In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
import tensorflow as tf
from tensorflow import keras
```

In [3]:
```
from sklearn.datasets import fetch_california_housing
```

In [4]:
```
df = fetch_california_housing()
```

In [5]:
```
print(df.feature_names)
```

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

In [6]:
```
from sklearn.model_selection import train_test_split
```

In [7]:
```
x_train_full,x_test,y_train_full,y_test =train_test_split(df.data,df.target,random_state = 42)
x_train,x_valid,y_train,y_valid = train_test_split(x_train_full,y_train_full,random_state=42)
```

In [8]:
```
from sklearn.preprocessing import StandardScaler
```

In [9]:
```
scaler = StandardScaler()
```

In [10]:
```
x_train = scaler.fit_transform(x_train)
x_valid = scaler.transform(x_valid)
x_test = scaler.transform(x_test)
```

In [11]:
```
np.random.seed(42)
tf.random.set_seed(42)
```

In [12]:
```
x_train.shape
```

Out[12]:
```
(11610, 8)
```

In [13]:
```
model = keras.models.Sequential([
    keras.layers.Dense(30,activation = 'relu',input_shape = [8]),
    keras.layers.Dense(30,activation = 'relu'),
    keras.layers.Dense(1)
])
```

In [14]:
```
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: 1,231
Trainable params: 1,231
Non-trainable params: 0
_________________________________________________________________
```

In [15]:
```
model.compile(loss = 'mean_squared_error',
              optimizer = keras.optimizers.SGD(),
              metrics = ['mae'])
```

In [16]:
```
model_history = model.fit(x_train,y_train,epochs = 30,validation_data = (x_valid,y_valid))
```

```
Epoch 1/30
363/363 [==============================] - 3s 4ms/step - loss: 0.6860 - mae: 0.5876 - val_loss: 0.6226 - val_ma
e: 0.4769
Epoch 2/30
363/363 [==============================] - 1s 3ms/step - loss: 0.4286 - mae: 0.4700 - val_loss: 3.4166 - val_ma
e: 0.4771
Epoch 3/30
363/363 [==============================] - 1s 3ms/step - loss: 0.4907 - mae: 0.4801 - val_loss: 2.1055 - val_ma
e: 0.4678
Epoch 4/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3964 - mae: 0.4451 - val_loss: 0.4980 - val_ma
e: 0.4321
Epoch 5/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3761 - mae: 0.4329 - val_loss: 0.3935 - val_ma
e: 0.4251
Epoch 6/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3636 - mae: 0.4265 - val_loss: 0.3986 - val_ma
```

```
e: 0.4172
Epoch 7/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3592 - mae: 0.4226 - val_loss: 0.3733 - val_ma
e: 0.4131
Epoch 8/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3504 - mae: 0.4175 - val_loss: 0.3645 - val_ma
e: 0.4113
Epoch 9/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3447 - mae: 0.4131 - val_loss: 0.3582 - val_ma
e: 0.3986
Epoch 10/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3416 - mae: 0.4117 - val_loss: 0.3622 - val_ma
e: 0.4074
Epoch 11/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3389 - mae: 0.4088 - val_loss: 0.3334 - val_ma
e: 0.3937
Epoch 12/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3371 - mae: 0.4072 - val_loss: 0.3537 - val_ma
e: 0.3943
Epoch 13/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3325 - mae: 0.4049 - val_loss: 0.3405 - val_ma
e: 0.3889
Epoch 14/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3280 - mae: 0.4017 - val_loss: 0.3269 - val_ma
e: 0.3876
Epoch 15/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3293 - mae: 0.4013 - val_loss: 0.3321 - val_ma
e: 0.3928
Epoch 16/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3234 - mae: 0.3994 - val_loss: 0.3235 - val_ma
e: 0.3912
Epoch 17/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3215 - mae: 0.3976 - val_loss: 0.3576 - val_ma
e: 0.4040
Epoch 18/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3210 - mae: 0.3962 - val_loss: 0.3118 - val_ma
e: 0.3799
Epoch 19/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3224 - mae: 0.3956 - val_loss: 0.3251 - val_ma
e: 0.3821
Epoch 20/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3171 - mae: 0.3931 - val_loss: 0.3686 - val_ma
e: 0.4010
Epoch 21/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3137 - mae: 0.3914 - val_loss: 0.3139 - val_ma
e: 0.3946
Epoch 22/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3120 - mae: 0.3906 - val_loss: 0.3211 - val_ma
e: 0.3843
Epoch 23/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3104 - mae: 0.3883 - val_loss: 0.3098 - val_ma
e: 0.3751
Epoch 24/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3101 - mae: 0.3891 - val_loss: 0.3160 - val_ma
e: 0.3832
Epoch 25/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3078 - mae: 0.3866 - val_loss: 0.3335 - val_ma
e: 0.3765
Epoch 26/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3066 - mae: 0.3858 - val_loss: 0.3140 - val_ma
e: 0.3801
Epoch 27/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3063 - mae: 0.3854 - val_loss: 0.3009 - val_ma
e: 0.3741
Epoch 28/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3062 - mae: 0.3838 - val_loss: 0.3166 - val_ma
e: 0.3818
Epoch 29/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3081 - mae: 0.3855 - val_loss: 0.3078 - val_ma
e: 0.3860
Epoch 30/30
363/363 [==============================] - 1s 3ms/step - loss: 0.3069 - mae: 0.3845 - val_loss: 0.3074 - val_ma
e: 0.3743
```

In [17]:
```
mae_test = model.evaluate(x_test,y_test)
```

