Problem Statement

You are a data scientist working for a school

You are asked to predict the GPA of the current students based on the following provided data:

0 StudentID int64
1 Age int64
2 Gender int64
3 Ethnicity int64
4 ParentalEducation int64
5 StudyTimeWeekly float64 6 Absences int64
7 Tutoring int64
8 ParentalSupport int64
9 Extracurricular int64
10 Sports int64
11 Music int64
12 Volunteering int64
13 GPA float64 14 GradeClass float64

The GPA is the Grade Point Average, typically ranges from 0.0 to 4.0 in most educational systems, with 4.0 representing an 'A' or excellent performance.

The minimum passing GPA can vary by institution, but it's often around 2.0. This usually corresponds to a 'C' grade, which is considered satisfactory.

You need to create a Deep Learning model capable to predict the GPA of a Student based on a set of provided features. The data provided represents 2,392 students.

In this excersice you will be requested to create a total of three models and select the most performant one.

1) Import Libraries

First let's import the following libraries, if there is any library that you need and is not in the list bellow feel free to include it

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
```

```python
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten
from tensorflow.keras.regularizers import l2
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

2) Load Data

- You will be provided with a cvs (comma separated value) file.
- You will need to add that file into a pandas dataframe, you can use the following code as reference
- The file will be available in canvas

In [3]:
```python
data = pd.read_csv("Student_performance_data _.csv")
data.head()
```

Out[3]:

	StudentID	Age	Gender	Ethnicity	ParentalEducation	StudyTimeWeekly	Absences	Tut
0	1001	17	1	0	2	19.833723	7	
1	1002	18	0	0	1	15.408756	0	
2	1003	15	0	2	3	4.210570	26	
3	1004	17	1	0	3	10.028829	14	
4	1005	17	1	0	2	4.672495	17	

3) Review you data:

Make sure you review your data. Place special attention of null or empty values.

In [4]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2392 entries, 0 to 2391
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   StudentID          2392 non-null   int64  
 1   Age                2392 non-null   int64  
 2   Gender             2392 non-null   int64  
 3   Ethnicity          2392 non-null   int64  
 4   ParentalEducation  2392 non-null   int64  
 5   StudyTimeWeekly    2392 non-null   float64
 6   Absences           2392 non-null   int64  
 7   Tutoring           2392 non-null   int64  
 8   ParentalSupport    2392 non-null   int64  
 9   Extracurricular    2392 non-null   int64  
 10  Sports             2392 non-null   int64  
 11  Music              2392 non-null   int64  
 12  Volunteering       2392 non-null   int64  
 13  GPA                2392 non-null   float64
 14  GradeClass         2392 non-null   float64
dtypes: float64(3), int64(12)
memory usage: 280.4 KB
```

4. Remove the columns not needed for Student performance prediction

- Choose only the columns you consider to be valuable for your model training.
- For example, StudentID might not be a good feature for your model, and thus should be removed from your main dataset, which other columns should also be removed?
- You can name that final dataset as 'dataset'

In [5]:

```
# Remove StudentID column
data = data.drop(columns=['StudentID'])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2392 entries, 0 to 2391
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Age                2392 non-null   int64  
 1   Gender             2392 non-null   int64  
 2   Ethnicity          2392 non-null   int64  
 3   ParentalEducation  2392 non-null   int64  
 4   StudyTimeWeekly    2392 non-null   float64
 5   Absences           2392 non-null   int64  
 6   Tutoring           2392 non-null   int64  
 7   ParentalSupport    2392 non-null   int64  
 8   Extracurricular    2392 non-null   int64  
 9   Sports             2392 non-null   int64  
 10  Music              2392 non-null   int64  
 11  Volunteering       2392 non-null   int64  
 12  GPA                2392 non-null   float64
 13  GradeClass         2392 non-null   float64
dtypes: float64(3), int64(11)
memory usage: 261.8 KB
```

5. Check if the columns has any null values:

- Here you now have your final dataset to use in your model training.
- Before moving foward review your data check for any null or empty value that might be needed to be removed

In [6]:
```
# Check for missing values
print(data.isnull().sum())
```

```
Age                  0
Gender               0
Ethnicity            0
ParentalEducation    0
StudyTimeWeekly      0
Absences             0
Tutoring             0
ParentalSupport      0
Extracurricular      0
Sports               0
Music                0
Volunteering         0
GPA                  0
GradeClass           0
dtype: int64
```

6. Prepare your data for training and for testing set:

- First create a dataset named X, with all columns but GPA. These are the features
- Next create another dataset named y, with only GPA column. This is the label

- If you go to your Imports, you will see the following import: **'from sklearn.model_selection import train_test_split'**

- Use that *train_test_split* function to create: X_train, X_test, y_train and y_test respectively. Use X and y datasets as parameters. Other parameters to use are: Test Size = 0.2, Random State = 42.

- Standarize your features (X_train and X_test) by using the StandardScaler (investigate how to use fit_transform and transform functions). This will help the training process by dealing with normilized data.

Note: Your X_train shape should be around (1913, 10). This means the dataset has 10 columns which should be the input.

In [7]:
```
# Your code here
x = data.drop(columns=['GPA'])
y = data['GPA']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_sta
# Scale features using StandardScaler (fit on train, transform on test)
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
print('x_train shape:', x_train.shape, 'x_test shape:', x_test.shape)
```

