In [1]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

In [2]:
```
# Load the dataset
df = pd.read_csv(r"C:\Users\junai\OneDrive - Middlesex University\ML, Regression\We
```

In [3]:
```
df.head()
```

Out[3]:

	Rooms	Age	Distance	Accessibility	Tax	DisadvantagedPosition	Crime	NitricOxides	Pupil
0	5.565	70.6	2.0635	24	666	17.16	8.79212	0.584	
1	6.879	77.7	3.2721	8	307	9.93	0.62356	0.507	
2	5.972	76.7	3.1025	4	304	9.97	0.34940	0.544	
3	6.943	97.4	1.8773	5	403	4.59	1.22358	0.605	
4	5.926	71.0	2.9084	24	666	18.13	15.57570	0.580	

In [4]:
```
# Task 1: Polynomial Regression
# Extract features and target variable
X = df[['Rooms', 'Age', 'Distance', 'Accessibility', 'Tax', 'DisadvantagedPosition'
y = df['Price']
```

In [6]:
```
X.head()
```

Out[6]:

	Rooms	Age	Distance	Accessibility	Tax	DisadvantagedPosition	Crime	NitricOxides	Pupil
0	5.565	70.6	2.0635	24	666	17.16	8.79212	0.584	
1	6.879	77.7	3.2721	8	307	9.93	0.62356	0.507	
2	5.972	76.7	3.1025	4	304	9.97	0.34940	0.544	
3	6.943	97.4	1.8773	5	403	4.59	1.22358	0.605	
4	5.926	71.0	2.9084	24	666	18.13	15.57570	0.580	

In [7]:
```
y.head()
```

Out[7]:
```
0    11.7
1    27.5
2    20.3
3    41.3
4    19.1
Name: Price, dtype: float64
```

In [ ]:

In [51]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [52]:
```
# Create polynomial features
poly = PolynomialFeatures(degree=3)
X_poly_train = poly.fit_transform(X_train)
X_poly_test = poly.transform(X_test)
```

In [53]:
```
# Standardize the features
scaler = StandardScaler()
X_poly_train_scaled = scaler.fit_transform(X_poly_train)
X_poly_test_scaled = scaler.transform(X_poly_test)
```

In [54]:
```
# Train polynomial regression model
poly_reg = Ridge(alpha=1.0)  # You can also use Lasso by replacing Ridge with Lassc
poly_reg.fit(X_poly_train_scaled, y_train)
```

Out[54]:
```
▾ Ridge
Ridge()
```

alphas: This is a list of different values for the regularization parameter (alpha).

best_alpha: This variable will be used to keep track of the alpha value that gives the best model performance.

best_model: This variable will store the model that performs the best in terms of minimizing the Mean Squared Error (MSE).

best_mse: This variable keeps track of the lowest MSE obtained among all the tested models

best_coefficients: This variable will store the coefficients of the features for the best model.

The purpose of this code snippet is to set up variables for tuning the regularization parameter and keeping track of the best-performing model during the tuning process. The values in alphas will be used to iterate over different strengths of regularization, and the model with the lowest average MSE (Mean Squared Error) across cross-validation folds will be selected as the best model.

In [88]:
```
# Task 2: Regularization
# Tune regularization parameter
alphas = [0.01, 0.1, 1, 10, 100]
best_alpha = None
best_model = None
best_mse = float('inf')
best_coefficients = None
```

In [90]:
```
for alpha in alphas:
    model = Ridge(alpha=alpha)
    model.fit(X_poly_train_scaled, y_train)
    y_pred = model.predict(X_poly_test_scaled)
    mse = mean_squared_error(y_test, y_pred)

    if mse < best_mse:
        best_mse = mse
        best_alpha = alpha
        best_model = model
        best_coefficients = model.coef_.copy()
```

In [ ]:

```

```

In [75]:

```
# Use the best alpha to train the final model
final_model = Ridge(alpha=best_alpha)
final_model.fit(X_poly_train_scaled, y_train)
```

Out[75]:

```
▾     Ridge
Ridge(alpha=1)
```

In [76]:

```
# Use the best model for predictions
y_pred_best = best_model.predict(X_poly_test_scaled)
```

