```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
```

```
C:\ProgramData\anaconda3\Lib\site-packages\numpy\
_distributor_init.py:30: UserWarning: loaded more than 1 DLL
from .libs:
C:\ProgramData\anaconda3\Lib\site-packages\numpy\.libs\
libopenblas.FB5AE2TYXYH2IJRDKGDGQ3XBKLKTF43H.gfortran-win_amd64.dll
C:\ProgramData\anaconda3\Lib\site-packages\numpy\.libs\
libopenblas64__v0.3.21-gcc_10_3_0.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
```

```
import tensorflow as tf
from tensorflow import keras
```

```
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing()
```

Data Set Characteristics:

Number of Instances:
20640

Number of Attributes:
8 numeric, predictive attributes and the target

Attribute Information:

- *MedInc*: median income in block
- *HouseAge*: median house age in block
- *AveRooms*: average number of rooms
- *AveBedrms*: average number of bedrooms
- *Population*: block population
- *AveOccup*: average house occupancy
- *Latitude*: house block latitude
- *Longitude*: house block longitude

Target

The target variable is the median house value in units of 100,000 for California districts.

```
print(housing.feature_names)

['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population',
'AveOccup', 'Latitude', 'Longitude']

from sklearn.model_selection import train_test_split
X_train_full, X_test, y_train_full, y_test =
train_test_split(housing.data, housing.target, random_state = 42)
X_train, X_valid, y_train, y_valid = train_test_split(X_train_full,
y_train_full, random_state = 42)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_valid = scaler.transform(X_valid)
X_test = scaler.transform(X_test)

np.random.seed(42)
tf.random.set_seed(42)

X_train.shape

(11610, 8)
```

Neural Network Diagram

```
model = keras.models.Sequential([
    keras.layers.Dense(30, activation = "relu", input_shape = [8]),
    keras.layers.Dense(30, activation = "relu"),
    keras.layers.Dense(1)
])

model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 30)                270

 dense_1 (Dense)             (None, 30)                930

 dense_2 (Dense)             (None, 1)                 31

=================================================================
Total params: 1231 (4.81 KB)
Trainable params: 1231 (4.81 KB)
```

```
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________

model.compile(loss = "mean_squared_error",
              optimizer = keras.optimizers.SGD(learning_rate=1e-3),
              metrics = ['mae'])

model_history = model.fit(X_train, y_train, epochs = 20,
validation_data = (X_valid, y_valid))

Epoch 1/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3943
- mae: 0.4493 - val_loss: 0.4064 - val_mae: 0.4394
Epoch 2/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3923
- mae: 0.4475 - val_loss: 0.4290 - val_mae: 0.4434
Epoch 3/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3906
- mae: 0.4466 - val_loss: 0.4194 - val_mae: 0.4376
Epoch 4/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3889
- mae: 0.4453 - val_loss: 0.3986 - val_mae: 0.4380
Epoch 5/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3873
- mae: 0.4442 - val_loss: 0.3901 - val_mae: 0.4356
Epoch 6/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3856
- mae: 0.4435 - val_loss: 0.4061 - val_mae: 0.4344
Epoch 7/20
363/363 [==============================] - 1s 4ms/step - loss: 0.3844
- mae: 0.4423 - val_loss: 0.4012 - val_mae: 0.4338
Epoch 8/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3830
- mae: 0.4414 - val_loss: 0.3900 - val_mae: 0.4316
Epoch 9/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3816
- mae: 0.4408 - val_loss: 0.4047 - val_mae: 0.4314
Epoch 10/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3803
- mae: 0.4398 - val_loss: 0.4205 - val_mae: 0.4316
Epoch 11/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3793
- mae: 0.4388 - val_loss: 0.3938 - val_mae: 0.4297
Epoch 12/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3779
- mae: 0.4377 - val_loss: 0.4301 - val_mae: 0.4317
Epoch 13/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3768
- mae: 0.4370 - val_loss: 0.4214 - val_mae: 0.4327
Epoch 14/20
```

```
363/363 [==============================] - 1s 3ms/step - loss: 0.3756
- mae: 0.4360 - val_loss: 0.4079 - val_mae: 0.4312
Epoch 15/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3746
- mae: 0.4356 - val_loss: 0.4115 - val_mae: 0.4302
Epoch 16/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3735
- mae: 0.4351 - val_loss: 0.3961 - val_mae: 0.4281
Epoch 17/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3726
- mae: 0.4341 - val_loss: 0.4180 - val_mae: 0.4281
Epoch 18/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3717
- mae: 0.4333 - val_loss: 0.4117 - val_mae: 0.4260
Epoch 19/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3705
- mae: 0.4324 - val_loss: 0.4033 - val_mae: 0.4250
Epoch 20/20
363/363 [==============================] - 1s 3ms/step - loss: 0.3698
- mae: 0.4317 - val_loss: 0.4320 - val_mae: 0.4288
```

