ouse-prices-with-linear-regression

August 15, 2024

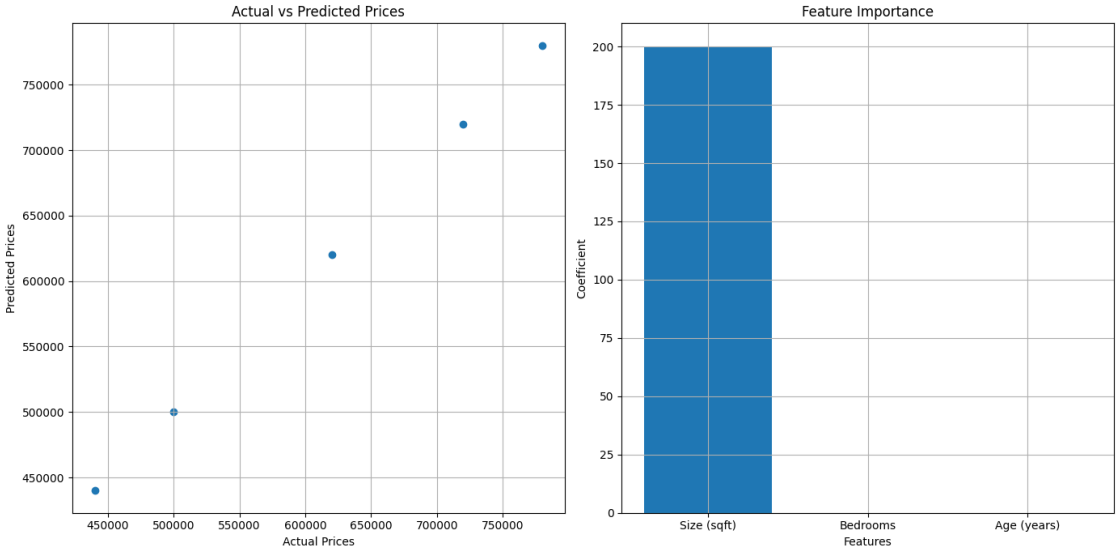
[87]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Step 1: Data Collection
def create_dataset():
    data = {
        'Size (sqft)': [1500, 1600, 1700, 1800, 1900, 2000, 2100, 2200, 2300,␣
 ↪2400,
                        2500, 2600, 2700, 2800, 2900, 3000, 3100, 3200, 3300,␣
 ↪3400,
                        3500, 3600, 3700, 3800, 3900],
        'Bedrooms': [3, 3, 4, 4, 4, 5, 5, 5, 6, 6,
                     3, 3, 4, 4, 4, 5, 5, 5, 6, 6,
                     3, 3, 4, 4, 4],
        'Age (years)': [10, 15, 20, 5, 8, 12, 14, 18, 10, 22,
                        11, 13, 17, 6, 9, 15, 19, 23, 12, 16,
                        13, 14, 21, 7, 20],
        'Price ($)': [400000, 420000, 440000, 460000, 480000, 500000, 520000,␣
 ↪540000, 560000, 580000,
                      600000, 620000, 640000, 660000, 680000, 700000, 720000,␣
 ↪740000, 760000, 780000,
                      800000, 820000, 840000, 860000, 880000]
    }
    df = pd.DataFrame(data)
    df.to_excel('house_prices.xlsx', index=False)
    print("Dataset created and saved as 'house_prices.xlsx'")

create_dataset()

# Step 2: Data Exploration and Cleaning
def explore_and_clean_data():
    # Load dataset
```

```
    df = pd.read_excel('house_prices.xlsx')

    # Explore dataset
    print("\nDataset Overview:")
    print(df.head())
    print("\nStatistical Summary:")
    print(df.describe())
    print("\nData Types and Missing Values:")
    print(df.info())
    print(df.isnull().sum())

    # Handle missing values if any (this dataset has none, but this is how you would do it)
    # df = df.dropna()  # Drop missing values
    # df.fillna(df.mean(), inplace=True)  # Fill missing values with mean (example)

    return df

df = explore_and_clean_data()

# Step 3: Feature Selection
def select_features(df):
    X = df[['Size (sqft)', 'Bedrooms', 'Age (years)']]
    y = df['Price ($)']
    return X, y

X, y = select_features(df)

# Step 4: Model Training
def train_model(X, y):
    # Split the dataset
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

    # Initialize the model
    model = LinearRegression()

    # Train the model
    model.fit(X_train, y_train)

    # Predict on the test set
    y_pred = model.predict(X_test)

    return model, X_test, y_test, y_pred

model, X_test, y_test, y_pred = train_model(X, y)
```

```python
# Step 5: Model Evaluation
def evaluate_model(y_test, y_pred):
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f"\nMean Squared Error: {mse}")
    print(f"R-squared: {r2}")
    print("Coefficients:", model.coef_)
    print("Intercept:", model.intercept_)

evaluate_model(y_test, y_pred)

# Step 6: Visualization
def visualize_results(X_test, y_test, y_pred):
    # Visualization of Actual vs Predicted prices
    plt.figure(figsize=(14, 7))

    plt.subplot(1, 2, 1)
    plt.scatter(y_test, y_pred)
    plt.xlabel('Actual Prices')
    plt.ylabel('Predicted Prices')
    plt.title('Actual vs Predicted Prices')
    plt.grid(True)

    # Plot feature importance
    plt.subplot(1, 2, 2)
    feature_names = X_test.columns
    coefficients = model.coef_
    plt.bar(feature_names, coefficients)
    plt.xlabel('Features')
    plt.ylabel('Coefficient')
    plt.title('Feature Importance')
    plt.grid(True)

    plt.tight_layout()
    plt.show()

visualize_results(X_test, y_test, y_pred)
```

```
Dataset created and saved as 'house_prices.xlsx'

Dataset Overview:
   Size (sqft)  Bedrooms  Age (years)  Price ($)
0         1500         3           10     400000
1         1600         3           15     420000
2         1700         4           20     440000
3         1800         4            5     460000
```

```
4         1900         4            8     480000

Statistical Summary:
       Size (sqft)   Bedrooms  Age (years)      Price ($)
count    25.000000  25.000000    25.000000      25.000000
mean   2700.000000   4.320000    14.000000  640000.000000
std     735.980072   1.029563     5.115336  147196.014439
min    1500.000000   3.000000     5.000000  400000.000000
25%    2100.000000   4.000000    10.000000  520000.000000
50%    2700.000000   4.000000    14.000000  640000.000000
75%    3300.000000   5.000000    18.000000  760000.000000
max    3900.000000   6.000000    23.000000  880000.000000

Data Types and Missing Values:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Size (sqft)  25 non-null     int64
 1   Bedrooms     25 non-null     int64
 2   Age (years)  25 non-null     int64
 3   Price ($)    25 non-null     int64
dtypes: int64(4)
memory usage: 928.0 bytes
None
Size (sqft)    0
Bedrooms       0
Age (years)    0
Price ($)      0
dtype: int64

Mean Squared Error: 1.3552527156068806e-21
R-squared: 1.0
Coefficients: [2.00000000e+02 6.98337613e-13 9.28417682e-14]
Intercept: 100000.00000000012
```


Actual vs Predicted Prices
Predicted Prices
Actual Prices
450000
500000
550000
600000
650000
700000
750000
Feature Importance
Coefficient
Features
Size (sqft)
Bedrooms
Age (years)
0
25
50
75
100
125
150
175
200