```
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from tqdm.notebook import tqdm
import warnings
warnings.filterwarnings("ignore")
```

```
boston = tf.keras.datasets.boston_housing
```

```
dir(boston)
```

```
['__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
 'load_data']
```

```
boston_data = boston.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/boston_housing.npz
57026/57026 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.boston_housing.load_data(path='boston_housing.npz', test_split=0.2, seed=42)
```

```
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

```
((404, 13), (404,), (102, 13), (102,))
```

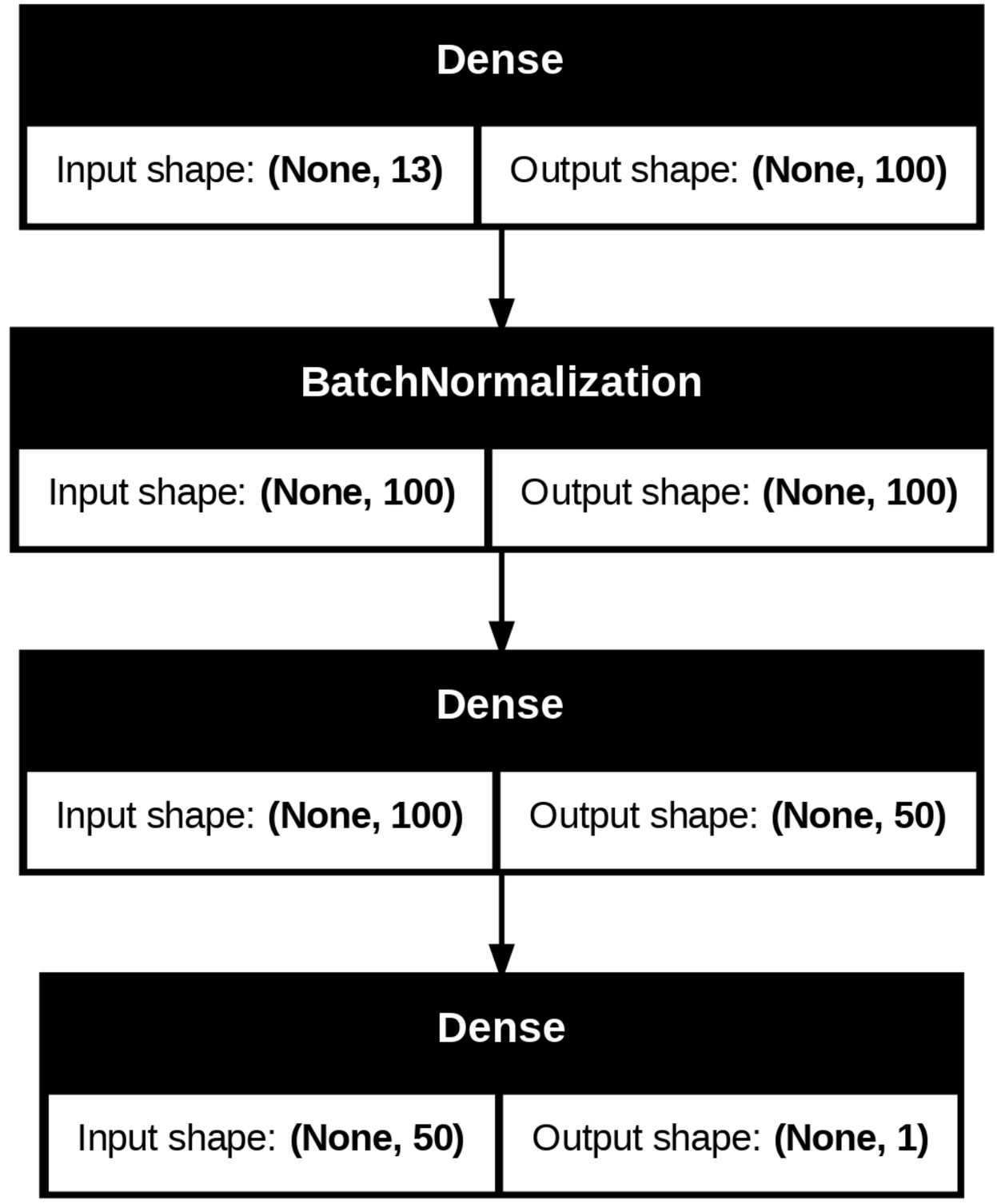
```
scaler = StandardScaler()
```

```
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

y_train_scaled = scaler.fit_transform(y_train.reshape(-1, 1))
y_test_scaled = scaler.transform(y_test.reshape(-1, 1))
```

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(13,), name='input-layer'),
    tf.keras.layers.Dense(100, activation='relu', name='hidden-layer-1'), # Added activation function
    tf.keras.layers.BatchNormalization(name='batch-norm-layer'), # More descriptive name
    tf.keras.layers.Dense(50, activation='relu', name='hidden-layer-2'), # Added activation function
    tf.keras.layers.Dense(1, name='output-layer')
])
```

```
tf.keras.utils.plot_model(model, show_shapes=True)
```




```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
hidden-layer-1 (Dense)	(None, 100)	1,400
batch-norm-layer (BatchNormalization)	(None, 100)	400
hidden-layer-2 (Dense)	(None, 50)	5,050
output-layer (Dense)	(None, 1)	51

Total params: 6,901 (26.96 KB)
Trainable params: 6,701 (26.18 KB)
Non-trainable params: 200 (800.00 B)

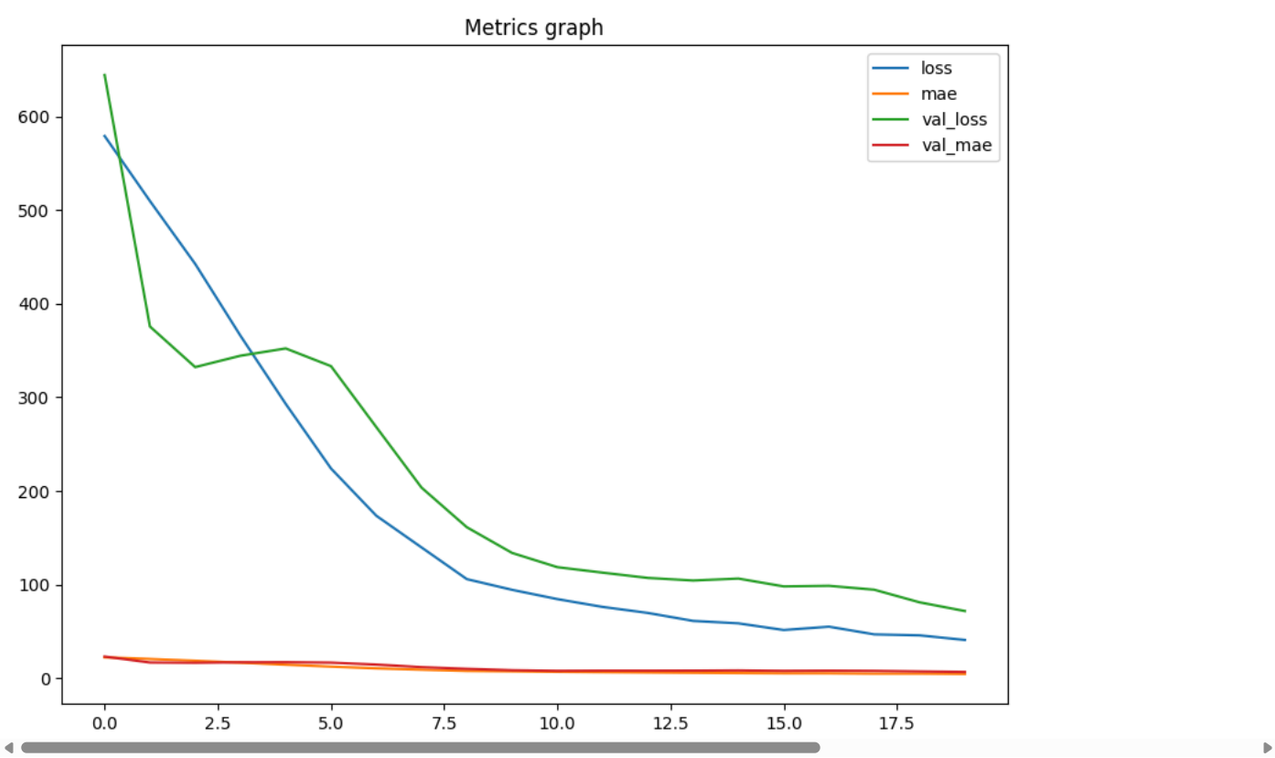
```
model.compile(
    optimizer='adam',
    loss='mse',
    metrics=['mae']
)
```