```
162/162 [==============================] - 0s 2ms/step - loss: 0.3095 - mae: 0.3819
```

In [18]:
```
model_history.history
```

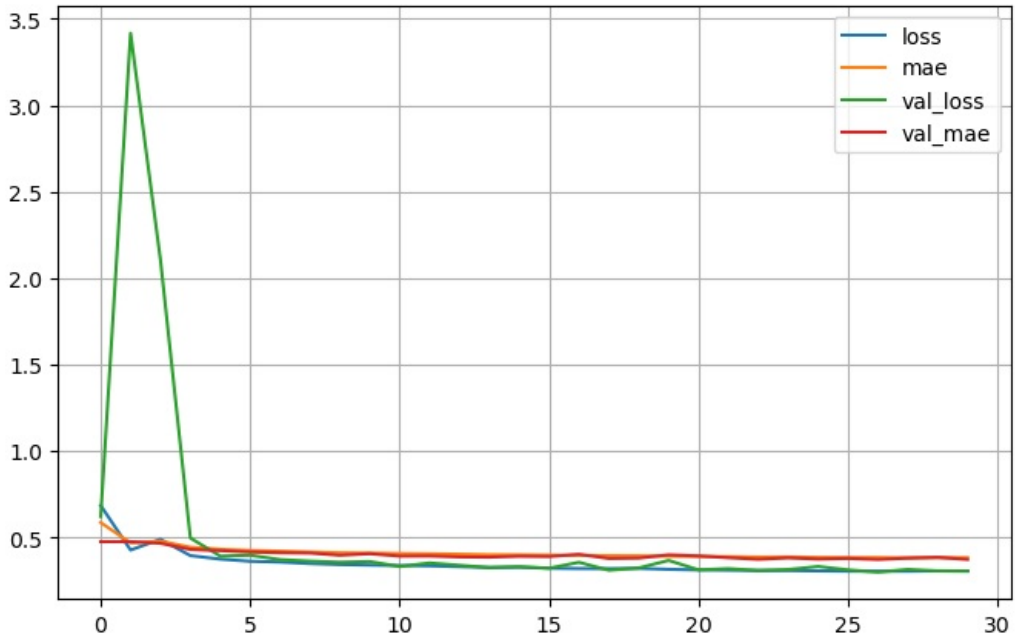
Out[18]:
```
{'loss': [0.6860350966453552,
  0.42857858538627625,
  0.49068236351013184,
  0.39637333154678345,
  0.37606489658355713,
  0.36363890767097473,
  0.3591632544994354,
  0.3504336178302765,
  0.3447352349758148,
```

