Dense+dropout+batch_normalization

November 7, 2024

Realizado por: Rodolfo Jesús Cruz Rebollar

Matrícula: A01368326

Grupo: 101

0.0.1 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.

0.0.2 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

[1]:
```
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, Dropout, BatchNormalization
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
c:\Users\Rodolfo\AppData\Local\Programs\Python\Python310\lib\site-
packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.17.3 and
<1.25.0 is required for this version of SciPy (detected version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

0.0.3 2) Load Data

[2]:
```
data = pd.read_csv("Student_performance_data _.csv")
data
```

[2]:

	StudentID	Age	Gender	Ethnicity	ParentalEducation	StudyTimeWeekly	Absences	Tutoring	ParentalSupport	Extracurricular	Sports	Music	Volunteering	GPA	GradeClass
0	1001	17	1	0	2	19.833723	7	1	2	0	0	1	0	2.929196	2.0
1	1002	18	0	0	1	15.408756	0	0	1	0	0	0	0		
2	1003	15	0	2	3	4.210570	26	0	2	0	0	0			
3	1004	17	1	0	3	10.028829	14	0	3	1	0	0			
4	1005	17	1	0	2	4.672495	17	1	3	0	0	0			
…	…	…	…	…	…	…	…	…	…	…	…	…			
2387	3388	18	1	0	3	10.680555	2	0	4	1	0	0			
2388	3389	17	0	0	1	7.583217	4	1	4	0	1	0			
2389	3390	16	1	0	2	6.805500	20	0	2	0	0	0			
2390	3391	16	1	1	0	12.416653	17	0	2	0	1	1			
2391	3392	16	1	0	2	17.819907	13	0	2	0	0	0			

```
1                0  3.042915         1.0
2                0  0.112602         4.0
3                0  2.054218         3.0
4                0  1.288061         4.0
…              …      …          …
2387             0  3.455509         0.0
2388             0  3.279150         4.0
2389             1  1.142333         2.0
2390             0  1.803297         1.0
2391             1  2.140014         1.0

[2392 rows x 15 columns]
```

0.0.4 3) Review you data:

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

[3]:
```
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
```

0.0.5 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'

[4]:
```
"""Columnas relevantes a seleccionar: Age, ParentalEducation, StudyTimeWeekly,␣
 ↪Absenses,
   ParentalSupport, Sports, GradeClass, GPA"""

dataset = data.iloc[:, [1, 4, 5, 6, 8, 10, 14, 13]]

dataset.head()
```

[4]:

	Age	ParentalEducation	StudyTimeWeekly	Absences	ParentalSupport	Sports	GradeClass	GPA
0	17	2	19.833723	7	2	0	2.0	2.929196
1	18	1	15.408756	0	1	0	1.0	3.042915
2	15	3	4.210570	26	2	0	4.0	0.112602
3	17	3	10.028829	14	3	0	3.0	2.054218
4	17	2	4.672495	17	3	0	4.0	1.288061

0.0.6 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

[5]:
```
# Revisar si existen valores nulos o faltantes en cada columna del dataset

dataset.isna().any()
```

[5]:
```
Age                 False
ParentalEducation   False
StudyTimeWeekly     False
Absences            False
ParentalSupport     False
Sports              False
GradeClass          False
GPA                 False
dtype: bool
```

0.0.7 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.

[6]:
```
# Dataset solo con variables predictoras sin la variable respuesta GPA

X = dataset.iloc[:, :7]

# Dataset solo con la variable respuesta GPA

y = dataset["GPA"]

# Crear conjuntos de entrenamiento y prueba para X, y

# Destinar 20% de los datos para probar el modelo y el 80% sobrante para␣
 ↪entrenarlo

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state=42)

# Estandarizar/normalizar los valores de las variables sin considerar a la␣
 ↪variable respuesta (GPA)

scaler = StandardScaler()

std_features = scaler.fit_transform(X_train, X_test)

std_features
```

[6]:
```
array([[ 1.37285117,  2.26211643,  1.46815853, …, -1.9146563 ,
         1.51039849,  0.82018081],
       [-0.40585814,  0.23853507, -1.27677348, …, -1.02021491,
        -0.66207693,  0.82018081],
       [ 0.48349652,  1.25032575, -1.10363153, …, -1.9146563 ,
        -0.66207693,  0.82018081],
       …,
       [-0.40585814,  0.23853507, -1.08325424, …, -0.12577352,
         1.51039849,  0.01141489],
       [ 1.37285117,  0.23853507, -0.93767144, …,  0.76866788,
```

```
 -0.66207693,  0.82018081],
[ 1.37285117,  0.23853507, -0.7578795 , …, -0.12577352,
 -0.66207693,  0.82018081]])
```

0.0.8 7. Different Neural Network Architectures

Nota: se utilizará un valor de dropout igual a 0.5 para los modelos 3 y 4.

Experiment 1: A single Dense Hidden Layer

[7]:
```
# Experiment 1: a single dense hidden layer

E1 = Sequential()

# Agregar capa de entrada con 64 unidades y activación Relu y dimensión de␣
 ↪entrada = 7

E1.add(Dense(64, activation="relu", input_dim = 7))

# Agregar capa densa oculta simple on 32 unidades y función de activación Relu

E1.add(Dense(32, activation="relu")) # single dense hidden layer

# Añadir capa de salida con 1 unidad

E1.add(Dense(1))
```

```
c:\Users\Rodolfo\AppData\Local\Programs\Python\Python310\lib\site-
packages\keras\src\layers\core\dense.py:87: 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]:
```
# Compilación del modelo 1

E1.compile(optimizer="adam", loss="mse", metrics=["mae"])
```