```
x_train shape: (1913, 13) x_test shape: (479, 13)
```

7. Define your Deep Neural Network.

- This will be a Sequential Neural Network.
- With a Dense input layer with 64 units, and input dimention of 10 and Relu as the activation function.
- A Dense hidden layer with 32 units, and Relu as the activation function.
- And a Dense output layer with 1 unit, do not define an activation function so it defaults to linear, suitable for regression tasks. e.g. Dense(1)

This last part of the output layer is super important, since we want to predict the GPA, this means that we want a regression and not a classification. Linear activation function is best for regression and Sigmoid is best for Binary Classification

In [8]:
```
# Your code here
model = Sequential()
# Use the number of features from x_train for the input shape
model.add(Dense(64, input_shape=(x_train.shape[1],), activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1))  # linear output for regression
```

```
c:\Users\Pansocrates03\Documents\7mo Semestre\DEEP LEARNING\act3\.venv\Lib\site-packages\keras\src\layers\core\dense.py:92: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

8. Compile your Neural Network

- Choose Adam as the optimizer
- And MSE as the Loss function
- Also add the following metrics: Mean Absolute Error

In [9]:
```
# Your code here
# For regression use MSE loss and track MAE as a metric
model.compile(loss='mse', optimizer='adam', metrics=['mae'])
```

9. Fit (or train) your model

- Use the X_train and y_train datasets for the training
- Do 50 data iterations
- Choose the batch size = 10
- Also select a validation_split of 0.2
- Save the result of the fit function in a variable called 'history'

In [10]:
```
# Your code here
# fit the keras model on the dataset
# Use a validation split and store the history
history = model.fit(x_train, y_train, epochs=50, batch_size=10, verbose=1, validati
```

```
Epoch 1/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 4s 10ms/step - loss: 1.1096 - mae: 0.7417 - val_loss:
0.1493 - val_mae: 0.3119
Epoch 2/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.1040 - mae: 0.2573 - val_loss: 0.
0949 - val_mae: 0.2516
Epoch 3/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0698 - mae: 0.2132 - val_loss: 0.
0749 - val_mae: 0.2238
Epoch 4/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0567 - mae: 0.1912 - val_loss: 0.
0723 - val_mae: 0.2169
Epoch 5/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0494 - mae: 0.1796 - val_loss: 0.
0602 - val_mae: 0.1984
Epoch 6/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0439 - mae: 0.1696 - val_loss: 0.
0565 - val_mae: 0.1912
Epoch 7/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0405 - mae: 0.1625 - val_loss: 0.
0520 - val_mae: 0.1837
Epoch 8/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0384 - mae: 0.1576 - val_loss: 0.
0531 - val_mae: 0.1845
Epoch 9/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0361 - mae: 0.1524 - val_loss: 0.
0543 - val_mae: 0.1864
Epoch 10/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0345 - mae: 0.1489 - val_loss: 0.
0503 - val_mae: 0.1801
Epoch 11/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0330 - mae: 0.1456 - val_loss: 0.
0505 - val_mae: 0.1797
Epoch 12/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0323 - mae: 0.1433 - val_loss: 0.
0504 - val_mae: 0.1770
Epoch 13/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0302 - mae: 0.1387 - val_loss: 0.
0495 - val_mae: 0.1770
Epoch 14/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0289 - mae: 0.1349 - val_loss: 0.
0484 - val_mae: 0.1772
Epoch 15/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0286 - mae: 0.1338 - val_loss: 0.
0498 - val_mae: 0.1768
Epoch 16/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0274 - mae: 0.1313 - val_loss: 0.
0464 - val_mae: 0.1723
Epoch 17/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0272 - mae: 0.1324 - val_loss: 0.
0486 - val_mae: 0.1770
Epoch 18/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0271 - mae: 0.1307 - val_loss: 0.
0455 - val_mae: 0.1711
Epoch 19/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0254 - mae: 0.1252 - val_loss: 0.
```

```
0479 - val_mae: 0.1742
Epoch 20/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0253 - mae: 0.1257 - val_loss: 0.
0469 - val_mae: 0.1734
Epoch 21/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0249 - mae: 0.1249 - val_loss: 0.
0478 - val_mae: 0.1756
Epoch 22/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0240 - mae: 0.1230 - val_loss: 0.
0469 - val_mae: 0.1736
Epoch 23/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0226 - mae: 0.1191 - val_loss: 0.
0519 - val_mae: 0.1831
Epoch 24/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0225 - mae: 0.1190 - val_loss: 0.
0458 - val_mae: 0.1690
Epoch 25/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - loss: 0.0219 - mae: 0.1175 - val_loss: 0.
0446 - val_mae: 0.1686
Epoch 26/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0214 - mae: 0.1162 - val_loss: 0.
0471 - val_mae: 0.1725
Epoch 27/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0211 - mae: 0.1147 - val_loss: 0.
0460 - val_mae: 0.1701
Epoch 28/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0207 - mae: 0.1144 - val_loss: 0.
0473 - val_mae: 0.1698
Epoch 29/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0200 - mae: 0.1121 - val_loss: 0.
0520 - val_mae: 0.1822
Epoch 30/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0201 - mae: 0.1127 - val_loss: 0.
0472 - val_mae: 0.1736
Epoch 31/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0186 - mae: 0.1088 - val_loss: 0.
0522 - val_mae: 0.1791
Epoch 32/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0192 - mae: 0.1097 - val_loss: 0.
0485 - val_mae: 0.1746
Epoch 33/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0187 - mae: 0.1072 - val_loss: 0.
0499 - val_mae: 0.1811
Epoch 34/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0177 - mae: 0.1045 - val_loss: 0.
0509 - val_mae: 0.1775
Epoch 35/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0176 - mae: 0.1048 - val_loss: 0.
0482 - val_mae: 0.1739
Epoch 36/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0170 - mae: 0.1032 - val_loss: 0.
0479 - val_mae: 0.1740
Epoch 37/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0187 - mae: 0.1081 - val_loss: 0.
0495 - val_mae: 0.1790
Epoch 38/50
```