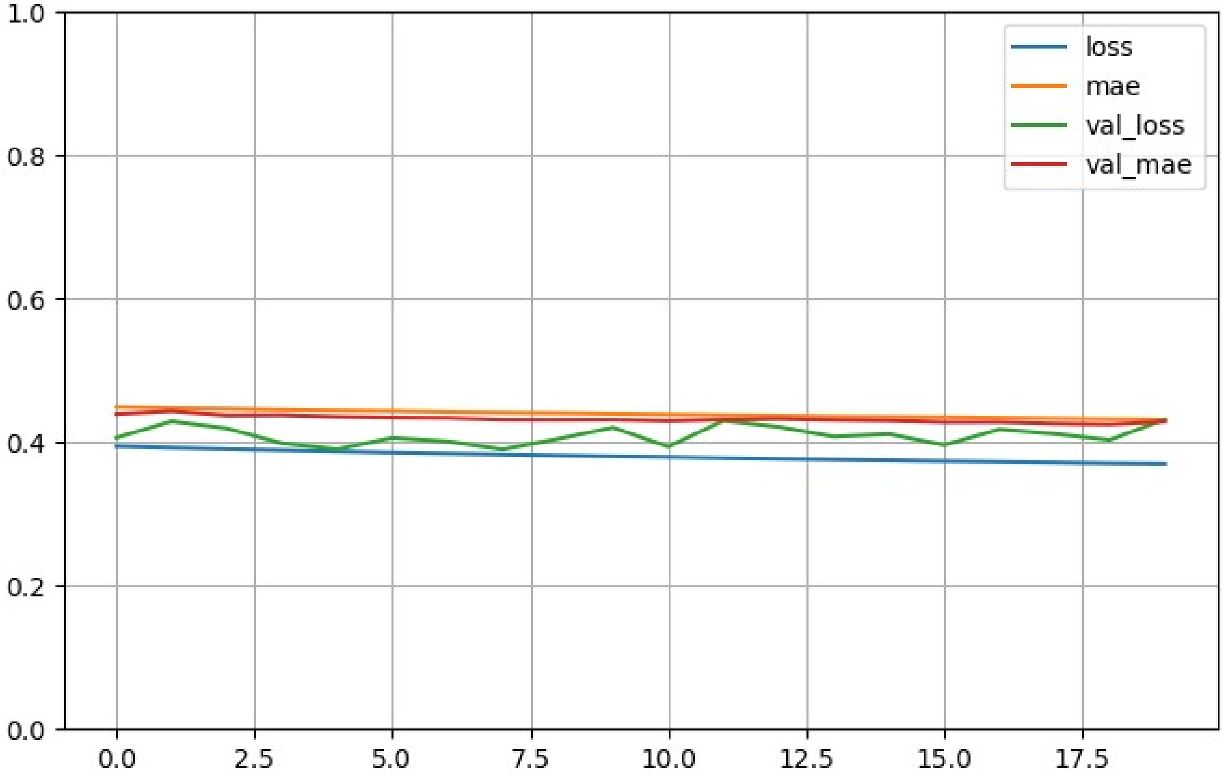
```
mae_test = model.evaluate(X_test, y_test)
```

```
162/162 [==============================] - 0s 2ms/step - loss:
30025.1699 - mae: 154.8190
```

```
model_history.history
```

```
{'loss': [0.3943287134170532,
  0.392260879278183,
  0.39056074619293213,
  0.3888949155807495,
  0.3873072564601898,
  0.38561171293258667,
  0.3843579590320587,
  0.3829883337020874,
  0.3816116750240326,
  0.38027894496917725,
  0.3792725205421448,
  0.37794122099876404,
  0.37678292393684387,
  0.3756231665611267,
  0.37464019656181335,
  0.3735232651233673,
  0.3726145327091217,
  0.37166276574134827,
  0.370490700006485,
  0.3697839379310608],
 'mae': [0.4492722451686859,
  0.447477787733078,
  0.44664981961250305,
  0.44532519578933716,
  0.4442335367202759,
  0.4435364007949829,
  0.4422949552536011,
  0.44139227271080017,
  0.44080784916877747,
  0.4397895336151123,
  0.4388081729412079,
  0.43770089745521545,
  0.43703022599220276,
  0.43598732352256775,
  0.4356386065483093,
  0.43511322140693665,
  0.4340752959251404,
  0.43330448865890503,
  0.4324326813220978,
  0.43172672390937805],
 'val_loss': [0.40637901425361633,
  0.4290466606616974,
  0.41938138008117676,
  0.3985860049724579,
  0.3901216685771942,
  0.40612098574638367,
  0.40118464827537537,
  0.3899715840816498,
  0.4046512544155121,
  0.42047297954559326,
  0.3937962055206299,
  0.43010565638542175,
  0.4213901460170746,
  0.40791672468185425,
  0.41147738695144653,
  0.39609864354133606,
  0.4179926812648773,
  0.41172268986701965,
  0.4032706022262573,
  0.4320087432861328],
 'val_mae': [0.439371258020401,
  0.4434032738208771,
  0.43762096762657166,
  0.4379856586456299,
  0.4356457591056824,
  0.43436649441719055,
  0.4338374733924866,
  0.4315696954727173,
  0.43137168884277344,
  0.43155765533447266,
  0.42969202995300293,
  0.4317205548286438,
  0.43273669481277466,
  0.43120279908180237,
  0.43015873432159424,
  0.4280836582183838,
  0.428141713142395,
  0.4259631931781769,
  0.42495104670524597,
  0.42875581979751587]}
```

```
pd.DataFrame(model_history.history).plot(figsize = (8,5))
plt.grid(True)
plt.gca().set_ylim(0,1)
plt.show()
```



```
X_new = X_test[:3]
```

```
y_pred = model.predict(X_new)
print(y_pred)
print(y_test[:3])
```

```
1/1 [==============================] - 0s 197ms/step
[[149.40912]
 [164.34145]
 [148.11191]]
[0.477   0.458   5.00001]

del model

keras.backend.clear_session()
```