In [77]:

```
# Print the results
print(f'Best Regularization Parameter (alpha): {best_alpha}')
print(f'Mean Squared Error on Test Data with Best Model: {best_mse}')

print(f'Coefficients of the Best Model: {best_coefficients}')
```

```
Best Regularization Parameter (alpha): 1
Mean Squared Error on Test Data with Best Model: 8.371145267611507
Coefficients of the Best Model: [ 0.00000000e+00  6.05623254e+00  1.05828771e+00 -
3.24470475e+00
  1.82455277e+00  4.06936550e-01 -7.03059267e-01 -3.36979980e-01
  1.25906258e-01  1.11962415e+00 -6.04306648e-01  5.88918358e-01
  3.94098890e+00  6.88196827e-01 -9.16746717e-01  1.11364306e+00
 -1.70176785e-01 -1.32110304e+00 -2.07730130e-02  2.17367584e+00
  1.33875562e+00 -2.95248452e-01  8.13546401e-01  2.87398175e-01
 -1.30508023e-01  1.97421712e+00  3.30474525e-01 -1.25255038e-01
 -3.89486037e-01 -1.89065240e-01  3.61584307e-01 -5.85762671e-01
  9.97555124e-01 -1.02346640e+00 -1.41552308e+00 -2.00861516e+00
 -1.48015584e+00 -1.85132981e+00 -2.31007737e+00 -1.66457323e+00
 -3.78543923e-01 -2.19056607e+00 -3.59058489e-01  7.13252994e-01
 -1.15140362e+00  4.42337350e-01  7.70554152e-01  1.60188397e+00
 -2.46943880e-01  8.51208074e-01  7.02102620e-01 -1.37803198e+00
  1.38239495e-01 -6.72375819e-02  1.26242340e+00 -1.11936335e-01
  1.10893490e+00  1.45379106e+00 -8.49545815e-02 -9.53397615e-02
  2.46874860e-01 -4.93791986e-02 -1.38937959e+00 -3.09770949e-01
 -3.03562624e-01 -1.88856565e-01 -2.86059905e-01 -6.08694297e-02
  1.03764722e-01 -7.61444864e-01 -1.55328212e-01 -3.69065557e-02
  1.06993483e+00  1.04422643e-01  1.17592663e-01 -3.69972874e-01
 -1.90609202e-01  5.59347190e-01  1.92933686e+00 -1.77271107e+00
  1.43604125e+00 -3.46747904e+00 -4.86364753e+00 -2.11347302e+00
 -3.41835412e-01 -4.48619698e-01 -2.48885279e+00  1.94247655e-01
 -1.34323182e+00  3.58995859e-01  1.40924737e+00  1.17798481e+00
 -4.59271822e-01 -5.28727002e-01 -3.83598079e-02 -5.61402355e-01
 -1.50669783e+00 -9.29886714e-01  1.50669022e+00 -2.93060816e-01
  1.14895124e+00 -6.92238592e-02 -8.54108020e-01 -2.16374403e-01
  1.06690246e+00 -1.48684904e-02 -3.61171103e-01  4.55924089e-01
 -2.19997780e+00 -8.48394197e-01  1.12895867e+00  5.69460666e-01
  5.15115946e-03  4.23350250e-01  1.02967255e+00 -6.14436570e-01
 -7.09070754e-01 -5.73860246e-01  3.54778656e-01 -8.66855625e-01
 -4.82209284e-01  1.59776188e-01  4.19120613e-01  1.85521166e+00
  1.39028667e+00 -1.67650417e-01  2.47559466e-01 -7.60496613e-01
 -1.83938592e+00  3.46549488e-01  2.23576385e-01  6.60662325e-02
 -7.25699502e-01  2.53965883e-01  4.81749043e-02 -1.53460009e+00
  1.60087854e-01 -3.65709598e-01 -1.45257245e+00  3.60323206e-01
 -1.07634658e+00  1.85859893e-02 -1.08414885e-01  8.84391033e-01
  8.88307221e-01 -1.62159736e+00  2.53144674e+00  4.92570869e-01
 -2.00621811e+00 -9.61827420e-01 -6.37189726e-01 -7.09189304e-01
  1.32858452e-01  8.46703181e-01  3.63564988e+00 -2.26262163e+00
 -3.40330512e+00  1.01831278e+00 -2.92051009e+00  9.50140635e-01
 -5.79596549e-01  8.49219389e-01 -7.77062408e-01  5.93882923e-01
  1.38674691e+00 -1.03544650e+00  3.20291946e-01  3.98995787e-01
