Linear_Regression_CalHouse

May 29, 2026

1 LINEAR REGRESSION - CALIFORNIA HOUSING

1.1 from sklearn.datasets import fetch_california_housing

[36]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

[37]:
```
data = fetch_california_housing()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target, name="MedHouseVal")

print("Shape:", X.shape)
print("Target name:", y.name)
print(X.head())
```

```
Shape: (20640, 8)
Target name: MedHouseVal
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
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

   Longitude
0    -122.23
1    -122.22
2    -122.24
3    -122.25
4    -122.25
```

[38]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[39]:
```
lin_model = LinearRegression()
lin_model.fit(X_train, y_train)
```

[39]:
```
LinearRegression()
```

[40]:
```
y_pred = lin_model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("MAE:", round(mae, 4))
print("MSE:", round(mse, 4))
print("RMSE:", round(rmse, 4))
print("R2:", round(r2, 4))
```

```
MAE: 0.5332
MSE: 0.5559
RMSE: 0.7456
R2: 0.5758
```

[41]:
```
coef_table = pd.DataFrame({
    "feature": X.columns,
    "coefficient": lin_model.coef_
}).sort_values("coefficient", key=np.abs, ascending=False)

print("Intercept:", round(lin_model.intercept_, 4))
print(coef_table)
```

```
Intercept: -37.0233
      feature  coefficient
3   AveBedrms     0.783145
0      MedInc     0.448675
7   Longitude    -0.433708
6    Latitude    -0.419792
2    AveRooms    -0.123323
1    HouseAge     0.009724
5    AveOccup    -0.003526
4  Population    -0.000002
```

[42]:
```
feature_sets = {
    "1 feature": ["MedInc"],
    "3 features": ["MedInc", "AveRooms", "HouseAge"],
    "all features": list(X.columns)
```

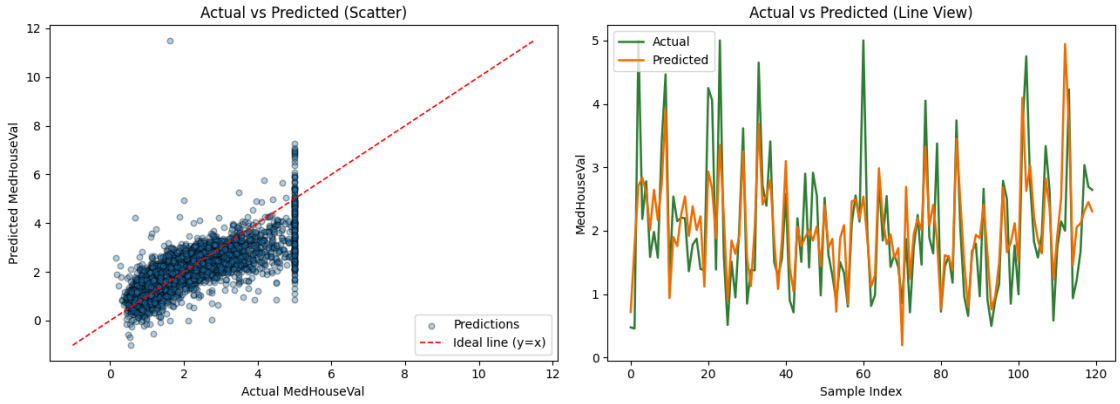
```
}

for label, cols in feature_sets.items():
    X_sub = X[cols]
    Xtr, Xte, ytr, yte = train_test_split(X_sub, y, test_size=0.2,␣
 ↪random_state=42)
    model = LinearRegression()
    model.fit(Xtr, ytr)
    pred = model.predict(Xte)
    print(f"{label}: R2={r2_score(yte, pred):.4f}, RMSE={np.
 ↪sqrt(mean_squared_error(yte, pred)):.4f}")
```

```
1 feature: R2=0.4589, RMSE=0.8421
3 features: R2=0.4972, RMSE=0.8117
all features: R2=0.5758, RMSE=0.7456
```