Functional API

```
input_ = keras.layers.Input(shape=X_train.shape[1:])
hidden1 = keras.layers.Dense(30, activation="relu")(input_)
hidden2 = keras.layers.Dense(30, activation="relu")(hidden1)
concat = keras.layers.concatenate([input_, hidden2])
output = keras.layers.Dense(1)(concat)
model = keras.models.Model(inputs = [input_], outputs=[output])

model.summary()

Model: "model"
______________________________________________________________________
____________________________
 Layer (type)                Output Shape                 Param #
Connected to
======================================================================
============================
 input_4 (InputLayer)        [(None, 8)]                  0          []



 dense_1 (Dense)             (None, 30)                   270
['input_4[0][0]']


 dense_2 (Dense)             (None, 30)                   930
['dense_1[0][0]']


 concatenate (Concatenate)   (None, 38)                   0
['input_4[0][0]',

'dense_2[0][0]']


 dense_3 (Dense)             (None, 1)                    39
['concatenate[0][0]']


======================================================================
============================
Total params: 1239 (4.84 KB)
Trainable params: 1239 (4.84 KB)
```

```
Non-trainable params: 0 (0.00 Byte)
_______________________________________________________________________
____________________________

model.compile(loss = "mean_squared_error",
              optimizer = keras.optimizers.SGD(learning_rate=1e-3),
              metrics = ['mae'])

model_history = model.fit(X_train, y_train, epochs = 40,
validation_data = (X_valid, y_valid))

Epoch 1/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3792
- mae: 0.4401 - val_loss: 0.3610 - val_mae: 0.4259
Epoch 2/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3779
- mae: 0.4393 - val_loss: 0.4297 - val_mae: 0.4341
Epoch 3/40
363/363 [==============================] - 1s 1ms/step - loss: 0.3774
- mae: 0.4396 - val_loss: 0.3974 - val_mae: 0.4279
Epoch 4/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3758
- mae: 0.4382 - val_loss: 0.3768 - val_mae: 0.4296
Epoch 5/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3755
- mae: 0.4377 - val_loss: 0.3515 - val_mae: 0.4244
Epoch 6/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3737
- mae: 0.4372 - val_loss: 0.3993 - val_mae: 0.4279
Epoch 7/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3737
- mae: 0.4366 - val_loss: 0.3639 - val_mae: 0.4246
Epoch 8/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3725
- mae: 0.4362 - val_loss: 0.3490 - val_mae: 0.4217
Epoch 9/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3715
- mae: 0.4358 - val_loss: 0.3728 - val_mae: 0.4238
Epoch 10/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3706
- mae: 0.4350 - val_loss: 0.3819 - val_mae: 0.4236
Epoch 11/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3704
- mae: 0.4343 - val_loss: 0.3475 - val_mae: 0.4202
Epoch 12/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3691
- mae: 0.4336 - val_loss: 0.4122 - val_mae: 0.4258
Epoch 13/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3688
- mae: 0.4333 - val_loss: 0.3601 - val_mae: 0.4228
```

```
Epoch 14/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3677
- mae: 0.4324 - val_loss: 0.3521 - val_mae: 0.4216
Epoch 15/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3670
- mae: 0.4324 - val_loss: 0.3628 - val_mae: 0.4225
Epoch 16/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3663
- mae: 0.4320 - val_loss: 0.3450 - val_mae: 0.4198
Epoch 17/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3651
- mae: 0.4312 - val_loss: 0.3844 - val_mae: 0.4226
Epoch 18/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3654
- mae: 0.4309 - val_loss: 0.3577 - val_mae: 0.4194
Epoch 19/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3642
- mae: 0.4301 - val_loss: 0.3585 - val_mae: 0.4200
Epoch 20/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3636
- mae: 0.4298 - val_loss: 0.4092 - val_mae: 0.4244
Epoch 21/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3631
- mae: 0.4292 - val_loss: 0.3702 - val_mae: 0.4228
Epoch 22/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3624
- mae: 0.4290 - val_loss: 0.3905 - val_mae: 0.4238
Epoch 23/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3617
- mae: 0.4290 - val_loss: 0.3438 - val_mae: 0.4152
Epoch 24/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3611
- mae: 0.4277 - val_loss: 0.3716 - val_mae: 0.4196
Epoch 25/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3604
- mae: 0.4278 - val_loss: 0.3822 - val_mae: 0.4213
Epoch 26/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3601
- mae: 0.4276 - val_loss: 0.3718 - val_mae: 0.4199
Epoch 27/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3600
- mae: 0.4273 - val_loss: 0.3402 - val_mae: 0.4157
Epoch 28/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3593
- mae: 0.4266 - val_loss: 0.3639 - val_mae: 0.4208
Epoch 29/40
363/363 [==============================] - 1s 1ms/step - loss: 0.3588
- mae: 0.4266 - val_loss: 0.3505 - val_mae: 0.4180
Epoch 30/40
```

```
363/363 [==============================] - 0s 1ms/step - loss: 0.3582
- mae: 0.4262 - val_loss: 0.3783 - val_mae: 0.4187
Epoch 31/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3572
- mae: 0.4250 - val_loss: 0.3417 - val_mae: 0.4152
Epoch 32/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3570
- mae: 0.4254 - val_loss: 0.3931 - val_mae: 0.4223
Epoch 33/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3573
- mae: 0.4258 - val_loss: 0.3394 - val_mae: 0.4161
Epoch 34/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3561
- mae: 0.4245 - val_loss: 0.3475 - val_mae: 0.4182
Epoch 35/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3561
- mae: 0.4249 - val_loss: 0.3769 - val_mae: 0.4205
Epoch 36/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3552
- mae: 0.4245 - val_loss: 0.3358 - val_mae: 0.4128
Epoch 37/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3553
- mae: 0.4239 - val_loss: 0.3467 - val_mae: 0.4158
Epoch 38/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3547
- mae: 0.4242 - val_loss: 0.3364 - val_mae: 0.4142
Epoch 39/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3539
- mae: 0.4235 - val_loss: 0.3394 - val_mae: 0.4136
Epoch 40/40
363/363 [==============================] - 0s 1ms/step - loss: 0.3534
- mae: 0.4233 - val_loss: 0.3472 - val_mae: 0.4126
```

