```
from sklearn.datasets import fetch_california_housing
import pandas as pd

data = fetch_california_housing(as_frame=True)
df = data.frame
df.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	L
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X = df.drop('MedHouseVal', axis=1)
y = df['MedHouseVal']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor

lr = LinearRegression()
dt = DecisionTreeRegressor(random_state=42)

lr.fit(X_train_scaled, y_train)
dt.fit(X_train_scaled, y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)
```

```
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

def evaluate(model, X, y):
    preds = model.predict(X)
    rmse = np.sqrt(mean_squared_error(y, preds))
    r2 = r2_score(y, preds)
    return rmse, r2

lr_rmse, lr_r2 = evaluate(lr, X_test_scaled, y_test)
dt_rmse, dt_r2 = evaluate(dt, X_test_scaled, y_test)

print("Linear Regression RMSE:", lr_rmse)
print("Linear Regression R2:", lr_r2)

print("Decision Tree RMSE:", dt_rmse)
print("Decision Tree R2:", dt_r2)
```

```
Linear Regression RMSE: 0.7455813830127763
Linear Regression R2: 0.575787706032451
Decision Tree RMSE: 0.7028289572288925
Decision Tree R2: 0.6230424613065773
```

```
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

def evaluate(model, X, y):
    preds = model.predict(X)
    rmse = np.sqrt(mean_squared_error(y, preds))
    r2 = r2_score(y, preds)
    return rmse, r2

lr_rmse, lr_r2 = evaluate(lr, X_test_scaled, y_test)
dt_rmse, dt_r2 = evaluate(dt, X_test_scaled, y_test)

print("Linear Regression RMSE:", lr_rmse)
print("Linear Regression R2:", lr_r2)

print("Decision Tree RMSE:", dt_rmse)
print("Decision Tree R2:", dt_r2)
```

```
Linear Regression RMSE: 0.7455813830127763
Linear Regression R2: 0.575787706032451
Decision Tree RMSE: 0.7028289572288925
Decision Tree R2: 0.6230424613065773
```

MODEL COMPARISON:

Linear Regression shows stable performance with reasonable RMSE and R^2. Decision Tree gives higher accuracy on training data but may overfit.

Based on RMSE and R^2 values, the model with lower RMSE and higher R^2 is preferred. In this experiment, Linear Regression performed better.