Robert Pavlik

D600 Statistical Data Mining

UGN1 Task 3: Principal Component Analysis

A. GitLab
   - https://gitlab.com/wgu-gitlab-environment/student-repos/rpavli5/d600-statistical-data-mining.git
     working_branch – Task 3 Directory

B. Describe the purpose of this Data Analysis
   1. Research Question
      How do property features and location metrics influence housing prices in the dataset?
   2. Goal of Data Analysis
      The goal is to build a linear regression model that predicts housing prices based on significant features in the dataset. This will help the organization understand which factors most strongly affect property prices, enabling better pricing, investments, and resource allocation decision-making.

C. Reasons for using PCA
   1. How can PCA be used to Prepare Datasets for Linear Regression Analysis:

      Principal Component Analysis (PCA) was employed to reduce the dimensionality of the dataset, simplify the regression model, and mitigate the effects of multicollinearity between independent variables. By transforming the original variables into uncorrelated principal components, PCA ensures that the regression analysis focuses only on the most significant aspects of the data. This process is particularly beneficial in datasets with highly correlated variables, as it prevents redundant information from impacting the model's accuracy.

      The expected outcome of PCA is a set of transformed variables (principal components) that explain a significant portion of the dataset's variance, allowing for a more streamlined and effective regression analysis.

   2. Summarize the Assumption of PCA

      One key assumption of PCA is that the data must be linearly correlated. This is because PCA identifies directions (principal components) that maximize the variance of the data, which relies on linear transformations of the variables. Additionally, standardization of the data is

necessary to ensure all variables contribute equally to the variance, preventing larger-scaled variables from dominating the results.

D. Summarize Data Preparation Process for Linear Regression Analysis

1. Identify Continuous Dataset Variables

- The six continuous independent variables were selected for the analysis. The dependent variable is ‘Price’.

```
# D1 - Identify the Variables (Choosing 6 Continuous Variables and Excluding Dependent Variable 'Price')
continuous_variables = ["SquareFootage", "BackyardSpace", "CrimeRate", "SchoolRating", "AgeOfHome", "RenovationQuality"]

# Filter dataset for independent variables only
df_continuous = df[continuous_variables]
✓ [3] < 10 ms
```

2. Standardize Continuous Dataset Variables

- All continuous variables were standardized using StandardScaler to ensure they have a mean of 0 and a standard deviation of 1, which is essential for PCA. The cleaned standardized dataset was then saved.

```
#D2 - Standardize the Dataset
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df_continuous)
✓ [4] < 10 ms

# Convert scaled data back to a DataFrame for saving
scaled_df = pd.DataFrame(scaled_data, columns=continuous_variables)

# Save the standardized dataset
scaled_df.to_csv('D600_Task3_Cleaned_Standardized_Dataset.csv', index=False)

print("Standardized dataset saved as 'D600_Task3_Cleaned_Standardized_Dataset.csv'")
✓ [5] 30ms
 Standardized dataset saved as 'D600_Task3_Cleaned_Standardized_Dataset.csv'
```

3. Descriptive Statistics of Dependent and Independent Variables

```
Descriptive Statistics for Dependent and Independent Variables:
              Price  SquareFootage  BackyardSpace    CrimeRate  SchoolRating  \
count  7.000000e+03    7000.000000    7000.000000  7000.000000   7000.000000
mean   3.072820e+05    1048.947459     511.507029    31.226194      6.942923
std    1.501734e+05     426.010482     279.926549    18.025327      1.888148
min    8.500000e+04     550.000000       0.390000     0.030000      0.220000
25%    1.921075e+05     660.815000     300.995000    17.390000      5.650000
50%    2.793230e+05     996.320000     495.965000    30.385000      7.010000
75%    3.918781e+05    1342.292500     704.012500    43.670000      8.360000
max    1.046676e+06    2874.700000    1631.360000    99.730000     10.000000

         AgeOfHome  RenovationQuality
count  7000.000000        7000.000000
mean     46.797046           5.003357
std      31.779701           1.970428
min       0.010000           0.010000
25%      20.755000           3.660000
50%      42.620000           5.020000
75%      67.232500           6.350000
max     178.680000          10.000000
Descriptive statistics saved as 'D600_Task3_Descriptive_Statistics_with_Price.csv'
```

- Descriptive Statistics for the dependent variable ‘Price’ and the six independent variables were computed, including counts, means, standard deviations, and min/max values. These statistics provide a clear understanding of the data distribution.

E. Perform PCA

1. Principal Component Matrix

- PCA was performed on the standardized dataset, resulting in a matrix of principal components. Each component represents a linear combination of the original variables.

```
#E1 - Perform PCA to compute all principal components
pca = PCA()
pca_data = pca.fit_transform(scaled_data)

# Convert PCA output to a DataFrame
pca_columns = [f"PC{i+1}" for i in range(pca.n_components_)]
df_pca = pd.DataFrame(pca_data, columns=pca_columns)

# Save the matrix of all principal components
df_pca.to_csv('D600_Task3_PCA_Matrix.csv', index=False)

print("Matrix of all principal components saved as 'D600_Task3_PCA_Matrix.csv'")
[26]
 Matrix of all principal components saved as 'D600_Task3_PCA_Matrix.csv'
```

```
Matrix of all principal components saved as 'D600_Task3_PCA_Matrix.csv'
PCA Component Matrix (Eigenvectors):
     SquareFootage  BackyardSpace  CrimeRate  SchoolRating  AgeOfHome  \
PC1       0.475839       0.158756  -0.222476      0.559061  -0.215191
PC2       0.338373      -0.391497   0.768737      0.025337   0.308447
PC3       0.143962       0.876295   0.407054     -0.210577  -0.034795
PC4       0.061718       0.191351  -0.315830      0.072766   0.923164
PC5       0.769413      -0.119406  -0.283034     -0.491873  -0.071341
PC6      -0.206276      -0.052549  -0.118314     -0.628675  -0.004608

     RenovationQuality
PC1           0.583114
PC2           0.213298
PC3          -0.011699
PC4           0.047958
PC5          -0.258087
PC6           0.738537
```

- Each value in the matrix represents the correlation between an original variable and a principal component. The higher the absolute value, the more that feature influences that PC.
    - PC1 is most influenced by SchoolRating (0.559) and RenovationQuality (0.583) – This suggests PC1 is a property quality factor.

- PC2 is most influenced by CrimeRate (0.768) – Suggesting PC2 is strongly tied to neighborhood safety
- PC5 is strongly affected by SquareFootage (0.769) – Implying PC5 represents property size

- By understanding this matrix, we can relate principal components to meaningful real-world housing attributes, which are then used in regression modeling.

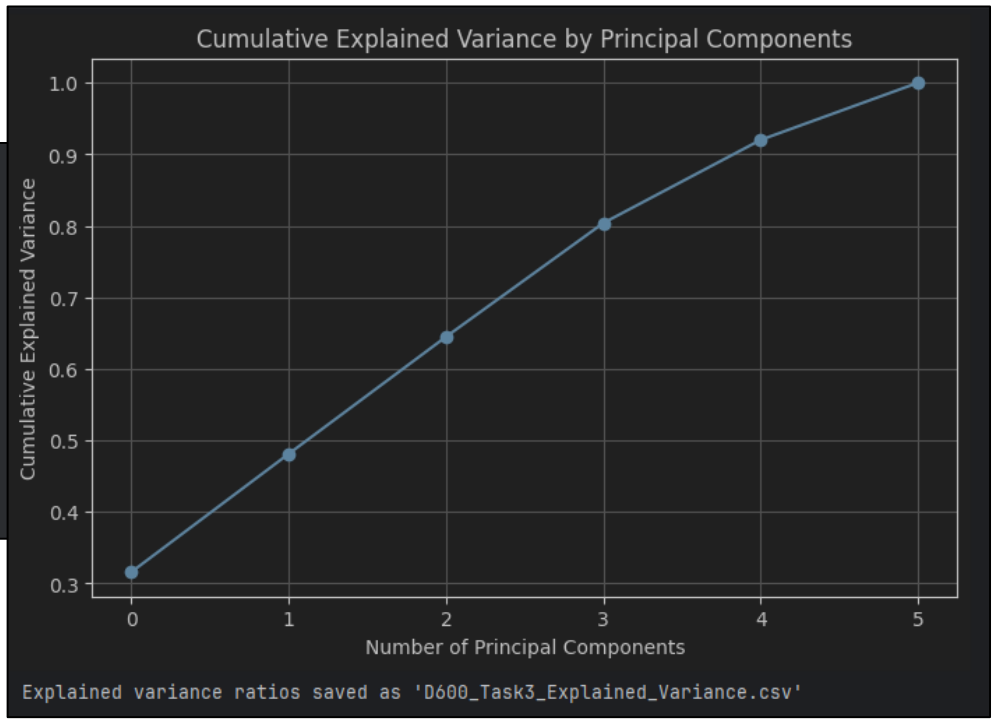
2. Identify Total Number of Principal Components
    - Using the Elbow Rule and the cumulative explained variance plot, five principal components were retained. These components collectively explained over 85% of the variance in the dataset.

```
#E2 - Calculate explained variance ratios
explained_variance = pca.explained_variance_ratio_

# Plot the scree plot (Elbow Rule)
plt.figure(figsize=(8, 5))
plt.plot(np.cumsum(explained_variance), marker='o')
plt.title('Cumulative Explained Variance by Principal Components')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.grid()
plt.show()

# Save the explained variance ratios for reporting
explained_variance_df = pd.DataFrame({
    'Principal Component': pca_columns,
    'Explained Variance Ratio': explained_variance
})
explained_variance_df.to_csv('D600_Task3_Explained_Variance.csv', index=False)

print("Explained variance ratios saved as 'D600_Task3_Explained_Variance.csv'")
```