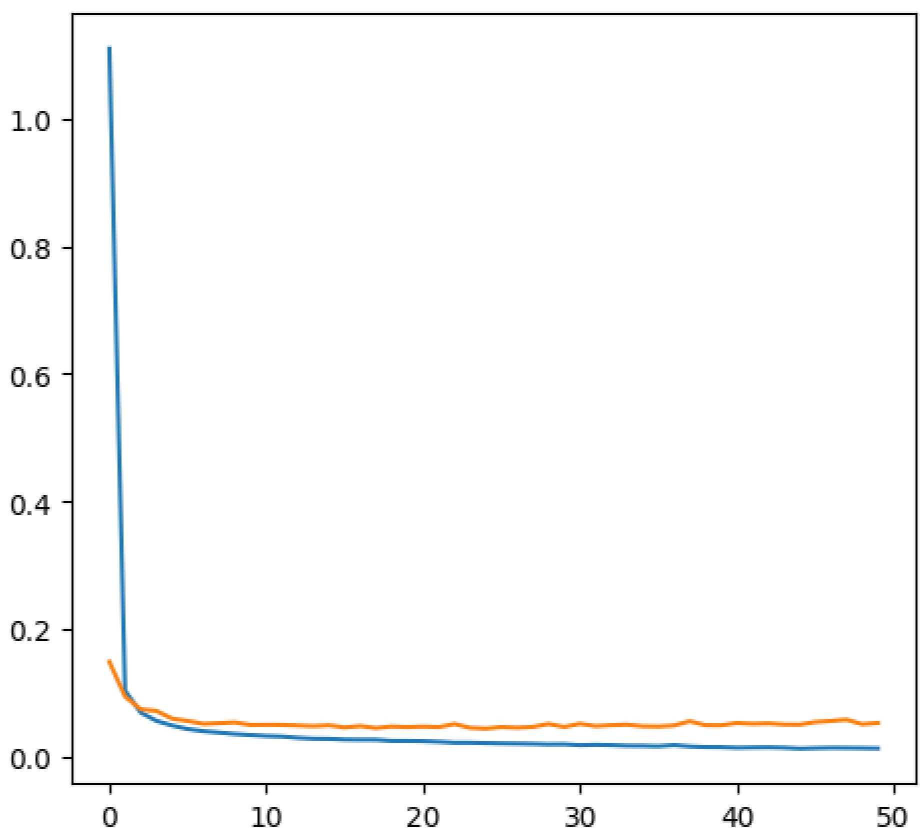
```
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0167 - mae: 0.1022 - val_loss: 0.
0561 - val_mae: 0.1849
Epoch 39/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0157 - mae: 0.0988 - val_loss: 0.
0500 - val_mae: 0.1779
Epoch 40/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0155 - mae: 0.0978 - val_loss: 0.
0498 - val_mae: 0.1757
Epoch 41/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 0.0146 - mae: 0.0957 - val_loss: 0.
0534 - val_mae: 0.1822
Epoch 42/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0150 - mae: 0.0978 - val_loss: 0.
0521 - val_mae: 0.1804
Epoch 43/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0153 - mae: 0.0975 - val_loss: 0.
0528 - val_mae: 0.1812
Epoch 44/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0147 - mae: 0.0955 - val_loss: 0.
0510 - val_mae: 0.1752
Epoch 45/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0134 - mae: 0.0905 - val_loss: 0.
0508 - val_mae: 0.1777
Epoch 46/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0140 - mae: 0.0942 - val_loss: 0.
0549 - val_mae: 0.1873
Epoch 47/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0145 - mae: 0.0954 - val_loss: 0.
0566 - val_mae: 0.1877
Epoch 48/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0143 - mae: 0.0948 - val_loss: 0.
0586 - val_mae: 0.1906
Epoch 49/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0140 - mae: 0.0930 - val_loss: 0.
0515 - val_mae: 0.1783
Epoch 50/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0137 - mae: 0.0911 - val_loss: 0.
0534 - val_mae: 0.1808
```