[9]:
```
# Entrenamiento del modelo 1

E1.fit(X_train, y_train, epochs=100, batch_size=10, validation_split=0.2)
```

```
Epoch 1/100
153/153             1s 3ms/step -
loss: 1.4415 - mae: 0.7374 - val_loss: 0.1245 - val_mae: 0.2772
Epoch 2/100
153/153             0s 2ms/step -
loss: 0.1208 - mae: 0.2763 - val_loss: 0.1206 - val_mae: 0.2810
Epoch 3/100
153/153             0s 2ms/step -
```

```
loss: 0.1037 - mae: 0.2628 - val_loss: 0.1306 - val_mae: 0.2953
Epoch 4/100
153/153             0s 2ms/step -
loss: 0.1067 - mae: 0.2661 - val_loss: 0.1083 - val_mae: 0.2636
Epoch 5/100
153/153             0s 2ms/step -
loss: 0.1097 - mae: 0.2660 - val_loss: 0.1131 - val_mae: 0.2734
Epoch 6/100
153/153             0s 2ms/step -
loss: 0.0986 - mae: 0.2502 - val_loss: 0.0924 - val_mae: 0.2438
Epoch 7/100
153/153             0s 2ms/step -
loss: 0.0929 - mae: 0.2446 - val_loss: 0.1059 - val_mae: 0.2624
Epoch 8/100
153/153             0s 2ms/step -
loss: 0.0942 - mae: 0.2486 - val_loss: 0.0959 - val_mae: 0.2505
Epoch 9/100
153/153             0s 2ms/step -
loss: 0.0883 - mae: 0.2388 - val_loss: 0.0981 - val_mae: 0.2533
Epoch 10/100
153/153             0s 2ms/step -
loss: 0.1028 - mae: 0.2536 - val_loss: 0.0879 - val_mae: 0.2400
Epoch 11/100
153/153             0s 2ms/step -
loss: 0.0915 - mae: 0.2464 - val_loss: 0.1296 - val_mae: 0.2895
Epoch 12/100
153/153             0s 2ms/step -
loss: 0.0880 - mae: 0.2341 - val_loss: 0.1150 - val_mae: 0.2711
Epoch 13/100
153/153             0s 2ms/step -
loss: 0.0885 - mae: 0.2409 - val_loss: 0.0868 - val_mae: 0.2371
Epoch 14/100
153/153             0s 2ms/step -
loss: 0.0808 - mae: 0.2270 - val_loss: 0.1074 - val_mae: 0.2658
Epoch 15/100
153/153             0s 2ms/step -
loss: 0.0899 - mae: 0.2424 - val_loss: 0.0863 - val_mae: 0.2338
Epoch 16/100
153/153             0s 2ms/step -
loss: 0.0868 - mae: 0.2321 - val_loss: 0.0821 - val_mae: 0.2269
Epoch 17/100
153/153             0s 2ms/step -
loss: 0.0777 - mae: 0.2245 - val_loss: 0.1180 - val_mae: 0.2836
Epoch 18/100
153/153             0s 3ms/step -
loss: 0.0829 - mae: 0.2306 - val_loss: 0.0790 - val_mae: 0.2254
Epoch 19/100
153/153             0s 2ms/step -
```

```
loss: 0.0709 - mae: 0.2144 - val_loss: 0.1200 - val_mae: 0.2776
Epoch 20/100
153/153             0s 2ms/step -
loss: 0.0842 - mae: 0.2337 - val_loss: 0.0727 - val_mae: 0.2138
Epoch 21/100
153/153             0s 2ms/step -
loss: 0.0757 - mae: 0.2221 - val_loss: 0.0860 - val_mae: 0.2356
Epoch 22/100
153/153             0s 2ms/step -
loss: 0.0745 - mae: 0.2174 - val_loss: 0.0715 - val_mae: 0.2119
Epoch 23/100
153/153             0s 2ms/step -
loss: 0.0783 - mae: 0.2234 - val_loss: 0.0965 - val_mae: 0.2525
Epoch 24/100
153/153             0s 2ms/step -
loss: 0.0858 - mae: 0.2312 - val_loss: 0.0748 - val_mae: 0.2166
Epoch 25/100
153/153             0s 2ms/step -
loss: 0.0800 - mae: 0.2242 - val_loss: 0.0737 - val_mae: 0.2173
Epoch 26/100
153/153             0s 2ms/step -
loss: 0.0686 - mae: 0.2090 - val_loss: 0.0688 - val_mae: 0.2076
Epoch 27/100
153/153             0s 2ms/step -
loss: 0.0693 - mae: 0.2141 - val_loss: 0.0770 - val_mae: 0.2203
Epoch 28/100
153/153             0s 2ms/step -
loss: 0.0686 - mae: 0.2092 - val_loss: 0.0854 - val_mae: 0.2326
Epoch 29/100
153/153             0s 2ms/step -
loss: 0.0738 - mae: 0.2151 - val_loss: 0.0688 - val_mae: 0.2087
Epoch 30/100
153/153             0s 2ms/step -
loss: 0.0692 - mae: 0.2081 - val_loss: 0.0669 - val_mae: 0.2064
Epoch 31/100
153/153             0s 2ms/step -
loss: 0.0634 - mae: 0.2010 - val_loss: 0.0814 - val_mae: 0.2259
Epoch 32/100
153/153             0s 2ms/step -
loss: 0.0830 - mae: 0.2311 - val_loss: 0.0790 - val_mae: 0.2304
Epoch 33/100
153/153             0s 2ms/step -
loss: 0.0663 - mae: 0.2071 - val_loss: 0.1089 - val_mae: 0.2715
Epoch 34/100
153/153             0s 2ms/step -
loss: 0.0749 - mae: 0.2193 - val_loss: 0.0755 - val_mae: 0.2222
Epoch 35/100
153/153             0s 2ms/step -
```

```
loss: 0.0702 - mae: 0.2115 - val_loss: 0.0687 - val_mae: 0.2105
Epoch 36/100
153/153             0s 2ms/step -
loss: 0.0628 - mae: 0.1975 - val_loss: 0.0786 - val_mae: 0.2221
Epoch 37/100
153/153             0s 2ms/step -
loss: 0.0692 - mae: 0.2114 - val_loss: 0.0881 - val_mae: 0.2386
Epoch 38/100
153/153             0s 2ms/step -
loss: 0.0696 - mae: 0.2116 - val_loss: 0.0766 - val_mae: 0.2252
Epoch 39/100
153/153             0s 2ms/step -
loss: 0.0669 - mae: 0.2045 - val_loss: 0.0882 - val_mae: 0.2389
Epoch 40/100
153/153             0s 2ms/step -
loss: 0.0634 - mae: 0.2011 - val_loss: 0.0855 - val_mae: 0.2356
Epoch 41/100
153/153             0s 2ms/step -
loss: 0.0657 - mae: 0.2023 - val_loss: 0.0667 - val_mae: 0.2052
Epoch 42/100
153/153             0s 2ms/step -
loss: 0.0629 - mae: 0.1993 - val_loss: 0.0636 - val_mae: 0.2030
Epoch 43/100
153/153             0s 2ms/step -
loss: 0.0622 - mae: 0.1979 - val_loss: 0.0627 - val_mae: 0.2003
Epoch 44/100
153/153             0s 2ms/step -
loss: 0.0564 - mae: 0.1892 - val_loss: 0.0918 - val_mae: 0.2491
Epoch 45/100
153/153             0s 2ms/step -
loss: 0.0661 - mae: 0.2054 - val_loss: 0.0609 - val_mae: 0.1985
Epoch 46/100
153/153             0s 2ms/step -
loss: 0.0596 - mae: 0.1935 - val_loss: 0.0621 - val_mae: 0.1995
Epoch 47/100
153/153             0s 2ms/step -
loss: 0.0628 - mae: 0.1996 - val_loss: 0.0847 - val_mae: 0.2402
Epoch 48/100
153/153             0s 2ms/step -
loss: 0.0626 - mae: 0.1970 - val_loss: 0.0689 - val_mae: 0.2119
Epoch 49/100
153/153             0s 2ms/step -
loss: 0.0586 - mae: 0.1917 - val_loss: 0.0659 - val_mae: 0.2045
Epoch 50/100
153/153             0s 2ms/step -
loss: 0.0606 - mae: 0.1965 - val_loss: 0.0579 - val_mae: 0.1919
Epoch 51/100
153/153             0s 2ms/step -
```

```
loss: 0.0562 - mae: 0.1912 - val_loss: 0.0692 - val_mae: 0.2136
Epoch 52/100
153/153             0s 2ms/step -
loss: 0.0519 - mae: 0.1810 - val_loss: 0.0684 - val_mae: 0.2135
Epoch 53/100
153/153             0s 2ms/step -
loss: 0.0580 - mae: 0.1922 - val_loss: 0.0594 - val_mae: 0.1931
Epoch 54/100
153/153             0s 2ms/step -
loss: 0.0619 - mae: 0.1979 - val_loss: 0.0662 - val_mae: 0.2035
Epoch 55/100
153/153             0s 2ms/step -
loss: 0.0609 - mae: 0.1943 - val_loss: 0.0558 - val_mae: 0.1884
Epoch 56/100
153/153             0s 2ms/step -
loss: 0.0558 - mae: 0.1892 - val_loss: 0.0672 - val_mae: 0.2046
Epoch 57/100
153/153             0s 2ms/step -
loss: 0.0610 - mae: 0.1965 - val_loss: 0.0565 - val_mae: 0.1886
Epoch 58/100
153/153             0s 2ms/step -
loss: 0.0540 - mae: 0.1843 - val_loss: 0.0614 - val_mae: 0.1975
Epoch 59/100
153/153             0s 2ms/step -
loss: 0.0514 - mae: 0.1792 - val_loss: 0.0652 - val_mae: 0.2077
Epoch 60/100
153/153             0s 2ms/step -
loss: 0.0584 - mae: 0.1908 - val_loss: 0.0817 - val_mae: 0.2341
Epoch 61/100
153/153             0s 2ms/step -
loss: 0.0567 - mae: 0.1898 - val_loss: 0.0652 - val_mae: 0.2084
Epoch 62/100
153/153             0s 2ms/step -
loss: 0.0542 - mae: 0.1862 - val_loss: 0.0701 - val_mae: 0.2076
Epoch 63/100
153/153             0s 2ms/step -
loss: 0.0540 - mae: 0.1834 - val_loss: 0.0667 - val_mae: 0.2132
Epoch 64/100
153/153             0s 2ms/step -
loss: 0.0518 - mae: 0.1792 - val_loss: 0.0666 - val_mae: 0.2018
Epoch 65/100
153/153             0s 2ms/step -
loss: 0.0630 - mae: 0.2008 - val_loss: 0.0713 - val_mae: 0.2120
Epoch 66/100
153/153             0s 2ms/step -
loss: 0.0542 - mae: 0.1864 - val_loss: 0.0577 - val_mae: 0.1909
Epoch 67/100
153/153             0s 2ms/step -
```

```
loss: 0.0546 - mae: 0.1856 - val_loss: 0.0584 - val_mae: 0.1963
Epoch 68/100
153/153             0s 2ms/step -
loss: 0.0496 - mae: 0.1768 - val_loss: 0.0805 - val_mae: 0.2240
Epoch 69/100
153/153             0s 2ms/step -
loss: 0.0542 - mae: 0.1877 - val_loss: 0.0579 - val_mae: 0.1878
Epoch 70/100
153/153             0s 2ms/step -
loss: 0.0511 - mae: 0.1794 - val_loss: 0.0601 - val_mae: 0.1982
Epoch 71/100
153/153             0s 2ms/step -
loss: 0.0557 - mae: 0.1866 - val_loss: 0.0584 - val_mae: 0.1965
Epoch 72/100
153/153             0s 2ms/step -
loss: 0.0496 - mae: 0.1721 - val_loss: 0.0622 - val_mae: 0.2001
Epoch 73/100
153/153             0s 2ms/step -
loss: 0.0481 - mae: 0.1739 - val_loss: 0.0697 - val_mae: 0.2077
Epoch 74/100
153/153             0s 2ms/step -
loss: 0.0552 - mae: 0.1837 - val_loss: 0.0601 - val_mae: 0.1947
Epoch 75/100
153/153             0s 2ms/step -
loss: 0.0501 - mae: 0.1774 - val_loss: 0.0574 - val_mae: 0.1908
Epoch 76/100
153/153             0s 2ms/step -
loss: 0.0503 - mae: 0.1769 - val_loss: 0.0591 - val_mae: 0.1909
Epoch 77/100
153/153             0s 2ms/step -
loss: 0.0460 - mae: 0.1684 - val_loss: 0.0627 - val_mae: 0.1965
Epoch 78/100
153/153             0s 2ms/step -
loss: 0.0479 - mae: 0.1722 - val_loss: 0.0590 - val_mae: 0.1934
Epoch 79/100
153/153             0s 2ms/step -
loss: 0.0506 - mae: 0.1767 - val_loss: 0.0563 - val_mae: 0.1872
Epoch 80/100
153/153             0s 2ms/step -
loss: 0.0517 - mae: 0.1799 - val_loss: 0.0655 - val_mae: 0.2124
Epoch 81/100
153/153             0s 2ms/step -
loss: 0.0488 - mae: 0.1736 - val_loss: 0.0531 - val_mae: 0.1816
Epoch 82/100
153/153             0s 2ms/step -
loss: 0.0451 - mae: 0.1665 - val_loss: 0.0678 - val_mae: 0.2169
Epoch 83/100
153/153             0s 2ms/step -
```

```
loss: 0.0561 - mae: 0.1870 - val_loss: 0.0701 - val_mae: 0.2058
Epoch 84/100
153/153             0s 2ms/step -
loss: 0.0507 - mae: 0.1787 - val_loss: 0.0696 - val_mae: 0.2127
Epoch 85/100
153/153             0s 2ms/step -
loss: 0.0510 - mae: 0.1795 - val_loss: 0.0559 - val_mae: 0.1903
Epoch 86/100
153/153             0s 2ms/step -
loss: 0.0477 - mae: 0.1717 - val_loss: 0.0529 - val_mae: 0.1812
Epoch 87/100
153/153             0s 2ms/step -
loss: 0.0481 - mae: 0.1721 - val_loss: 0.0559 - val_mae: 0.1893
Epoch 88/100
153/153             0s 2ms/step -
loss: 0.0474 - mae: 0.1718 - val_loss: 0.0588 - val_mae: 0.1912
Epoch 89/100
153/153             0s 2ms/step -
loss: 0.0493 - mae: 0.1764 - val_loss: 0.0529 - val_mae: 0.1836
Epoch 90/100
153/153             0s 2ms/step -
loss: 0.0446 - mae: 0.1695 - val_loss: 0.0853 - val_mae: 0.2254
Epoch 91/100
153/153             0s 2ms/step -
loss: 0.0464 - mae: 0.1667 - val_loss: 0.0671 - val_mae: 0.2124
Epoch 92/100
153/153             0s 2ms/step -
loss: 0.0511 - mae: 0.1784 - val_loss: 0.0545 - val_mae: 0.1864
Epoch 93/100
153/153             0s 2ms/step -
loss: 0.0524 - mae: 0.1785 - val_loss: 0.0537 - val_mae: 0.1853
Epoch 94/100
153/153             0s 2ms/step -
loss: 0.0445 - mae: 0.1632 - val_loss: 0.0552 - val_mae: 0.1893
Epoch 95/100
153/153             0s 2ms/step -
loss: 0.0484 - mae: 0.1732 - val_loss: 0.0578 - val_mae: 0.1890
Epoch 96/100
153/153             0s 2ms/step -
loss: 0.0492 - mae: 0.1724 - val_loss: 0.0598 - val_mae: 0.1996
Epoch 97/100
153/153             0s 2ms/step -
loss: 0.0458 - mae: 0.1692 - val_loss: 0.0536 - val_mae: 0.1844
Epoch 98/100
153/153             0s 2ms/step -
loss: 0.0440 - mae: 0.1633 - val_loss: 0.0522 - val_mae: 0.1786
Epoch 99/100
153/153             0s 2ms/step -
```

```
loss: 0.0445 - mae: 0.1664 - val_loss: 0.0499 - val_mae: 0.1759
Epoch 100/100
153/153             0s 2ms/step -
loss: 0.0442 - mae: 0.1646 - val_loss: 0.0582 - val_mae: 0.1939
```

