```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping


data = fetch_california_housing()
X = data.data
y = data.target

population = data.data[:, 4]


scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


X_train, X_val, y_train, y_val, population_train, population_val = train_test_split(
    X_scaled, y, population, test_size=0.2, random_state=42)

def build_single_task_model(input_size):
    input_layer = Input(shape=(input_size,))

    x = Dense(64, activation='relu')(input_layer)
    x = Dropout(0.5)(x)

    x = Dense(32, activation='relu')(x)
    x = Dropout(0.5)(x)


    output_layer = Dense(1, name='output')(x)


    model = Model(inputs=input_layer, outputs=output_layer)


    model.compile(optimizer=Adam(clipvalue=5.0),
                  loss='mse',
                  metrics=['mae'])

    return model


input_size = X_train.shape[1]
model = build_single_task_model(input_size)


model.summary()


early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)


history = model.fit(X_train,
                    y_train,
                    validation_data=(X_val, y_val),
                    epochs=10,
                    batch_size=64,
                    callbacks=[early_stopping])


plt.figure(figsize=(10, 6))
plt.plot(history.history['loss'], label='Train Loss', color='blue')
plt.plot(history.history['val_loss'], label='Validation Loss', color='red')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```

```
plt.figure(figsize=(10, 6))
plt.plot(history.history['mae'], label='Training MAE', color='blue')
plt.plot(history.history['val_mae'], label='Validation MAE', color='red')
plt.xlabel('Epochs')
plt.ylabel('Mean Absolute Error (MAE)')
plt.title('Training and Validation MAE')
plt.legend()
plt.show()
```

Model: "functional_3"

Layer (type)	Output Shape	Param #
input_layer_3 (InputLayer)	(None, 8)	0
dense_6 (Dense)	(None, 64)	576
dropout_6 (Dropout)	(None, 64)	0
dense_7 (Dense)	(None, 32)	2,080
dropout_7 (Dropout)	(None, 32)	0
output (Dense)	(None, 1)	33

Total params: 2,689 (10.50 KB)
Trainable params: 2,689 (10.50 KB)
Non-trainable params: 0 (0.00 B)

```
Epoch 1/10
258/258 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 3.2420 - mae: 1.2883 - val_loss: 0.7522 - val_mae: 0.6102
Epoch 2/10
258/258 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 1.2470 - mae: 0.8004 - val_loss: 0.6098 - val_mae: 0.5307
Epoch 3/10
258/258 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 1.1299 - mae: 0.7087 - val_loss: 0.4976 - val_mae: 0.4820
Epoch 4/10
258/258 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.7870 - mae: 0.6362 - val_loss: 0.4721 - val_mae: 0.4692
Epoch 5/10
258/258 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.7352 - mae: 0.6065 - val_loss: 0.4460 - val_mae: 0.4603
```

Double-click (or enter) to edit

Start coding or generate with AI.

Start coding or generate with AI.

```
Epoch 10/10
258/258 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.5664 - mae: 0.5377 - val_loss: 0.3993 - val_mae: 0.4482
```