The California Housing Dataset in Python - Multiple Linear Regression

1. Importing necessary libraries

First, we need to import the libraries required for data manipulation, modeling, and visualization.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

2. Loading the California housing dataset

Next, we load the California housing dataset and create a Pandas DataFrame for easier manipulation.

```
california = fetch_california_housing()
data = pd.DataFrame(california.data, columns = california.feature_names)
data['MEDIAN_PRICE'] = california.target    # target variable

# Displaying the first few rows of the dataset
data.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MEDIAN_PRICE
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 data | View recommended plots | New interactive sheet

3. Preparing the data for modeling

We now select features for the model and split the dataset into training and testing sets.

```
X = data.drop('MEDIAN_PRICE', axis=1)    # selecting features, using all but the last column
y = data['MEDIAN_PRICE']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Displaying the shapes of the training and testing sets
X_train.shape, X_test.shape
```

```
((16512, 8), (4128, 8))
```

4. Building the multiple linear regression model

Here, we create and fit the multiple linear regression model.

```
model = LinearRegression()
model.fit(X_train, y_train)

# Displaying the regression coefficients
coefficients = model.coef_
coefficients
```

```
array([ 4.33333407e-01,  9.29324337e-03, -9.86433739e-02,  5.93215487e-01,
       -7.56192502e-06, -4.74516383e-03, -4.21449336e-01, -4.34166041e-01])
```

5. Making predictions using the model

With the model built, we can make predictions on the test dataset, and compare the predicted values with the actual ones.

```
y_pred = model.predict(X_test)

compare = pd.DataFrame({'Actual value': y_test, 'Predicted value': y_pred})
compare.head()
```

	Actual value	Predicted value
14740	1.369	2.281107
10101	2.413	2.790091
20566	2.007	1.903328
2670	0.725	1.017603
15709	4.600	2.948524

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

6. Evaluating the model's performance

Next, we evaluate the performance of the model using metrics like Mean Squared Error (MSE) and R-squared score.

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

mse, r2
```

```
(0.5289841670367209, 0.5943232652466202)
```

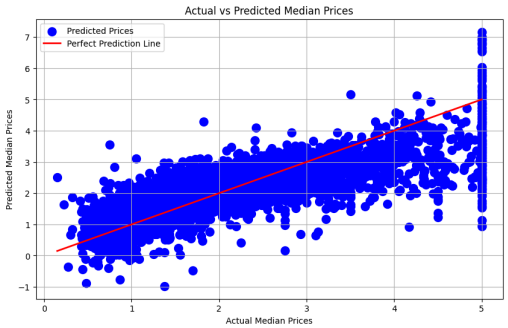
7. Visualising the results

Finally, we visualise the actual vs predicted prices and include the regression equation.

```
# Preparing the regression equation
eqn_parts = [f"{coef:.2f}*{name}" for coef, name in zip(coefficients, X.columns)]
eqn = "MEDIAN_PRICE = " + " + ".join(eqn_parts)

# Plotting the results with the regression equation
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Predicted Prices', s=100)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linewidth=2, label='Perfect Prediction Line')
plt.title('Actual vs Predicted Median Prices')
plt.xlabel('Actual Median Prices')
plt.ylabel('Predicted Median Prices')
plt.legend()
plt.grid()

# Displaying the regression equation
plt.gca().text(1.05, 0.5, eqn, fontsize=12, color='green', transform=plt.gca().transAxes, verticalalignment='center')
plt.show()
```



MEDIAN_PRICE = 0.43*MedInc + 0.01*HouseAge + -0.10*AveRooms + 0.59*AveBedrms + -0.00*Population + -0.00*AveOccup + -0.42*Latitude + -0.43*Longitude