se-dropout-batch-normalization

November 9, 2024

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
from tensorflow.keras.layers import Dropout
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
```

0.0.3 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

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

[2]:

	StudentID	Age	Gender	Ethnicity	ParentalEducation	StudyTimeWeekly	\
0	1001	17	1	0	2	19.833723	
1	1002	18	0	0	1	15.408756	
2	1003	15	0	2	3	4.210570	
3	1004	17	1	0	3	10.028829	
4	1005	17	1	0	2	4.672495	
…	…	…	…	…	…	…	
2387	3388	18	1	0	3	10.680555	
2388	3389	17	0	0	1	7.583217	
2389	3390	16	1	0	2	6.805500	
2390	3391	16	1	1	0	12.416653	
2391	3392	16	1	0	2	17.819907	

	Absences	Tutoring	ParentalSupport	Extracurricular	Sports	Music	\
0	7	1	2	0	0	1	
1	0	0	1	0	0	0	
2	26	0	2	0	0	0	
3	14	0	3	1	0	0	
4	17	1	3	0	0	0	
…	…	…	…	…	…	…	
2387	2	0	4	1	0	0	
2388	4	1	4	0	1	0	
2389	20	0	2	0	0	0	
2390	17	0	2	0	1	1	
2391	13	0	2	0	0	0	

	Volunteering	GPA	GradeClass
0	0	2.929196	2.0
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]:
```
# Your code here
dataset = data.drop(columns=['StudentID','Age','Gender','Ethnicity'])
```

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]:
```
# Your code here
dataset.isnull().sum()
```

[5]:
```
ParentalEducation    0
StudyTimeWeekly      0
Absences             0
Tutoring             0
ParentalSupport      0
Extracurricular      0
Sports               0
Music                0
Volunteering         0
GPA                  0
GradeClass           0
dtype: int64
```

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]:
```
# Your code here
X = dataset.drop(columns=['GPA'])
y = dataset['GPA']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
print(X_train.shape)
```

```
(1913, 10)
```

1 Experimento 1

A single Dense Hidden Layer

[13]:
```
# Experiment 1: A single Dense Hidden Layer
model_exp1 = Sequential()
model_exp1.add(Dense(64, input_dim=10, activation='relu'))
model_exp1.add(Dense(1))  # Output layer
model_exp1.compile(optimizer='Adam', loss='mse', metrics=['mae'])
history_exp1 = model_exp1.fit(X_train_scaled, y_train, epochs=50,␣
 ↪batch_size=10, validation_split=0.2)
```

```
Epoch 1/50
c:\Users\andre\AppData\Local\Programs\Python\Python311\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)
153/153            1s 1ms/step -
loss: 2.5068 - mae: 1.2658 - val_loss: 0.2918 - val_mae: 0.4406
Epoch 2/50
153/153            0s 860us/step -
loss: 0.1800 - mae: 0.3321 - val_loss: 0.1323 - val_mae: 0.2884
Epoch 3/50
153/153            0s 860us/step -
loss: 0.0995 - mae: 0.2507 - val_loss: 0.0858 - val_mae: 0.2358
Epoch 4/50
153/153            0s 931us/step -
loss: 0.0668 - mae: 0.2042 - val_loss: 0.0674 - val_mae: 0.2104
Epoch 5/50
153/153            0s 2ms/step -
loss: 0.0533 - mae: 0.1880 - val_loss: 0.0584 - val_mae: 0.1979
Epoch 6/50
153/153            0s 1ms/step -
loss: 0.0497 - mae: 0.1797 - val_loss: 0.0534 - val_mae: 0.1901
Epoch 7/50
153/153            0s 1ms/step -
loss: 0.0429 - mae: 0.1690 - val_loss: 0.0496 - val_mae: 0.1831
Epoch 8/50
```

```
153/153             0s 1ms/step -
loss: 0.0412 - mae: 0.1653 - val_loss: 0.0484 - val_mae: 0.1790
Epoch 9/50
153/153             0s 1ms/step -
loss: 0.0391 - mae: 0.1612 - val_loss: 0.0474 - val_mae: 0.1763
Epoch 10/50
153/153             0s 1ms/step -
loss: 0.0384 - mae: 0.1605 - val_loss: 0.0458 - val_mae: 0.1732
Epoch 11/50
153/153             0s 802us/step -
loss: 0.0363 - mae: 0.1546 - val_loss: 0.0447 - val_mae: 0.1726
Epoch 12/50
153/153             0s 850us/step -
loss: 0.0368 - mae: 0.1556 - val_loss: 0.0451 - val_mae: 0.1726
Epoch 13/50
153/153             0s 1ms/step -
loss: 0.0364 - mae: 0.1523 - val_loss: 0.0445 - val_mae: 0.1713
Epoch 14/50
153/153             0s 986us/step -
loss: 0.0347 - mae: 0.1489 - val_loss: 0.0462 - val_mae: 0.1721
Epoch 15/50
153/153             0s 876us/step -
loss: 0.0373 - mae: 0.1542 - val_loss: 0.0418 - val_mae: 0.1657
Epoch 16/50
153/153             0s 958us/step -
loss: 0.0331 - mae: 0.1439 - val_loss: 0.0439 - val_mae: 0.1696
Epoch 17/50
153/153             0s 1ms/step -
loss: 0.0334 - mae: 0.1463 - val_loss: 0.0419 - val_mae: 0.1655
Epoch 18/50
153/153             0s 883us/step -
loss: 0.0324 - mae: 0.1417 - val_loss: 0.0418 - val_mae: 0.1629
Epoch 19/50
153/153             0s 872us/step -
loss: 0.0297 - mae: 0.1385 - val_loss: 0.0434 - val_mae: 0.1678
Epoch 20/50
153/153             0s 867us/step -
loss: 0.0332 - mae: 0.1458 - val_loss: 0.0415 - val_mae: 0.1640
Epoch 21/50
153/153             0s 865us/step -
loss: 0.0319 - mae: 0.1414 - val_loss: 0.0423 - val_mae: 0.1665
Epoch 22/50
153/153             0s 941us/step -
loss: 0.0307 - mae: 0.1385 - val_loss: 0.0422 - val_mae: 0.1615
Epoch 23/50
153/153             0s 820us/step -
loss: 0.0295 - mae: 0.1358 - val_loss: 0.0389 - val_mae: 0.1588
Epoch 24/50
```