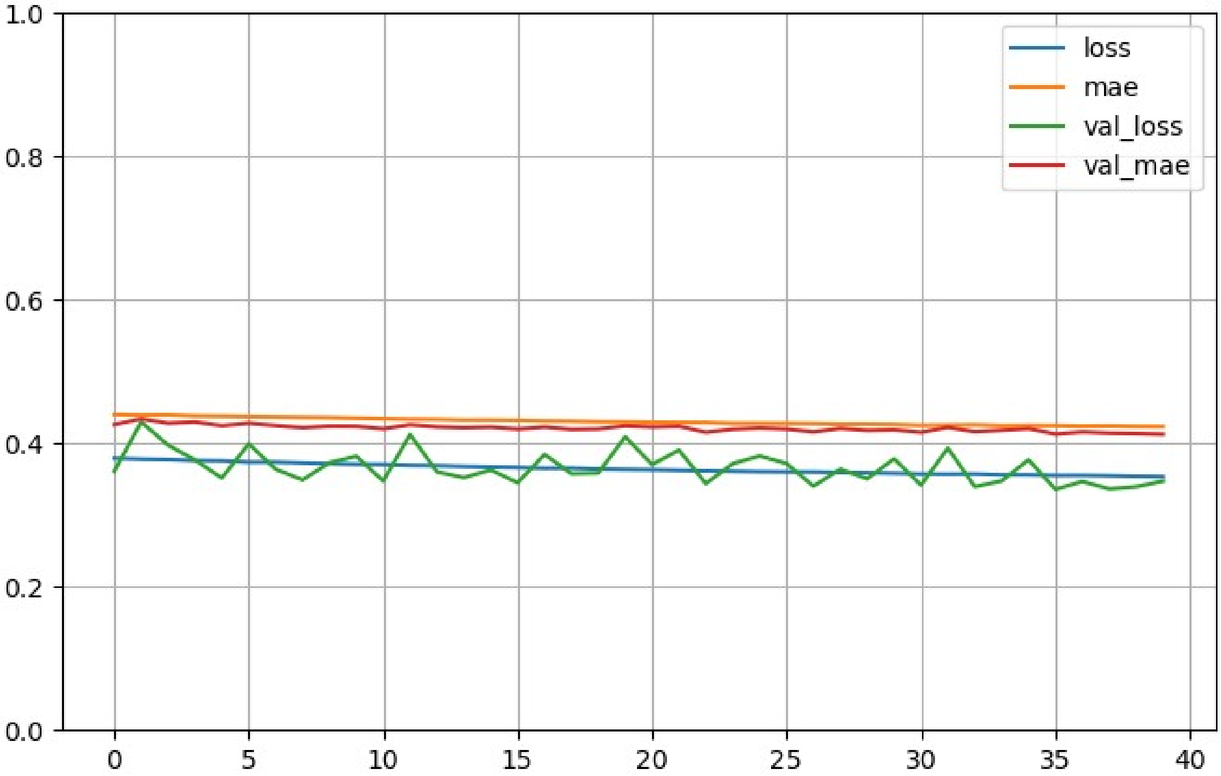
```
mae_test = model.evaluate(X_test, y_test)
```

```
162/162 [==============================] - 0s 754us/step - loss:
21324.0938 - mae: 143.7520
```

```
model_history.history
```

```
{'loss': [0.3791651129722595,
  0.3779199421405792,
  0.37737831473350525,
  0.3758224546909332,
  0.3754596710205078,
  0.37367740273475647,
  0.3736606538295746,
  0.37251195311546326,
  0.3714843690395355,
  0.3706281781196594,
  0.37035220861434937,
  0.3691158890724182,
  0.3687695562839508,
  0.36765122413635254,
  0.3669997453689575,
  0.3663310408592224,
  0.3651280999183655,
  0.36537042260169983,
  0.36415883898735046,
  0.36359429359436035,
  0.36311763525009155,
  0.36238008737564087,
  0.36166810989379883,
  0.36108213663101196,
  0.36044538021087646,
  0.3601008951663971,
  0.35995352268218994,
  0.3592613935470581,
  0.3588206171989441,
  0.35816359519958496,
  0.35721734166145325,
  0.35702767968177795,
  0.35726189613342285,
  0.35608306527137756,
  0.356071412563324,
  0.3552153706550598,
  0.35529372096061707,
  0.35473746061325073,
  0.35391050577163696,
  0.35342705249786377],
 'mae': [0.4400605857372284,
  0.43932223320007324,
  0.43959736824035645,
  0.4381815791130066,
  0.43773388862609863,
  0.4372248649597168,
  0.4366455674171448,
  0.43616679310798645,
  0.43578991293907166,
  0.43496841192245483,
  0.434280127286911,
  0.43357688188552856,
  0.4333229959011078,
  0.43238675594329834,
  0.4324222505092621,
  0.4320008456707001,
  0.4312053322792053,
  0.430869460105896,
  0.4301030933856964,
  0.4298229515552521,
  0.4292203187942505,
  0.4290238320827484,
  0.4289743900299072,
  0.42767271399497986,
  0.4277515709400177,
  0.4275660216808319,
  0.4272894561290741,
  0.4265604317188263,
  0.42664891481399536,
  0.4261969327926636,
  0.4249957203865051,
  0.4253928065299988,
  0.42577478289604187,
  0.42451199889183044,
  0.4248548746109009,
  0.4245174527168274,
  0.42388394474983215,
  0.42416316270828247,
  0.4234876036643982,
  0.42332592606544495],
 'val_loss': [0.36101147532463074,
  0.429714173078537,
  0.3973986506462097,
  0.37677887082099915,
  0.3515484929084778,
  0.39925622940063477,
  0.36392971873283386,
  0.34897130727767944,
  0.3728080987930298,
  0.3819088935852051,
  0.34749144315719604,
  0.41221147775650024,
  0.3601033389568329,
  0.35214152932167053,
  0.36280009150505066,
  0.34495946764945984,
  0.38438481092453003,
  0.35769668221473694,
  0.3585398197174072,
  0.4091602563858032,
  0.37022486329078674,
  0.39049458503723145,
  0.3437998592853546,
  0.37159380316734314,
  0.3822316527366638,
  0.3717559278011322,
  0.3401782810688019,
  0.3639087677001953,
  0.35050398111343384,
  0.3783251941204071,
  0.3416934907436371,
  0.3931162357330322,
  0.3394372761249542,
  0.34752190113067627,
  0.3768715560436249,
  0.33583441376686096,
  0.3467106819152832,
  0.3364410102367401,
  0.3393889367580414,
  0.34720656275749207],
 'val_mae': [0.4259410500526428,
  0.4340710937976837,
  0.42793911695480347,
  0.4296204745769501,
  0.42442557215690613,
  0.42789626121520996,
  0.42460450530052185,
  0.42171528935432434,
  0.42383986711502075,
  0.423625648021698,
  0.42019957304000854,
  0.4257792830467224,
  0.42284780740737915,
  0.4216395914554596,
  0.4224943518638611,
  0.41979730129241943,
  0.422561913728714,
  0.41940534114837646,
  0.4199816584587097,
  0.4244321882724762,
  0.42278802394866943,
  0.423784464597702,
  0.4152480959892273,
  0.4195975959300995,
  0.4212793707847595,
  0.4198818802833557,
  0.4157438278198242,
  0.42079848051071167,
  0.4179823696613312,
  0.4187231957912445,
  0.4152330458164215,
  0.42227181792259216,
  0.4160514771938324,
  0.4181712865829468,
  0.42051830887794495,
  0.4128258228302002,
  0.4157634377479553,
  0.4142112135887146,
  0.4136275351047516,
  0.4126463234424591]}
```

```
pd.DataFrame(model_history.history).plot(figsize = (8,5))
plt.grid(True)
plt.gca().set_ylim(0,1)
plt.show()
```