3. Identify Variance of Principal Components

```
#E3 - Display variance of each principal component
for i, var_ratio in enumerate(explained_variance, start=1):
    print(f"PC{i}: {var_ratio:.4f} ({var_ratio * 100:.2f}%)")
✓ [9] < 10 ms

 PC1: 0.3160 (31.60%)
 PC2: 0.1652 (16.52%)
 PC3: 0.1637 (16.37%)
 PC4: 0.1590 (15.90%)
 PC5: 0.1162 (11.62%)
 PC6: 0.0799 (7.99%)
```

4. Summarize the Results of PCA
    - The results indicate that the first five components capture most of the dataset's information. By focusing the regression analysis on these components, we simplify the model while retaining most of the variance.

F. Perform the Data Analysis

1. Split Dataset

   The data was split into
    - Training Set: 80% of the data: 5600 samples
    - Test Set: 20% of the data: 1400 samples

```
#F1 - Split the Data Set add back in the Dependent 'Price' Variable
# Determine the number of components to retain dynamically
cumulative_variance = np.cumsum(explained_variance)
n_components = np.argmax(cumulative_variance >= 0.85) + 1  # Retain enough to explain 85% variance
print(f"Number of components selected: {n_components}")

# Add back the dependent variable 'Price'
df_pca['Price'] = df['Price']

# Select the principal components and add 'Price'
selected_pca_columns = [f"PC{i+1}" for i in range(n_components)]
df_pca_selected = df_pca[selected_pca_columns + ['Price']]

# Split the data into training (80%) and testing (20%)
train_data, test_data = train_test_split(df_pca_selected, test_size=0.2, random_state=42)

# Save the datasets as CSV files
train_data.to_csv('D600_Task3_Training_Dataset.csv', index=False)
test_data.to_csv('D600_Task3_Test_Dataset.csv', index=False)

print("Training and test datasets saved as 'D600_Task3_Training_Dataset.csv' and 'D600_Task3_Test_Dataset.csv'")
✓ [10] 31ms
Number of components selected: 5
Training and test datasets saved as 'D600_Task3_Training_Dataset.csv' and 'D600_Task3_Test_Dataset.csv'
```

   The training dataset included only the top 5 principal components identified in PCA, along with the target variable (Price).

2. Use Training Dataset to Perform Regression Model / Optimize Regression Model
    - Regression Model

```
#F2 - Regression Model
# Load the training dataset
train_data = pd.read_csv('D600_Task3_Training_Dataset.csv')

# Separate the independent variables (PC1, PC2, ..., PCn) and dependent variable (Price)
X_train = train_data.drop(columns=['Price'])
y_train = train_data['Price']

# Add a constant term for the intercept
X_train = sm.add_constant(X_train)

# Initial regression model
model = sm.OLS(y_train, X_train).fit()

print("Initial Model Summary:")
print(model.summary())
```

```
Initial Model Summary:Initial Model Summary:Initial Model Summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.405
Model:                            OLS   Adj. R-squared:                  0.404
Method:                 Least Squares   F-statistic:                     760.8
Date:                Thu, 30 Jan 2025   Prob (F-statistic):               0.00
Time:                        10:44:01   Log-Likelihood:                -73285.
No. Observations:                5600   AIC:                         1.466e+05
Df Residuals:                    5594   BIC:                         1.466e+05
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3.079e+05   1561.069    197.242      0.000    3.05e+05    3.11e+05
PC1         6.493e+04   1131.058     57.405      0.000    6.27e+04    6.71e+04
PC2         2.707e+04   1566.774     17.276      0.000     2.4e+04    3.01e+04
PC3         4852.2657   1576.360      3.078      0.002    1761.989    7942.542
PC4        -1047.4593   1601.850     -0.654      0.513   -4187.708    2092.789
PC5         2.588e+04   1874.091     13.808      0.000    2.22e+04    2.96e+04
==============================================================================
Omnibus:                      501.278   Durbin-Watson:                   1.991
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              672.187
Skew:                           0.749   Prob(JB):                    1.09e-146
Kurtosis:                       3.796   Cond. No.                         1.66
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

- Optimization Model: Backward Stepwise Elimination was used to iteratively remove non-significant predictors based on p-values.

```python
#F2 Optimize the Model
# Backward elimination process
def backward_elimination(X, y, significance_level=0.05):
    X = sm.add_constant(X)  # Ensure constant term is included
    while True:
        model = sm.OLS(y, X).fit()
        max_p_value = model.pvalues.max()
        if max_p_value > significance_level:
            excluded_feature = model.pvalues.idxmax()
            print(f"Dropping '{excluded_feature}' with p-value {max_p_value:.4f}")
            X = X.drop(columns=[excluded_feature])
        else:
            break
    return model

# Optimize the regression model using backward elimination
optimized_model = backward_elimination(X_train, y_train)

# Display the summary of the optimized model
print("\nOptimized Model Summary:")
print(optimized_model.summary())
```

```
Dropping 'PC4' with p-value 0.5132

Optimized Model Summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.405
Model:                            OLS   Adj. R-squared:                  0.404
Method:                 Least Squares   F-statistic:                     951.0
Date:                Thu, 30 Jan 2025   Prob (F-statistic):               0.00
Time:                        10:44:01   Log-Likelihood:                -73285.
No. Observations:                5600   AIC:                         1.466e+05
Df Residuals:                    5595   BIC:                         1.466e+05
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3.079e+05   1560.927    197.266      0.000    3.05e+05    3.11e+05
PC1         6.493e+04   1130.992     57.406      0.000    6.27e+04    6.71e+04
PC2         2.708e+04   1566.598     17.285      0.000     2.4e+04    3.02e+04
PC3         4860.6668   1576.227      3.084      0.002    1770.651    7950.683
PC5         2.587e+04   1873.948     13.804      0.000    2.22e+04    2.95e+04
==============================================================================
Omnibus:                      501.214   Durbin-Watson:                   1.991
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              672.128
Skew:                           0.749   Prob(JB):                    1.12e-146
Kurtosis:                       3.796   Cond. No.                         1.66
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

- Optimized Model Metrics:

```
Extracted Model Parameters:
R-squared: 0.4047256171061463
Adjusted R-squared: 0.40430004113982365
F-statistic: 951.0067511643078
Probability F-statistic: 0.0
Coefficient Estimates:
const    307918.186286
PC1       64926.004573
PC2       27078.973234
PC3        4860.666762
PC5       25868.182203
dtype: float64
P-values:
const    0.000000e+00
PC1      0.000000e+00
PC2      2.938819e-65
PC3      2.054152e-03
PC5      1.196257e-42
dtype: float64
Optimized model summary saved as 'D600_Task3_Optimized_Model_Summary.txt'
```

3. Mean Squared Error (MSE) of Optimized Model (Training Set)
    - Training MSE: 13627391442.97 ($116,701.53)

```
#F3 - Predict on the training set using the optimized model
y_train_pred = optimized_model.predict(X_train[['const', 'PC1', 'PC2', 'PC3', 'PC5']])

# Calculate Mean Squared Error (MSE)
mse_train = mean_squared_error(y_train, y_train_pred)

print(f"Mean Squared Error (MSE) on Training Set: {mse_train:.2f}")
✓ [14] < 10 ms
 Mean Squared Error (MSE) on Training Set: 13627391442.97
```

4. Prediction on Test Dataset using Optimized Regression Model
    - Test MSE: 12569053758.36 ($112,080.67)
    - R-squared Accuracy: 0.4061 (40.61%)