10. View your history variable:

- Use Matplotlib.pyplot to show graphs of your model traning history
- In one graph:
    - Plot the Training Loss and the Validation Loss
    - X Label = Epochs
    - Y Label = Loss
    - Title = Training and Validation Loss over Epochs
- In a second graph:
    - Plot the Training MAE and the Validation MAE
    - X Label = Epochs
    - Y Label = Mean Absolute Error (MAE)
    - Title = Training and Validation MAE over Epochs

In [11]:
```
# Matplotlib code to show graphs of model training history
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
```

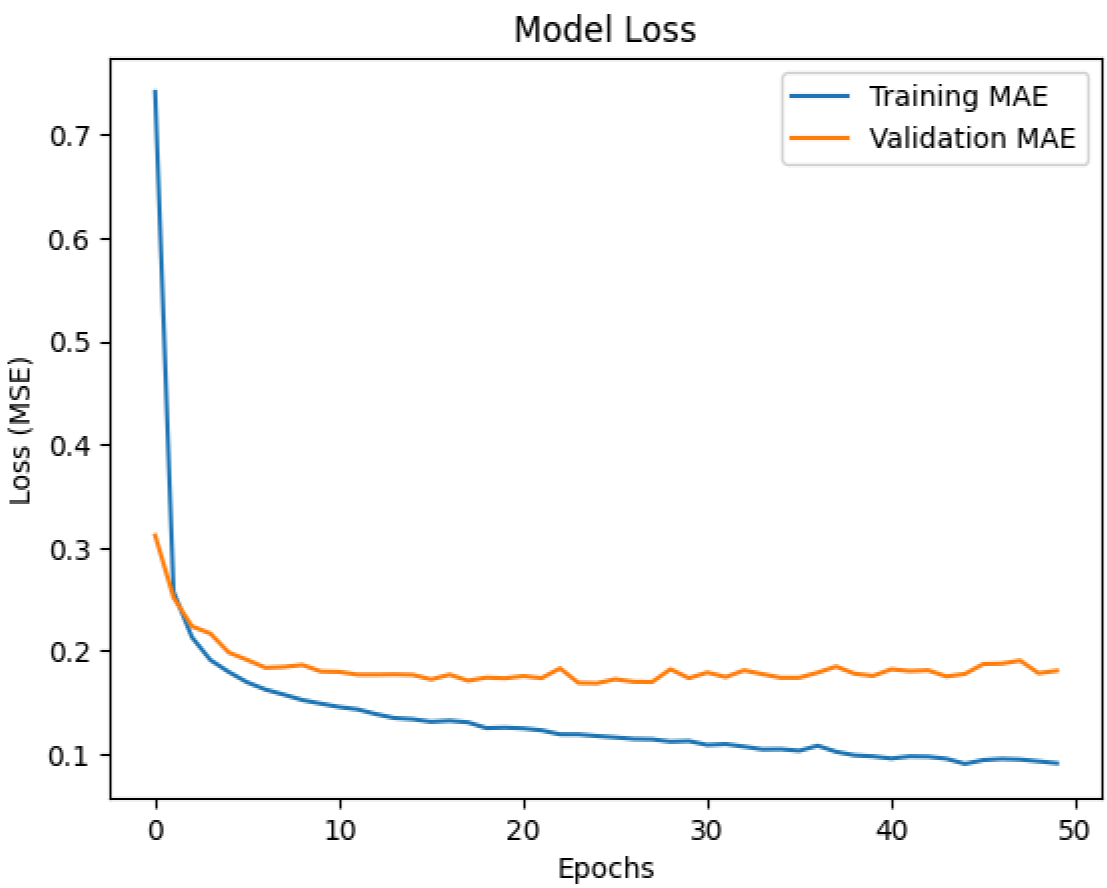
Out[11]:
```
[<matplotlib.lines.Line2D at 0x1c805feed50>]
```



11. Evaluate your model:

- See the result of your loss function.
- What can you deduct from there?

In [33]:
```
# Result of loss function
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss (MSE)')
plt.plot(history.history['mae'], label='Training MAE')
plt.plot(history.history['val_mae'], label='Validation MAE')
plt.legend(["Training MAE", "Validation MAE"])
plt.show()
```



12. Use your model to make some predictions:

- Make predictions of your X_test dataset
- Print the each of the predictions and the actual value (which is in y_test)
- How good was your model?

In [13]:
```
# Make predictions and evaluate the model
y_pred = model.predict(x_test)
from sklearn.metrics import mean_squared_error, mean_absolute_error
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
print(f'Test MSE: {mse}, Test MAE: {mae}')
```

```
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step
Test MSE: 0.05553159080897823, Test MAE: 0.1814346866158936
```

13. Compete against this model:

- Create two more different models to compete with this model
- Here are a few ideas of things you can change:
  - During Dataset data engineering:
    - You can remove features that you think do not help in the training and prediction
    - Feature Scaling: Ensure all features are on a similar scale (as you already did with StandardScaler)
  - During Model Definition:
    - You can change the Model Architecture (change the type or number of layers or the number of units)
    - You can add dropout layers to prevent overfitting
  - During Model Compile:
    - You can try other optimizer when compiling your model, here some optimizer samples: Adam, RMSprop, or Adagrad.
    - Try another Loss Function
  - During Model Training:
    - Encrease the number of Epochs
    - Adjust the size of your batch
- Explain in a Markdown cell which changes are you implementing
- Show the comparison of your model versus the original model