  1.91293290e+00  5.55012169e-01  1.80667851e+00  6.48174369e-01
 -1.58791988e+00  5.17619846e-02 -6.60229968e-01  5.63175754e-01
  5.74932282e-02  1.28365079e+00  1.72735635e+00 -4.19923392e-01
 -4.96515063e-01 -7.86513534e-02  3.79399432e-01 -1.46563119e+00
 -1.04904643e+00 -6.23176288e-01 -2.83382696e-01 -5.09170585e-01
 -6.55177718e-02 -4.70578503e-01 -1.33119309e+00  8.27892853e-02
  1.66880905e-01 -2.00182244e-02 -9.37860631e-01  3.28810721e-01
 -1.69948682e-01  1.36814445e-01  1.08690852e+00 -4.14492462e-01
  2.00759692e+00  1.31430309e+00 -7.17463480e-01  3.95929060e+00
  4.11149672e-02  9.88395628e-01 -6.21751979e-01  1.05900626e+00
 -1.59957791e+00 -5.66677295e-01  3.85769809e+00 -3.95271942e-01
  7.40340049e-01 -1.28735558e+00 -1.20169025e+00 -3.08371517e-01
  2.32103191e-01  2.74424580e+00 -8.72341269e-01  4.98452022e-01
  2.66424527e-01 -6.51746100e-01  1.76066948e+00 -2.54699405e+00
 -1.46025927e-01  2.12924597e+00  4.62089528e-01  1.31493456e+00
  2.98295545e+00 -2.40224652e-01 -9.28766928e-01 -1.69256516e+00
 -5.83571687e-01 -9.95966183e-01  1.08393374e+00 -1.23848805e+00
 -4.63243220e-02  7.05242927e-01 -3.40654480e-01  1.02428946e+00
 -2.01734883e+00 -4.61934683e-01  1.42197800e-01  7.78626093e-01
```

```
-7.47486706e-01 -4.80562596e-01 -2.00037132e+00  5.79871835e-01
-1.03159991e+00 -3.86783274e-01 -2.89468665e-01 -4.52796851e-01
 1.48012696e-01 -1.55743748e+00  5.25848215e-01 -1.40765238e-01
 6.72293392e-01 -3.66725557e-01  1.82540732e-01 -2.11932050e-01
 9.00699007e-01 -8.79708282e-01 -1.00203865e+00 -1.64448051e+00
-1.69822349e+00 -2.41406602e-01  5.48767603e-01  4.72940407e-01
-1.35621757e-01  4.83636532e-01  1.85405421e-01  5.32520111e-01
 3.96158713e-01 -3.32445484e-02  1.50342975e+00 -7.64970579e-01
 7.69461291e-01  6.61599247e-01 -7.32093135e-02  2.74825652e-01
 2.18488899e-01 -1.68512606e+00  3.71373241e-01 -1.07105840e-01
 1.06747459e+00  8.99374205e-01  2.63392075e-01  7.02920722e-01
 4.89991098e-01 -8.63490661e-01 -6.46053218e-01  3.67377850e-02
-2.11535337e+00 -2.72446329e-01  1.90167409e-01  2.29268691e-01
 1.56173793e+00  2.51957994e-01 -2.26907069e-01  1.17581077e-01
 6.97571772e-01  1.72575229e-01  1.84846209e+00  7.67232668e-01
 8.58148852e-01 -1.79918540e+00  1.97653578e+00 -1.48702827e-01
 2.31238004e-01 -4.46148517e-01  1.66737803e+00  1.06388521e+00
-1.15018987e+00  1.97816550e+00  1.83966525e+00 -2.36958701e-01
 7.71222469e-02  8.37540216e-01  2.09042505e-01  1.05289203e+00
 1.61780529e-01  3.39828290e-01 -8.45250880e-01  7.50970363e-01
 7.85280835e-01 -7.91068336e-01  9.49058727e-01 -7.72984247e-02
-2.55798535e+00  1.23328154e+00  4.45851935e-02 -2.92147369e-01
-1.05767745e+00 -3.01939623e-01 -1.97782344e-01 -1.02018875e-01
 3.23787841e-01 -1.13911791e-01 -8.15793910e-02 -8.12151966e-01
 1.04292100e-01  9.41170012e-01  5.46183879e-01  3.58419974e-02
 3.79331662e-01 -9.82765377e-01  4.38757865e-01 -2.38170506e-01
-1.13466369e+00  2.45978615e-01 -6.74421278e-01 -5.04837953e-01
 3.79759574e-01 -4.23031126e-02  1.08019203e+00  4.13876071e-01
-1.87487336e-01  5.44633384e-01 -1.36702457e+00  1.76458132e-01
 1.60682482e+00 -1.03391898e+00 -2.39955362e-01 -2.25216333e-01]
```