[9]:
```
<keras.src.callbacks.history.History at 0x1e0a281cac0>
```

[26]:
```
# Evaluar el modelo 1

eval1 = E1.evaluate(X_test, y_test)

print(f'Valores de pérdida modelo 1: {eval1}')
```

```
15/15             0s 2ms/step - loss:
0.0587 - mae: 0.1782
Valores de pérdida modelo 1: [0.05687776580452919, 0.1798904687166214]
```

Experiment 2: A set of three Dense Hidden Layers

[11]:
```
# Experiment 2: a single dense hidden layer

E2 = Sequential()

# Agregar capa de entrada con 64 unidades y activación Relu y dimensión de␣
 ↪entrada = 7

E2.add(Dense(64, activation="relu", input_dim = 7))

# Agregar capa densa oculta 1 con 32 unidades y función de activación Relu

E2.add(Dense(32, activation="relu")) # dense hidden layer 1

# Agregar capa densa oculta 2 con 16 unidades y función de activación Relu

E2.add(Dense(16, activation="relu")) # dense hidden layer 2

# Agregar capa densa oculta 3 con 8 unidades y función de activación Relu

E2.add(Dense(8, activation="relu")) # dense hidden layer 3

# Añadir capa de salida con 1 unidad

E2.add(Dense(1))
```

[12]:
```
# Compilación del modelo 2 (con 3 capas densas ocultas)

E2.compile(optimizer = "adam", loss = "mse", metrics = ["mae"])
```

