 \\ 1.00870 \end{bmatrix}$$

We have now completed a one batch gradient descent update.

Homework III – Group 001

(ist1102082, ist1103252)

II. Programming and critical analysis

1)

Exercise 1

Loading the data from the csv file and converting it into a dataframe

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_absolute_error
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('winequality-red.csv', sep=';')
df.head()

X = df.drop('quality', axis=1)
y = df['quality']
```

[1]

Aprendizagem 2023/24

Homework III – Group 001

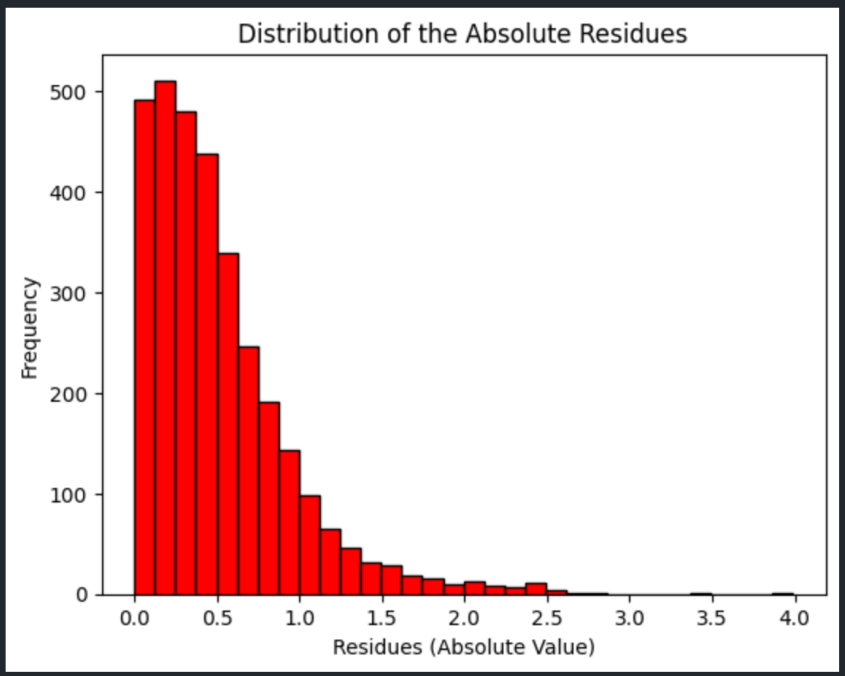
(ist1102082, ist1103252)

Perform a Multi Layer Perceptron Regression using the given parameters

```
# Set the MLPRegressor parameters
activation = "relu"  # Rectified Linear Unit (ReLU) activation
hidden_layer_sizes = (10, 10)  # Two hidden layers with 10 neurons each
validation_fraction = 0.2  # Use 20% of the training data for validation
residuals = []

# Split the dataset into a training and test set with a fixed seed
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=0)

# Perform 10 runs with different random states and collect residuals
for random_state in range(1, 11):
    # Create an MLP Regressor with the specified parameters
    mlp_regressor = MLPRegressor(hidden_layer_sizes=hidden_layer_sizes, activation=activation,
                                 validation_fraction=validation_fraction, random_state=random_state, early_stopping=True)

    # Train the MLP regressor
    mlp_regressor.fit(X_train, y_train)

    # Make predictions
    y_pred = mlp_regressor.predict(X_test)

    # Calculate residuals (absolute errors)
    residuals.extend(np.abs(y_test - y_pred))
```

Plot the distribution of residuals using a histogram

```
# Plot the distribution of residuals using a histogram
plt.hist(residuals, bins=32, edgecolor='black', color='red')
plt.title("Distribution of the Absolute Residues")
plt.xlabel("Residues (Absolute Value)")
plt.ylabel("Frequency")
plt.show()
```



Homework III – Group 001

(ist1102082, ist1103252)

2)

Exercise 2

We will perform the regression again with the same parameters, but we will calculate the MAE (Mean Absolute Error) with and without rounding and bounding. For the bounds we have decided to go within an interval of (5, 6) and we will round to one decimal place.

```
mae_round_bound = []    # Mean absolute error rounded and bounded
mae = []                # Mean absolute error without rounding and bounding


# Split the dataset into a training and test set with a fixed seed
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=0)

# Perform 10 runs with different random states and collect residuals
for random_state in range(1, 11):

    # Create an MLP Regressor with the specified parameters
    mlp_regressor = MLPRegressor(hidden_layer_sizes=(10, 10), activation="relu",
                                 validation_fraction=0.2, random_state=random_state,
                                 early_stopping=True)

    # Train the MLP regressor
    mlp_regressor.fit(X_train, y_train)

    # Make predictions
    y_pred = mlp_regressor.predict(X_test)

    mae.append(mean_absolute_error(y_test, y_pred))

    # Round and Bound
    round_pred = np.round(y_pred)
    y_pred_round_bound = np.clip(round_pred, 5, 6)
    mae_round_bound.append(mean_absolute_error(y_test, y_pred_round_bound))


    # Calculate residuals (absolute errors)
    residuals.extend(np.abs(y_test - y_pred))

print("Average MAE without rounding and bounding:", np.mean(mae))
print("Average MAE with rounding and bounding: ", np.mean(mae_round_bound))
```

```
Average MAE without rounding and bounding: 0.5097171955009514
Average MAE with rounding and bounding:  0.42125
```

As we can observe, the MAE is lower when we round and bound the values. By rounding the predictions to the nearest integer, we are now conforming the model's outputs to the problem's requirement. This can lead to more accurate predictions and thus lower MAE because the predictions align with the target variable's nature.Furthermore, bounding the values eliminates outliers that may be present in the data. This can also lead to more accurate predictions and thus lower MAE.

3)

Exercise 3

```
# Specify the number of iterations (in terms of batches)
num_iterations = [20, 50, 100, 200]

# Initialize lists to store RMSE values for each number of iterations
rmse_values = []

# Split the dataset into a training and test set with a fixed seed
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

for n_iterations in num_iterations:
    # Perform 10 runs with different random states and assess the impact
    random_states = range(1, 11)
    rmse_iterations = []


    for random_state in random_states:

        # Create an MLP Regressor with the specified number of iterations
        mlp_regressor = MLPRegressor(hidden_layer_sizes=(10, 10), activation = "relu",
                                     max_iter=n_iterations,
                                     random_state=random_state)

        # Train the MLP regressor
        mlp_regressor.fit(X_train, y_train)

        # Make predictions
        y_pred = mlp_regressor.predict(X_test)

        # Calculate RMSE
        rmse = np.sqrt(np.mean((y_test-y_pred)**2))

        rmse_iterations.append(rmse)

    # Calculate the average RMSE for the current number of iterations
    average_rmse = np.mean(rmse_iterations)
    rmse_values.append(average_rmse)

# Print the RMSE values for different numbers of iterations
for i, n_iterations in enumerate(num_iterations):
    print(f"Average RMSE for {n_iterations} iterations: {rmse_values[i]:.2f}")
```

```
Average RMSE for 20 iterations: 1.40
Average RMSE for 50 iterations: 0.80
Average RMSE for 100 iterations: 0.69
Average RMSE for 200 iterations: 0.66
```

Homework III – Group 001

(ist1102082, ist1103252)

4)

Exercise 4

A lower RMSE value indicates a better fit of the model to the data, meaning that the model's predictions are closer to the actual values. Following the same logic, a higher RMSE suggests that the model's predictions are further from the actual values, indicating a less accurate model.

By analyzing our results, we can see that the RMSE value is lower when we perform more iterations, meaning that the model is more accurate. This is because the model has more time to learn the patterns in the data, and thus it can make more accurate predictions.

Now, early stopping can definitely impact the performance of the model, in both positive and negative ways.

In the positive side, early stopping can prevent overfitting. Overfitting occurs when the model is too complex, and it starts to memorize the training data, instead of learning the general patterns. This can lead to a model that performs very well on the training data, but it performs poorly on the test data. Early stopping can prevent this from happening, by stopping the training process when the model starts to overfit.

In the negative side, early stopping can prevent the model from reaching its full potential. If the number of iterations set for early stopping is too low, it may lead to underfitting. Early stopping can stop training before the model has fully converged, resulting in a suboptimal solution.

END