```
 0.3416411280632019,
 0.3388660252094269,
 0.33707621693611145,
 0.3325381278991699,
 0.32801133394241333,
 0.32931452989578247,
 0.3233952522277832,
 0.32146942615509033,
 0.3209802508354187,
 0.32241880893707275,
 0.3171079456806183,
 0.3137245774269104,
 0.3120401203632355,
 0.3104282021522522,
 0.3100728392601013,
 0.30784472823143005,
 0.30662301182746887,
 0.30626380443573,
 0.3062289357185364,
 0.30806034803390503,
 0.30694344639778137],
'mae': [0.5875542759895325,
 0.4699855148792267,
 0.48008817434310913,
 0.44509610533714294,
 0.4329044818878174,
 0.426518052816391,
 0.4226357340812683,
 0.41745978593826294,
 0.4131312370300293,
 0.41174769401550293,
 0.40878793597221375,
 0.4072113037109375,
 0.40491777658462524,
 0.40173640847206116,
 0.40128058195114136,
 0.39936044812202454,
 0.39757245779037476,
 0.39616405963897705,
 0.3955683708190918,
 0.39313799142837524,
 0.3913708031177521,
 0.3906291127204895,
 0.38829725980758667,
 0.38911840319633484,
 0.38659846782684326,
 0.38575366139411926,
 0.3853800892829895,
 0.38375401496887207,
 0.3855398893356323,
 0.3844830095767975],
'val_loss': [0.6225500702857971,
 3.416584014892578,
 2.105531930923462,
 0.4979768991470337,
 0.39347290992736816,
 0.3986354172229767,
 0.3732934296131134,
 0.3645224869251251,
 0.35822364687919617,
 0.3621854782104492,
 0.33341464400291443,
 0.35373955965042114,
 0.3404729962348938,
 0.32689765095710754,
 0.3320634663105011,
 0.3235388696193695,
 0.3575971722602844,
 0.3117719292640686,
 0.32507187128067017,
 0.36857613921165466,
 0.31391409039497375,
 0.32108721137046814,
 0.30976879596710205,
 0.31599611043930054,
 0.3334956765174866,
 0.3139708936214447,
 0.30089184641838074,
 0.3165510594844818,
 0.3077872395515442,
 0.3073861300945282],
'val_mae': [0.47691401839256287,
 0.4771106243133545,
 0.46782588958740234,
 0.43206143379211426,
 0.4250621199607849,
 0.4172133207321167,
 0.4130727946758270,
 0.4112563729286194,
```

```
  0.39858952164649963,
  0.40735992789268494,
  0.39371392130851746,
  0.394314706325531,
  0.38892585039138794,
  0.3875541388988495,
  0.39283493161201477,
  0.3912133276462555,
  0.403969407081604,
  0.37989693880081177,
  0.38205376267433167,
  0.4010195732116699,
  0.3946186304092407,
  0.38433516025543213,
  0.3751460611820221,
  0.3832392990589142,
  0.37651559710502625,
  0.3801248073577881,
  0.3740726113319397,
  0.3817773759365082,
  0.38596710562705994,
  0.3742929697036743]}
```

In [19]:
```
pd.DataFrame(model_history.history).plot(figsize = (8,5))
plt.grid(True)

plt.show()
```



In [20]:
```
x_new = x_test[:3]
```

In [21]:
```
y_pred = model.predict(x_new)
```

```
1/1 [==============================] - 0s 236ms/step
```

In [22]:
```
y_pred
```

Out[22]:
```
array([[0.6329897],
       [1.3350961],
       [4.173784 ]], dtype=float32)
```

In [23]:
```
y_test[:3]
```

Out[23]:
```
array([0.477  , 0.458  , 5.00001])
```

Functional API

Not all neural network models are simply sequential. Some may have complex tropologies.Some may have multiple imputs and/or multiple outputs. For example a wide and deep learning connections all or path of inputs directly to the output layer

In [24]:
```
del model
```

In [25]:
```
keras.backend.clear_session()
```

In [29]:
```
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])
```

In [30]:
```
model.summary()
```

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

 dense_7 (Dense)                (None, 30)           270         ['input_4[0][0]']

 dense_8 (Dense)                (None, 30)           930         ['dense_7[0][0]']

 concatenate_2 (Concatenate)    (None, 38)           0           ['input_4[0][0]',
                                                                  'dense_8[0][0]']

 dense_9 (Dense)                (None, 1)            39          ['concatenate_2[0][0]']

==================================================================================================
Total params: 1,239
Trainable params: 1,239
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [ ]: