```
import numpy as np
import pandas as pd
from tensorflow import keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
import matplotlib.pyplot as plt
```

```
## Load the data set
from sklearn.datasets import fetch_california_housing
data = fetch_california_housing(as_frame=True)
df = data.frame
df.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

Next steps: Generate code with df View recommended plots New interactive sheet

```
## Data preprocessing
X = df.drop(columns=['MedHouseVal'])
y = df['MedHouseVal']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
## Normalize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
## Build the Model
model = keras.Sequential([
    keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(1)
])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argumen
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

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

```
## Train the model
history = model.fit(X_train, y_train, validation_split=0.2, epochs=50, batch_size=32)
```

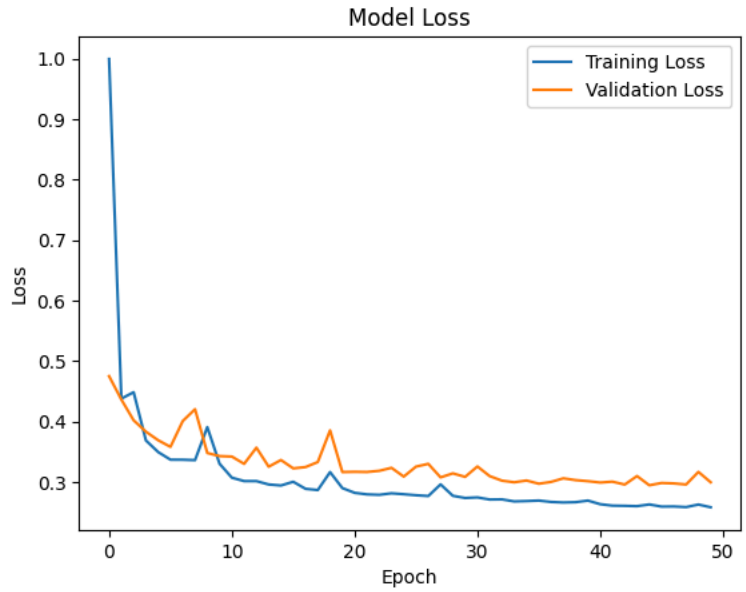
```
Epoch 21/50
```

```
Epoch 31/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2663 - mae: 0.3540 - val_loss: 0.3253 - val_mae: 0.3997
Epoch 32/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.2630 - mae: 0.3534 - val_loss: 0.3094 - val_mae: 0.3826
Epoch 33/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.2746 - mae: 0.3622 - val_loss: 0.3018 - val_mae: 0.3744
Epoch 34/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 0.2623 - mae: 0.3533 - val_loss: 0.2992 - val_mae: 0.3703
Epoch 35/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2659 - mae: 0.3537 - val_loss: 0.3021 - val_mae: 0.3706
Epoch 36/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.2702 - mae: 0.3577 - val_loss: 0.2968 - val_mae: 0.3730
Epoch 37/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2605 - mae: 0.3533 - val_loss: 0.2998 - val_mae: 0.3767
Epoch 38/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2680 - mae: 0.3542 - val_loss: 0.3057 - val_mae: 0.3779
Epoch 39/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2620 - mae: 0.3521 - val_loss: 0.3027 - val_mae: 0.3702
Epoch 40/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2564 - mae: 0.3475 - val_loss: 0.3010 - val_mae: 0.3722
Epoch 41/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.2695 - mae: 0.3554 - val_loss: 0.2988 - val_mae: 0.3825
Epoch 42/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.2532 - mae: 0.3462 - val_loss: 0.3002 - val_mae: 0.3669
Epoch 43/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 0.2539 - mae: 0.3464 - val_loss: 0.2954 - val_mae: 0.3665
Epoch 44/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2649 - mae: 0.3507 - val_loss: 0.3095 - val_mae: 0.3742
Epoch 45/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2591 - mae: 0.3524 - val_loss: 0.2942 - val_mae: 0.3678
Epoch 46/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2639 - mae: 0.3529 - val_loss: 0.2980 - val_mae: 0.3830
Epoch 47/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2580 - mae: 0.3512 - val_loss: 0.2973 - val_mae: 0.3807
Epoch 48/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2534 - mae: 0.3446 - val_loss: 0.2955 - val_mae: 0.3755
Epoch 49/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.2574 - mae: 0.3505 - val_loss: 0.3163 - val_mae: 0.4060
Epoch 50/50
413/413 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.2608 - mae: 0.3479 - val_loss: 0.2992 - val_mae: 0.3663
```