Model 2:

- Changes:
  - Dataset Data Engineering
  - Model Definition
  - Model Compile
  - Model Training

In [19]:
```
# Model 2
model2 = Sequential()
model2.add(Dense(128, input_shape=(x_train.shape[1],), activation='relu', kernel_re
model2.add(Dropout(0.3))
model2.add(Dense(64, activation='relu', kernel_regularizer=l2(0.001)))
model2.add(Dropout(0.3))
model2.add(Dense(32, activation='relu', kernel_regularizer=l2(0.001)))
model2.add(Dense(1))  # Linear output for regression

# Compile the second model
model2.compile(loss='mse', optimizer='adam', metrics=['mae'])

# Fit the second model
```

```python
history2 = model2.fit(x_train, y_train, epochs=100, batch_size=10, verbose=1, valid

# Make predictions and evaluate the model
y_pred = model2.predict(x_test)
from sklearn.metrics import mean_squared_error, mean_absolute_error
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
print(f'Test MSE: {mse}, Test MAE: {mae}')
```

```
Epoch 1/100
```

```
c:\Users\Pansocrates03\Documents\7mo Semestre\DEEP LEARNING\act3\.venv\Lib\site-pack
ages\keras\src\layers\core\dense.py:92: UserWarning: Do not pass an `input_shape`/`i
nput_dim` argument to a layer. When using Sequential models, prefer using an `Input
(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
153/153 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 0.6743 - mae: 0.5546 - val_loss: 0.
3671 - val_mae: 0.3881
Epoch 2/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.3908 - mae: 0.3907 - val_loss: 0.
3115 - val_mae: 0.3466
Epoch 3/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.3076 - mae: 0.3247 - val_loss: 0.
2624 - val_mae: 0.3009
Epoch 4/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.2770 - mae: 0.3003 - val_loss: 0.
3260 - val_mae: 0.3781
Epoch 5/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.2530 - mae: 0.2796 - val_loss: 0.
2428 - val_mae: 0.2876
Epoch 6/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.2322 - mae: 0.2689 - val_loss: 0.
3009 - val_mae: 0.3658
Epoch 7/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.2146 - mae: 0.2568 - val_loss: 0.
1879 - val_mae: 0.2311
Epoch 8/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.1960 - mae: 0.2386 - val_loss: 0.
2085 - val_mae: 0.2709
Epoch 9/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.1815 - mae: 0.2264 - val_loss: 0.
1649 - val_mae: 0.2142
Epoch 10/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.1667 - mae: 0.2156 - val_loss: 0.
1548 - val_mae: 0.2056
Epoch 11/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.1588 - mae: 0.2118 - val_loss: 0.
1365 - val_mae: 0.1794
Epoch 12/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.1539 - mae: 0.2051 - val_loss: 0.
1254 - val_mae: 0.1686
Epoch 13/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.1502 - mae: 0.2104 - val_loss: 0.
1669 - val_mae: 0.2513
Epoch 14/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.1397 - mae: 0.2031 - val_loss: 0.
1695 - val_mae: 0.2558
Epoch 15/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.1325 - mae: 0.1963 - val_loss: 0.
1168 - val_mae: 0.1766
Epoch 16/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.1292 - mae: 0.1961 - val_loss: 0.
1543 - val_mae: 0.2469
Epoch 17/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.1246 - mae: 0.1987 - val_loss: 0.
1144 - val_mae: 0.1884
Epoch 18/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.1163 - mae: 0.1929 - val_loss: 0.
1169 - val_mae: 0.1996
Epoch 19/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.1143 - mae: 0.1907 - val_loss: 0.
0925 - val_mae: 0.1552
```

```
Epoch 20/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.1062 - mae: 0.1822 - val_loss: 0.
0872 - val_mae: 0.1514
Epoch 21/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.1023 - mae: 0.1832 - val_loss: 0.
1126 - val_mae: 0.2078
Epoch 22/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.1001 - mae: 0.1822 - val_loss: 0.
0962 - val_mae: 0.1858
Epoch 23/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0920 - mae: 0.1752 - val_loss: 0.
0776 - val_mae: 0.1475
Epoch 24/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0895 - mae: 0.1768 - val_loss: 0.
0983 - val_mae: 0.1989
Epoch 25/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0900 - mae: 0.1814 - val_loss: 0.
0884 - val_mae: 0.1810
Epoch 26/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0827 - mae: 0.1689 - val_loss: 0.
0733 - val_mae: 0.1531
Epoch 27/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0832 - mae: 0.1762 - val_loss: 0.
0704 - val_mae: 0.1549
Epoch 28/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0800 - mae: 0.1733 - val_loss: 0.
0707 - val_mae: 0.1524
Epoch 29/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0804 - mae: 0.1732 - val_loss: 0.
0697 - val_mae: 0.1596
Epoch 30/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0755 - mae: 0.1687 - val_loss: 0.
0683 - val_mae: 0.1566
Epoch 31/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0782 - mae: 0.1758 - val_loss: 0.
0756 - val_mae: 0.1730
Epoch 32/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0744 - mae: 0.1711 - val_loss: 0.
0668 - val_mae: 0.1593
Epoch 33/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0728 - mae: 0.1732 - val_loss: 0.
0746 - val_mae: 0.1805
Epoch 34/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0693 - mae: 0.1681 - val_loss: 0.
0797 - val_mae: 0.1898
Epoch 35/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0707 - mae: 0.1724 - val_loss: 0.
0682 - val_mae: 0.1680
Epoch 36/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0732 - mae: 0.1778 - val_loss: 0.
0612 - val_mae: 0.1532
Epoch 37/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0664 - mae: 0.1672 - val_loss: 0.
0737 - val_mae: 0.1832
Epoch 38/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0659 - mae: 0.1645 - val_loss: 0.
```

```
0671 - val_mae: 0.1669
Epoch 39/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0622 - mae: 0.1612 - val_loss: 0.
0720 - val_mae: 0.1840
Epoch 40/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0616 - mae: 0.1614 - val_loss: 0.
0685 - val_mae: 0.1738
Epoch 41/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0626 - mae: 0.1630 - val_loss: 0.
0616 - val_mae: 0.1595
Epoch 42/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0610 - mae: 0.1612 - val_loss: 0.
0587 - val_mae: 0.1572
Epoch 43/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0614 - mae: 0.1640 - val_loss: 0.
0731 - val_mae: 0.1843
Epoch 44/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0598 - mae: 0.1600 - val_loss: 0.
0638 - val_mae: 0.1627
Epoch 45/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0597 - mae: 0.1613 - val_loss: 0.
0557 - val_mae: 0.1535
Epoch 46/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0569 - mae: 0.1567 - val_loss: 0.
0576 - val_mae: 0.1562
Epoch 47/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0578 - mae: 0.1586 - val_loss: 0.
0761 - val_mae: 0.1979
Epoch 48/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0619 - mae: 0.1676 - val_loss: 0.
0537 - val_mae: 0.1494
Epoch 49/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0564 - mae: 0.1577 - val_loss: 0.
0618 - val_mae: 0.1646
Epoch 50/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0580 - mae: 0.1602 - val_loss: 0.
0631 - val_mae: 0.1657
Epoch 51/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0543 - mae: 0.1544 - val_loss: 0.
0732 - val_mae: 0.1878
Epoch 52/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0547 - mae: 0.1544 - val_loss: 0.
0552 - val_mae: 0.1547
Epoch 53/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0573 - mae: 0.1616 - val_loss: 0.
0739 - val_mae: 0.1983
Epoch 54/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0547 - mae: 0.1571 - val_loss: 0.
0687 - val_mae: 0.1850
Epoch 55/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0585 - mae: 0.1613 - val_loss: 0.
0634 - val_mae: 0.1717
Epoch 56/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0563 - mae: 0.1596 - val_loss: 0.
0578 - val_mae: 0.1607
Epoch 57/100
```

```
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0542 - mae: 0.1574 - val_loss: 0.
0627 - val_mae: 0.1744
Epoch 58/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0515 - mae: 0.1509 - val_loss: 0.
0559 - val_mae: 0.1575
Epoch 59/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0551 - mae: 0.1583 - val_loss: 0.
0613 - val_mae: 0.1673
Epoch 60/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0542 - mae: 0.1565 - val_loss: 0.
0586 - val_mae: 0.1609
Epoch 61/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0540 - mae: 0.1586 - val_loss: 0.
0866 - val_mae: 0.2215
Epoch 62/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0536 - mae: 0.1561 - val_loss: 0.
0705 - val_mae: 0.1862
Epoch 63/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0556 - mae: 0.1586 - val_loss: 0.
0838 - val_mae: 0.2118
Epoch 64/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0552 - mae: 0.1583 - val_loss: 0.