[13]:
```
# Entrenamiento del modelo 2

E2.fit(X_train, y_train, epochs=100, batch_size=10, validation_split=0.2)
```

```
Epoch 1/100
153/153             2s 3ms/step -
loss: 1.0956 - mae: 0.6838 - val_loss: 0.1627 - val_mae: 0.3234
Epoch 2/100
153/153             0s 2ms/step -
loss: 0.1252 - mae: 0.2833 - val_loss: 0.1401 - val_mae: 0.3048
Epoch 3/100
153/153             0s 2ms/step -
loss: 0.0974 - mae: 0.2575 - val_loss: 0.1196 - val_mae: 0.2815
Epoch 4/100
153/153             0s 2ms/step -
loss: 0.0999 - mae: 0.2539 - val_loss: 0.1139 - val_mae: 0.2712
Epoch 5/100
153/153             0s 2ms/step -
loss: 0.0912 - mae: 0.2432 - val_loss: 0.1526 - val_mae: 0.3190
Epoch 6/100
153/153             0s 2ms/step -
loss: 0.1055 - mae: 0.2603 - val_loss: 0.1051 - val_mae: 0.2596
Epoch 7/100
153/153             0s 2ms/step -
loss: 0.0904 - mae: 0.2425 - val_loss: 0.0917 - val_mae: 0.2423
Epoch 8/100
153/153             0s 2ms/step -
loss: 0.0874 - mae: 0.2375 - val_loss: 0.1007 - val_mae: 0.2557
Epoch 9/100
153/153             0s 2ms/step -
loss: 0.0876 - mae: 0.2348 - val_loss: 0.0982 - val_mae: 0.2522
Epoch 10/100
153/153             0s 2ms/step -
loss: 0.0847 - mae: 0.2305 - val_loss: 0.0983 - val_mae: 0.2533
Epoch 11/100
153/153             0s 2ms/step -
loss: 0.0789 - mae: 0.2264 - val_loss: 0.0988 - val_mae: 0.2550
Epoch 12/100
153/153             0s 2ms/step -
loss: 0.0806 - mae: 0.2306 - val_loss: 0.0920 - val_mae: 0.2435
Epoch 13/100
153/153             0s 2ms/step -
loss: 0.0786 - mae: 0.2248 - val_loss: 0.0844 - val_mae: 0.2324
Epoch 14/100
153/153             0s 2ms/step -
loss: 0.0724 - mae: 0.2159 - val_loss: 0.0982 - val_mae: 0.2544
Epoch 15/100
153/153             0s 2ms/step -
```

```
loss: 0.0781 - mae: 0.2255 - val_loss: 0.0840 - val_mae: 0.2320
Epoch 16/100
153/153             0s 2ms/step -
loss: 0.0724 - mae: 0.2156 - val_loss: 0.0801 - val_mae: 0.2266
Epoch 17/100
153/153             0s 2ms/step -
loss: 0.0736 - mae: 0.2156 - val_loss: 0.0808 - val_mae: 0.2277
Epoch 18/100
153/153             0s 2ms/step -
loss: 0.0766 - mae: 0.2210 - val_loss: 0.0803 - val_mae: 0.2276
Epoch 19/100
153/153             0s 2ms/step -
loss: 0.0681 - mae: 0.2093 - val_loss: 0.0938 - val_mae: 0.2463
Epoch 20/100
153/153             0s 2ms/step -
loss: 0.0745 - mae: 0.2201 - val_loss: 0.0862 - val_mae: 0.2354
Epoch 21/100
153/153             0s 2ms/step -
loss: 0.0744 - mae: 0.2156 - val_loss: 0.0827 - val_mae: 0.2322
Epoch 22/100
153/153             0s 2ms/step -
loss: 0.0700 - mae: 0.2097 - val_loss: 0.0839 - val_mae: 0.2333
Epoch 23/100
153/153             0s 2ms/step -
loss: 0.0691 - mae: 0.2116 - val_loss: 0.0825 - val_mae: 0.2308
Epoch 24/100
153/153             0s 2ms/step -
loss: 0.0643 - mae: 0.2036 - val_loss: 0.0959 - val_mae: 0.2467
Epoch 25/100
153/153             0s 2ms/step -
loss: 0.0684 - mae: 0.2091 - val_loss: 0.0831 - val_mae: 0.2299
Epoch 26/100
153/153             0s 2ms/step -
loss: 0.0684 - mae: 0.2062 - val_loss: 0.0698 - val_mae: 0.2107
Epoch 27/100
153/153             0s 2ms/step -
loss: 0.0690 - mae: 0.2061 - val_loss: 0.0720 - val_mae: 0.2146
Epoch 28/100
153/153             0s 2ms/step -
loss: 0.0605 - mae: 0.1952 - val_loss: 0.0749 - val_mae: 0.2190
Epoch 29/100
153/153             0s 2ms/step -
loss: 0.0578 - mae: 0.1921 - val_loss: 0.0704 - val_mae: 0.2123
Epoch 30/100
153/153             0s 2ms/step -
loss: 0.0646 - mae: 0.2018 - val_loss: 0.0766 - val_mae: 0.2221
Epoch 31/100
153/153             0s 2ms/step -
```

```
loss: 0.0628 - mae: 0.1992 - val_loss: 0.0728 - val_mae: 0.2136
Epoch 32/100
153/153             0s 2ms/step -
loss: 0.0620 - mae: 0.1992 - val_loss: 0.0687 - val_mae: 0.2089
Epoch 33/100
153/153             0s 2ms/step -
loss: 0.0602 - mae: 0.1946 - val_loss: 0.0669 - val_mae: 0.2054
Epoch 34/100
153/153             0s 2ms/step -
loss: 0.0581 - mae: 0.1926 - val_loss: 0.0676 - val_mae: 0.2091
Epoch 35/100
153/153             0s 2ms/step -
loss: 0.0585 - mae: 0.1933 - val_loss: 0.0671 - val_mae: 0.2049
Epoch 36/100
153/153             0s 2ms/step -
loss: 0.0594 - mae: 0.1953 - val_loss: 0.0626 - val_mae: 0.1987
Epoch 37/100
153/153             0s 2ms/step -
loss: 0.0580 - mae: 0.1935 - val_loss: 0.0680 - val_mae: 0.2070
Epoch 38/100
153/153             0s 2ms/step -
loss: 0.0625 - mae: 0.1977 - val_loss: 0.0689 - val_mae: 0.2096
Epoch 39/100
153/153             0s 2ms/step -
loss: 0.0573 - mae: 0.1893 - val_loss: 0.0671 - val_mae: 0.2079
Epoch 40/100
153/153             0s 2ms/step -
loss: 0.0506 - mae: 0.1747 - val_loss: 0.0609 - val_mae: 0.1980
Epoch 41/100
153/153             0s 2ms/step -
loss: 0.0585 - mae: 0.1919 - val_loss: 0.0661 - val_mae: 0.2060
Epoch 42/100
153/153             0s 2ms/step -
loss: 0.0584 - mae: 0.1906 - val_loss: 0.0623 - val_mae: 0.1975
Epoch 43/100
153/153             0s 2ms/step -
loss: 0.0568 - mae: 0.1903 - val_loss: 0.0616 - val_mae: 0.1985
Epoch 44/100
153/153             0s 2ms/step -
loss: 0.0618 - mae: 0.1949 - val_loss: 0.0718 - val_mae: 0.2196
Epoch 45/100
153/153             0s 2ms/step -
loss: 0.0587 - mae: 0.1934 - val_loss: 0.0762 - val_mae: 0.2189
Epoch 46/100
153/153             0s 2ms/step -
loss: 0.0521 - mae: 0.1760 - val_loss: 0.0644 - val_mae: 0.2041
Epoch 47/100
153/153             0s 2ms/step -
```

```
loss: 0.0524 - mae: 0.1809 - val_loss: 0.0620 - val_mae: 0.1980
Epoch 48/100
153/153             0s 2ms/step -
loss: 0.0565 - mae: 0.1884 - val_loss: 0.0556 - val_mae: 0.1883
Epoch 49/100
153/153             0s 2ms/step -
loss: 0.0530 - mae: 0.1818 - val_loss: 0.0568 - val_mae: 0.1895
Epoch 50/100
153/153             0s 2ms/step -
loss: 0.0481 - mae: 0.1728 - val_loss: 0.0681 - val_mae: 0.2132
Epoch 51/100
153/153             0s 2ms/step -
loss: 0.0569 - mae: 0.1884 - val_loss: 0.0581 - val_mae: 0.1926
Epoch 52/100
153/153             0s 2ms/step -
loss: 0.0499 - mae: 0.1748 - val_loss: 0.0576 - val_mae: 0.1903
Epoch 53/100
153/153             0s 2ms/step -
loss: 0.0511 - mae: 0.1752 - val_loss: 0.0629 - val_mae: 0.1963
Epoch 54/100
153/153             0s 2ms/step -
loss: 0.0500 - mae: 0.1742 - val_loss: 0.0550 - val_mae: 0.1839
Epoch 55/100
153/153             0s 2ms/step -
loss: 0.0480 - mae: 0.1719 - val_loss: 0.0647 - val_mae: 0.1984
Epoch 56/100
153/153             0s 2ms/step -
loss: 0.0465 - mae: 0.1689 - val_loss: 0.0552 - val_mae: 0.1852
Epoch 57/100
153/153             0s 2ms/step -
loss: 0.0519 - mae: 0.1778 - val_loss: 0.0582 - val_mae: 0.1887
Epoch 58/100
153/153             0s 2ms/step -
loss: 0.0479 - mae: 0.1727 - val_loss: 0.0548 - val_mae: 0.1894
Epoch 59/100
153/153             0s 2ms/step -
loss: 0.0499 - mae: 0.1762 - val_loss: 0.0591 - val_mae: 0.1951
Epoch 60/100
153/153             0s 2ms/step -
loss: 0.0490 - mae: 0.1735 - val_loss: 0.0613 - val_mae: 0.1957
Epoch 61/100
153/153             0s 2ms/step -
loss: 0.0463 - mae: 0.1706 - val_loss: 0.0554 - val_mae: 0.1896
Epoch 62/100
153/153             0s 2ms/step -
loss: 0.0501 - mae: 0.1759 - val_loss: 0.0625 - val_mae: 0.2017
Epoch 63/100
153/153             0s 2ms/step -
```

```
loss: 0.0477 - mae: 0.1720 - val_loss: 0.0553 - val_mae: 0.1878
Epoch 64/100
153/153             0s 2ms/step -
loss: 0.0459 - mae: 0.1708 - val_loss: 0.0519 - val_mae: 0.1803
Epoch 65/100
153/153             0s 2ms/step -
loss: 0.0472 - mae: 0.1709 - val_loss: 0.0563 - val_mae: 0.1911
Epoch 66/100
153/153             0s 2ms/step -
loss: 0.0491 - mae: 0.1747 - val_loss: 0.0571 - val_mae: 0.1904
Epoch 67/100
153/153             0s 2ms/step -
loss: 0.0464 - mae: 0.1705 - val_loss: 0.0541 - val_mae: 0.1831
Epoch 68/100
153/153             0s 2ms/step -
loss: 0.0456 - mae: 0.1675 - val_loss: 0.0556 - val_mae: 0.1903
Epoch 69/100
153/153             0s 2ms/step -
loss: 0.0414 - mae: 0.1595 - val_loss: 0.0522 - val_mae: 0.1762
Epoch 70/100
153/153             0s 2ms/step -
loss: 0.0427 - mae: 0.1631 - val_loss: 0.0533 - val_mae: 0.1855
Epoch 71/100
153/153             0s 2ms/step -
loss: 0.0430 - mae: 0.1603 - val_loss: 0.0591 - val_mae: 0.1908
Epoch 72/100
153/153             0s 2ms/step -
loss: 0.0420 - mae: 0.1612 - val_loss: 0.0552 - val_mae: 0.1865
Epoch 73/100
153/153             0s 2ms/step -
loss: 0.0458 - mae: 0.1671 - val_loss: 0.0521 - val_mae: 0.1823
Epoch 74/100
153/153             0s 2ms/step -
loss: 0.0453 - mae: 0.1640 - val_loss: 0.0511 - val_mae: 0.1798
Epoch 75/100
153/153             0s 2ms/step -
loss: 0.0456 - mae: 0.1685 - val_loss: 0.0498 - val_mae: 0.1743
Epoch 76/100
153/153             0s 2ms/step -
loss: 0.0435 - mae: 0.1659 - val_loss: 0.0660 - val_mae: 0.2023
Epoch 77/100
153/153             0s 2ms/step -
loss: 0.0427 - mae: 0.1632 - val_loss: 0.0622 - val_mae: 0.2051
Epoch 78/100
153/153             0s 2ms/step -
loss: 0.0449 - mae: 0.1664 - val_loss: 0.0497 - val_mae: 0.1767
Epoch 79/100
153/153             0s 2ms/step -
```

```
loss: 0.0456 - mae: 0.1691 - val_loss: 0.0538 - val_mae: 0.1884
Epoch 80/100
153/153             0s 2ms/step -
loss: 0.0480 - mae: 0.1725 - val_loss: 0.0588 - val_mae: 0.1947
Epoch 81/100
153/153             0s 2ms/step -
loss: 0.0414 - mae: 0.1606 - val_loss: 0.0580 - val_mae: 0.1918
Epoch 82/100
153/153             0s 2ms/step -
loss: 0.0450 - mae: 0.1631 - val_loss: 0.0517 - val_mae: 0.1782
Epoch 83/100
153/153             0s 2ms/step -
loss: 0.0442 - mae: 0.1653 - val_loss: 0.0494 - val_mae: 0.1747
Epoch 84/100
153/153             0s 2ms/step -
loss: 0.0472 - mae: 0.1699 - val_loss: 0.0492 - val_mae: 0.1724
Epoch 85/100
153/153             0s 2ms/step -
loss: 0.0418 - mae: 0.1598 - val_loss: 0.0509 - val_mae: 0.1794
Epoch 86/100
153/153             0s 2ms/step -
loss: 0.0450 - mae: 0.1652 - val_loss: 0.0505 - val_mae: 0.1784
Epoch 87/100
153/153             0s 2ms/step -
loss: 0.0433 - mae: 0.1637 - val_loss: 0.0563 - val_mae: 0.1817
Epoch 88/100
153/153             0s 2ms/step -
loss: 0.0398 - mae: 0.1576 - val_loss: 0.0576 - val_mae: 0.1876
Epoch 89/100
153/153             0s 2ms/step -
loss: 0.0427 - mae: 0.1615 - val_loss: 0.0527 - val_mae: 0.1825
Epoch 90/100
153/153             0s 2ms/step -
loss: 0.0448 - mae: 0.1655 - val_loss: 0.0537 - val_mae: 0.1837
Epoch 91/100
153/153             0s 2ms/step -
loss: 0.0445 - mae: 0.1681 - val_loss: 0.0495 - val_mae: 0.1750
Epoch 92/100
153/153             0s 2ms/step -
loss: 0.0403 - mae: 0.1572 - val_loss: 0.0510 - val_mae: 0.1743
Epoch 93/100
153/153             0s 2ms/step -
loss: 0.0449 - mae: 0.1676 - val_loss: 0.0502 - val_mae: 0.1759
Epoch 94/100
153/153             0s 2ms/step -
loss: 0.0431 - mae: 0.1635 - val_loss: 0.0546 - val_mae: 0.1902
Epoch 95/100
153/153             0s 2ms/step -
```

```
loss: 0.0409 - mae: 0.1576 - val_loss: 0.0557 - val_mae: 0.1854
Epoch 96/100
153/153             0s 2ms/step -
loss: 0.0418 - mae: 0.1615 - val_loss: 0.0511 - val_mae: 0.1765
Epoch 97/100
153/153             0s 2ms/step -
loss: 0.0418 - mae: 0.1583 - val_loss: 0.0513 - val_mae: 0.1760
Epoch 98/100
153/153             0s 2ms/step -
loss: 0.0421 - mae: 0.1602 - val_loss: 0.0545 - val_mae: 0.1897
Epoch 99/100
153/153             0s 2ms/step -
loss: 0.0390 - mae: 0.1557 - val_loss: 0.0504 - val_mae: 0.1774
Epoch 100/100
153/153             0s 2ms/step -
loss: 0.0380 - mae: 0.1547 - val_loss: 0.0507 - val_mae: 0.1768
```

[13]:
```
<keras.src.callbacks.history.History at 0x1e0a5045b10>
```

[14]:
```
# Evaluar el modelo 2

eval2 = E2.evaluate(X_test, y_test)

print(f'Valores de pérdida modelo 2: {eval2}')
```

```
15/15             0s 1ms/step - loss:
0.0534 - mae: 0.1711
15/15             0s 1ms/step - loss:
0.0534 - mae: 0.1711
Valores de pérdida modelo 2: [0.05030996352434158, 0.16920119524002075]
```

Experiment 3: Add a dropout layer after each Dense Hidden Layer

[15]:
```
# Experiment 3: a single dense hidden layer

E3 = Sequential()

# Agregar capa de entrada con 64 unidades y activación Relu y dimensión de␣
 ↪entrada = 7

E3.add(Dense(64, activation="relu", input_dim = 7))

# Agregar capa densa oculta 1 con 32 unidades y función de activación Relu

E3.add(Dense(32, activation="relu")) # dense hidden layer 1

# Agregar una capa de dropout después del dense hidden layer 1

E3.add(Dropout(0.5)) # dropout layer after dense hidden layer 1
```

```
# Agregar capa densa oculta 2 con 16 unidades y función de activación Relu

E3.add(Dense(16, activation="relu")) # dense hidden layer 2

# Agregar capa de dropout después del dense hidden layer 2

E3.add(Dropout(0.5)) # dropout layer after dense hidden layer 2

# Agregar capa densa oculta 3 con 8 unidades y función de activación Relu

E3.add(Dense(8, activation="relu")) # dense hidden layer 3

# Agregar capa de dropout después del dense hidden layer 3

E3.add(Dropout(0.5)) # dropout layer after dense hidden layer 3

# Añadir capa de salida con 1 unidad

E3.add(Dense(1))
```

[16]:
```
# Compilación del modelo 3 (con dropout layer después de cada hidden layer)

E3.compile(optimizer="adam", loss="mse", metrics=["mae"])
```

[17]:
```
# Entrenamiento del modelo 3

E3.fit(X_train, y_train, epochs=100, batch_size=10, validation_split=0.2)
```

```
Epoch 1/100
153/153             2s 3ms/step -
loss: 17.4613 - mae: 3.0271 - val_loss: 2.7426 - val_mae: 1.4520
Epoch 2/100
153/153             0s 2ms/step -
loss: 3.0943 - mae: 1.4541 - val_loss: 1.8519 - val_mae: 1.1751
Epoch 3/100
153/153             0s 2ms/step -
loss: 2.1910 - mae: 1.1876 - val_loss: 1.3071 - val_mae: 0.9670
Epoch 4/100
153/153             0s 2ms/step -
loss: 1.6676 - mae: 1.0221 - val_loss: 1.0046 - val_mae: 0.8487
Epoch 5/100
153/153             0s 2ms/step -
loss: 1.4910 - mae: 0.9591 - val_loss: 0.8829 - val_mae: 0.7992
Epoch 6/100
153/153             0s 2ms/step -
loss: 1.1649 - mae: 0.8479 - val_loss: 0.9469 - val_mae: 0.8215
Epoch 7/100
```

```
153/153             0s 2ms/step -
loss: 1.1817 - mae: 0.8428 - val_loss: 0.8188 - val_mae: 0.7659
Epoch 8/100
153/153             0s 2ms/step -
loss: 1.1083 - mae: 0.8395 - val_loss: 0.5884 - val_mae: 0.6489
Epoch 9/100
153/153             0s 2ms/step -
loss: 1.0054 - mae: 0.7809 - val_loss: 0.7008 - val_mae: 0.7107
Epoch 10/100
153/153             0s 2ms/step -
loss: 0.9831 - mae: 0.7736 - val_loss: 0.5494 - val_mae: 0.6385
Epoch 11/100
153/153             0s 2ms/step -
loss: 0.9273 - mae: 0.7525 - val_loss: 0.6844 - val_mae: 0.7050
Epoch 12/100
153/153             0s 2ms/step -
loss: 0.8844 - mae: 0.7408 - val_loss: 0.5810 - val_mae: 0.6404
Epoch 13/100
153/153             0s 2ms/step -
loss: 0.7949 - mae: 0.6893 - val_loss: 0.4390 - val_mae: 0.5566
Epoch 14/100
153/153             0s 2ms/step -
loss: 0.7574 - mae: 0.6810 - val_loss: 0.4914 - val_mae: 0.5955
Epoch 15/100
153/153             0s 2ms/step -
loss: 0.7478 - mae: 0.6708 - val_loss: 0.5740 - val_mae: 0.6393
Epoch 16/100
153/153             0s 2ms/step -
loss: 0.7565 - mae: 0.6839 - val_loss: 0.4611 - val_mae: 0.5742
Epoch 17/100
153/153             0s 2ms/step -
loss: 0.7230 - mae: 0.6616 - val_loss: 0.4653 - val_mae: 0.5687
Epoch 18/100
153/153             0s 2ms/step -
loss: 0.6341 - mae: 0.6210 - val_loss: 0.5572 - val_mae: 0.6321
Epoch 19/100
153/153             0s 2ms/step -
loss: 0.5784 - mae: 0.5922 - val_loss: 0.3470 - val_mae: 0.4944
Epoch 20/100
153/153             0s 2ms/step -
loss: 0.6608 - mae: 0.6229 - val_loss: 0.4524 - val_mae: 0.5642
Epoch 21/100
153/153             0s 2ms/step -
loss: 0.6123 - mae: 0.6170 - val_loss: 0.5279 - val_mae: 0.6119
Epoch 22/100
153/153             0s 2ms/step -
loss: 0.5612 - mae: 0.5921 - val_loss: 0.4531 - val_mae: 0.5699
Epoch 23/100
```

```
153/153            0s 2ms/step -
loss: 0.5613 - mae: 0.5909 - val_loss: 0.3673 - val_mae: 0.5102
Epoch 24/100
153/153            0s 2ms/step -
loss: 0.5932 - mae: 0.6092 - val_loss: 0.4927 - val_mae: 0.5918
Epoch 25/100
153/153            0s 2ms/step -
loss: 0.5053 - mae: 0.5549 - val_loss: 0.4421 - val_mae: 0.5565
Epoch 26/100
153/153            0s 2ms/step -
loss: 0.6065 - mae: 0.6062 - val_loss: 0.4432 - val_mae: 0.5575
Epoch 27/100
153/153            0s 3ms/step -
loss: 0.5229 - mae: 0.5615 - val_loss: 0.4949 - val_mae: 0.6003
Epoch 28/100
153/153            0s 2ms/step -
loss: 0.5128 - mae: 0.5570 - val_loss: 0.4529 - val_mae: 0.5660
Epoch 29/100
153/153            0s 2ms/step -
loss: 0.5265 - mae: 0.5634 - val_loss: 0.4570 - val_mae: 0.5683
Epoch 30/100
153/153            0s 2ms/step -
loss: 0.4872 - mae: 0.5498 - val_loss: 0.4612 - val_mae: 0.5728
Epoch 31/100
153/153            0s 2ms/step -
loss: 0.5006 - mae: 0.5527 - val_loss: 0.4505 - val_mae: 0.5624
Epoch 32/100
153/153            0s 2ms/step -
loss: 0.4827 - mae: 0.5526 - val_loss: 0.4664 - val_mae: 0.5736
Epoch 33/100
153/153            0s 2ms/step -
loss: 0.4745 - mae: 0.5464 - val_loss: 0.4684 - val_mae: 0.5704
Epoch 34/100
153/153            0s 2ms/step -
loss: 0.4587 - mae: 0.5287 - val_loss: 0.4125 - val_mae: 0.5373
Epoch 35/100
153/153            0s 2ms/step -
loss: 0.4936 - mae: 0.5554 - val_loss: 0.4775 - val_mae: 0.5827
Epoch 36/100
153/153            0s 2ms/step -
loss: 0.4563 - mae: 0.5320 - val_loss: 0.4801 - val_mae: 0.5727
Epoch 37/100
153/153            0s 2ms/step -
loss: 0.5336 - mae: 0.5680 - val_loss: 0.4391 - val_mae: 0.5507
Epoch 38/100
153/153            0s 2ms/step -
loss: 0.5011 - mae: 0.5539 - val_loss: 0.4677 - val_mae: 0.5710
Epoch 39/100
```

```
153/153            0s 2ms/step -
loss: 0.4571 - mae: 0.5287 - val_loss: 0.4468 - val_mae: 0.5668
Epoch 40/100
153/153            0s 2ms/step -
loss: 0.4782 - mae: 0.5480 - val_loss: 0.4329 - val_mae: 0.5480
Epoch 41/100
153/153            0s 2ms/step -
loss: 0.4610 - mae: 0.5335 - val_loss: 0.3942 - val_mae: 0.5238
Epoch 42/100
153/153            0s 2ms/step -
loss: 0.4603 - mae: 0.5228 - val_loss: 0.4282 - val_mae: 0.5454
Epoch 43/100
153/153            0s 2ms/step -
loss: 0.4868 - mae: 0.5469 - val_loss: 0.3877 - val_mae: 0.5169
Epoch 44/100
153/153            0s 2ms/step -
loss: 0.4752 - mae: 0.5400 - val_loss: 0.4047 - val_mae: 0.5356
Epoch 45/100
153/153            0s 2ms/step -
loss: 0.4662 - mae: 0.5287 - val_loss: 0.4487 - val_mae: 0.5646
Epoch 46/100
153/153            0s 2ms/step -
loss: 0.4392 - mae: 0.5207 - val_loss: 0.4431 - val_mae: 0.5573
Epoch 47/100
153/153            0s 2ms/step -
loss: 0.4081 - mae: 0.5029 - val_loss: 0.4224 - val_mae: 0.5431
Epoch 48/100
153/153            0s 2ms/step -
loss: 0.4686 - mae: 0.5362 - val_loss: 0.4092 - val_mae: 0.5320
Epoch 49/100
153/153            0s 2ms/step -
loss: 0.4359 - mae: 0.5146 - val_loss: 0.3812 - val_mae: 0.5139
Epoch 50/100
153/153            0s 2ms/step -
loss: 0.4422 - mae: 0.5199 - val_loss: 0.4247 - val_mae: 0.5412
Epoch 51/100
153/153            0s 3ms/step -
loss: 0.4723 - mae: 0.5394 - val_loss: 0.3969 - val_mae: 0.5275
Epoch 52/100
153/153            0s 3ms/step -
loss: 0.4428 - mae: 0.5227 - val_loss: 0.4113 - val_mae: 0.5323
Epoch 53/100
153/153            0s 2ms/step -
loss: 0.4091 - mae: 0.4940 - val_loss: 0.3711 - val_mae: 0.5081
Epoch 54/100
153/153            0s 3ms/step -
loss: 0.4228 - mae: 0.4978 - val_loss: 0.3949 - val_mae: 0.5192
Epoch 55/100
```

```
153/153             0s 3ms/step -
loss: 0.4301 - mae: 0.5097 - val_loss: 0.4087 - val_mae: 0.5315
Epoch 56/100
153/153             0s 3ms/step -
loss: 0.4404 - mae: 0.5200 - val_loss: 0.4066 - val_mae: 0.5288
Epoch 57/100
153/153             0s 2ms/step -
loss: 0.4436 - mae: 0.5159 - val_loss: 0.4711 - val_mae: 0.5697
Epoch 58/100
153/153             0s 2ms/step -
loss: 0.4383 - mae: 0.5200 - val_loss: 0.3831 - val_mae: 0.5137
Epoch 59/100
153/153             0s 2ms/step -
loss: 0.4439 - mae: 0.5252 - val_loss: 0.4089 - val_mae: 0.5316
Epoch 60/100
153/153             0s 2ms/step -
loss: 0.3765 - mae: 0.4783 - val_loss: 0.3867 - val_mae: 0.5171
Epoch 61/100
153/153             0s 2ms/step -
loss: 0.4060 - mae: 0.4919 - val_loss: 0.3481 - val_mae: 0.4884
Epoch 62/100
153/153             0s 2ms/step -
loss: 0.4170 - mae: 0.4922 - val_loss: 0.3552 - val_mae: 0.4916
Epoch 63/100
153/153             0s 2ms/step -
loss: 0.4282 - mae: 0.5094 - val_loss: 0.3170 - val_mae: 0.4676
Epoch 64/100
153/153             0s 2ms/step -
loss: 0.4169 - mae: 0.5011 - val_loss: 0.3012 - val_mae: 0.4581
Epoch 65/100
153/153             0s 2ms/step -
loss: 0.4178 - mae: 0.5020 - val_loss: 0.3378 - val_mae: 0.4884
Epoch 66/100
153/153             0s 2ms/step -
loss: 0.3855 - mae: 0.4836 - val_loss: 0.3261 - val_mae: 0.4738
Epoch 67/100
153/153             0s 2ms/step -
loss: 0.4108 - mae: 0.4949 - val_loss: 0.3661 - val_mae: 0.4966
Epoch 68/100
153/153             0s 2ms/step -
loss: 0.4158 - mae: 0.5017 - val_loss: 0.3363 - val_mae: 0.4759
Epoch 69/100
153/153             0s 2ms/step -
loss: 0.3855 - mae: 0.4848 - val_loss: 0.3420 - val_mae: 0.4845
Epoch 70/100
153/153             0s 2ms/step -
loss: 0.3949 - mae: 0.4873 - val_loss: 0.3285 - val_mae: 0.4826
Epoch 71/100
```

```
153/153            0s 2ms/step -
loss: 0.3776 - mae: 0.4770 - val_loss: 0.3266 - val_mae: 0.4751
Epoch 72/100
153/153            0s 2ms/step -
loss: 0.4301 - mae: 0.5140 - val_loss: 0.2847 - val_mae: 0.4380
Epoch 73/100
153/153            0s 2ms/step -
loss: 0.3802 - mae: 0.4684 - val_loss: 0.3325 - val_mae: 0.4778
Epoch 74/100
153/153            0s 2ms/step -
loss: 0.3957 - mae: 0.4903 - val_loss: 0.3401 - val_mae: 0.4786
Epoch 75/100
153/153            0s 2ms/step -
loss: 0.3905 - mae: 0.4834 - val_loss: 0.3318 - val_mae: 0.4764
Epoch 76/100
153/153            0s 2ms/step -
loss: 0.3714 - mae: 0.4691 - val_loss: 0.2974 - val_mae: 0.4524
Epoch 77/100
153/153            0s 2ms/step -
loss: 0.3658 - mae: 0.4635 - val_loss: 0.2959 - val_mae: 0.4469
Epoch 78/100
153/153            0s 2ms/step -
loss: 0.3624 - mae: 0.4643 - val_loss: 0.2709 - val_mae: 0.4294
Epoch 79/100
153/153            0s 2ms/step -
loss: 0.3831 - mae: 0.4664 - val_loss: 0.3355 - val_mae: 0.4882
Epoch 80/100
153/153            0s 2ms/step -
loss: 0.3726 - mae: 0.4763 - val_loss: 0.3156 - val_mae: 0.4648
Epoch 81/100
153/153            0s 2ms/step -
loss: 0.3640 - mae: 0.4763 - val_loss: 0.3540 - val_mae: 0.4932
Epoch 82/100
153/153            0s 2ms/step -
loss: 0.3784 - mae: 0.4787 - val_loss: 0.2849 - val_mae: 0.4417
Epoch 83/100
153/153            0s 2ms/step -
loss: 0.3935 - mae: 0.4850 - val_loss: 0.2509 - val_mae: 0.4112
Epoch 84/100
153/153            0s 2ms/step -
loss: 0.3851 - mae: 0.4918 - val_loss: 0.2667 - val_mae: 0.4282
Epoch 85/100
153/153            0s 2ms/step -
loss: 0.3585 - mae: 0.4651 - val_loss: 0.2882 - val_mae: 0.4440
Epoch 86/100
153/153            0s 2ms/step -
loss: 0.3525 - mae: 0.4514 - val_loss: 0.2479 - val_mae: 0.4109
Epoch 87/100
```

```
153/153             0s 2ms/step -
loss: 0.3686 - mae: 0.4558 - val_loss: 0.2656 - val_mae: 0.4268
Epoch 88/100
153/153             0s 2ms/step -
loss: 0.3623 - mae: 0.4580 - val_loss: 0.2861 - val_mae: 0.4424
Epoch 89/100
153/153             0s 2ms/step -
loss: 0.3598 - mae: 0.4656 - val_loss: 0.2543 - val_mae: 0.4142
Epoch 90/100
153/153             0s 2ms/step -
loss: 0.3987 - mae: 0.4838 - val_loss: 0.2885 - val_mae: 0.4423
Epoch 91/100
153/153             0s 2ms/step -
loss: 0.3363 - mae: 0.4464 - val_loss: 0.3217 - val_mae: 0.4738
Epoch 92/100
153/153             0s 2ms/step -
loss: 0.3721 - mae: 0.4635 - val_loss: 0.2913 - val_mae: 0.4497
Epoch 93/100
153/153             0s 2ms/step -
loss: 0.3660 - mae: 0.4707 - val_loss: 0.2809 - val_mae: 0.4387
Epoch 94/100
153/153             0s 2ms/step -
loss: 0.3879 - mae: 0.4859 - val_loss: 0.2808 - val_mae: 0.4437
Epoch 95/100
153/153             0s 2ms/step -
loss: 0.3505 - mae: 0.4568 - val_loss: 0.2955 - val_mae: 0.4534
Epoch 96/100
153/153             0s 2ms/step -
loss: 0.3766 - mae: 0.4772 - val_loss: 0.2869 - val_mae: 0.4462
Epoch 97/100
153/153             0s 2ms/step -
loss: 0.3657 - mae: 0.4559 - val_loss: 0.3033 - val_mae: 0.4594
Epoch 98/100
153/153             0s 2ms/step -
loss: 0.3869 - mae: 0.4830 - val_loss: 0.2785 - val_mae: 0.4408
Epoch 99/100
153/153             0s 2ms/step -
loss: 0.3674 - mae: 0.4756 - val_loss: 0.2865 - val_mae: 0.4461
Epoch 100/100
153/153             0s 2ms/step -
loss: 0.3459 - mae: 0.4598 - val_loss: 0.3096 - val_mae: 0.4606
```