```
history = model.fit(x_train, y_train, batch_size=32, epochs=20, validation_data=(x_test, y_test))
```

```
Epoch 1/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - loss: 613.3219 - mae: 22.7953 - val_loss: 644.1995 - val_mae: 22.9610
Epoch 2/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 492.4562 - mae: 19.9995 - val_loss: 375.5863 - val_mae: 16.5817
Epoch 3/20
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 450.3040 - mae: 18.8613 - val_loss: 332.1854 - val_mae: 16.3065
Epoch 4/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 380.9872 - mae: 16.7515 - val_loss: 344.3435 - val_mae: 16.8123
Epoch 5/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 320.7765 - mae: 15.2770 - val_loss: 352.1318 - val_mae: 16.9017
Epoch 6/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 261.0700 - mae: 13.2325 - val_loss: 333.1944 - val_mae: 16.4613
Epoch 7/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 213.6038 - mae: 11.7599 - val_loss: 268.1405 - val_mae: 14.3820
Epoch 8/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 165.4708 - mae: 9.7457 - val_loss: 203.4223 - val_mae: 11.6654
Epoch 9/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 108.4503 - mae: 7.7173 - val_loss: 161.1279 - val_mae: 9.8096
Epoch 10/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 107.4608 - mae: 7.4724 - val_loss: 133.6502 - val_mae: 8.3531
Epoch 11/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 90.6716 - mae: 6.9324 - val_loss: 118.4454 - val_mae: 7.6937
Epoch 12/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 75.6380 - mae: 6.4644 - val_loss: 112.6140 - val_mae: 7.8732
Epoch 13/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 67.3138 - mae: 5.9995 - val_loss: 106.9482 - val_mae: 7.8693
Epoch 14/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 65.0389 - mae: 5.7154 - val_loss: 104.2100 - val_mae: 7.9409
Epoch 15/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 65.7142 - mae: 5.2730 - val_loss: 106.2745 - val_mae: 8.1390
Epoch 16/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 46.6264 - mae: 4.8246 - val_loss: 97.8811 - val_mae: 7.6954
Epoch 17/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 43.5111 - mae: 4.6192 - val_loss: 98.5237 - val_mae: 7.8843
Epoch 18/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 51.5562 - mae: 4.8405 - val_loss: 94.3702 - val_mae: 7.6696
Epoch 19/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 48.7143 - mae: 4.7820 - val_loss: 80.9103 - val_mae: 7.0743
Epoch 20/20
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 42.9282 - mae: 4.4450 - val_loss: 71.5890 - val_mae: 6.5232
```

```
pd.DataFrame(history.history).plot(figsize=(10,7))
plt.title("Metrics graph")
plt.show()
```




```
y_pred = model.predict(x_test)
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
```

```
sns.regplot(x=y_test, y=y_pred)
plt.title("Regression Line for Predicted values")
plt.show()
```