0822 - val_mae: 0.2060
Epoch 65/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0523 - mae: 0.1530 - val_loss: 0.
0700 - val_mae: 0.1884
Epoch 66/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0529 - mae: 0.1548 - val_loss: 0.
0676 - val_mae: 0.1871
Epoch 67/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0539 - mae: 0.1560 - val_loss: 0.
0639 - val_mae: 0.1784
Epoch 68/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0556 - mae: 0.1593 - val_loss: 0.
0734 - val_mae: 0.1919
Epoch 69/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0531 - mae: 0.1557 - val_loss: 0.
0685 - val_mae: 0.1848
Epoch 70/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0534 - mae: 0.1562 - val_loss: 0.
0546 - val_mae: 0.1581
Epoch 71/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0508 - mae: 0.1521 - val_loss: 0.
0788 - val_mae: 0.2050
Epoch 72/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0543 - mae: 0.1597 - val_loss: 0.
0729 - val_mae: 0.1949
Epoch 73/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0528 - mae: 0.1561 - val_loss: 0.
0711 - val_mae: 0.1883
Epoch 74/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0533 - mae: 0.1563 - val_loss: 0.
0602 - val_mae: 0.1709
Epoch 75/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0529 - mae: 0.1559 - val_loss: 0.
0832 - val_mae: 0.2138
```

```
Epoch 76/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0515 - mae: 0.1534 - val_loss: 0.
0728 - val_mae: 0.1980
Epoch 77/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0541 - mae: 0.1586 - val_loss: 0.
0609 - val_mae: 0.1703
Epoch 78/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0552 - mae: 0.1618 - val_loss: 0.
0683 - val_mae: 0.1867
Epoch 79/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0514 - mae: 0.1550 - val_loss: 0.
0813 - val_mae: 0.2106
Epoch 80/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0514 - mae: 0.1533 - val_loss: 0.
0620 - val_mae: 0.1733
Epoch 81/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0528 - mae: 0.1572 - val_loss: 0.
0982 - val_mae: 0.2369
Epoch 82/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0510 - mae: 0.1527 - val_loss: 0.
0666 - val_mae: 0.1767
Epoch 83/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0538 - mae: 0.1576 - val_loss: 0.
0929 - val_mae: 0.2296
Epoch 84/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0568 - mae: 0.1634 - val_loss: 0.
0727 - val_mae: 0.1919
Epoch 85/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0502 - mae: 0.1525 - val_loss: 0.
0611 - val_mae: 0.1714
Epoch 86/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0529 - mae: 0.1569 - val_loss: 0.
0999 - val_mae: 0.2413
Epoch 87/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0537 - mae: 0.1560 - val_loss: 0.
0707 - val_mae: 0.1956
Epoch 88/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0508 - mae: 0.1545 - val_loss: 0.
0585 - val_mae: 0.1693
Epoch 89/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0521 - mae: 0.1558 - val_loss: 0.
0980 - val_mae: 0.2340
Epoch 90/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0526 - mae: 0.1554 - val_loss: 0.
0786 - val_mae: 0.2052
Epoch 91/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0496 - mae: 0.1499 - val_loss: 0.
0603 - val_mae: 0.1691
Epoch 92/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0518 - mae: 0.1542 - val_loss: 0.
0887 - val_mae: 0.2179
Epoch 93/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0520 - mae: 0.1549 - val_loss: 0.
0681 - val_mae: 0.1824
Epoch 94/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0529 - mae: 0.1560 - val_loss: 0.
```

```
0810 - val_mae: 0.2094
Epoch 95/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0535 - mae: 0.1595 - val_loss: 0.
0888 - val_mae: 0.2245
Epoch 96/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0522 - mae: 0.1578 - val_loss: 0.
0708 - val_mae: 0.1890
Epoch 97/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0504 - mae: 0.1524 - val_loss: 0.
0755 - val_mae: 0.1991
Epoch 98/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0520 - mae: 0.1565 - val_loss: 0.
0689 - val_mae: 0.1829
Epoch 99/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0502 - mae: 0.1524 - val_loss: 0.
0542 - val_mae: 0.1563
Epoch 100/100
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0501 - mae: 0.1530 - val_loss: 0.
0679 - val_mae: 0.1845
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
Test MSE: 0.05294810358332716, Test MAE: 0.18103118290938466
```

