In [1]:
```
import pandas as pd
import tensorflow as tf
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers
```

In [2]:
```
url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data'

column_names = ['MPG', 'Cylinders', 'Displacement', 'Horsepower', 'Weight',
                'Acceleration', 'Model Year', 'Origin']

raw_dataset = pd.read_csv(url, names=column_names,
                          na_values='?', comment='\t',
                          sep=' ', skipinitialspace=True)
```

In [3]:
```
dataset = raw_dataset.copy()
dataset.tail()
```

Out[3]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
393	27.0	4	140.0	86.0	2790.0	15.6	82	1
394	44.0	4	97.0	52.0	2130.0	24.6	82	2
395	32.0	4	135.0	84.0	2295.0	11.6	82	1
396	28.0	4	120.0	79.0	2625.0	18.6	82	1
397	31.0	4	119.0	82.0	2720.0	19.4	82	1

In [4]:
```
dataset.isna().sum()
```

Out[4]:
```
MPG             0
Cylinders       0
Displacement    0
Horsepower      6
Weight          0
Acceleration    0
Model Year      0
Origin          0
dtype: int64
```

In [5]:
```
dataset = dataset.dropna()
```

In [6]:
```
dataset.isna().sum()
```

Out[6]:
```
MPG             0
Cylinders       0
Displacement    0
Horsepower      0
Weight          0
Acceleration    0
Model Year      0
Origin          0
dtype: int64
```

In [7]:
```
dataset.head(15)
```

Out[7]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
0	18.0	8	307.0	130.0	3504.0	12.0	70	1
1	15.0	8	350.0	165.0	3693.0	11.5	70	1
2	18.0	8	318.0	150.0	3436.0	11.0	70	1
3	16.0	8	304.0	150.0	3433.0	12.0	70	1
4	17.0	8	302.0	140.0	3449.0	10.5	70	1
5	15.0	8	429.0	198.0	4341.0	10.0	70	1
6	14.0	8	454.0	220.0	4354.0	9.0	70	1
7	14.0	8	440.0	215.0	4312.0	8.5	70	1
8	14.0	8	455.0	225.0	4425.0	10.0	70	1
9	15.0	8	390.0	190.0	3850.0	8.5	70	1
10	15.0	8	383.0	170.0	3563.0	10.0	70	1
11	14.0	8	340.0	160.0	3609.0	8.0	70	1
12	15.0	8	400.0	150.0	3761.0	9.5	70	1
13	14.0	8	455.0	225.0	3086.0	10.0	70	1
14	24.0	4	113.0	95.0	2372.0	15.0	70	3

In [8]:
```
dataset['Origin'].unique()
```

Out[8]:
```
array([1, 3, 2], dtype=int64)
```

In [9]:
```
dataset['Origin'] = dataset['Origin'].map({1: 'USA', 2: 'Europe', 3: 'Japan'})
```

In [10]:
```
dataset.head()
```

Out[10]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
0	18.0	8	307.0	130.0	3504.0	12.0	70	USA
1	15.0	8	350.0	165.0	3693.0	11.5	70	USA
2	18.0	8	318.0	150.0	3436.0	11.0	70	USA
3	16.0	8	304.0	150.0	3433.0	12.0	70	USA
4	17.0	8	302.0	140.0	3449.0	10.5	70	USA

In [11]:
```
dataset = pd.get_dummies(dataset, columns=['Origin'])
```

In [12]:
```
dataset.head(10)
```

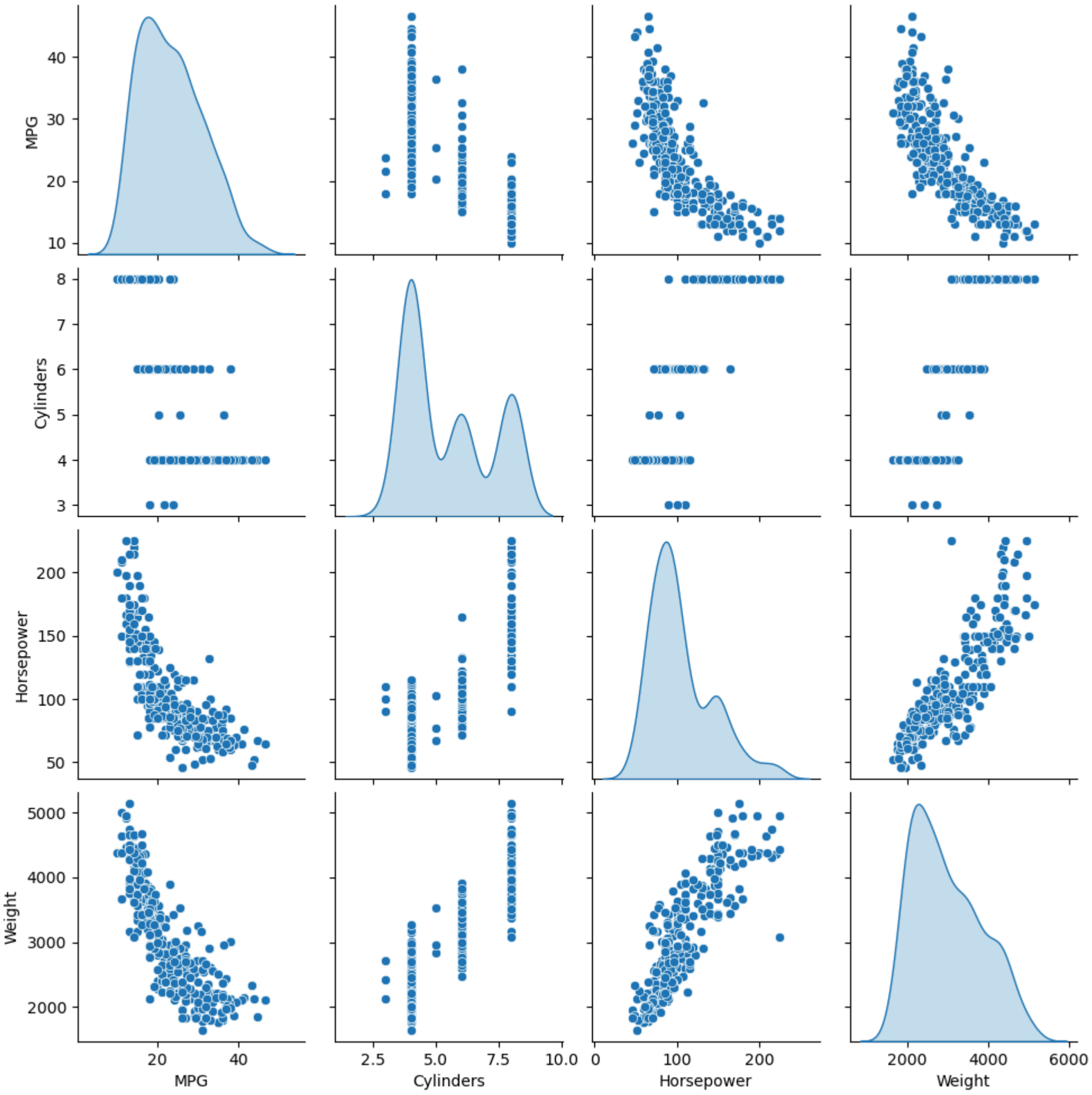
Out[12]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin_Europe	Origin_Japan	Origin_USA
0	18.0	8	307.0	130.0	3504.0	12.0	70	0	0	1
1	15.0	8	350.0	165.0	3693.0	11.5	70	0	0	1
2	18.0	8	318.0	150.0	3436.0	11.0	70	0	0	1
3	16.0	8	304.0	150.0	3433.0	12.0	70	0	0	1
4	17.0	8	302.0	140.0	3449.0	10.5	70	0	0	1
5	15.0	8	429.0	198.0	4341.0	10.0	70	0	0	1
6	14.0	8	454.0	220.0	4354.0	9.0	70	0	0	1
7	14.0	8	440.0	215.0	4312.0	8.5	70	0	0	1
8	14.0	8	455.0	225.0	4425.0	10.0	70	0	0	1
9	15.0	8	390.0	190.0	3850.0	8.5	70	0	0	1

In [13]:
```
train_dataset = dataset.sample(frac=0.8, random_state=0)
test_dataset = dataset.drop(train_dataset.index)
```

In [14]:
```
sns.pairplot(train_dataset[['MPG', 'Cylinders', 'Horsepower', 'Weight']], diag_kind='kde')
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x212cbed8750>
```



In [15]:
```
train_dataset.describe().transpose()
```

Out[15]:

	count	mean	std	min	25%	50%	75%	max
MPG	314.0	23.310510	7.728652	10.0	17.00	22.0	28.95	46.6
Cylinders	314.0	5.477707	1.699788	3.0	4.00	4.0	8.00	8.0
Displacement	314.0	195.318471	104.331589	68.0	105.50	151.0	265.75	455.0
Horsepower	314.0	104.869427	38.096214	46.0	76.25	94.5	128.00	225.0
Weight	314.0	2990.251592	843.898596	1649.0	2256.50	2822.5	3608.00	5140.0
Acceleration	314.0	15.559236	2.789230	8.0	13.80	15.5	17.20	24.8
Model Year	314.0	75.898089	3.675642	70.0	73.00	76.0	79.00	82.0
Origin_Europe	314.0	0.178344	0.383413	0.0	0.00	0.0	0.00	1.0
Origin_Japan	314.0	0.197452	0.398712	0.0	0.00	0.0	0.00	1.0
Origin_USA	314.0	0.624204	0.485101	0.0	0.00	1.0	1.00	1.0

In [16]:
```
train_features = train_dataset.copy()
test_features = test_dataset.copy()

train_labels = train_features.pop('MPG')
test_labels = test_features.pop('MPG')
```

In [17]:
```
train_dataset.describe().transpose()[['mean', 'std']]
```

Out[17]:

	mean	std
MPG	23.310510	7.728652
Cylinders	5.477707	1.699788
Displacement	195.318471	104.331589
Horsepower	104.869427	38.096214
Weight	2990.251592	843.898596
Acceleration	15.559236	2.789230
Model Year	75.898089	3.675642
Origin_Europe	0.178344	0.383413
Origin_Japan	0.197452	0.398712
Origin_USA	0.624204	0.485101

In [18]:
```
normalizer = tf.keras.layers.Normalization(axis=-1)
normalizer.adapt(np.array(train_features))
print(normalizer.mean.numpy())
```

```
[[5.47770691e+00 1.95318497e+02 1.04869446e+02 2.99025171e+03
  1.55592356e+01 7.58980942e+01 1.78343967e-01 1.97452217e-01
  6.24203861e-01]]
```

In [19]:
```
first = np.array(train_features[:1])


with np.printoptions(precision=2, suppress=True):
  print('First example:', first)
  print()
  print('Normalized:', normalizer(first).numpy())
```

```
First example: [[   4.    90.    75.  2125.    14.5   74.     0.     0.     1. ]]

Normalized: [[-0.87 -1.01 -0.79 -1.03 -0.38 -0.52 -0.47 -0.5   0.78]]
```

SLR

In [20]:
```
horsepower = np.array(train_features['Horsepower'])

horsepower_normalizer = layers.Normalization(input_shape=[1,], axis=None)
horsepower_normalizer.adapt(horsepower)
```

In [21]:
```
horsepower_model = tf.keras.Sequential([
    horsepower_normalizer,
    layers.Dense(units=1)
])

horsepower_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization_1 (Normaliza  (None, 1)                 3
 tion)

 dense (Dense)               (None, 1)                 2

=================================================================
Total params: 5 (24.00 Byte)
Trainable params: 2 (8.00 Byte)
Non-trainable params: 3 (16.00 Byte)
_________________________________________________________________
```

In [22]:
```
horsepower_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.1),
    loss='mean_absolute_error')
```

In [23]:
```
%%time
history = horsepower_model.fit(
    train_features['Horsepower'],
    train_labels,
    epochs=100,
    verbose=0,
    validation_split = 0.2)
```

```
CPU times: total: 6 s
Wall time: 4.89 s
```

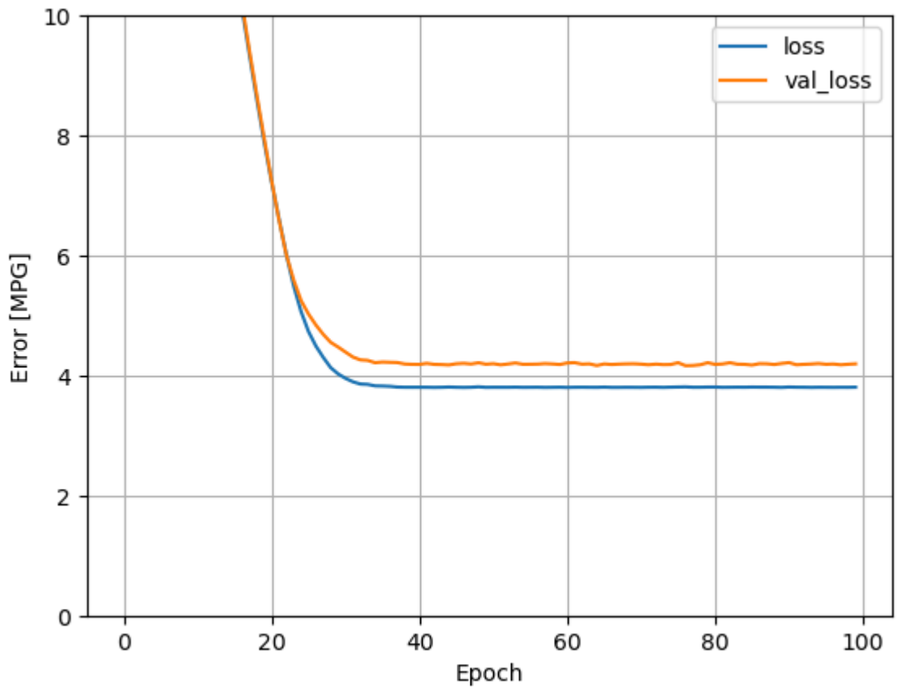
In [24]:
```
hist = pd.DataFrame(history.history)
hist['epoch'] = history.epoch
hist.tail()
```

Out[24]:

	loss	val_loss	epoch
95	3.803825	4.187245	95
96	3.802713	4.191550	96
97	3.803993	4.179204	97
98	3.803464	4.187901	98
99	3.805697	4.194726	99

In [25]:
```
def plot_loss(history):
  plt.plot(history.history['loss'], label='loss')
  plt.plot(history.history['val_loss'], label='val_loss')
  plt.ylim([0, 10])
  plt.xlabel('Epoch')
  plt.ylabel('Error [MPG]')
  plt.legend()
  plt.grid(True)
```

In [26]:
```
plot_loss(history)
```



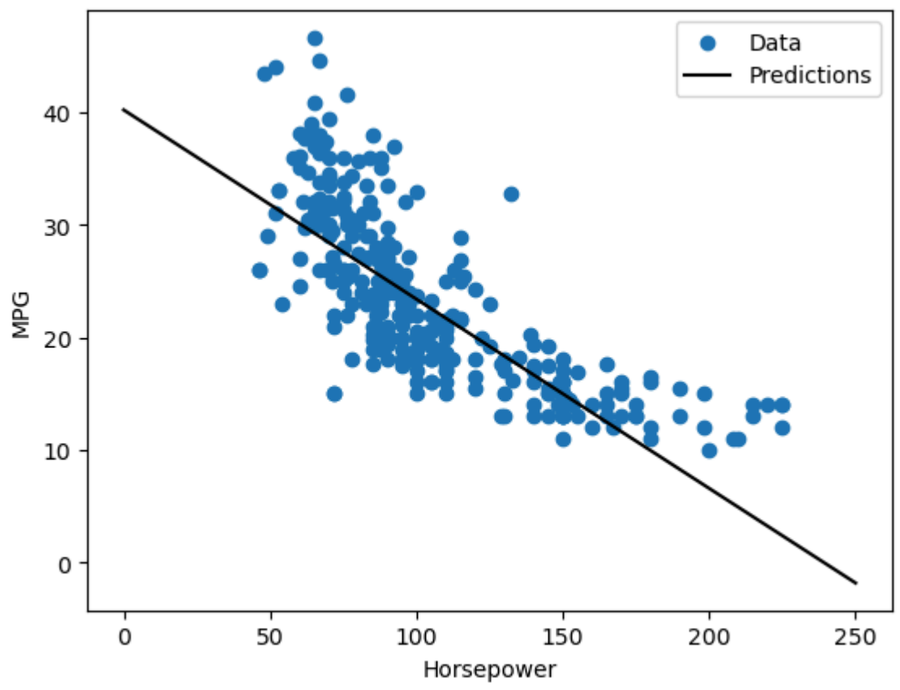
In [27]:
```
test_results = {}

test_results['horsepower_model'] = horsepower_model.evaluate(
    test_features['Horsepower'],
    test_labels, verbose=0)
```

In [28]:
```
x = tf.linspace(0.0, 250, 251)
y = horsepower_model.predict(x)
```

```
8/8 [==============================] - 0s 2ms/step
```

In [29]:
```
def plot_horsepower(x, y):
  plt.scatter(train_features['Horsepower'], train_labels, label='Data')
  plt.plot(x, y, color='k', label='Predictions')
  plt.xlabel('Horsepower')
  plt.ylabel('MPG')
  plt.legend()
```

In [30]:
```
plot_horsepower(x, y)
```



MLR

In [31]:
```
linear_model = tf.keras.Sequential([
    normalizer,
    layers.Dense(units=1)
])
```

In [32]:
```
linear_model.layers[1].kernel
```

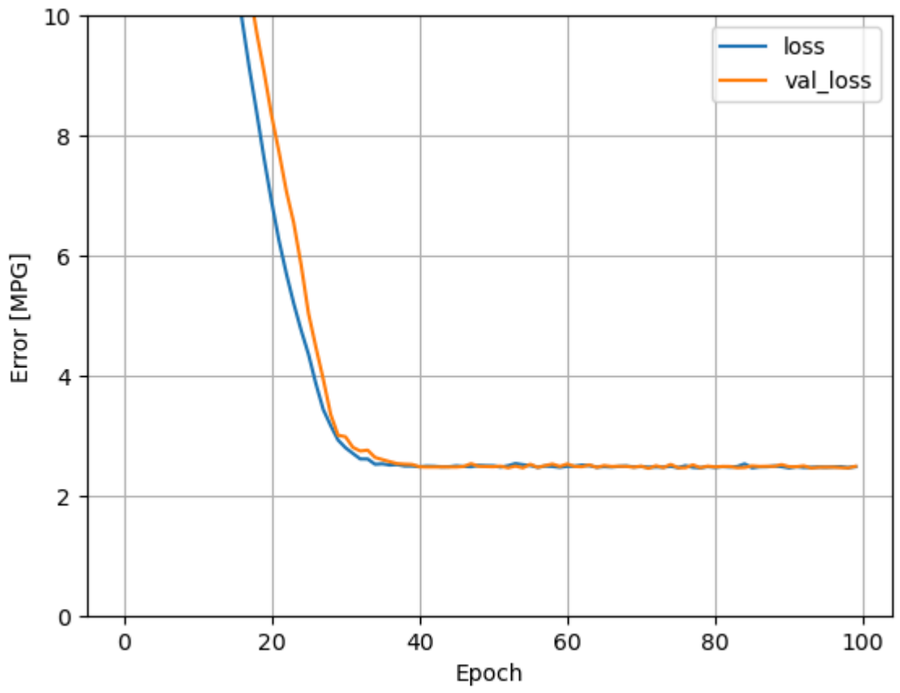
Out[32]:
```
<tf.Variable 'dense_1/kernel:0' shape=(9, 1) dtype=float32, numpy=
array([[-0.5449934 ],
       [ 0.08895987],
       [-0.11711317],
       [ 0.66201925],
       [ 0.3909036 ],
       [ 0.71676576],
       [ 0.57576394],
       [ 0.26156402],
       [ 0.7015773 ]], dtype=float32)>
```

In [33]:
```
linear_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.1),
    loss='mean_absolute_error')
```

In [34]:
```
%%time
history = linear_model.fit(
    train_features,
    train_labels,
    epochs=100,
    verbose=0,
    validation_split = 0.2)
```

```
CPU times: total: 5.39 s
Wall time: 4.51 s
```

In [35]:
```
plot_loss(history)
```



In [36]:
```
test_results['linear_model'] = linear_model.evaluate(
    test_features, test_labels, verbose=0)
```

In [ ]:
```
# MIT License
#
# Copyright (c) 2017 François Chollet
#
# Permission is hereby granted, free of charge, to any person obtaining a
# copy of this software and associated documentation files (the "Software"),
# to deal in the Software without restriction, including without limitation
# the rights to use, copy, modify, merge, publish, distribute, sublicense,
# and/or sell copies of the Software, and to permit persons to whom the
# Software is furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
# THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
# FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
# DEALINGS IN THE SOFTWARE.
```