Saving and Restoring

```
model.save("my_func_model.h5")
```

```
C:\ProgramData\anaconda3\Lib\site-packages\keras\src\engine\
training.py:3000: UserWarning: You are saving your model as an HDF5
file via `model.save()`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(
```

```
%pwd
```

```
'C:\\Users\\hp\\Downloads\\Deep Learning'
```

```
del model
```

```
keras.backend.clear_session()
```

```
model = keras.models.load_model("my_func_model.h5")

model.summary()

Model: "model"
______________________________________________________________________
____________________________
 Layer (type)                Output Shape                 Param #
Connected to
======================================================================
============================
 input_4 (InputLayer)        [(None, 8)]                  0         []



 dense_1 (Dense)             (None, 30)                   270
['input_4[0][0]']


 dense_2 (Dense)             (None, 30)                   930
['dense_1[0][0]']


 concatenate (Concatenate)   (None, 38)                   0
['input_4[0][0]',

'dense_2[0][0]']


 dense_3 (Dense)             (None, 1)                    39
['concatenate[0][0]']


======================================================================
============================
Total params: 1239 (4.84 KB)
Trainable params: 1239 (4.84 KB)
Non-trainable params: 0 (0.00 Byte)
______________________________________________________________________
____________________________

y_pred = model.predict(X_new)
print(y_pred)

1/1 [==============================] - 0s 166ms/step
[[138.0817 ]
 [146.99583]
 [151.09116]]

del model
```

```
keras.backend.clear_session()
```