Realizar tests al modelo 2

In [20]:
```
# Make predictions and evaluate the model
y_pred = model2.predict(x_test)
from sklearn.metrics import mean_squared_error, mean_absolute_error
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
print(f'Test MSE: {mse}, Test MAE: {mae}')
```

```
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Test MSE: 0.05294810358332716, Test MAE: 0.18103118290938466
```

Model 3:

- Changes:
  - Dataset Data Engineering
  - Model Definition
  - Model Compile
  - Model Training

In [17]:
```
# Model 3
model3 = Sequential()
model3.add(Conv1D(64, kernel_size=2, activation='relu', input_shape=(x_train.shape[
model3.add(MaxPooling1D(pool_size=2))
model3.add(Flatten())
model3.add(Dense(50, activation='relu'))
model3.add(Dense(1))  # linear output for regression
# Your code here
model3.compile(loss='mse', optimizer='adam', metrics=['mae'])

# Fit the third model
history3 = model3.fit(x_train, y_train, epochs=50, batch_size=10, verbose=1, valida
```

```
c:\Users\Pansocrates03\Documents\7mo Semestre\DEEP LEARNING\act3\.venv\Lib\site-pack
ages\keras\src\layers\convolutional\base_conv.py:113: UserWarning: Do not pass an `i
nput_shape`/`input_dim` argument to a layer. When using Sequential models, prefer us
ing an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
Epoch 1/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - loss: 0.4360 - mae: 0.4902 - val_loss: 
0.1589 - val_mae: 0.3194
Epoch 2/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.1256 - mae: 0.2801 - val_loss: 0.
1053 - val_mae: 0.2589
Epoch 3/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0965 - mae: 0.2470 - val_loss: 0.
0968 - val_mae: 0.2454
Epoch 4/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 0.0832 - mae: 0.2286 - val_loss: 0.
0824 - val_mae: 0.2242
Epoch 5/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 0.0778 - mae: 0.2206 - val_loss: 0.
0699 - val_mae: 0.2128
Epoch 6/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0756 - mae: 0.2193 - val_loss: 0.
0726 - val_mae: 0.2129
Epoch 7/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0634 - mae: 0.2009 - val_loss: 0.
0577 - val_mae: 0.1903
Epoch 8/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0601 - mae: 0.1933 - val_loss: 0.
0733 - val_mae: 0.2170
Epoch 9/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0611 - mae: 0.1953 - val_loss: 0.
0651 - val_mae: 0.2019
Epoch 10/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0568 - mae: 0.1890 - val_loss: 0.
0817 - val_mae: 0.2253
Epoch 11/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0561 - mae: 0.1874 - val_loss: 0.
0573 - val_mae: 0.1893
Epoch 12/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0561 - mae: 0.1885 - val_loss: 0.
0536 - val_mae: 0.1803
Epoch 13/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 0.0493 - mae: 0.1777 - val_loss: 0.
0668 - val_mae: 0.2039
Epoch 14/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0499 - mae: 0.1764 - val_loss: 0.
0626 - val_mae: 0.1993
Epoch 15/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0502 - mae: 0.1787 - val_loss: 0.
0468 - val_mae: 0.1685
Epoch 16/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0484 - mae: 0.1764 - val_loss: 0.
0505 - val_mae: 0.1760
Epoch 17/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0499 - mae: 0.1771 - val_loss: 0.
0470 - val_mae: 0.1708
Epoch 18/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0462 - mae: 0.1686 - val_loss: 0.
0494 - val_mae: 0.1756
Epoch 19/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0463 - mae: 0.1714 - val_loss: 0.
```

```
0492 - val_mae: 0.1719
Epoch 20/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0477 - mae: 0.1742 - val_loss: 0.
0518 - val_mae: 0.1795
Epoch 21/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0439 - mae: 0.1670 - val_loss: 0.
0488 - val_mae: 0.1739
Epoch 22/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0420 - mae: 0.1632 - val_loss: 0.
0498 - val_mae: 0.1753
Epoch 23/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0425 - mae: 0.1652 - val_loss: 0.
0513 - val_mae: 0.1781
Epoch 24/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0456 - mae: 0.1711 - val_loss: 0.
0570 - val_mae: 0.1914
Epoch 25/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 0.0411 - mae: 0.1613 - val_loss: 0.
0552 - val_mae: 0.1850
Epoch 26/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0405 - mae: 0.1613 - val_loss: 0.
0514 - val_mae: 0.1765
Epoch 27/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0421 - mae: 0.1648 - val_loss: 0.
0516 - val_mae: 0.1787
Epoch 28/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0416 - mae: 0.1643 - val_loss: 0.
0500 - val_mae: 0.1739
Epoch 29/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0397 - mae: 0.1600 - val_loss: 0.
0460 - val_mae: 0.1683
Epoch 30/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0413 - mae: 0.1640 - val_loss: 0.
0476 - val_mae: 0.1706
Epoch 31/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0428 - mae: 0.1657 - val_loss: 0.
0489 - val_mae: 0.1752
Epoch 32/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0380 - mae: 0.1574 - val_loss: 0.
0446 - val_mae: 0.1660
Epoch 33/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0391 - mae: 0.1575 - val_loss: 0.
0463 - val_mae: 0.1682
Epoch 34/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0408 - mae: 0.1611 - val_loss: 0.
0560 - val_mae: 0.1894
Epoch 35/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0374 - mae: 0.1531 - val_loss: 0.
0465 - val_mae: 0.1710
Epoch 36/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - loss: 0.0366 - mae: 0.1520 - val_loss: 0.
0462 - val_mae: 0.1661
Epoch 37/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 0.0358 - mae: 0.1516 - val_loss: 0.
0547 - val_mae: 0.1870
Epoch 38/50
```

```
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0401 - mae: 0.1588 - val_loss: 0.
0525 - val_mae: 0.1817
Epoch 39/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0429 - mae: 0.1668 - val_loss: 0.
0496 - val_mae: 0.1762
Epoch 40/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0384 - mae: 0.1573 - val_loss: 0.
0481 - val_mae: 0.1719
Epoch 41/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0374 - mae: 0.1546 - val_loss: 0.
0449 - val_mae: 0.1677
Epoch 42/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0340 - mae: 0.1471 - val_loss: 0.
0489 - val_mae: 0.1726
Epoch 43/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0357 - mae: 0.1500 - val_loss: 0.
0461 - val_mae: 0.1677
Epoch 44/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0353 - mae: 0.1501 - val_loss: 0.
0447 - val_mae: 0.1636
Epoch 45/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0366 - mae: 0.1533 - val_loss: 0.
0512 - val_mae: 0.1792
Epoch 46/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0362 - mae: 0.1509 - val_loss: 0.
0513 - val_mae: 0.1782
Epoch 47/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0349 - mae: 0.1491 - val_loss: 0.
0424 - val_mae: 0.1612
Epoch 48/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0335 - mae: 0.1455 - val_loss: 0.
0452 - val_mae: 0.1667
Epoch 49/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0325 - mae: 0.1436 - val_loss: 0.
0427 - val_mae: 0.1623
Epoch 50/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0348 - mae: 0.1502 - val_loss: 0.
0446 - val_mae: 0.1635
```

In [21]:

```python
# Make predictions and evaluate the model
y_pred = model3.predict(x_test)
from sklearn.metrics import mean_squared_error, mean_absolute_error
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
print(f'Test MSE: {mse}, Test MAE: {mae}')
```

```
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
Test MSE: 0.04503756368444725, Test MAE: 0.16592758621392015
```