In [ ]:

Normalization Coefficient

Positive vs. Negative Values: A positive coefficient means that as the corresponding feature increases, the predicted house price tends to increase, and vice versa for negative coefficients. Magnitude of Coefficients: Features with larger magnitude coefficients have a more significant impact on the predicted house prices.

Features with High Positive Coefficients: Features with high positive coefficients contribute more to increasing house prices.

Features with High Negative Coefficients: Features with high negative coefficients contribute more to decreasing house prices.

In [78]:

```
# Calculate L2 norm of coefficients
l2_norm = np.linalg.norm(best_coefficients)

# Normalize coefficients
normalized_coefficients = best_coefficients / l2_norm

# Print normalized coefficients
print(f'Normalized Coefficients of the Best Model: {normalized_coefficients}')
```

```
Normalized Coefficients of the Best Model: [ 0.00000000e+00  2.70356627e-01  4.724
30830e-02 -1.44847052e-01
  8.14499659e-02  1.81660781e-02 -3.13853095e-02 -1.50431428e-02
  5.62058856e-03  4.99812065e-02 -2.69768881e-02  2.62899385e-02
  1.75929914e-01  3.07218344e-02 -4.09245433e-02  4.97142042e-02
 -7.59687169e-03 -5.89754375e-02 -9.27329275e-04  9.70351889e-02
  5.97634672e-02 -1.31802032e-02  3.63175720e-02  1.28297586e-02
 -5.82601621e-03  8.81311408e-02  1.47527324e-02 -5.59151741e-03
 -1.73870687e-02 -8.44007235e-03  1.61415060e-02 -2.61490653e-02
  4.45319160e-02 -4.56886230e-02 -6.31904479e-02 -8.96667056e-02
 -6.60757226e-02 -8.26453211e-02 -1.03124297e-01 -7.43083100e-02
 -1.68986012e-02 -9.77891868e-02 -1.60287508e-02  3.18403682e-02
 -5.13998755e-02  1.97464073e-02  3.43983525e-02  7.15097948e-02
 -1.10238360e-02  3.79988289e-02  3.13426037e-02 -6.15168053e-02
  6.17115729e-03 -3.00155679e-03  5.63559163e-02 -4.99695643e-03
  4.95040272e-02  6.48987710e-02 -3.79246239e-03 -4.25606781e-03
  1.10207549e-02 -2.20433966e-03 -6.20233746e-02 -1.38285029e-02
 -1.35513567e-02 -8.43075684e-03 -1.27700168e-02 -2.71727573e-03
  4.63216694e-03 -3.39917042e-02 -6.93401571e-03 -1.64754769e-03
  4.77630228e-02  4.66153724e-03  5.24946088e-03 -1.65159805e-02
 -8.50899643e-03  2.49698503e-02  8.61276383e-02 -7.91356973e-02
  6.41064005e-02 -1.54791933e-01 -2.17118371e-01 -9.43476714e-02
 -1.52598944e-02 -2.00268579e-02 -1.11105022e-01  8.67142079e-03
 -5.99632895e-02  1.60259549e-02  6.29102934e-02  5.25864882e-02
 -2.05023800e-02 -2.36029327e-02 -1.71242240e-03 -2.50615950e-02
 -6.72605851e-02 -4.15111266e-02  6.72602454e-02 -1.30825448e-02
  5.12903988e-02 -3.09022630e-03 -3.81282856e-02 -9.65918226e-03
  4.76276543e-02 -6.63745140e-04 -1.61230601e-02  2.03529336e-02
 -9.82093362e-02 -3.78732144e-02  5.03979093e-02  2.54213266e-02
  2.29953208e-04  1.88988030e-02  4.59656717e-02 -2.74290985e-02
 -3.16536686e-02 -2.56177285e-02  1.58376945e-02 -3.86973521e-02
 -2.15263326e-02  7.13257807e-03  1.87099876e-02  8.28186113e-02
  6.20638678e-02 -7.48409195e-03  1.10513164e-02 -3.39493733e-02
 -8.21121332e-02  1.54703358e-02  9.98068633e-03  2.94926651e-03
 -3.23959935e-02  1.13373057e-02  2.15057869e-03 -6.85061715e-02
  7.14649117e-03 -1.63256633e-02 -6.48443707e-02  1.60852091e-02
 -4.80492502e-02  8.29698226e-04 -4.83975514e-03  3.94801513e-02
  3.96549741e-02 -7.23898215e-02  1.13006461e-01  2.19888847e-02
 -8.95596986e-02 -4.29369935e-02 -2.84448234e-02 -3.16589608e-02