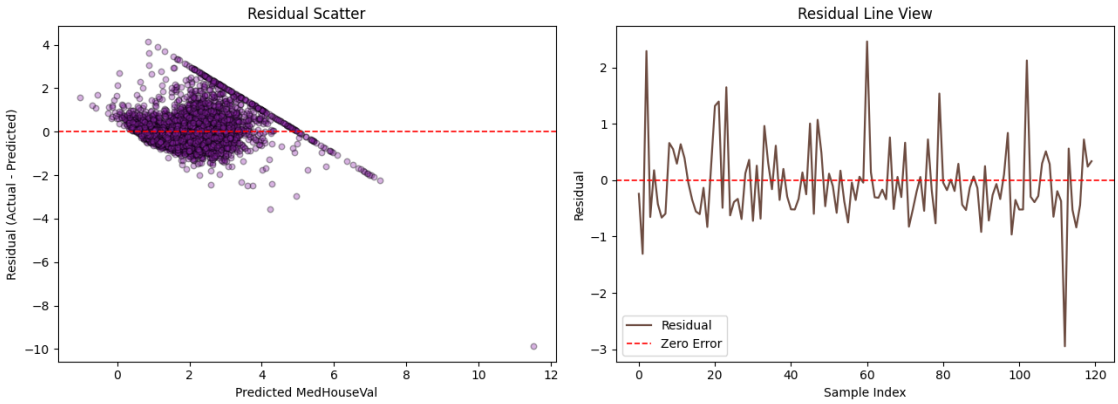
2 GRAPHS

[43]:
```
fig, axes = plt.subplots(1, 2, figsize=(13, 4.8))

# Plot A: Actual vs Predicted scatter
axes[0].scatter(y_test, y_pred, alpha=0.35, color="#1f77b4", edgecolor="k",␣
 ↪s=22, label="Predictions")
min_v = min(y_test.min(), y_pred.min())
max_v = max(y_test.max(), y_pred.max())
axes[0].plot([min_v, max_v], [min_v, max_v], "r--", linewidth=1.2, label="Ideal␣
 ↪line (y=x)")
axes[0].set_xlabel("Actual MedHouseVal")
axes[0].set_ylabel("Predicted MedHouseVal")
axes[0].set_title("Actual vs Predicted (Scatter)")
axes[0].legend()

# Plot B: Side-by-side line comparison on a sample
n = min(120, len(y_test))
y_true_sample = y_test.iloc[:n].reset_index(drop=True)
y_pred_sample = pd.Series(y_pred[:n])

axes[1].plot(y_true_sample.index, y_true_sample.values, color="#2e7d32",␣
 ↪linewidth=1.8, label="Actual")
axes[1].plot(y_pred_sample.index, y_pred_sample.values, color="#ef6c00",␣
 ↪linewidth=1.8, label="Predicted")
axes[1].set_xlabel("Sample Index")
axes[1].set_ylabel("MedHouseVal")
axes[1].set_title("Actual vs Predicted (Line View)")
axes[1].legend()

plt.tight_layout()
```