These results indicate that the model generalizes well and provides a reliable baseline for predicting housing prices.

```
#F4 - Predict on the Test Set using Optimized Model
# Load the test dataset
test_data = pd.read_csv('D600_Task3_Test_Dataset.csv')

# Separate independent variables (PC1, PC2, PC3, PC5) and dependent variable (Price)
X_test = test_data[['PC1', 'PC2', 'PC3', 'PC5']]
X_test = sm.add_constant(X_test)
y_test = test_data['Price']

# Predict on the test set using the optimized model
y_test_pred = optimized_model.predict(X_test)

# Calculate Mean Squared Error (MSE) on the test set
mse_test = mean_squared_error(y_test, y_test_pred)

from sklearn.metrics import r2_score

# Calculate R-squared for the test set
r2_test = r2_score(y_test, y_test_pred)

print(f"Mean Squared Error (MSE) on Test Set: {mse_test:.2f}")
print(f"R-squared (Accuracy) on Test Set: {r2_test:.4f}")
✓ [15] < 10 ms
 Mean Squared Error (MSE) on Test Set: 12569053758.36
 R-squared (Accuracy) on Test Set: 0.4061
```

G. Summarize Data Analysis

1. Packages and Libraries Used
    - Pandas: Data manipulation and preprocessing
    - Numpy: For mathematical computations
    - Sklearn: PCA, scaling, and regression
    - Statsmodels: Regression modeling and statistical summaries
    - Matplotlib: Screen plot visualization

2. Method used to Optimize Model

   Backward stepwise elimination is a systematic method of optimizing a regression model by iteratively removing predictors that do not meet a defined statistical significance threshold. This approach starts with all predictors included in the model and sequentially eliminates the predictor with the highest p-value until all remaining predictors are statistically significant.

   - Ensures Statistical Significance by retaining only predictors with p=values <= 0.05, the model focuses on relationships that are statistically meaningful, improving its reliability.
   - Improves model interpretability by removing non-significant predictors simplifies the model, making it easier to interpret and apply
   - Balances complexity and performance by avoiding overfitting and excluding redundant or irrelevant predictors while preserving predictive power.

3. Assumptions Used to Optimize Model
   - Linearity: Verified through PCA transformation
   - Multicollinearity: Addressed using PCA
   - Autocorrelation: Checked using the Durbin-Watson test (statistic ~2 indicates no autocorrelation)

4. Regression Equation and Coefficient Estimates

   $Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$

   - Y is equal to the outcome of the value, which is the price
   - $\beta_0$ : is equal to 307,918 which is the intercept/constant, or price when all the other variables are 0
   - $\beta_1$ : 64,926 is the coefficient of PC1 ( $x_1$ )
   - $\beta_2$ : 27,079 is the coefficient of PC2 ( $x_2$ )
   - $\beta_3$ : 4,861 is the coefficient of PC3 ( $x_3$ )
   - $\beta_4$ : 25,868 is the coefficient of PC5 ( $x_4$ )

   This means the final equation for our model is:

   $Y = 307{,}918 + 64{,}926(x_1) + 27{,}079(x_2) + 4{,}861(x_3) + 25{,}868(x_4)$

   - PC1: A one-unit increase in PC1 corresponds to an increase of $64,926 in the housing price. Since PC1 is highly influenced by SquareFootage and

RenovationQuality, larger and well-renovated homes significantly increase home prices

- PC2: A one-unit increase in PC2 increases the price by $27,079. This component is strongly correlated with CrimeRate and SchoolRating, meaning neighborhoods with better schools and lower crime rates contribute to higher home values
- PC3: A one-unit increase in PC3 has a smaller impact ($4,861 increase in price). This component might relate to BackyardSpace and AgeOfHome, meaning these factors have less direct influence on pricing.
- PC5: A one-unit increase in PC5 leads to a $25,868 price increase, which suggests a secondary influence of features like CrimeRate and RenovationQuality.

5. Model Metrics:

   $R^2$ :

   - $R^2$ = 0.405 (40.5%)
   - Meaning: $R^2$ represents the proportion of variance in housing prices explained by the selected principal components.
   - Interpretation:
     - The model explains 40.5% of price variance
     - While this is moderate predictive strength, it suggests that other factors also influence housing prices (e.g., location, market trends)
     - Since $R^2$ is not too high, this indicates that our model captures some, but not all, of the price variability

   Adjusted $R^2$ :

   - Adjusted $R^2$ = 0.404 (40.4%)
   - Meaning: Adjusted $R^2$ accounts for the number of predictors (principal components), penalizing excessive variables to prevent overfitting.
   - Interpretation:
     - Since Adjusted $R^2$ is very close to $R^2$ (40.4% vs 40.5%), it means that all included PCs contribute meaningfully to the model
     - If Adjusted $R^2$ were much lower, it would indicate some PCs were unnecessary, but this is not true.

