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

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

In [3]:
```
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing()
```

In [4]:
```
print(housing.feature_names)
```

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

In [5]:
```
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)
```

In [7]:
```
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)
```

In [8]:
```
np.random.seed(42)
tf.random.set_seed(42)
```

In [9]:
```
X_train.shape
```

Out[9]:
```
(11610, 8)
```

In [10]:
```
model = keras.models.Sequential([
    keras.layers.Dense(30, activation = "relu", input_shape=[8]),
    keras.layers.Dense(30, activation = "relu"),
    keras.layers.Dense(1)
])
```

In [12]:
```
model.compile(loss = "mean_squared_error",
              optimizer = keras.optimizers.SGD(lr=1e-3),
              metrics = ["mae"])
```

In [13]:
```
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 [14]:
```
model_history = model.fit(X_train, y_train, epochs = 20, validation_data = (X_valid, y_valid))
```

```
Epoch 1/20
363/363 [==============================] - 0s 845us/step - loss: 1.8866 - mae: 0.9900 - val_loss: 0.7126 - val_mae: 0.6368
Epoch 2/20
363/363 [==============================] - 0s 575us/step - loss: 0.6577 - mae: 0.6042 - val_loss: 0.6880 - val_mae: 0.5704
Epoch 3/20
363/363 [==============================] - 0s 570us/step - loss: 0.5934 - mae: 0.5618 - val_loss: 0.5803 - val_mae: 0.5352
Epoch 4/20
363/363 [==============================] - 0s 599us/step - loss: 0.5557 - mae: 0.5398 - val_loss: 0.5166 - val_mae: 0.5207
Epoch 5/20
363/363 [==============================] - 0s 600us/step - loss: 0.5272 - mae: 0.5237 - val_loss: 0.4895 - val_mae: 0.5022
Epoch 6/20
363/363 [==============================] - 0s 626us/step - loss: 0.5033 - mae: 0.5113 - val_loss: 0.4951 - val_mae: 0.4934
Epoch 7/20
363/363 [==============================] - 0s 757us/step - loss: 0.4854 - mae: 0.5010 - val_loss: 0.4861 - val_mae: 0.4838
Epoch 8/20
363/363 [==============================] - 0s 647us/step - loss: 0.4709 - mae: 0.4924 - val_loss: 0.4554 - val_mae: 0.4753
Epoch 9/20
363/363 [==============================] - 0s 574us/step - loss: 0.4578 - mae: 0.4857 - val_loss: 0.4413 - val_mae: 0.4671
Epoch 10/20
363/363 [==============================] - 0s 578us/step - loss: 0.4474 - mae: 0.4797 - val_loss: 0.4379 - val_mae: 0.4623
Epoch 11/20
363/363 [==============================] - 0s 607us/step - loss: 0.4393 - mae: 0.4744 - val_loss: 0.4396 - val_mae: 0.4638
Epoch 12/20
363/363 [==============================] - 0s 581us/step - loss: 0.4318 - mae: 0.4703 - val_loss: 0.4507 - val_mae: 0.4573
Epoch 13/20
363/363 [==============================] - 0s 591us/step - loss: 0.4261 - mae: 0.4674 - val_loss: 0.3997 - val_mae: 0.4517
Epoch 14/20
363/363 [==============================] - 0s 590us/step - loss: 0.4202 - mae: 0.4636 - val_loss: 0.3956 - val_mae: 0.4497
Epoch 15/20
363/363 [==============================] - 0s 580us/step - loss: 0.4155 - mae: 0.4613 - val_loss: 0.3916 - val_mae: 0.4464
Epoch 16/20
363/363 [==============================] - 0s 593us/step - loss: 0.4112 - mae: 0.4591 - val_loss: 0.3937 - val_mae: 0.4445
Epoch 17/20
363/363 [==============================] - 0s 595us/step - loss: 0.4077 - mae: 0.4569 - val_loss: 0.3809 - val_mae: 0.4390
Epoch 18/20
363/363 [==============================] - 0s 596us/step - loss: 0.4040 - mae: 0.4545 - val_loss: 0.3793 - val_mae: 0.4368
Epoch 19/20
363/363 [==============================] - 0s 588us/step - loss: 0.4004 - mae: 0.4521 - val_loss: 0.3850 - val_mae: 0.4369
Epoch 20/20
363/363 [==============================] - 0s 662us/step - loss: 0.3980 - mae: 0.4508 - val_loss: 0.3809 - val_mae: 0.4368
```

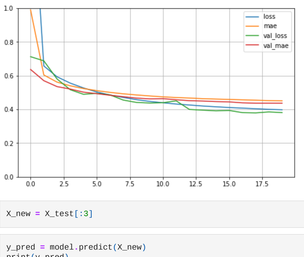
In [15]:
```
mae_test = model.evaluate(X_test, y_test)
```

```
162/162 [==============================] - 0s 378us/step - loss: 0.3942 - mae: 0.4502
```

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

Out[18]:
```
{'loss': [1.8866397142410278,
  0.6577126979827881,
  0.5934418538570404,
  0.5557191371917725,
  0.5271904468536377,
  0.5032975673675537,
  0.4853555262088776,
  0.4709180891513824,
  0.45779937505722046,
  0.4474469721317291,
  0.4393136501312256,
  0.43176087737083435,
  0.4260516464710236,
  0.42017653584480286,
  0.41549986600875854,
  0.4111650288105011,
  0.4077068865299225,
  0.40395283699035645,
  0.4004494547843933,
  0.39796024560928345],
 'mae': [0.9900256991386414,
  0.6041509590986609,
  0.5618006587028503,
  0.5398454070091248,
  0.5237293038954651,
  0.5112563371658325,
  0.5010154247283936,
  0.4924480091506958,
  0.4857262969017029,
  0.4797375202178955,
  0.4744163453578949,
  0.4703480303287506,
  0.46740883588790894,
  0.4636020950842361,
  0.4612664428232193,
  0.4591343402862549,
  0.4568700492382049,
  0.4545365273952484,
  0.4521065056324005,
  0.4508314728736877],
 'val_loss': [0.7126054763793945,
  0.6880008448524475,
  0.5803286433321991,
  0.5166085362434387,
  0.4895076453685765,
  0.4950792193412781,
  0.4861253798007965,
  0.4553801715373993,
  0.4413371384143829,
  0.4378637969493866,
  0.4396438896656036,
  0.4506688871407405,
  0.3997234702110290 5,
  0.3955893516540527 3,
  0.3915725052356672,
  0.3936831057071686,
  0.3809485137462616,
  0.3793475329875946,
  0.3850175738334656,
  0.3809465467929841],
 'val_mae': [0.6368111968040466,
  0.5703656673431396,
  0.5351505279541016,
  0.5206614136695862,
  0.5022227168083191,
  0.4933752417564392,
  0.4838490018821 2 7 9 5 7,
  0.4752774536096497,
  0.4670593440532684 3,
  0.4623450875583648 7,
  0.4637791812419891 4,
  0.4572649896144867,
  0.4516655371656641785,
  0.4496610760688782,
  0.4463699758052826,
  0.4444962739944458,
  0.4389741718769073 5,
  0.4368110895156860 4,
  0.4369049666831207 3,
  0.4367616176605224 6]}
```

In [19]:
```
import pandas as pd
pd.DataFrame(model_history.history).plot(figsize = (8,5))
plt.grid(True)
plt.gca().set_ylim(0,1)
plt.show()
```



In [20]:
```
X_new = X_test[:3]
```

In [23]:
```
y_pred = model.predict(X_new)
print(y_pred)
print(y_test[:3])
```

```
[[0.5328768]
 [1.8915398]
 [3.404087 ]]
[0.477   0.458   5.00001]
```

In [ ]:
```
```