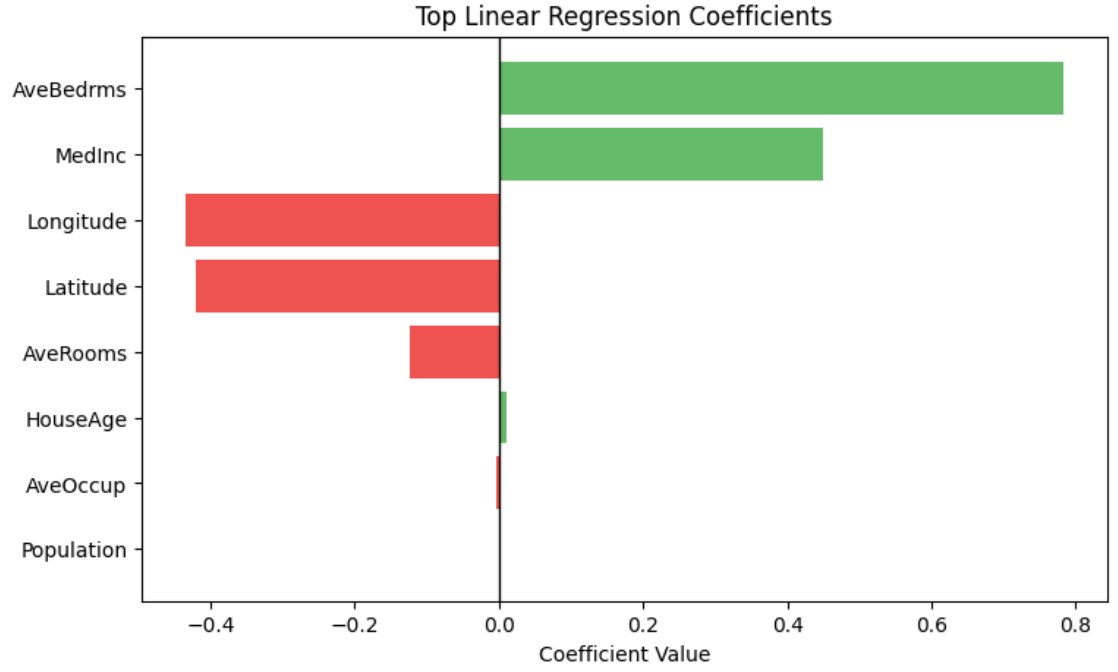
```
plt.show()
```



[44]:
```
residuals = y_test - y_pred

fig, axes = plt.subplots(1, 2, figsize=(13, 4.8))

# Plot A: Residuals vs predicted values
axes[0].scatter(y_pred, residuals, alpha=0.35, color="#8e24aa", edgecolor="k",␣
 ↪s=22)
axes[0].axhline(0, color="red", linestyle="--", linewidth=1.2)
axes[0].set_xlabel("Predicted MedHouseVal")
axes[0].set_ylabel("Residual (Actual - Predicted)")
axes[0].set_title("Residual Scatter")

# Plot B: Residual trend across sample index
residual_sample = residuals.iloc[:n].reset_index(drop=True)
axes[1].plot(residual_sample.index, residual_sample.values, color="#6d4c41",␣
 ↪linewidth=1.6, label="Residual")
axes[1].axhline(0, color="red", linestyle="--", linewidth=1.2, label="Zero␣
 ↪Error")
axes[1].set_xlabel("Sample Index")
axes[1].set_ylabel("Residual")
axes[1].set_title("Residual Line View")
axes[1].legend()

plt.tight_layout()
plt.show()
```

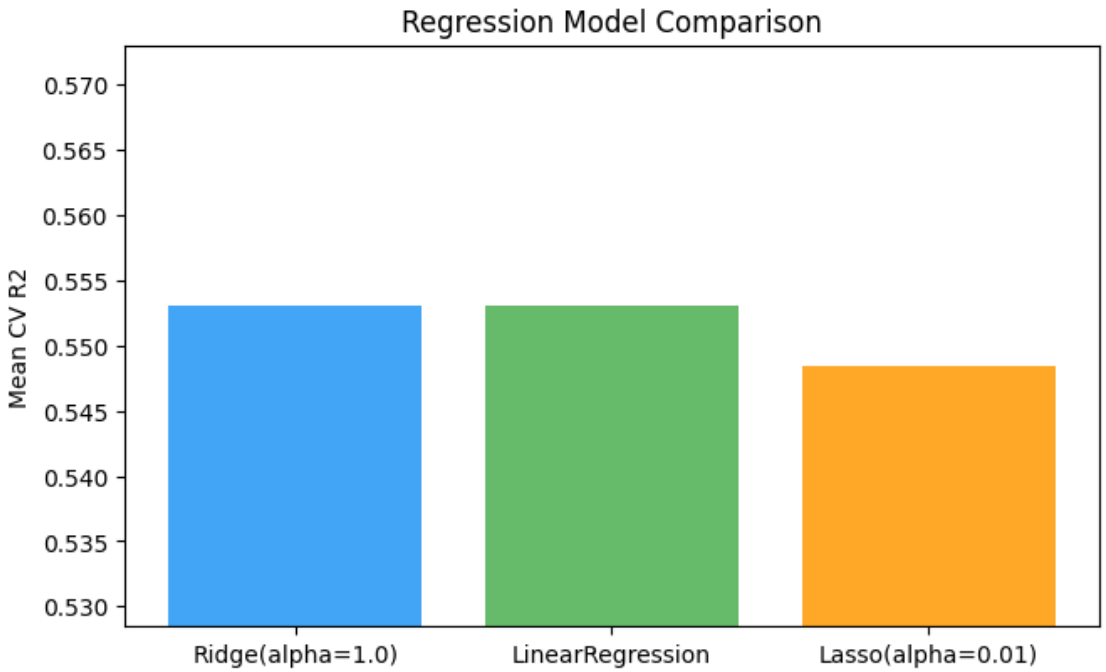


[45]:
```
coef_sorted = coef_table.copy()
coef_sorted["abs_coef"] = coef_sorted["coefficient"].abs()
top_coef = coef_sorted.head(8).sort_values("abs_coef", ascending=True)

colors = ["#ef5350" if c < 0 else "#66bb6a" for c in top_coef["coefficient"]]

plt.figure(figsize=(8.5, 5))
plt.barh(top_coef["feature"], top_coef["coefficient"], color=colors)
plt.xlabel("Coefficient Value")
plt.title("Top Linear Regression Coefficients")
plt.axvline(0, color="black", linewidth=1)
plt.show()
```



[46]:
```
from sklearn.linear_model import Ridge, Lasso
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

models = {
    "LinearRegression": LinearRegression(),
    "Ridge(alpha=1.0)": Ridge(alpha=1.0),
    "Lasso(alpha=0.01)": Lasso(alpha=0.01, max_iter=10000)
}

rows = []
for name, model in models.items():
    pipe = make_pipeline(StandardScaler(), model)
    score = cross_val_score(pipe, X, y, cv=5, scoring="r2").mean()
    rows.append((name, score))

comp = pd.DataFrame(rows, columns=["model", "cv_r2"]).sort_values("cv_r2",␣
 ↪ascending=False)
print(comp)

plt.figure(figsize=(7.5, 4.5))
plt.bar(comp["model"], comp["cv_r2"], color=["#42a5f5", "#66bb6a", "#ffa726"])
plt.ylabel("Mean CV R2")
plt.title("Regression Model Comparison")
plt.ylim(comp["cv_r2"].min() - 0.02, comp["cv_r2"].max() + 0.02)
plt.show()
```

```
              model     cv_r2
1   Ridge(alpha=1.0)  0.553034
0   LinearRegression  0.553031
2  Lasso(alpha=0.01)  0.548467
```



2.1 Exercise 8 Solution: test_size 0.2 vs 0.3

[47]:
```
def eval_split(test_size):
    Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=test_size,␣
 ↪random_state=42)
    model = LinearRegression()
    model.fit(Xtr, ytr)
    pred = model.predict(Xte)
    return {
        "test_size": test_size,
        "MAE": mean_absolute_error(yte, pred),
        "RMSE": np.sqrt(mean_squared_error(yte, pred)),
        "R2": r2_score(yte, pred)
    }

result_02 = eval_split(0.2)
result_03 = eval_split(0.3)

compare = pd.DataFrame([result_02, result_03])
print(compare.round(4))
```

```
   test_size     MAE    RMSE      R2
0        0.2  0.5332  0.7456  0.5758
1        0.3  0.5272  0.7284  0.5958
```

3 Second Practical Example (Single Block): Diabetes Dataset

[48]:
```
from sklearn.datasets import load_diabetes

# 1) Load
db = load_diabetes()
X_db = pd.DataFrame(db.data, columns=db.feature_names)
y_db = pd.Series(db.target, name="disease_progression")

# 2) Split
Xtr_db, Xte_db, ytr_db, yte_db = train_test_split(
    X_db, y_db, test_size=0.2, random_state=42
)

# 3) Train
db_model = LinearRegression()
db_model.fit(Xtr_db, ytr_db)

# 4) Predict + Evaluate
pred_db = db_model.predict(Xte_db)
db_mae = mean_absolute_error(yte_db, pred_db)
db_rmse = np.sqrt(mean_squared_error(yte_db, pred_db))
db_r2 = r2_score(yte_db, pred_db)

print("Diabetes MAE:", round(db_mae, 4))
print("Diabetes RMSE:", round(db_rmse, 4))
print("Diabetes R2:", round(db_r2, 4))

# 5) Quick coefficient interpretation support
db_coef = pd.DataFrame({
    "feature": X_db.columns,
    "coefficient": db_model.coef_
})
db_coef["abs_coef"] = db_coef["coefficient"].abs()
print("\nTop coefficients by magnitude:")
print(db_coef.sort_values("abs_coef", ascending=False).head(6)[["feature",␣
 ↪"coefficient"]])
```

```
Diabetes MAE: 42.7941
Diabetes RMSE: 53.8534
Diabetes R2: 0.4526

Top coefficients by magnitude:
  feature  coefficient
4      s1  -931.488846
8      s5   736.198859
2     bmi   542.428759
5      s2   518.062277
```

```
3      bp   347.703844
7      s4   275.317902
```

3.1 Exercise 9 Solution: Same Workflow on load_diabetes()

[49]:
```
from sklearn.datasets import load_diabetes

db = load_diabetes()
X_db = pd.DataFrame(db.data, columns=db.feature_names)
y_db = pd.Series(db.target, name="disease_progression")

Xtr_db, Xte_db, ytr_db, yte_db = train_test_split(
    X_db, y_db, test_size=0.2, random_state=42
)

db_model = LinearRegression()
db_model.fit(Xtr_db, ytr_db)
pred_db = db_model.predict(Xte_db)

print("MAE:", round(mean_absolute_error(yte_db, pred_db), 4))
print("RMSE:", round(np.sqrt(mean_squared_error(yte_db, pred_db)), 4))
print("R2:", round(r2_score(yte_db, pred_db), 4))

db_coef = pd.DataFrame({
    "feature": X_db.columns,
    "coefficient": db_model.coef_
})
db_coef["abs_coef"] = db_coef["coefficient"].abs()
db_coef = db_coef.sort_values("abs_coef", ascending=False)
print(db_coef[["feature", "coefficient"]])
```

```
MAE: 42.7941
RMSE: 53.8534
R2: 0.4526
  feature  coefficient
4      s1  -931.488846
8      s5   736.198859
2     bmi   542.428759
5      s2   518.062277
3      bp   347.703844
7      s4   275.317902
1     sex  -241.964362
6      s3   163.419983
9      s6    48.670657
0     age    37.904021
```