```
153/153            0s 900us/step -
loss: 0.0306 - mae: 0.1398 - val_loss: 0.0403 - val_mae: 0.1603
Epoch 25/50
153/153            0s 1ms/step -
loss: 0.0297 - mae: 0.1345 - val_loss: 0.0410 - val_mae: 0.1618
Epoch 26/50
153/153            0s 956us/step -
loss: 0.0292 - mae: 0.1340 - val_loss: 0.0393 - val_mae: 0.1581
Epoch 27/50
153/153            0s 920us/step -
loss: 0.0282 - mae: 0.1315 - val_loss: 0.0403 - val_mae: 0.1626
Epoch 28/50
153/153            0s 860us/step -
loss: 0.0299 - mae: 0.1336 - val_loss: 0.0415 - val_mae: 0.1619
Epoch 29/50
153/153            0s 1ms/step -
loss: 0.0276 - mae: 0.1301 - val_loss: 0.0405 - val_mae: 0.1599
Epoch 30/50
153/153            0s 1ms/step -
loss: 0.0279 - mae: 0.1346 - val_loss: 0.0414 - val_mae: 0.1607
Epoch 31/50
153/153            0s 1ms/step -
loss: 0.0279 - mae: 0.1322 - val_loss: 0.0394 - val_mae: 0.1600
Epoch 32/50
153/153            0s 863us/step -
loss: 0.0275 - mae: 0.1325 - val_loss: 0.0394 - val_mae: 0.1592
Epoch 33/50
153/153            0s 842us/step -
loss: 0.0262 - mae: 0.1278 - val_loss: 0.0394 - val_mae: 0.1587
Epoch 34/50
153/153            0s 909us/step -
loss: 0.0266 - mae: 0.1290 - val_loss: 0.0395 - val_mae: 0.1593
Epoch 35/50
153/153            0s 1ms/step -
loss: 0.0279 - mae: 0.1332 - val_loss: 0.0379 - val_mae: 0.1588
Epoch 36/50
153/153            0s 1000us/step -
loss: 0.0270 - mae: 0.1292 - val_loss: 0.0381 - val_mae: 0.1572
Epoch 37/50
153/153            0s 1ms/step -
loss: 0.0278 - mae: 0.1307 - val_loss: 0.0399 - val_mae: 0.1606
Epoch 38/50
153/153            0s 898us/step -
loss: 0.0270 - mae: 0.1297 - val_loss: 0.0390 - val_mae: 0.1565
Epoch 39/50
153/153            0s 895us/step -
loss: 0.0263 - mae: 0.1277 - val_loss: 0.0398 - val_mae: 0.1591
Epoch 40/50
```

```
153/153             0s 883us/step -
loss: 0.0287 - mae: 0.1342 - val_loss: 0.0398 - val_mae: 0.1612
Epoch 41/50
153/153             0s 1ms/step -
loss: 0.0266 - mae: 0.1287 - val_loss: 0.0376 - val_mae: 0.1558
Epoch 42/50
153/153             0s 911us/step -
loss: 0.0250 - mae: 0.1252 - val_loss: 0.0386 - val_mae: 0.1562
Epoch 43/50
153/153             0s 956us/step -
loss: 0.0242 - mae: 0.1229 - val_loss: 0.0400 - val_mae: 0.1588
Epoch 44/50
153/153             0s 1ms/step -
loss: 0.0271 - mae: 0.1281 - val_loss: 0.0372 - val_mae: 0.1549
Epoch 45/50
153/153             0s 885us/step -
loss: 0.0253 - mae: 0.1263 - val_loss: 0.0378 - val_mae: 0.1554
Epoch 46/50
153/153             0s 889us/step -
loss: 0.0271 - mae: 0.1272 - val_loss: 0.0399 - val_mae: 0.1583
Epoch 47/50
153/153             0s 886us/step -
loss: 0.0250 - mae: 0.1238 - val_loss: 0.0380 - val_mae: 0.1543
Epoch 48/50
153/153             0s 990us/step -
loss: 0.0247 - mae: 0.1200 - val_loss: 0.0413 - val_mae: 0.1618
Epoch 49/50
153/153             0s 898us/step -
loss: 0.0259 - mae: 0.1280 - val_loss: 0.0401 - val_mae: 0.1601
Epoch 50/50
153/153             0s 914us/step -
loss: 0.0245 - mae: 0.1226 - val_loss: 0.0400 - val_mae: 0.1571
```

2 Experimento 2

A set of three Dense Hidden Layers

[14]:
```
# Experiment 2: A set of three Dense Hidden Layers
model_exp2 = Sequential()
model_exp2.add(Dense(64, input_dim=10, activation='relu'))
model_exp2.add(Dense(32, activation='relu'))
model_exp2.add(Dense(16, activation='relu'))
model_exp2.add(Dense(1))  # Output layer
model_exp2.compile(optimizer='Adam', loss='mse', metrics=['mae'])
history_exp2 = model_exp2.fit(X_train_scaled, y_train, epochs=50,␣
 ↪batch_size=10, validation_split=0.2)
```

```
Epoch 1/50
```

```
c:\Users\andre\AppData\Local\Programs\Python\Python311\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)
153/153            1s 2ms/step -
loss: 1.0619 - mae: 0.7522 - val_loss: 0.1208 - val_mae: 0.2831
Epoch 2/50
153/153            0s 1ms/step -
loss: 0.0844 - mae: 0.2312 - val_loss: 0.0676 - val_mae: 0.2056
Epoch 3/50
153/153            0s 2ms/step -
loss: 0.0572 - mae: 0.1910 - val_loss: 0.0571 - val_mae: 0.1932
Epoch 4/50
153/153            0s 1ms/step -
loss: 0.0461 - mae: 0.1728 - val_loss: 0.0598 - val_mae: 0.1969
Epoch 5/50
153/153            0s 967us/step -
loss: 0.0443 - mae: 0.1708 - val_loss: 0.0473 - val_mae: 0.1728
Epoch 6/50
153/153            0s 2ms/step -
loss: 0.0388 - mae: 0.1581 - val_loss: 0.0489 - val_mae: 0.1803
Epoch 7/50
153/153            0s 1ms/step -
loss: 0.0352 - mae: 0.1511 - val_loss: 0.0436 - val_mae: 0.1671
Epoch 8/50
153/153            0s 1ms/step -
loss: 0.0376 - mae: 0.1562 - val_loss: 0.0459 - val_mae: 0.1679
Epoch 9/50
153/153            0s 922us/step -
loss: 0.0343 - mae: 0.1474 - val_loss: 0.0451 - val_mae: 0.1641
Epoch 10/50
153/153            0s 908us/step -
loss: 0.0293 - mae: 0.1371 - val_loss: 0.0439 - val_mae: 0.1646
Epoch 11/50
153/153            0s 1ms/step -
loss: 0.0296 - mae: 0.1381 - val_loss: 0.0441 - val_mae: 0.1698
Epoch 12/50
153/153            0s 917us/step -
loss: 0.0300 - mae: 0.1352 - val_loss: 0.0438 - val_mae: 0.1611
Epoch 13/50
153/153            0s 915us/step -
loss: 0.0289 - mae: 0.1353 - val_loss: 0.0434 - val_mae: 0.1629
Epoch 14/50
153/153            0s 936us/step -
loss: 0.0268 - mae: 0.1294 - val_loss: 0.0471 - val_mae: 0.1751
Epoch 15/50
```

```
153/153             0s 929us/step -
loss: 0.0269 - mae: 0.1292 - val_loss: 0.0432 - val_mae: 0.1663
Epoch 16/50
153/153             0s 1ms/step -
loss: 0.0256 - mae: 0.1252 - val_loss: 0.0466 - val_mae: 0.1676
Epoch 17/50
153/153             0s 904us/step -
loss: 0.0254 - mae: 0.1258 - val_loss: 0.0423 - val_mae: 0.1633
Epoch 18/50
153/153             0s 1ms/step -
loss: 0.0271 - mae: 0.1319 - val_loss: 0.0471 - val_mae: 0.1710
Epoch 19/50
153/153             0s 970us/step -
loss: 0.0272 - mae: 0.1334 - val_loss: 0.0418 - val_mae: 0.1607
Epoch 20/50
153/153             0s 882us/step -
loss: 0.0214 - mae: 0.1144 - val_loss: 0.0451 - val_mae: 0.1653
Epoch 21/50
153/153             0s 881us/step -
loss: 0.0253 - mae: 0.1263 - val_loss: 0.0437 - val_mae: 0.1639
Epoch 22/50
153/153             0s 881us/step -
loss: 0.0233 - mae: 0.1210 - val_loss: 0.0450 - val_mae: 0.1637
Epoch 23/50
153/153             0s 975us/step -
loss: 0.0225 - mae: 0.1187 - val_loss: 0.0450 - val_mae: 0.1636
Epoch 24/50
153/153             0s 904us/step -
loss: 0.0239 - mae: 0.1237 - val_loss: 0.0469 - val_mae: 0.1691
Epoch 25/50
153/153             0s 884us/step -
loss: 0.0228 - mae: 0.1188 - val_loss: 0.0454 - val_mae: 0.1676
Epoch 26/50
153/153             0s 1ms/step -
loss: 0.0228 - mae: 0.1176 - val_loss: 0.0505 - val_mae: 0.1708
Epoch 27/50
153/153             0s 914us/step -
loss: 0.0223 - mae: 0.1152 - val_loss: 0.0452 - val_mae: 0.1661
Epoch 28/50
153/153             0s 880us/step -
loss: 0.0214 - mae: 0.1145 - val_loss: 0.0456 - val_mae: 0.1683
Epoch 29/50
153/153             0s 906us/step -
loss: 0.0216 - mae: 0.1158 - val_loss: 0.0424 - val_mae: 0.1569
Epoch 30/50
153/153             0s 1ms/step -
loss: 0.0203 - mae: 0.1124 - val_loss: 0.0454 - val_mae: 0.1643
Epoch 31/50
```

```
153/153             0s 876us/step -
loss: 0.0211 - mae: 0.1127 - val_loss: 0.0421 - val_mae: 0.1564
Epoch 32/50
153/153             0s 911us/step -
loss: 0.0205 - mae: 0.1142 - val_loss: 0.0420 - val_mae: 0.1568
Epoch 33/50
153/153             0s 925us/step -
loss: 0.0188 - mae: 0.1079 - val_loss: 0.0426 - val_mae: 0.1618
Epoch 34/50
153/153             0s 899us/step -
loss: 0.0201 - mae: 0.1134 - val_loss: 0.0455 - val_mae: 0.1641
Epoch 35/50
153/153             0s 970us/step -
loss: 0.0221 - mae: 0.1156 - val_loss: 0.0418 - val_mae: 0.1595
Epoch 36/50
153/153             0s 940us/step -
loss: 0.0212 - mae: 0.1151 - val_loss: 0.0477 - val_mae: 0.1713
Epoch 37/50
153/153             0s 928us/step -
loss: 0.0211 - mae: 0.1145 - val_loss: 0.0439 - val_mae: 0.1629
Epoch 38/50
153/153             0s 970us/step -
loss: 0.0199 - mae: 0.1094 - val_loss: 0.0413 - val_mae: 0.1578
Epoch 39/50
153/153             0s 882us/step -
loss: 0.0199 - mae: 0.1100 - val_loss: 0.0429 - val_mae: 0.1619
Epoch 40/50
153/153             0s 915us/step -
loss: 0.0201 - mae: 0.1101 - val_loss: 0.0419 - val_mae: 0.1595
Epoch 41/50
153/153             0s 929us/step -
loss: 0.0177 - mae: 0.1041 - val_loss: 0.0424 - val_mae: 0.1599
Epoch 42/50
153/153             0s 864us/step -
loss: 0.0175 - mae: 0.1027 - val_loss: 0.0422 - val_mae: 0.1587
Epoch 43/50
153/153             0s 944us/step -
loss: 0.0198 - mae: 0.1114 - val_loss: 0.0419 - val_mae: 0.1591
Epoch 44/50
153/153             0s 871us/step -
loss: 0.0172 - mae: 0.1051 - val_loss: 0.0415 - val_mae: 0.1587
Epoch 45/50
153/153             0s 1ms/step -
loss: 0.0193 - mae: 0.1096 - val_loss: 0.0458 - val_mae: 0.1644
Epoch 46/50
153/153             0s 938us/step -
loss: 0.0205 - mae: 0.1148 - val_loss: 0.0415 - val_mae: 0.1580
Epoch 47/50
```

```
153/153             0s 922us/step -
loss: 0.0173 - mae: 0.1034 - val_loss: 0.0423 - val_mae: 0.1607
Epoch 48/50
153/153             0s 892us/step -
loss: 0.0187 - mae: 0.1082 - val_loss: 0.0447 - val_mae: 0.1648
Epoch 49/50
153/153             0s 898us/step -
loss: 0.0185 - mae: 0.1057 - val_loss: 0.0418 - val_mae: 0.1581
Epoch 50/50
153/153             0s 1ms/step -
loss: 0.0179 - mae: 0.1053 - val_loss: 0.0425 - val_mae: 0.1593
```

3 Experimento 3

Add a dropout layer after each Dense Hidden Layer

[15]:
```
# Experiment 3: Add a dropout layer after each Dense Hidden Layer
model_exp3 = Sequential()
model_exp3.add(Dense(64, input_dim=10, activation='relu'))
model_exp3.add(Dropout(0.5))
model_exp3.add(Dense(32, activation='relu'))
model_exp3.add(Dropout(0.5))
model_exp3.add(Dense(16, activation='relu'))
model_exp3.add(Dropout(0.5))
model_exp3.add(Dense(1))  # Output layer
model_exp3.compile(optimizer='Adam', loss='mse', metrics=['mae'])
history_exp3 = model_exp3.fit(X_train_scaled, y_train, epochs=50,␣
 ↪batch_size=10, validation_split=0.2)
```

```
Epoch 1/50

c:\Users\andre\AppData\Local\Programs\Python\Python311\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)

153/153             1s 2ms/step -
loss: 2.9602 - mae: 1.4158 - val_loss: 0.9856 - val_mae: 0.8514
Epoch 2/50
153/153             0s 1ms/step -
loss: 1.3742 - mae: 0.9290 - val_loss: 0.6539 - val_mae: 0.7010
Epoch 3/50
153/153             0s 2ms/step -
loss: 1.0486 - mae: 0.7918 - val_loss: 0.5123 - val_mae: 0.6241
Epoch 4/50
153/153             0s 2ms/step -
loss: 0.8677 - mae: 0.7312 - val_loss: 0.4952 - val_mae: 0.6156
Epoch 5/50
```

```
153/153             0s 1ms/step -
loss: 0.8365 - mae: 0.7046 - val_loss: 0.5735 - val_mae: 0.6597
Epoch 6/50
153/153             0s 1ms/step -
loss: 0.7294 - mae: 0.6587 - val_loss: 0.3436 - val_mae: 0.5159
Epoch 7/50
153/153             0s 1ms/step -
loss: 0.6558 - mae: 0.6419 - val_loss: 0.4206 - val_mae: 0.5710
Epoch 8/50
153/153             0s 1ms/step -
loss: 0.6715 - mae: 0.6220 - val_loss: 0.3413 - val_mae: 0.5142
Epoch 9/50
153/153             0s 1ms/step -
loss: 0.5969 - mae: 0.5902 - val_loss: 0.3921 - val_mae: 0.5522
Epoch 10/50
153/153             0s 1ms/step -
loss: 0.5391 - mae: 0.5803 - val_loss: 0.3419 - val_mae: 0.5131
Epoch 11/50
153/153             0s 1ms/step -
loss: 0.5465 - mae: 0.5692 - val_loss: 0.3257 - val_mae: 0.4986
Epoch 12/50
153/153             0s 1ms/step -
loss: 0.5680 - mae: 0.5821 - val_loss: 0.3306 - val_mae: 0.5029
Epoch 13/50
153/153             0s 1ms/step -
loss: 0.4855 - mae: 0.5344 - val_loss: 0.3048 - val_mae: 0.4823
Epoch 14/50
153/153             0s 1ms/step -
loss: 0.4408 - mae: 0.5177 - val_loss: 0.2301 - val_mae: 0.4169
Epoch 15/50
153/153             0s 1ms/step -
loss: 0.4437 - mae: 0.5245 - val_loss: 0.2899 - val_mae: 0.4680
Epoch 16/50
153/153             0s 1ms/step -
loss: 0.4187 - mae: 0.5021 - val_loss: 0.2117 - val_mae: 0.3957
Epoch 17/50
153/153             0s 1ms/step -
loss: 0.3965 - mae: 0.4799 - val_loss: 0.2027 - val_mae: 0.3850
Epoch 18/50
153/153             0s 1ms/step -
loss: 0.3578 - mae: 0.4690 - val_loss: 0.1758 - val_mae: 0.3567
Epoch 19/50
153/153             0s 1ms/step -
loss: 0.3722 - mae: 0.4657 - val_loss: 0.1651 - val_mae: 0.3446
Epoch 20/50
153/153             0s 1ms/step -
loss: 0.3693 - mae: 0.4627 - val_loss: 0.1748 - val_mae: 0.3525
Epoch 21/50
```

```
153/153            0s 1ms/step -
loss: 0.3054 - mae: 0.4266 - val_loss: 0.1900 - val_mae: 0.3693
Epoch 22/50
153/153            0s 1ms/step -
loss: 0.2956 - mae: 0.4119 - val_loss: 0.1688 - val_mae: 0.3445
Epoch 23/50
153/153            0s 1ms/step -
loss: 0.2877 - mae: 0.4181 - val_loss: 0.1332 - val_mae: 0.2998
Epoch 24/50
153/153            0s 1ms/step -
loss: 0.2968 - mae: 0.4164 - val_loss: 0.1668 - val_mae: 0.3412
Epoch 25/50
153/153            0s 1ms/step -
loss: 0.3076 - mae: 0.4294 - val_loss: 0.1748 - val_mae: 0.3480
Epoch 26/50
153/153            0s 1ms/step -
loss: 0.2684 - mae: 0.3957 - val_loss: 0.1783 - val_mae: 0.3524
Epoch 27/50
153/153            0s 1ms/step -
loss: 0.2749 - mae: 0.3980 - val_loss: 0.1553 - val_mae: 0.3240
Epoch 28/50
153/153            0s 1ms/step -
loss: 0.3081 - mae: 0.4079 - val_loss: 0.1497 - val_mae: 0.3153
Epoch 29/50
153/153            0s 1ms/step -
loss: 0.2466 - mae: 0.3695 - val_loss: 0.1631 - val_mae: 0.3340
Epoch 30/50
153/153            0s 1ms/step -
loss: 0.2530 - mae: 0.3781 - val_loss: 0.1598 - val_mae: 0.3254
Epoch 31/50
153/153            0s 1ms/step -
loss: 0.2948 - mae: 0.4136 - val_loss: 0.1290 - val_mae: 0.2884
Epoch 32/50
153/153            0s 1ms/step -
loss: 0.2644 - mae: 0.3891 - val_loss: 0.1660 - val_mae: 0.3317
Epoch 33/50
153/153            0s 1ms/step -
loss: 0.2613 - mae: 0.3860 - val_loss: 0.1246 - val_mae: 0.2815
Epoch 34/50
153/153            0s 1ms/step -
loss: 0.2688 - mae: 0.3889 - val_loss: 0.1449 - val_mae: 0.3037
Epoch 35/50
153/153            0s 1ms/step -
loss: 0.2404 - mae: 0.3698 - val_loss: 0.1616 - val_mae: 0.3321
Epoch 36/50
153/153            0s 966us/step -
loss: 0.2406 - mae: 0.3661 - val_loss: 0.1501 - val_mae: 0.3210
Epoch 37/50
```

```
153/153            0s 1ms/step -
loss: 0.2209 - mae: 0.3593 - val_loss: 0.1082 - val_mae: 0.2448
Epoch 38/50
153/153            0s 1ms/step -
loss: 0.2581 - mae: 0.3783 - val_loss: 0.1064 - val_mae: 0.2522
Epoch 39/50
153/153            0s 1ms/step -
loss: 0.2438 - mae: 0.3716 - val_loss: 0.1576 - val_mae: 0.3270
Epoch 40/50
153/153            0s 1ms/step -
loss: 0.2534 - mae: 0.3788 - val_loss: 0.1319 - val_mae: 0.2875
Epoch 41/50
153/153            0s 1ms/step -
loss: 0.2282 - mae: 0.3562 - val_loss: 0.1459 - val_mae: 0.3155
Epoch 42/50
153/153            0s 1ms/step -
loss: 0.2527 - mae: 0.3738 - val_loss: 0.1025 - val_mae: 0.2460
Epoch 43/50
153/153            0s 1ms/step -
loss: 0.2169 - mae: 0.3513 - val_loss: 0.0970 - val_mae: 0.2355
Epoch 44/50
153/153            0s 1ms/step -
loss: 0.2434 - mae: 0.3714 - val_loss: 0.1013 - val_mae: 0.2433
Epoch 45/50
153/153            0s 1ms/step -
loss: 0.2310 - mae: 0.3657 - val_loss: 0.1189 - val_mae: 0.2750
Epoch 46/50
153/153            0s 1ms/step -
loss: 0.2616 - mae: 0.3853 - val_loss: 0.1060 - val_mae: 0.2554
Epoch 47/50
153/153            0s 1ms/step -
loss: 0.2257 - mae: 0.3575 - val_loss: 0.1113 - val_mae: 0.2657
Epoch 48/50
153/153            0s 1ms/step -
loss: 0.2233 - mae: 0.3576 - val_loss: 0.1133 - val_mae: 0.2736
Epoch 49/50
153/153            0s 1ms/step -
loss: 0.2136 - mae: 0.3540 - val_loss: 0.1532 - val_mae: 0.3232
Epoch 50/50
153/153            0s 1ms/step -
loss: 0.2406 - mae: 0.3781 - val_loss: 0.1254 - val_mae: 0.2863
```

4 Experimento 4

Add a Batch Normalization Layer after each Dropout Layer.

[16]:
```
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
# Experiment 4: Add a Batch Normalization Layer after each Dropout Layer
model_exp4 = Sequential()
model_exp4.add(Dense(64, input_dim=10, activation='relu'))
model_exp4.add(Dropout(0.5))
model_exp4.add(BatchNormalization())
model_exp4.add(Dense(32, activation='relu'))
model_exp4.add(Dropout(0.5))
model_exp4.add(BatchNormalization())
model_exp4.add(Dense(16, activation='relu'))
model_exp4.add(Dropout(0.5))
model_exp4.add(BatchNormalization())
model_exp4.add(Dense(1))  # Output layer
model_exp4.compile(optimizer='Adam', loss='mse', metrics=['mae'])
history_exp4 = model_exp4.fit(X_train_scaled, y_train, epochs=50,␣
 ↪batch_size=10, validation_split=0.2)
```

```
Epoch 1/50
```

```
c:\Users\andre\AppData\Local\Programs\Python\Python311\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)
```

```
153/153             2s 2ms/step -
loss: 5.3591 - mae: 1.9083 - val_loss: 2.7433 - val_mae: 1.4228
Epoch 2/50
153/153             0s 2ms/step -
loss: 2.8096 - mae: 1.3657 - val_loss: 1.3131 - val_mae: 0.9668
Epoch 3/50
153/153             0s 1ms/step -
loss: 1.7056 - mae: 1.0407 - val_loss: 0.6652 - val_mae: 0.7013
Epoch 4/50
153/153             0s 1ms/step -
loss: 1.1644 - mae: 0.8824 - val_loss: 0.5073 - val_mae: 0.6179
Epoch 5/50
153/153             0s 1ms/step -
loss: 0.9857 - mae: 0.8052 - val_loss: 0.4171 - val_mae: 0.5596
Epoch 6/50
153/153             0s 1ms/step -
loss: 0.8815 - mae: 0.7486 - val_loss: 0.3729 - val_mae: 0.5285
Epoch 7/50
153/153             0s 2ms/step -
loss: 0.7924 - mae: 0.7213 - val_loss: 0.3477 - val_mae: 0.5077
Epoch 8/50
153/153             0s 1ms/step -
loss: 0.6673 - mae: 0.6603 - val_loss: 0.3103 - val_mae: 0.4796
```

```
Epoch 9/50
153/153             0s 2ms/step -
loss: 0.7404 - mae: 0.6990 - val_loss: 0.2907 - val_mae: 0.4588
Epoch 10/50
153/153             0s 1ms/step -
loss: 0.6584 - mae: 0.6603 - val_loss: 0.2778 - val_mae: 0.4478
Epoch 11/50
153/153             0s 2ms/step -
loss: 0.6529 - mae: 0.6519 - val_loss: 0.2533 - val_mae: 0.4215
Epoch 12/50
153/153             0s 1ms/step -
loss: 0.6107 - mae: 0.6268 - val_loss: 0.2144 - val_mae: 0.3826
Epoch 13/50
153/153             0s 1ms/step -
loss: 0.5308 - mae: 0.5928 - val_loss: 0.2143 - val_mae: 0.3827
Epoch 14/50
153/153             0s 1ms/step -
loss: 0.5404 - mae: 0.6008 - val_loss: 0.1981 - val_mae: 0.3611
Epoch 15/50
153/153             0s 1ms/step -
loss: 0.4945 - mae: 0.5669 - val_loss: 0.1777 - val_mae: 0.3420
Epoch 16/50
153/153             0s 1ms/step -
loss: 0.4626 - mae: 0.5489 - val_loss: 0.1754 - val_mae: 0.3447
Epoch 17/50
153/153             0s 1ms/step -
loss: 0.4785 - mae: 0.5599 - val_loss: 0.1627 - val_mae: 0.3207
Epoch 18/50
153/153             0s 1ms/step -
loss: 0.5131 - mae: 0.5684 - val_loss: 0.1471 - val_mae: 0.3081
Epoch 19/50
153/153             0s 1ms/step -
loss: 0.4505 - mae: 0.5393 - val_loss: 0.1482 - val_mae: 0.3167
Epoch 20/50
153/153             0s 1ms/step -
loss: 0.4437 - mae: 0.5379 - val_loss: 0.1528 - val_mae: 0.3227
Epoch 21/50
153/153             0s 2ms/step -
loss: 0.4133 - mae: 0.5139 - val_loss: 0.1313 - val_mae: 0.2894
Epoch 22/50
153/153             0s 3ms/step -
loss: 0.4737 - mae: 0.5542 - val_loss: 0.1333 - val_mae: 0.2866
Epoch 23/50
153/153             0s 1ms/step -
loss: 0.4426 - mae: 0.5294 - val_loss: 0.1295 - val_mae: 0.2917
Epoch 24/50
153/153             0s 1ms/step -
loss: 0.3992 - mae: 0.5075 - val_loss: 0.1185 - val_mae: 0.2779
```

```
Epoch 25/50
153/153             0s 1ms/step -
loss: 0.3968 - mae: 0.5001 - val_loss: 0.1073 - val_mae: 0.2519
Epoch 26/50
153/153             0s 1ms/step -
loss: 0.3736 - mae: 0.4880 - val_loss: 0.1102 - val_mae: 0.2564
Epoch 27/50
153/153             0s 1ms/step -
loss: 0.3488 - mae: 0.4720 - val_loss: 0.1136 - val_mae: 0.2731
Epoch 28/50
153/153             0s 1ms/step -
loss: 0.3720 - mae: 0.4876 - val_loss: 0.1015 - val_mae: 0.2511
Epoch 29/50
153/153             0s 1ms/step -
loss: 0.3625 - mae: 0.4806 - val_loss: 0.1013 - val_mae: 0.2507
Epoch 30/50
153/153             0s 1ms/step -
loss: 0.3979 - mae: 0.5075 - val_loss: 0.0971 - val_mae: 0.2477
Epoch 31/50
153/153             0s 1ms/step -
loss: 0.4061 - mae: 0.5039 - val_loss: 0.0931 - val_mae: 0.2339
Epoch 32/50
153/153             0s 1ms/step -
loss: 0.3510 - mae: 0.4707 - val_loss: 0.1063 - val_mae: 0.2698
Epoch 33/50
153/153             0s 2ms/step -
loss: 0.3795 - mae: 0.4934 - val_loss: 0.0856 - val_mae: 0.2322
Epoch 34/50
153/153             0s 1ms/step -
loss: 0.3893 - mae: 0.5014 - val_loss: 0.0722 - val_mae: 0.2055
Epoch 35/50
153/153             0s 1ms/step -
loss: 0.3712 - mae: 0.4824 - val_loss: 0.0908 - val_mae: 0.2477
Epoch 36/50
153/153             0s 1ms/step -
loss: 0.3700 - mae: 0.4905 - val_loss: 0.0708 - val_mae: 0.2078
Epoch 37/50
153/153             0s 1ms/step -
loss: 0.3702 - mae: 0.4813 - val_loss: 0.0831 - val_mae: 0.2362
Epoch 38/50
153/153             0s 1ms/step -
loss: 0.3539 - mae: 0.4783 - val_loss: 0.0708 - val_mae: 0.2143
Epoch 39/50
153/153             0s 1ms/step -
loss: 0.3492 - mae: 0.4774 - val_loss: 0.0710 - val_mae: 0.2138
Epoch 40/50
153/153             0s 1ms/step -
loss: 0.3209 - mae: 0.4586 - val_loss: 0.0685 - val_mae: 0.2112
```

```
Epoch 41/50
153/153             0s 1ms/step -
loss: 0.3655 - mae: 0.4817 - val_loss: 0.0765 - val_mae: 0.2267
Epoch 42/50
153/153             0s 1ms/step -
loss: 0.3405 - mae: 0.4587 - val_loss: 0.0629 - val_mae: 0.2022
Epoch 43/50
153/153             0s 1ms/step -
loss: 0.3113 - mae: 0.4456 - val_loss: 0.0633 - val_mae: 0.2033
Epoch 44/50
153/153             0s 1ms/step -
loss: 0.3284 - mae: 0.4531 - val_loss: 0.0585 - val_mae: 0.1950
Epoch 45/50
153/153             0s 1ms/step -
loss: 0.3189 - mae: 0.4561 - val_loss: 0.0583 - val_mae: 0.1975
Epoch 46/50
153/153             0s 1ms/step -
loss: 0.3387 - mae: 0.4625 - val_loss: 0.0560 - val_mae: 0.1918
Epoch 47/50
153/153             0s 1ms/step -
loss: 0.3144 - mae: 0.4517 - val_loss: 0.0510 - val_mae: 0.1811
Epoch 48/50
153/153             0s 1ms/step -
loss: 0.3285 - mae: 0.4617 - val_loss: 0.0687 - val_mae: 0.2165
Epoch 49/50
153/153             0s 1ms/step -
loss: 0.3156 - mae: 0.4591 - val_loss: 0.0522 - val_mae: 0.1855
Epoch 50/50
153/153             0s 1ms/step -
loss: 0.3255 - mae: 0.4656 - val_loss: 0.0555 - val_mae: 0.1915
```

[17]:
```python
import pandas as pd

# Extraer las métricas de cada experimento
# Experiment 1
train_mae_exp1 = history_exp1.history['mae'][-1]
val_mae_exp1 = history_exp1.history['val_mae'][-1]
train_mse_exp1 = history_exp1.history['loss'][-1]
val_mse_exp1 = history_exp1.history['val_loss'][-1]

# Experiment 2
train_mae_exp2 = history_exp2.history['mae'][-1]
val_mae_exp2 = history_exp2.history['val_mae'][-1]
train_mse_exp2 = history_exp2.history['loss'][-1]
val_mse_exp2 = history_exp2.history['val_loss'][-1]

# Experiment 3
```

```
train_mae_exp3 = history_exp3.history['mae'][-1]
val_mae_exp3 = history_exp3.history['val_mae'][-1]
train_mse_exp3 = history_exp3.history['loss'][-1]
val_mse_exp3 = history_exp3.history['val_loss'][-1]

# Experiment 4
train_mae_exp4 = history_exp4.history['mae'][-1]
val_mae_exp4 = history_exp4.history['val_mae'][-1]
train_mse_exp4 = history_exp4.history['loss'][-1]
val_mse_exp4 = history_exp4.history['val_loss'][-1]

# Crear la tabla comparativa
data = {
    "Experiment": ["Single Dense Layer", "Three Dense Layers", "Dense +␣
 ↪Dropout", "Dense + Dropout + BatchNorm"],
    "Train MAE": [train_mae_exp1, train_mae_exp2, train_mae_exp3,␣
 ↪train_mae_exp4],
    "Validation MAE": [val_mae_exp1, val_mae_exp2, val_mae_exp3, val_mae_exp4],
    "Train MSE": [train_mse_exp1, train_mse_exp2, train_mse_exp3,␣
 ↪train_mse_exp4],
    "Validation MSE": [val_mse_exp1, val_mse_exp2, val_mse_exp3, val_mse_exp4]
}

# Convertir a DataFrame y mostrar
comparative_table = pd.DataFrame(data)
print(comparative_table)
```

```
                  Experiment  Train MAE  Validation MAE  Train MSE  \
0           Single Dense Layer   0.127547        0.157150   0.026370
1           Three Dense Layers   0.108935        0.159287   0.019483
2              Dense + Dropout   0.363670        0.286266   0.226369
3  Dense + Dropout + BatchNorm   0.473906        0.191514   0.339518

   Validation MSE
0        0.040039
1        0.042471
2        0.125392
3        0.055512
```