  5.93094185e-03  3.77977257e-02  1.62299257e-01 -1.01005823e-01
 -1.51927141e-01  4.54585598e-02 -1.30374660e-01  4.24152831e-02
 -2.58738031e-02  3.79100519e-02 -3.46888879e-02  2.65115619e-02
  6.19058493e-02 -4.62234272e-02  1.42981713e-02  1.78115940e-02
  8.53953484e-02  2.47763304e-02  8.06520401e-02  2.89351895e-02
 -7.08864233e-02  2.31070975e-03 -2.94733642e-02  2.51407614e-02
  2.56655853e-03  5.73035290e-02  7.71110147e-02 -1.87458244e-02
 -2.21649577e-02 -3.51107962e-03  1.69367920e-02 -6.54273269e-02
 -4.68305423e-02 -2.78192488e-02 -1.26505033e-02 -2.27299136e-02
 -2.92478265e-03 -2.10071222e-02 -5.94258676e-02  3.69580128e-03
  7.44974012e-03 -8.93634716e-04 -4.18670907e-02  1.46784584e-02
 -7.58668897e-03  6.10754155e-03  4.85207460e-02 -1.85033820e-02
  8.96212503e-02  5.86718802e-02 -3.20283287e-02  1.76746921e-01
  1.83541563e-03  4.41230265e-02 -2.77556661e-02  4.72751598e-02
 -7.14068502e-02 -2.52970739e-02  1.72211724e-01 -1.76453577e-02
  3.30495630e-02 -5.74689151e-02 -5.36447241e-02 -1.37660307e-02
  1.03613320e-02  1.22506036e-01 -3.89422371e-02  2.22514256e-02
  1.18934727e-02 -2.90946354e-02  7.85981483e-02 -1.13700509e-01
 -6.51875187e-03  9.50517927e-02  2.06281654e-02  5.87000699e-02
  1.33162287e-01 -1.07238826e-02 -4.14611382e-02 -7.55578993e-02
 -2.60512575e-02 -4.44609841e-02  4.83879489e-02 -5.52874169e-02
 -2.06796674e-03  3.14827903e-02 -1.52071764e-02  4.57253650e-02
 -9.00565859e-02 -2.06212529e-02  6.34786022e-03  3.47586925e-02
 -3.33685973e-02 -2.14528227e-02 -8.92986920e-02  2.58860921e-02
 -4.60517113e-02 -1.72664145e-02 -1.29221874e-02 -2.02133304e-02
```

```
  6.60744332e-03 -6.95256566e-02  2.34744206e-02 -6.28390914e-03
  3.00118882e-02 -1.63710168e-02  8.14881140e-03 -9.46087093e-03
  4.02081565e-02 -3.92711083e-02 -4.47320654e-02 -7.34113498e-02
 -7.58104931e-02 -1.07766461e-02  2.44975663e-02  2.11125601e-02
 -6.05429870e-03  2.15900464e-02  8.27669413e-03  2.37722611e-02
  1.76849441e-02 -1.48407181e-03  6.71146946e-02 -3.41490959e-02
  3.43495660e-02  2.95344903e-02 -3.26814120e-03  1.22685078e-02
  9.75357559e-03 -7.52258101e-02  1.65784943e-02 -4.78131802e-03
  4.76531948e-02  4.01490160e-02  1.17581009e-02  3.13791246e-02
  2.18737210e-02 -3.85471366e-02 -2.88404991e-02  1.64001359e-03
 -9.44316124e-02 -1.21622923e-02  8.48927432e-03  1.02347969e-02
  6.97176332e-02  1.12476714e-02 -1.01293716e-02  5.24894367e-03
  3.11403419e-02  7.70394081e-03  8.25173033e-02  3.42500780e-02
  3.83086726e-02 -8.03175395e-02  8.82346483e-02 -6.63825151e-03
  1.03227091e-02 -1.99165418e-02  7.44335190e-02  4.74929611e-02
 -5.13456926e-02  8.83074007e-02  8.21246027e-02 -1.05780871e-02
  3.44281869e-03  3.73886813e-02  9.33187856e-03  4.70022140e-02
  7.22205397e-03  1.51702944e-02 -3.77328935e-02  3.35241114e-02
  3.50557672e-02 -3.53141274e-02  4.23669856e-02 -3.45068345e-03
 -1.14191172e-01  5.50549926e-02  1.99033020e-03 -1.30417676e-02
 -4.72158402e-02 -1.34789042e-02 -8.82921308e-03 -4.55423052e-03
  1.44542318e-02 -5.08514286e-03 -3.64179032e-03 -3.62553229e-02
  4.65570968e-03  4.20148250e-02  2.43822261e-02  1.60002468e-03
  1.69337667e-02 -4.38716861e-02  1.95866153e-02 -1.06321833e-02
 -5.06525876e-02  1.09807456e-02 -3.01068793e-02 -2.25365003e-02
  1.69528691e-02 -1.88845570e-03  4.82209151e-02  1.84758656e-02
 -8.36963301e-03  2.43130104e-02 -6.10254229e-02  7.87727768e-03
  7.17303599e-02 -4.61552369e-02 -1.07118612e-02 -1.00538953e-02]
```