Using Callbacks during Training

```
np.random.seed(42)
tf.random.set_seed(42)

model = keras.models.Sequential([
    keras.layers.Dense(30, activation = "relu", input_shape = [8]),
    keras.layers.Dense(30, activation = "relu"),
    keras.layers.Dense(1)
])

model.compile(loss="mse",
optimizer=keras.optimizers.SGD(learning_rate=1e-3))

checkpoint_cb = keras.callbacks.ModelCheckpoint("Model-
{epoch:02d}.h5")

history = model.fit(X_train, y_train, epochs = 10,
                    validation_data = (X_valid, y_valid),
                    callbacks = [checkpoint_cb])
```

```
Epoch 1/10
363/363 [==============================] - 1s 2ms/step - loss: 2.0994
- val_loss: 1.1882
Epoch 2/10
363/363 [==============================] - 1s 2ms/step - loss: 0.7151
- val_loss: 0.6380
Epoch 3/10
363/363 [==============================] - 1s 2ms/step - loss: 0.6317
- val_loss: 0.5768
Epoch 4/10
363/363 [==============================] - 1s 2ms/step - loss: 0.5899
- val_loss: 0.5845
Epoch 5/10
363/363 [==============================] - 1s 2ms/step - loss: 0.5576
- val_loss: 0.5559
Epoch 6/10
363/363 [==============================] - 1s 2ms/step - loss: 0.5298
- val_loss: 0.5099
Epoch 7/10
363/363 [==============================] - 1s 2ms/step - loss: 0.5069
- val_loss: 0.4714
Epoch 8/10
363/363 [==============================] - 1s 2ms/step - loss: 0.4875
- val_loss: 0.4803
Epoch 9/10
363/363 [==============================] - 1s 1ms/step - loss: 0.4706
- val_loss: 0.4530
```

```
Epoch 10/10
363/363 [==============================] - 1s 2ms/step - loss: 0.4566
- val_loss: 0.4454
```

```python
del model
keras.backend.clear_session()

model = keras.models.load_model("Model-10.h5")

mse_test = model.evaluate(X_test, y_test)
```

```
162/162 [==============================] - 1s 2ms/step - loss:
67096.9141
```

Best Model Only

```python
del model
keras.backend.clear_session()

model = keras.models.Sequential([
    keras.layers.Dense(30, activation = "relu", input_shape = [8]),
    keras.layers.Dense(30, activation = "relu"),
    keras.layers.Dense(1)
])

model.compile(loss="mse",
optimizer=keras.optimizers.SGD(learning_rate=1e-3))

checkpoint_cb = keras.callbacks.ModelCheckpoint("Best_Model.h5",
save_best_only = True)

history = model.fit(X_train, y_train, epochs = 10,
                    validation_data = (X_valid, y_valid),
                    callbacks = [checkpoint_cb])
```

```
Epoch 1/10
363/363 [==============================] - 2s 4ms/step - loss: 1.6932
- val_loss: 0.8270
Epoch 2/10
363/363 [==============================] - 1s 3ms/step - loss: 0.7516
- val_loss: 0.6691
Epoch 3/10
363/363 [==============================] - 1s 3ms/step - loss: 0.6662
- val_loss: 0.6065
Epoch 4/10
363/363 [==============================] - 1s 3ms/step - loss: 0.6126
- val_loss: 0.5602
Epoch 5/10
363/363 [==============================] - 1s 3ms/step - loss: 0.5701
- val_loss: 0.5217
Epoch 6/10
```

```
363/363 [==============================] - 1s 3ms/step - loss: 0.5356
- val_loss: 0.4925
Epoch 7/10
363/363 [==============================] - 1s 3ms/step - loss: 0.5087
- val_loss: 0.4692
Epoch 8/10
363/363 [==============================] - 1s 3ms/step - loss: 0.4878
- val_loss: 0.4520
Epoch 9/10
363/363 [==============================] - 1s 3ms/step - loss: 0.4711
- val_loss: 0.4397
Epoch 10/10
363/363 [==============================] - 1s 3ms/step - loss: 0.4575
- val_loss: 0.4332
```

```
model = keras.models.load_model("Best_Model.h5") # rollback to best
model
mse_test = model.evaluate(X_test, y_test)
```

```
162/162 [==============================] - 0s 2ms/step - loss:
80629.0469
```