[17]:
```
<keras.src.callbacks.history.History at 0x1e0a6b6abc0>
```

[28]:
```
# Evaluación del modelo 3

eval3 = E3.evaluate(X_test, y_test)
```

```
print(f'Valores de pérdida modelo 3: {eval3}')
```

```
15/15            0s 1ms/step - loss:
0.2755 - mae: 0.4203
15/15            0s 1ms/step - loss:
0.2755 - mae: 0.4203
Valores de pérdida modelo 3: [0.27483677864074707, 0.41845667362213135]
```

Experiment 4: Add a Batch Normalization Layer after each Dropout Layer

[19]:
```
# Experiment 4: a single dense hidden layer

E4 = Sequential()

# Agregar capa de entrada con 64 unidades y activación Relu y dimensión de␣
 ↪entrada = 7

E4.add(Dense(64, activation="relu", input_dim = 7))

# Agregar capa densa oculta 1 con 32 unidades y función de activación Relu

E4.add(Dense(32, activation="relu")) # dense hidden layer 1

# Agregar una capa de dropout después del dense hidden layer 1

E4.add(Dropout(0.5)) # dropout layer after dense hidden layer 1

# Agregar batch normalization layer después del dropout layer 1

E4.add(BatchNormalization()) # Batch Normalization Layer after Dropout Layer 1

# Agregar capa densa oculta 2 con 16 unidades y función de activación Relu

E4.add(Dense(16, activation="relu")) # dense hidden layer 2

# Agregar capa de dropout después del dense hidden layer 2

E4.add(Dropout(0.5)) # dropout layer after dense hidden layer 2

# Agregar batch normalization layer después del dropout layer 2

E4.add(BatchNormalization()) # Batch Normalization Layer after Dropout Layer 2

# Agregar capa densa oculta 3 con 8 unidades y función de activación Relu

E4.add(Dense(8, activation="relu")) # dense hidden layer 3
```

```
# Agregar capa de dropout después del dense hidden layer 3

E4.add(Dropout(0.5)) # dropout layer after dense hidden layer 3

# Agregar batch normalization layer después del dropout layer 3

E4.add(BatchNormalization()) # Batch Normalization Layer after Dropout Layer 3

# Añadir capa de salida con 1 unidad

E4.add(Dense(1))
```

[20]:
```
# Compilar el modelo 4 (con batchnormalization después de cada dropout layer)

E4.compile(optimizer="adam", loss="mse", metrics=["mae"])
```

[21]:
```
# Entrenamiento del modelo 4

E4.fit(X_train, y_train, epochs=100, batch_size=10, validation_split=0.2)
```

```
Epoch 1/100
153/153             3s 4ms/step -
loss: 4.9914 - mae: 1.9054 - val_loss: 2.9514 - val_mae: 1.5834
Epoch 2/100
153/153             0s 3ms/step -
loss: 2.7381 - mae: 1.3778 - val_loss: 1.5947 - val_mae: 1.1185
Epoch 3/100
153/153             0s 3ms/step -
loss: 1.6927 - mae: 1.0570 - val_loss: 0.6834 - val_mae: 0.7019
Epoch 4/100
153/153             0s 3ms/step -
loss: 1.2140 - mae: 0.8828 - val_loss: 0.5100 - val_mae: 0.6068
Epoch 5/100
153/153             0s 3ms/step -
loss: 0.9923 - mae: 0.8036 - val_loss: 0.4442 - val_mae: 0.5673
Epoch 6/100
153/153             0s 3ms/step -
loss: 0.9045 - mae: 0.7581 - val_loss: 0.4991 - val_mae: 0.6043
Epoch 7/100
153/153             0s 3ms/step -
loss: 0.8158 - mae: 0.7293 - val_loss: 0.4318 - val_mae: 0.5564
Epoch 8/100
153/153             0s 3ms/step -
loss: 0.7120 - mae: 0.6852 - val_loss: 0.4299 - val_mae: 0.5555
Epoch 9/100
153/153             0s 3ms/step -
loss: 0.7332 - mae: 0.6874 - val_loss: 0.5006 - val_mae: 0.6028
Epoch 10/100
```

```
153/153            0s 3ms/step -
loss: 0.6511 - mae: 0.6519 - val_loss: 0.3818 - val_mae: 0.5209
Epoch 11/100
153/153            0s 3ms/step -
loss: 0.6933 - mae: 0.6790 - val_loss: 0.4209 - val_mae: 0.5493
Epoch 12/100
153/153            1s 3ms/step -
loss: 0.6396 - mae: 0.6489 - val_loss: 0.3371 - val_mae: 0.4721
Epoch 13/100
153/153            1s 3ms/step -
loss: 0.5473 - mae: 0.5988 - val_loss: 0.3544 - val_mae: 0.4967
Epoch 14/100
153/153            0s 3ms/step -
loss: 0.5883 - mae: 0.6174 - val_loss: 0.3739 - val_mae: 0.5200
Epoch 15/100
153/153            1s 3ms/step -
loss: 0.6136 - mae: 0.6288 - val_loss: 0.3647 - val_mae: 0.5127
Epoch 16/100
153/153            1s 4ms/step -
loss: 0.5571 - mae: 0.6114 - val_loss: 0.3189 - val_mae: 0.4749
Epoch 17/100
153/153            1s 4ms/step -
loss: 0.5606 - mae: 0.6050 - val_loss: 0.3338 - val_mae: 0.4868
Epoch 18/100
153/153            1s 3ms/step -
loss: 0.5313 - mae: 0.5920 - val_loss: 0.3068 - val_mae: 0.4692
Epoch 19/100
153/153            1s 3ms/step -
loss: 0.5733 - mae: 0.6036 - val_loss: 0.2583 - val_mae: 0.4169
Epoch 20/100
153/153            1s 3ms/step -
loss: 0.5525 - mae: 0.6008 - val_loss: 0.2762 - val_mae: 0.4371
Epoch 21/100
153/153            1s 3ms/step -
loss: 0.5193 - mae: 0.5851 - val_loss: 0.2546 - val_mae: 0.4129
Epoch 22/100
153/153            0s 3ms/step -
loss: 0.5170 - mae: 0.5841 - val_loss: 0.2803 - val_mae: 0.4408
Epoch 23/100
153/153            0s 3ms/step -
loss: 0.4715 - mae: 0.5461 - val_loss: 0.2666 - val_mae: 0.4263
Epoch 24/100
153/153            0s 3ms/step -
loss: 0.5046 - mae: 0.5742 - val_loss: 0.2537 - val_mae: 0.4153
Epoch 25/100
153/153            0s 3ms/step -
loss: 0.5067 - mae: 0.5756 - val_loss: 0.2246 - val_mae: 0.3752
Epoch 26/100
```

```
153/153            0s 3ms/step -
loss: 0.4393 - mae: 0.5327 - val_loss: 0.2514 - val_mae: 0.4011
Epoch 27/100
153/153            0s 3ms/step -
loss: 0.5006 - mae: 0.5631 - val_loss: 0.2875 - val_mae: 0.4461
Epoch 28/100
153/153            0s 3ms/step -
loss: 0.4986 - mae: 0.5672 - val_loss: 0.2272 - val_mae: 0.3966
Epoch 29/100
153/153            0s 3ms/step -
loss: 0.5207 - mae: 0.5757 - val_loss: 0.2224 - val_mae: 0.3888
Epoch 30/100
153/153            0s 3ms/step -
loss: 0.5198 - mae: 0.5828 - val_loss: 0.2217 - val_mae: 0.3723
Epoch 31/100
153/153            0s 3ms/step -
loss: 0.5199 - mae: 0.5804 - val_loss: 0.1958 - val_mae: 0.3546
Epoch 32/100
153/153            0s 3ms/step -
loss: 0.4444 - mae: 0.5252 - val_loss: 0.2046 - val_mae: 0.3651
Epoch 33/100
153/153            0s 3ms/step -
loss: 0.4687 - mae: 0.5522 - val_loss: 0.1943 - val_mae: 0.3603
Epoch 34/100
153/153            0s 3ms/step -
loss: 0.4281 - mae: 0.5196 - val_loss: 0.2418 - val_mae: 0.4005
Epoch 35/100
153/153            0s 3ms/step -
loss: 0.4914 - mae: 0.5638 - val_loss: 0.2756 - val_mae: 0.4320
Epoch 36/100
153/153            0s 3ms/step -
loss: 0.4568 - mae: 0.5408 - val_loss: 0.1887 - val_mae: 0.3485
Epoch 37/100
153/153            0s 3ms/step -
loss: 0.4502 - mae: 0.5348 - val_loss: 0.2223 - val_mae: 0.3857
Epoch 38/100
153/153            0s 3ms/step -
loss: 0.4434 - mae: 0.5214 - val_loss: 0.2117 - val_mae: 0.3771
Epoch 39/100
153/153            0s 3ms/step -
loss: 0.4531 - mae: 0.5381 - val_loss: 0.2142 - val_mae: 0.3813
Epoch 40/100
153/153            0s 3ms/step -
loss: 0.4494 - mae: 0.5400 - val_loss: 0.2055 - val_mae: 0.3731
Epoch 41/100
153/153            0s 3ms/step -
loss: 0.4595 - mae: 0.5425 - val_loss: 0.1974 - val_mae: 0.3524
Epoch 42/100
```

```
153/153            0s 3ms/step -
loss: 0.4691 - mae: 0.5525 - val_loss: 0.1622 - val_mae: 0.3289
Epoch 43/100
153/153            0s 3ms/step -
loss: 0.4554 - mae: 0.5458 - val_loss: 0.1996 - val_mae: 0.3668
Epoch 44/100
153/153            0s 3ms/step -
loss: 0.4775 - mae: 0.5550 - val_loss: 0.2011 - val_mae: 0.3538
Epoch 45/100
153/153            0s 3ms/step -
loss: 0.4789 - mae: 0.5598 - val_loss: 0.2156 - val_mae: 0.3915
Epoch 46/100
153/153            0s 3ms/step -
loss: 0.4425 - mae: 0.5381 - val_loss: 0.2308 - val_mae: 0.3922
Epoch 47/100
153/153            0s 3ms/step -
loss: 0.4409 - mae: 0.5294 - val_loss: 0.1393 - val_mae: 0.3042
Epoch 48/100
153/153            0s 3ms/step -
loss: 0.4581 - mae: 0.5450 - val_loss: 0.1602 - val_mae: 0.3288
Epoch 49/100
153/153            0s 3ms/step -
loss: 0.4369 - mae: 0.5355 - val_loss: 0.1912 - val_mae: 0.3656
Epoch 50/100
153/153            0s 3ms/step -
loss: 0.4466 - mae: 0.5383 - val_loss: 0.1632 - val_mae: 0.3342
Epoch 51/100
153/153            0s 3ms/step -
loss: 0.4276 - mae: 0.5251 - val_loss: 0.1795 - val_mae: 0.3529
Epoch 52/100
153/153            0s 3ms/step -
loss: 0.4508 - mae: 0.5454 - val_loss: 0.1886 - val_mae: 0.3562
Epoch 53/100
153/153            0s 3ms/step -
loss: 0.4853 - mae: 0.5609 - val_loss: 0.1669 - val_mae: 0.3344
Epoch 54/100
153/153            1s 3ms/step -
loss: 0.4146 - mae: 0.5106 - val_loss: 0.1417 - val_mae: 0.3088
Epoch 55/100
153/153            0s 3ms/step -
loss: 0.4250 - mae: 0.5179 - val_loss: 0.1852 - val_mae: 0.3522
Epoch 56/100
153/153            0s 3ms/step -
loss: 0.4111 - mae: 0.5098 - val_loss: 0.1607 - val_mae: 0.3327
Epoch 57/100
153/153            0s 3ms/step -
loss: 0.4303 - mae: 0.5312 - val_loss: 0.1928 - val_mae: 0.3698
Epoch 58/100
```

```
153/153            0s 3ms/step -
loss: 0.4155 - mae: 0.5132 - val_loss: 0.1823 - val_mae: 0.3559
Epoch 59/100
153/153            0s 3ms/step -
loss: 0.4476 - mae: 0.5266 - val_loss: 0.1605 - val_mae: 0.3314
Epoch 60/100
153/153            0s 3ms/step -
loss: 0.4169 - mae: 0.5138 - val_loss: 0.2047 - val_mae: 0.3673
Epoch 61/100
153/153            0s 3ms/step -
loss: 0.4221 - mae: 0.5250 - val_loss: 0.2461 - val_mae: 0.4126
Epoch 62/100
153/153            0s 3ms/step -
loss: 0.3931 - mae: 0.4993 - val_loss: 0.1653 - val_mae: 0.3409
Epoch 63/100
153/153            0s 3ms/step -
loss: 0.4262 - mae: 0.5280 - val_loss: 0.1633 - val_mae: 0.3367
Epoch 64/100
153/153            0s 3ms/step -
loss: 0.4139 - mae: 0.5160 - val_loss: 0.1565 - val_mae: 0.3291
Epoch 65/100
153/153            0s 3ms/step -
loss: 0.4177 - mae: 0.5243 - val_loss: 0.1950 - val_mae: 0.3690
Epoch 66/100
153/153            0s 3ms/step -
loss: 0.4650 - mae: 0.5444 - val_loss: 0.2064 - val_mae: 0.3648
Epoch 67/100
153/153            0s 3ms/step -
loss: 0.3892 - mae: 0.5019 - val_loss: 0.1811 - val_mae: 0.3596
Epoch 68/100
153/153            0s 3ms/step -
loss: 0.4024 - mae: 0.5098 - val_loss: 0.1628 - val_mae: 0.3360
Epoch 69/100
153/153            0s 3ms/step -
loss: 0.4273 - mae: 0.5209 - val_loss: 0.1534 - val_mae: 0.3253
Epoch 70/100
153/153            0s 3ms/step -
loss: 0.4434 - mae: 0.5379 - val_loss: 0.1474 - val_mae: 0.3189
Epoch 71/100
153/153            0s 3ms/step -
loss: 0.4577 - mae: 0.5314 - val_loss: 0.2343 - val_mae: 0.3999
Epoch 72/100
153/153            0s 3ms/step -
loss: 0.4289 - mae: 0.5260 - val_loss: 0.1573 - val_mae: 0.3280
Epoch 73/100
153/153            0s 3ms/step -
loss: 0.4070 - mae: 0.5089 - val_loss: 0.1614 - val_mae: 0.3331
Epoch 74/100
```

```
153/153            0s 3ms/step -
loss: 0.4415 - mae: 0.5327 - val_loss: 0.1915 - val_mae: 0.3661
Epoch 75/100
153/153            0s 3ms/step -
loss: 0.3973 - mae: 0.5025 - val_loss: 0.2222 - val_mae: 0.3852
Epoch 76/100
153/153            0s 3ms/step -
loss: 0.4259 - mae: 0.5236 - val_loss: 0.2107 - val_mae: 0.3635
Epoch 77/100
153/153            0s 3ms/step -
loss: 0.4554 - mae: 0.5321 - val_loss: 0.2134 - val_mae: 0.3898
Epoch 78/100
153/153            0s 3ms/step -
loss: 0.4445 - mae: 0.5301 - val_loss: 0.2585 - val_mae: 0.4066
Epoch 79/100
153/153            0s 3ms/step -
loss: 0.4400 - mae: 0.5312 - val_loss: 0.2406 - val_mae: 0.4128
Epoch 80/100
153/153            0s 3ms/step -
loss: 0.4563 - mae: 0.5366 - val_loss: 0.2493 - val_mae: 0.4141
Epoch 81/100
153/153            0s 3ms/step -
loss: 0.4231 - mae: 0.5250 - val_loss: 0.5196 - val_mae: 0.4668
Epoch 82/100
153/153            0s 3ms/step -
loss: 0.4613 - mae: 0.5424 - val_loss: 0.3526 - val_mae: 0.4297
Epoch 83/100
153/153            0s 3ms/step -
loss: 0.4272 - mae: 0.5211 - val_loss: 0.3612 - val_mae: 0.3669
Epoch 84/100
153/153            0s 3ms/step -
loss: 0.4186 - mae: 0.5209 - val_loss: 0.3535 - val_mae: 0.4265
Epoch 85/100
153/153            0s 3ms/step -
loss: 0.4547 - mae: 0.5459 - val_loss: 0.3744 - val_mae: 0.4614
Epoch 86/100
153/153            0s 3ms/step -
loss: 0.4284 - mae: 0.5245 - val_loss: 0.1992 - val_mae: 0.3611
Epoch 87/100
153/153            0s 3ms/step -
loss: 0.3901 - mae: 0.4985 - val_loss: 0.1759 - val_mae: 0.3493
Epoch 88/100
153/153            0s 3ms/step -
loss: 0.4212 - mae: 0.5263 - val_loss: 0.2209 - val_mae: 0.3872
Epoch 89/100
153/153            0s 3ms/step -
loss: 0.4074 - mae: 0.5080 - val_loss: 0.1568 - val_mae: 0.3325
Epoch 90/100
```

```
153/153             0s 3ms/step -
loss: 0.3947 - mae: 0.5027 - val_loss: 0.1540 - val_mae: 0.3263
Epoch 91/100
153/153             0s 3ms/step -
loss: 0.3874 - mae: 0.5030 - val_loss: 0.1369 - val_mae: 0.3085
Epoch 92/100
153/153             0s 3ms/step -
loss: 0.3997 - mae: 0.4923 - val_loss: 0.1516 - val_mae: 0.3282
Epoch 93/100
153/153             0s 3ms/step -
loss: 0.4509 - mae: 0.5423 - val_loss: 0.1441 - val_mae: 0.3155
Epoch 94/100
153/153             0s 3ms/step -
loss: 0.4407 - mae: 0.5257 - val_loss: 0.1663 - val_mae: 0.3423
Epoch 95/100
153/153             0s 3ms/step -
loss: 0.4165 - mae: 0.5094 - val_loss: 0.1674 - val_mae: 0.3435
Epoch 96/100
153/153             0s 3ms/step -
loss: 0.3848 - mae: 0.5036 - val_loss: 0.2043 - val_mae: 0.3821
Epoch 97/100
153/153             0s 3ms/step -
loss: 0.3682 - mae: 0.4741 - val_loss: 0.2476 - val_mae: 0.4121
Epoch 98/100
153/153             0s 3ms/step -
loss: 0.4588 - mae: 0.5417 - val_loss: 0.2673 - val_mae: 0.4418
Epoch 99/100
153/153             0s 3ms/step -
loss: 0.4076 - mae: 0.5082 - val_loss: 0.1895 - val_mae: 0.3654
Epoch 100/100
153/153             0s 3ms/step -
loss: 0.4258 - mae: 0.5286 - val_loss: 0.2669 - val_mae: 0.4417
```

[21]:
```
<keras.src.callbacks.history.History at 0x1e0a8f154b0>
```

[22]:
```
# Evaluación del modelo 4

eval4 = E4.evaluate(X_test, y_test)

print(f'Valores de pérdida modelo 4: {eval4}')
```

```
15/15             0s 1ms/step - loss:
0.2264 - mae: 0.4028
Valores de pérdida modelo 4: [0.24065794050693512, 0.4098511040210724]
```

0.0.9 8. Predictions with the 4 models

[23]:
```
pred_model1 = E1.predict(X_test) # predictions with model 1 (single dense layer)

pred_model2 = E2.predict(X_test) # predictions with model 2 (3 dense hidden␣
 ↪layers)

pred_model3 = E3.predict(X_test) # predictions with model 3 (3 Dense hidden␣
 ↪layers + dropout)

pred_model4 = E4.predict(X_test) # predictions with model 4 (Dense hidden␣
 ↪layers + dropout + batch normalization)
```

```
15/15             0s 4ms/step
15/15             0s 4ms/step
15/15             0s 4ms/step
15/15             0s 9ms/step
```

0.0.10 9. Tabla comparativa de valores reales de GPA contra predicciones de los 4 modelos

[24]:
```
pred_model1 = pd.Series(np.reshape(pred_model1, len(pred_model1)))
pred_model1.index = y_test.index

pred_model2 = pd.Series(np.reshape(pred_model2, len(pred_model2)))
pred_model2.index = y_test.index

pred_model3 = pd.Series(np.reshape(pred_model3, len(pred_model3)))
pred_model3.index = y_test.index

pred_model4 = pd.Series(np.reshape(pred_model4, len(pred_model4)))
pred_model4.index = y_test.index
```

[25]:
```
GPA_realvspred = pd.DataFrame({"Modelo 1": pd.Series(np.reshape(pred_model1,␣
 ↪len(pred_model1))),
                               "Modelo 2": pd.Series(np.reshape(pred_model2,␣
 ↪len(pred_model2))),
                               "Modelo 3": pd.Series(np.reshape(pred_model3,␣
 ↪len(pred_model3))),
                               "Modelo 4": pd.Series(np.reshape(pred_model4,␣
 ↪len(pred_model4))),
                               "GPA real": y_test})

GPA_realvspred
```

[25]:

	Modelo 1	Modelo 2	Modelo 3	Modelo 4	GPA real
1004	1.362935	1.535858	1.487540	1.663879	1.427724
196	3.048964	3.150845	2.423644	2.798783	3.117354

```
2342  2.081095  2.187081  1.970699  1.885336  2.037769
1708  3.716757  3.860939  2.666003  4.180348  3.548521
435   0.451198  0.494626  1.243160  1.042828  0.248977
…          …         …         …         …         …
986   1.853950  1.775873  1.997903  1.894092  1.562360
120   2.074698  2.103497  2.024846  1.915603  2.174903
283   2.169771  2.261521  2.133455  1.925295  2.332540
1740  2.725360  2.729002  2.193378  1.917401  2.777967
1726  0.986181  0.859922  1.364492  1.509829  0.863545

[479 rows x 5 columns]
```

0.0.11 10. Tabla comparativa de métricas (MSE y MAE) de los 4 modelos

[27]:
```
metrics_table = pd.DataFrame({"MSE": [0.0587,0.0534,0.2755,0.2264],
                              "MAE":[0.1782,0.1711,0.4203,0.4028]},
                              index=["Single Dense Hidden Layer", "3 Dense␣
 ↪Hidden Layers", "Dense + Dropout", "Dense + Dropout + Batch Normalization"])

# Asignar nombre al índice del dataframe anterior

metrics_table.index.name = "Modelo"

# Mostrar los valores de las métricas para los 4 modelos: valores de pérdida␣
 ↪MSE Y MAE

metrics_table
```

[27]:

Modelo	MSE	MAE
Single Dense Hidden Layer	0.0587	0.1782
3 Dense Hidden Layers	0.0534	0.1711
Dense + Dropout	0.2755	0.4203
Dense + Dropout + Batch Normalization	0.2264	0.4028

0.0.12 11. Conclusión final sobre el mejor modelo

En conclusión, es posible afirmar que de los 4 modelos probados anteriormente, aquel que resulta tener el mejor desempeño es el modelo 2 (aquel con 3 capas densas ocultas) y sin dropout ni batch normalization, esto principalmente debido a que éste modelo posee un valor de MSE igual a 0.0534 junto con un MAE de 0.1711, siendo éstos los menores valores posibles de MSE y MAE, de entre todos los modelos probados, lo cual es un indicativo de que el modelo con 3 capas densas ocultas (modelo 2) es aquel que arroja las predicciones con el mayor grado de precisión y confiabilidad posibles, mientras que los demás modelos, entre ellos, a los que se les aplicó batch normalization y dropout muestran tener una mayor margen de error en cuanto a su capacidad predictiva, esto principalmente porque poseen un mayor valor de MSE y de MAE que el modelo con triple capa densa oculta (3 Dense Hidden Layers), además, cabe mencionar que los modelos con el dropout y

batch normalization aplicados es probable que tengan un mayor valor de MSE y MAE, debido a que el dataset de prueba para los modelos no contiene un volumen suficientemente alto de datos como para que las técnicas de dropout y batch normalization produzcan una mejoría significativa en la precisión de las predicciones de los modelos, además de que el hecho de utilizar alguna de éstas 2 técnicas, tales como el dropout, ocasiona por el contrario, que el MSE y el MAE de los modelos incremente, indicando mayor margen de error en las predicciones derivadas de los mismos, por lo que al tener una cantidad relativamente baja de datos en el dataset de prueba y aplicar técnicas como dropout y batch normalization, al momento de eliminar aleatoriamente ciertas neuronas de las redes, las predicciones de los modelos se sesgan más y por tanto pierden precisión y en consecuencia también su confiabilidad resulta ser menor, por lo que es más recomendable utilizar las técnicas de batch normalization y dropout en modelos que se pongan a prueba con una cantidad mayormente alta de datos en el dataset de prueba, esto para asegurar que al quitar neuronas de forma aleatoria mediante dropout, o bien, al aplicar batch normalization a los modelos en un intento por mejorar el desempeño de los mismos, éstos mismos modelos presenten una mejoría significativa en cuanto a su rendimiento predictivo, lo cual se traducirá en predicciones más precisas y confiables y en una mayor cantidad existente de alternativas para lograrlas, puesto que el dropout permitirá a los modelos explorar alternativas que probablemente no hayan sido exploradas anteriormente y que una de ellas sea la que arroje las predicciones más precisas y confiables posible de entre todas las posibilidades, además de que al haber muchos más datos, el quitar neuronas de los modelos tendrá un menor impacto negativo en en la precisión del mismo para realizar predicciones, por lo que las métricas como el MSE y el MAE ya no aumentarán significativamente al emplear tanto dropout como batch normalization.