Rank Coefficient

In [79]:

```python
# Get the absolute values of coefficients
absolute_coefficients = np.abs(best_coefficients)

# Create a dataframe for better visualization
coefficients_df = pd.DataFrame({'Feature': poly.get_feature_names_out(X.columns), '

# Sort the dataframe by absolute coefficient values in descending order
coefficients_df = coefficients_df.sort_values(by='Absolute Coefficient', ascending=

# Print the ranked coefficients
print('Ranked Coefficients:')
print(coefficients_df)
```

```
Ranked Coefficients:
                                       Feature  Coefficient  \
1                                        Rooms     6.056233   
82                                   Rooms^2 Tax    -4.863648   
203                             Distance^2 Crime     3.959291   
12                                     Rooms^2     3.940989   
210  Distance Accessibility DisadvantagedPosition     3.857698   
..                                         ...          ...   
193                           Age PupilTeacher^2    -0.020018   
141                          Rooms Residential^2     0.018586   
105                  Rooms Distance PupilTeacher    -0.014868   
112                Rooms Accessibility NitricOxides     0.005151   
0                                            1     0.000000   

     Absolute Coefficient  
1                6.056233  
82               4.863648  
203              3.959291  
12               3.940989  
210              3.857698  
..                    ...  
193              0.020018  
141              0.018586  
105              0.014868  
112              0.005151  
0                0.000000  

[364 rows x 3 columns]
```

Top Positive Coefficient (Largest Increase in Price):

Feature: Rooms Coefficient: 6.056233 Absolute Coefficient: 6.056233

Top Negative Coefficient (Largest Decrease in Price): Feature: Rooms^2 Tax Coefficient: -4.863648 Absolute Coefficient: 4.863648

Other Features with Significant Impact: Distance^2 Crime: Positive coefficient (3.959291) Rooms^2: Positive coefficient (3.940989) Distance Accessibility DisadvantagedPosition: Positive coefficient (3.857698) Features with Little Impact:

Features with coefficients close to zero, such as the intercept (1) with a coefficient of 0.000000, have little impact on the predicted house prices.

Interpretation: A one-unit increase in the Rooms feature results in a substantial increase in the predicted house price. The interaction term Rooms^2 Tax has a large negative impact, indicating that this combination of features leads to a significant decrease in the predicted price.