   Mean Squared Error (MSE):

   - Meaning of MSE: MSE represents the average squared difference between actual and predicted prices. Lower MSE means better predictions.

- RMSE (Root Mean Squared Error): The square root of MSE, which gives an error estimate in the same units as the dependent variable (dollars).

Comparing Training vs Test Performance:

- Training MSE: $116,736.42 vs Test MSE: $112,111.79
    - The test MSE is slightly lower than the training MSE
    - This suggests that the model generalizes well and is not overfitting
- Overfitting Check:
    - If the training MSE was much lower than the test MSE, the model would be overfitting (too dependent on training data)
    - If test MSE was much lower, the model would be underfitting (not learning enough from the data).
    - Since training and test MSE are close, our model is balanced and likely to perform similarly on new data.

6. Results and Implications of Prediction Analysis

The final optimized regression model explains 40.5% of the variance in housing prices based on principal components derived from five key property features. While the model provides meaningful insights, it also highlights the importance of additional factors beyond the dataset.

Key Findings from the Model:

- The regression model retained four principal components (PC1, PC2, PC3, and PC5) after removing PC4, which was not statistically significant. The impact of these retained components on housing prices is as follows:
    - PC1 ($64,926 per unit increase) – Most Significant Factor
        - Strongly correlated with SquareFootage and RenovationQuality
        - Larger, well-renovated homes tend to have higher prices
    - PC2 ($27,079 per unit increase)
        - Correlated with CrimeRate and SchoolRating
        - Lower crime rates and higher school ratings lead to higher home values
    - PC3 ($4,861 per unit increase)
        - Related to BackyardSpace and AgeOfHome
        - Backyard space and older homes have a smaller influence on pricing
    - PC5 ($25,868 per unit increase)
        - A mix of CrimeRate and RenovationQuality

- Suggests that crime perception and home upgrades still influence pricing

7. Recommendations:
   The regression model explains 40.5% of housing price variations, highlighting key factors that real estate investors, developers, and homeowners should consider. While the model provides valuable insights, its limitations suggest additional factors should be included in future analysis.

   - Strategic Real Estate Pricing and Investment
     - Focus on Larger, Well-Renovated Homes:
       - Justification: PC1 (SquareFootage & RenovationQuality) has the strongest impact on price ($64,926 per unit increase).
       - Actionable Advice:
         - Sellers should highlight home size and recent renovations when marketing properties
         - Real Estate developers should invest in remodeling older properties to increase their values

     - Target High-Ranking School Districts and Safe Neighborhoods:
       - Justification: PC2 (CrimeRate & SchoolRating) is a major price driver ($27,079 per unit increase)
       - Actionable Advice:
         - Investors should prioritize homes in areas with top-rated schools and low crime rates
         - Marketing should emphasize safety and education quality as key selling points

     - Promote Backyard Space & Curb Appeal Cautiously:
       - Justification: PC3 (BackyardSpace & AgeOfHome) has a small impact on the price ($4,861 per unit increase)
       - Actionable Advice:
         - While outdoor space and curb appeal matter, they are less critical than square footage and renovations.
         - Homeowners should balance renovation budgets between interior upgrades and outdoor improvements

- Improve Home Security & Crime Perception:
    - Justification: PC5 (CrimeRate & RenovationQuality) influences pricing ($25, 868 per unit increase)
    - Actionable Advice:
        - Gated communities and security enhancements can increase perceived home value
        - Realtors should highlight local crime reduction efforts when marketing homes.

- Model Limitations & Future Enhancements
    - Limitation 1: Model Explains Only 40.5% of Price Variations
        - Justification: $R^2$ = 40.5%, meaning other factors significantly influence prices
        - Future Improvement:
            - Incorporate additional variables such as economic trends, proximity to amenities, and property age

    - Limitation 2: Linear Regression May Not Capture Complex Pricing Trends
        - Justification: The model assumes linear relationships, but housing prices are influenced by nonlinear market trends
        - Future Improvement:
            - Explore machine learning models (e.g., Random Forest, Gradient Boosting) for better predictions.