Best Model Only

```
del model
keras.backend.clear_session()

model = keras.models.Sequential([
    keras.layers.Dense(30, activation = "relu", input_shape = [8]),
    keras.layers.Dense(30, activation = "relu"),
    keras.layers.Dense(1)
])

model.compile(loss="mse",
optimizer=keras.optimizers.SGD(learning_rate=1e-3))

checkpoint_cb = keras.callbacks.ModelCheckpoint("early_stop_Model.h5",
save_best_only = True)

early_stopping_cb = keras.callbacks.EarlyStopping(patience=10,

restore_best_weights=True)
# patience : Number of epochs with no improvement after which training
will be stopped.

history = model.fit(X_train, y_train, epochs = 200,
                    validation_data = (X_valid, y_valid),
                    callbacks = [checkpoint_cb, early_stopping_cb])
```

```
Epoch 1/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4663
```

```
- val_loss: 0.4430
Epoch 2/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4567
- val_loss: 0.4457
Epoch 3/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4485
- val_loss: 0.4300
Epoch 4/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4415
- val_loss: 0.4394
Epoch 5/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4353
- val_loss: 0.4314
Epoch 6/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4299
- val_loss: 0.4335
Epoch 7/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4251
- val_loss: 0.4429
Epoch 8/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4209
- val_loss: 0.4293
Epoch 9/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4172
- val_loss: 0.4257
Epoch 10/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4138
- val_loss: 0.4453
Epoch 11/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4110
- val_loss: 0.4348
Epoch 12/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4080
- val_loss: 0.4530
Epoch 13/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4055
- val_loss: 0.4616
Epoch 14/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4031
- val_loss: 0.4468
Epoch 15/200
363/363 [==============================] - 1s 3ms/step - loss: 0.4008
- val_loss: 0.4406
Epoch 16/200
363/363 [==============================] - 1s 3ms/step - loss: 0.3987
- val_loss: 0.4360
Epoch 17/200
363/363 [==============================] - 1s 3ms/step - loss: 0.3968
- val_loss: 0.4532
```

```
Epoch 18/200
363/363 [==============================] - 1s 3ms/step - loss: 0.3949
- val_loss: 0.4483
Epoch 19/200
363/363 [==============================] - 1s 3ms/step - loss: 0.3927
- val_loss: 0.4332
```

```
model = keras.models.load_model("early_stop_Model.h5")
mse_test = model.evaluate(X_test, y_test)
```

```
162/162 [==============================] - 1s 2ms/step - loss:
86827.9844
```