In [ ]:

In [61]:

```
# Task 3: Performance Assessment
y_pred_final = final_model.predict(X_poly_test_scaled)
mse_final = mean_squared_error(y_test, y_pred_final)
print(f'Mean Squared Error on Test Data: {mse_final}')
```

```
Mean Squared Error on Test Data: 8.371145267611507
```

MSE

The MSE represents the average of the squared differences between the predicted and actual values. Lower MSE values indicate better performance, as they suggest that the model's predictions are closer to the true values.

In [62]:
```
# Task 4: Stability
# Evaluate the stability by examining coefficients
coefficients = final_model.coef_
poly_feature_names = poly.get_feature_names_out(X.columns)

# Create feature names for polynomial features
coefficients_names = [name.replace(' ', '*') for name in poly_feature_names]
```

In [ ]:
```

```

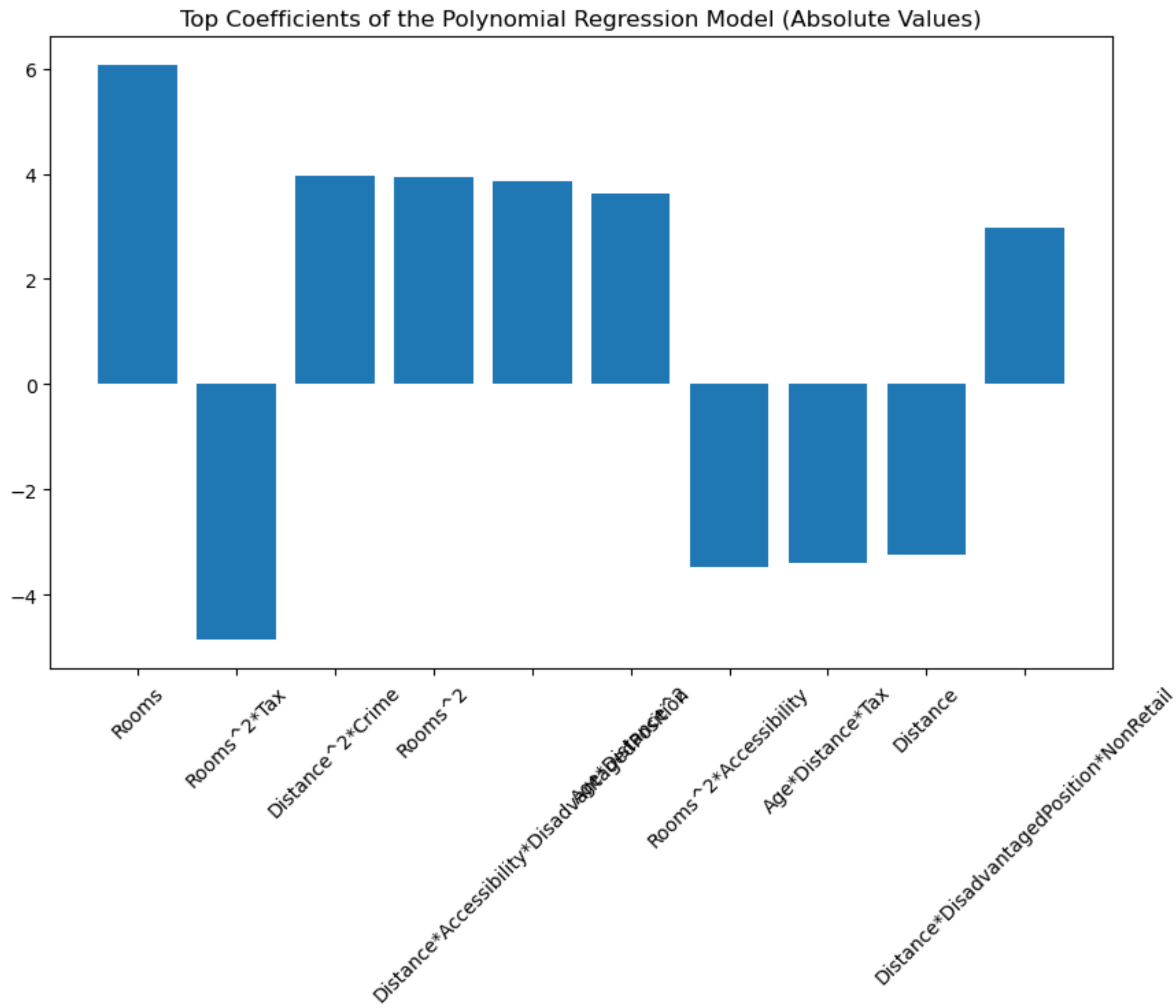
In [80]:
```
# Task 5: Interpretation
# Simplified Visualization of Coefficients
absolute_coefficients = np.abs(coefficients)
sorted_indices = np.argsort(absolute_coefficients)[::-1]  # Sort in descending orde
```