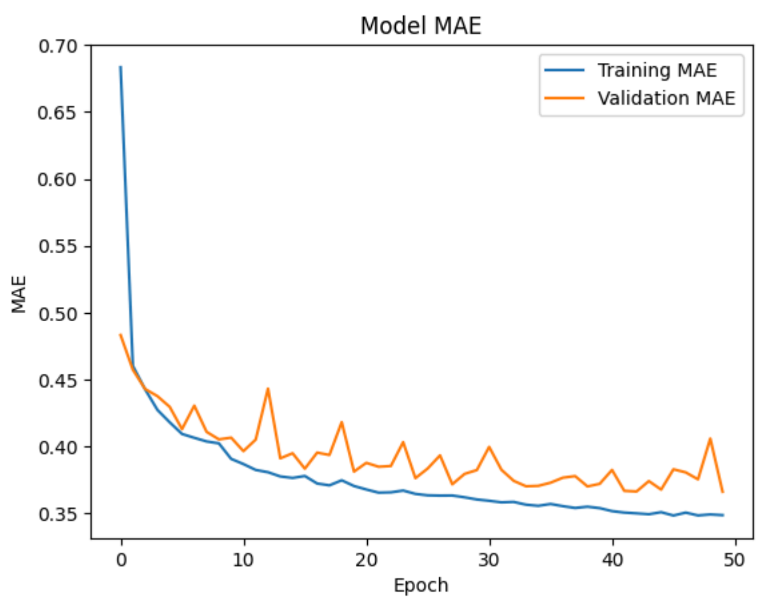
```
## Evaluate the model performance using Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R² Score.

import numpy as np
# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2) Score: {r2}")
```

```
129/129 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Mean Absolute Error (MAE): 0.36160070117514953
Root Mean Squared Error (RMSE): 0.5394142701374033
R-squared (R2) Score: 0.7779565313888003
```

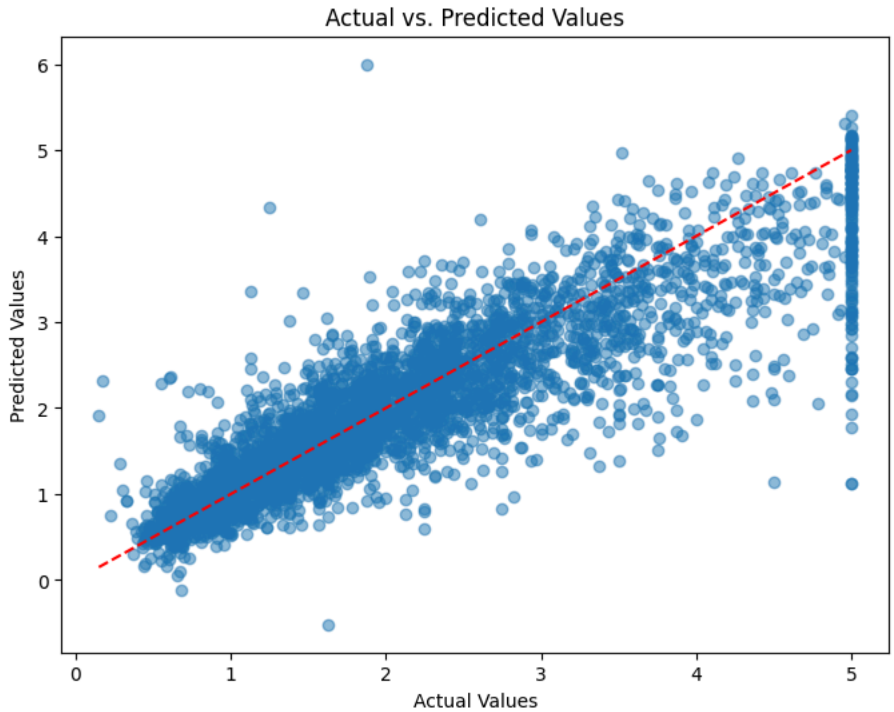
```
## plot training history
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(loc='upper right')
plt.show()
```




```
# prompt: give me plot the training and validation mae

import matplotlib.pyplot as plt
plt.plot(history.history['mae'], label='Training MAE')
plt.plot(history.history['val_mae'], label='Validation MAE')
plt.title('Model MAE')
plt.ylabel('MAE')
plt.xlabel('Epoch')
plt.legend(loc='upper right')
plt.show()
```




```
## Visualize the predictions vs. actual values using a scatter plot.

plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs. Predicted Values")
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--') # Add a diagonal line
plt.show()
```


Actual vs. Predicted Values
Predicted Values
Actual Values
6
5
4
3
2
1
0
0
1
2
3
4
5