In [ ]:
```

```

In [81]:
```
# Select top N features for visualization (adjust N as needed)
top_n = 10
selected_indices = sorted_indices[:top_n]
selected_coefficients = coefficients[selected_indices]
selected_names = np.array(coefficients_names)[selected_indices]
```

In [82]:
```
# Plot the simplified visualization
plt.figure(figsize=(10, 6))
plt.bar(selected_names, selected_coefficients)
plt.xticks(rotation=45)
plt.title('Top Coefficients of the Polynomial Regression Model (Absolute Values)')
plt.show()
```



Positive Coefficients:

The positive coefficient for the Rooms feature (6.056233) suggests that an increase in the number of rooms leads to a substantial increase in the predicted house price. This is consistent with common intuition.

The positive coefficient for the interaction term Distance^2 Crime (3.959291) indicates that the squared distance to a high-crime area has a positive impact on house prices. Although this might seem counterintuitive, it suggests that, within certain ranges, proximity to a high-crime area might be associated with higher-priced houses.

Positive coefficients for other features like Rooms^2 and Distance Accessibility DisadvantagedPosition suggest positive contributions to house prices.

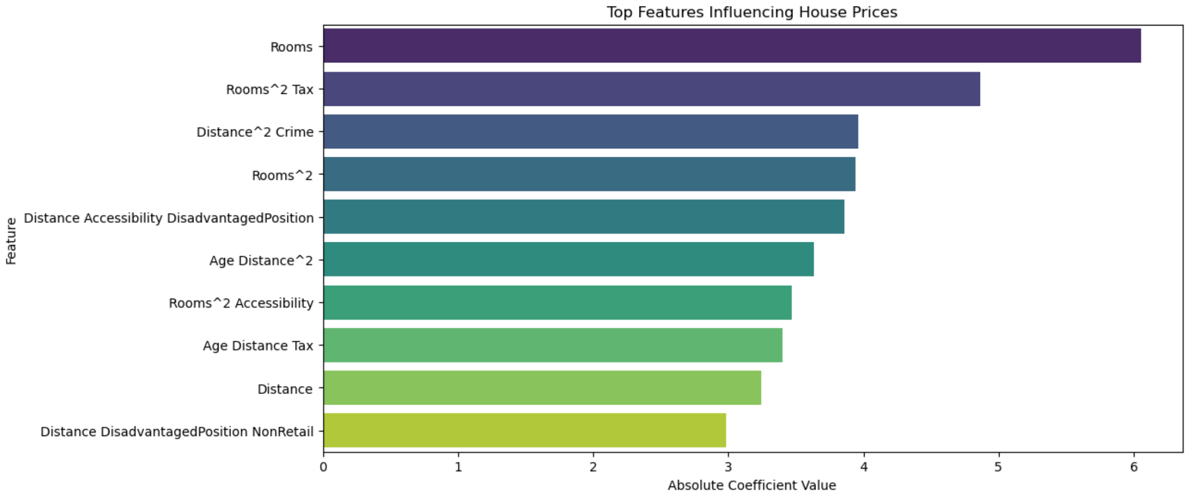
Negative Coefficients:

The negative coefficient for the feature Rooms^2 Tax (-4.863648) implies that the interaction between the squared number of rooms and the tax rate has a significant negative impact on house prices. In specific scenarios, an increase in both the number of rooms and the tax rate might lead to a notable decrease in the predicted house price.

In [83]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Visualize the top N features based on absolute coefficient values
top_features = coefficients_df.head(10)  # Adjust N as needed
```

```
plt.figure(figsize=(12, 6))
sns.barplot(x='Absolute Coefficient', y='Feature', data=top_features, palette='viri
plt.title('Top Features Influencing House Prices')
plt.xlabel('Absolute Coefficient Value')
plt.ylabel('Feature')
plt.show()
```



In [ ]:

In [ ]: