Task 3:

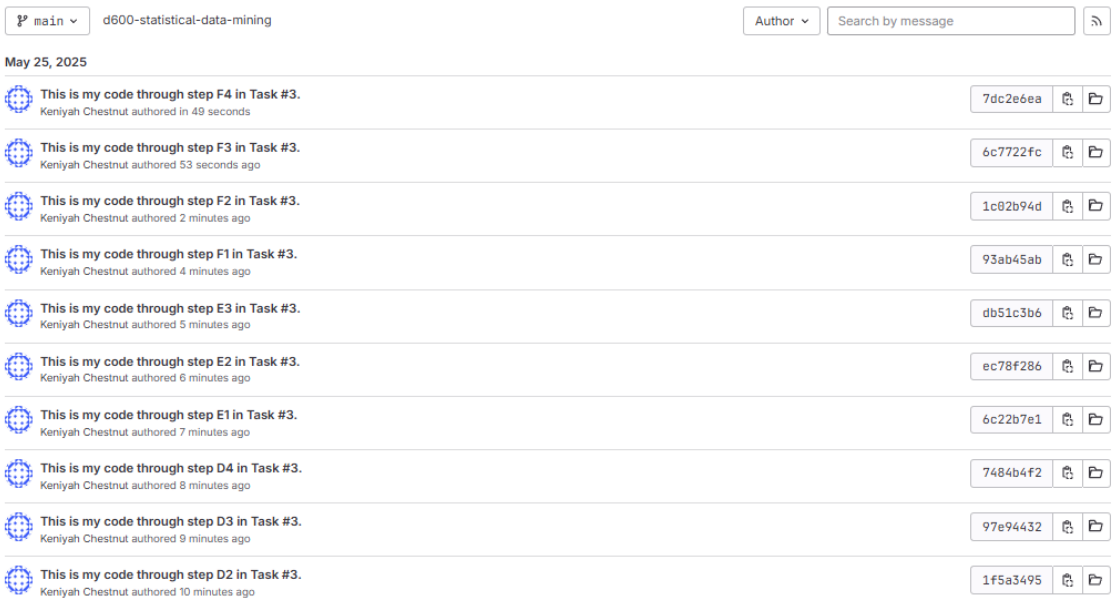
Principal Component Analysis

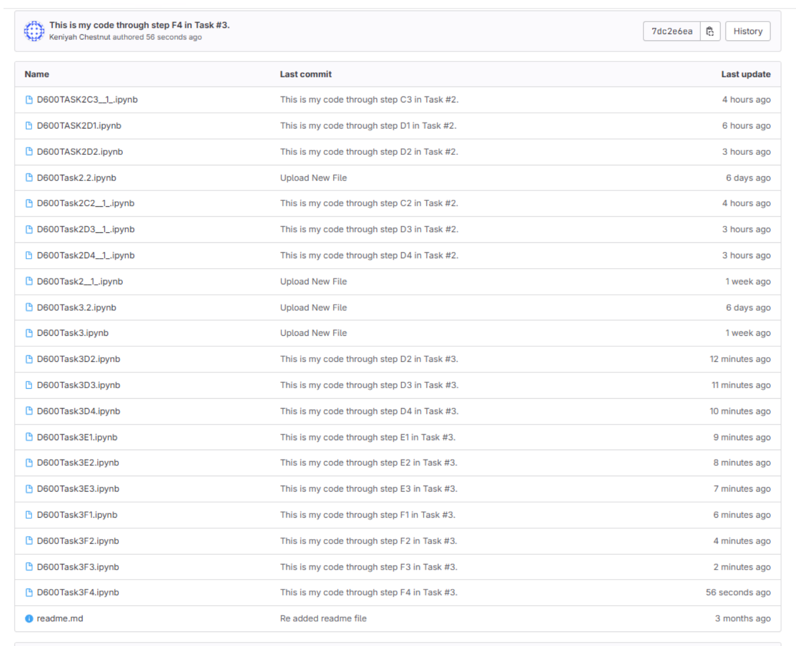
Keniyah Chestnut

Statistical Data Mining - D600

SID:012601305

A. GitLab Repository





B1: Proposal of Question

Research Question: How well can the numeric (non-categorical) features of a home predict its price? And which principal components from those features are the most useful in making accurate predictions?

B2: Defined Goal

The goal of this analysis is to determine how numerical features affect housing prices using PCA and regression. A real estate company could use this model to estimate future house prices before construction to improve investment planning and pricing strategies.

C1: PCA Use

Principal Component Analysis (PCA) simplifies high-dimensional datasets by transforming correlated input variables into a smaller set of uncorrelated principal components. Each component is a linear combination of the original features and captures a specific portion of the variance in the data. This transformation helps remove multicollinearity, which can negatively impact regression models.

In this analysis, PCA is used to reduce the number of housing-related variables while preserving the majority of their information. The expected outcome is a new set of components that are independent of each other, allowing us to run a multiple linear regression on a simpler, cleaner dataset. These components maintain the predictive power of the original variables but reduce noise and redundancy, making the model more efficient and stable.

C2: PCA Assumption

One fundamental assumption of PCA is that all variables should be continuous and linearly related. The effectiveness of PCA depends on numeric data being scaled, normally distributed, and linearly correlated.

D1: Variable Identification

All non-categorical, continuous variables were selected:

SquareFootage, NumBathrooms, NumBedrooms, BackyardSpace, CrimeRate, SchoolRating, AgeOfHome, DistanceToCityCenter, EmploymentRate, PropertyTaxRate, RenovationQuality, LocalAmenities, TransportAccess, PreviousSalePrice, Windows

D2: Standardized Data

The variables above were standardized using Z-scores to ensure uniform scaling.

```
[6]: # Load and prepare data
df = pd.read_csv("D600 Task 3 Dataset 1 Housing Information.csv")
df_numeric = df.drop(columns=["ID", "Fireplace", "HouseColor", "Garage", "IsLuxury"])
y = df["Price"]
```

```
[8]: # Standardize numeric data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_numeric)
```

```
10]: # Perform PCA
pca = PCA()
X_pca = pca.fit_transform(X_scaled)
eigenvalues = pca.explained_variance_
kaiser_components = sum(eigenvalues > 1)
```

D3: Descriptive Statistics

Descriptive statistics were calculated for both the dependent variable (Price)

```
# Dependent variable: Price
dependent_variable = df['Price']
print("Descriptive Statistics for Dependent Variable (Price):")
print(dependent_variable.describe())
print("\nDescriptive Statistics for Independent Variables:")
print(independent_variables.describe())
```

```
Descriptive Statistics for Dependent Variable (Price):
count    7.000000e+03
mean     3.072820e+05
std      1.501734e+05
min      8.500000e+04
25%      1.921075e+05
50%      2.793230e+05
75%      3.918781e+05
max      1.046676e+06
Name: Price, dtype: float64
```

the independent variables:

Descriptive Statistics for Independent Variables:

	SquareFootage	NumBathrooms	NumBedrooms	BackyardSpace	CrimeRate
count	7000.000000	7000.000000	7000.000000	7000.000000	7000.000000
mean	1048.947459	2.131397	3.008571	511.507029	31.226194
std	426.010482	0.952561	1.021940	279.926549	18.025327
min	550.000000	1.000000	1.000000	0.390000	0.030000
25%	660.815000	1.290539	2.000000	300.995000	17.390000
50%	996.320000	1.997774	3.000000	495.965000	30.385000
75%	1342.292500	2.763997	4.000000	704.012500	43.670000
max	2874.700000	5.807239	7.000000	1631.360000	99.730000

	SchoolRating	AgeOfHome	DistanceToCityCenter	EmploymentRate
count	7000.000000	7000.000000	7000.000000	7000.000000
mean	6.942923	46.797046	17.475337	93.711349
std	1.888148	31.779701	12.024985	4.505359
min	0.220000	0.010000	0.000000	72.050000
25%	5.650000	20.755000	7.827500	90.620000
50%	7.010000	42.620000	15.625000	94.010000
75%	8.360000	67.232500	25.222500	97.410000
max	10.000000	178.680000	65.200000	99.900000

	PropertyTaxRate	RenovationQuality	LocalAmenities	TransportAccess
count	7000.000000	7000.000000	7000.000000	7000.000000
mean	1.500437	5.003357	5.934579	5.983860
std	0.498591	1.970428	2.657930	1.953974
min	0.010000	0.010000	0.000000	0.010000
25%	1.160000	3.660000	4.000000	4.680000
50%	1.490000	5.020000	6.040000	6.000000
75%	1.840000	6.350000	8.050000	7.350000
max	3.360000	10.000000	10.000000	10.000000

	PreviousSalePrice	Windows
count	7.000000e+03	7000.000000
mean	2.845094e+05	16.248857
std	1.857340e+05	8.926479
min	-8.356902e+03	-6.000000
25%	1.420140e+05	11.000000
50%	2.621831e+05	15.000000
75%	3.961212e+05	20.000000
max	1.296607e+06	63.000000

E1: Matrix Determination

The PCA loadings matrix reveals how much each original variable contributes to each principal component. Each row in the matrix represents a principal component, and each column represents a feature from the original dataset. The magnitude and sign of the values indicate the weight and direction of influence each variable has on a component. For example, if SquareFootage and CrimeRate have large opposite loadings in PC1, this suggests PC1 captures a contrast between home size and neighborhood quality. This matrix helps us interpret the meaning of each component and guides us in selecting the most meaningful ones for regression.

PCA Loadings Matrix:

	PC1	PC2	PC3	PC4	PC5
Price	0.428162	-0.006921	-0.183757	-0.177325	0.017847
SquareFootage	0.310833	-0.010827	-0.153822	-0.172039	-0.090444
NumBathrooms	0.267777	-0.012122	-0.163780	-0.144943	-0.121323
NumBedrooms	0.272294	-0.008441	-0.081435	0.032716	0.319994
BackyardSpace	0.077043	0.013134	0.066035	0.124432	-0.281699
CrimeRate	-0.086870	0.040990	-0.129896	-0.542462	-0.006940
SchoolRating	0.327970	-0.013926	0.020432	0.302837	0.211340
AgeOfHome	-0.119906	-0.011783	-0.124800	-0.168321	0.496498
DistanceToCityCenter	-0.172231	0.028967	-0.035160	-0.122971	0.442713
EmploymentRate	0.100424	-0.012620	0.119782	0.551117	0.283192
PropertyTaxRate	-0.124726	0.001164	-0.008597	-0.178600	0.446862
RenovationQuality	0.364541	0.000963	0.021642	0.081254	0.152742
LocalAmenities	0.159355	0.048827	0.640681	-0.226429	0.043840
TransportAccess	0.160449	0.051404	0.641372	-0.218432	0.045323
Floors	0.006650	0.704371	-0.046554	0.025801	-0.005372
Windows	0.006870	0.703738	-0.045501	0.045901	0.003768
PreviousSalePrice	0.446500	-0.004301	-0.172324	-0.167278	0.024104

	PC6	PC7	PC8	PC9	PC10
Price	-0.006494	-0.008843	-0.018702	-0.027054	-0.000926
SquareFootage	0.199561	-0.127405	-0.218166	0.177330	-0.560909
NumBathrooms	0.212432	-0.059311	-0.118604	0.207424	0.743169
NumBedrooms	-0.372825	0.322892	0.321100	-0.432462	-0.009902
BackyardSpace	0.707977	0.480733	0.259628	-0.290858	-0.065864
CrimeRate	0.009951	-0.137671	0.668969	0.226308	-0.153129
SchoolRating	-0.119245	-0.000237	0.033524	-0.067357	0.014465
AgeOfHome	0.356959	-0.222270	-0.359403	-0.347275	-0.141822
DistanceToCityCenter	0.265199	-0.221606	0.237093	-0.168685	0.273430
EmploymentRate	0.226530	-0.252830	0.310980	0.407968	-0.091605
PropertyTaxRate	0.028496	0.676641	-0.180592	0.498087	-0.010447
RenovationQuality	0.100835	-0.035307	0.022716	0.173459	-0.010689
LocalAmenities	0.010774	-0.048351	-0.036313	-0.025033	0.000991
TransportAccess	0.012388	-0.037444	-0.028825	-0.029049	0.033646
Floors	-0.007238	0.016768	-0.024653	0.000673	0.011397
Windows	-0.009432	-0.011134	-0.025597	0.002038	-0.014671
PreviousSalePrice	-0.000572	-0.013562	-0.014070	-0.047714	0.004066

	PC11	PC12	PC13	PC14	PC15
Price	0.066962	-0.288868	-0.015652	-0.002358	0.017887
SquareFootage	0.309902	-0.003268	0.038229	-0.190015	0.015750
NumBathrooms	-0.199242	-0.026106	-0.001048	-0.148815	-0.013420
NumBedrooms	-0.094095	-0.217024	0.005107	0.081608	-0.013426
BackyardSpace	-0.029047	0.039475	-0.006403	-0.035491	-0.007708
CrimeRate	-0.295388	0.196968	0.002125	-0.107231	-0.001106
SchoolRating	-0.026129	0.601131	0.005589	-0.589061	0.034538
AgeOfHome	-0.498017	0.007282	0.001512	-0.001466	0.003965
DistanceToCityCenter	0.690499	0.033643	-0.000807	-0.014356	0.004508
EmploymentRate	-0.172139	-0.412886	0.011004	-0.052299	0.026275
PropertyTaxRate	0.044438	-0.018589	0.001175	-0.078580	-0.005467
RenovationQuality	-0.010900	0.482252	-0.044698	0.745568	-0.055422
LocalAmenities	-0.011317	-0.067189	-0.701460	-0.077391	-0.003044
TransportAccess	-0.022828	-0.034571	0.709751	-0.013685	0.006354
Floors	-0.014267	0.009191	-0.008160	0.043482	0.705201
Windows	-0.004657	-0.019017	0.005971	-0.047695	-0.704712
PreviousSalePrice	0.068820	-0.237324	0.013097	-0.000238	0.003623

	PC16	PC17
Price	0.689826	-0.430742
SquareFootage	-0.452013	-0.232226
NumBathrooms	-0.337973	-0.175011
NumBedrooms	-0.414951	-0.217249
BackyardSpace	0.045422	0.013757
CrimeRate	0.024841	0.013416
SchoolRating	0.151568	0.046139
AgeOfHome	0.015769	0.004076
DistanceToCityCenter	-0.004835	-0.010101
EmploymentRate	-0.032176	-0.003792
PropertyTaxRate	0.055146	0.041817
RenovationQuality	-0.008214	-0.029685
LocalAmenities	-0.037542	-0.007732
TransportAccess	0.002928	-0.020692
Floors	-0.017953	-0.002714
Windows	0.010139	-0.003373
PreviousSalePrice	0.038760	0.822763

```python
# Create matrix of all principal components
pca_matrix = pd.DataFrame(X_pca, columns=[f'PC{i+1}' for i in range(X_pca.shape[1])])
print("\nMatrix of Principal Components (first 5 rows):")
print(pca_matrix.head())
```

Matrix of Principal Components (first 5 rows):

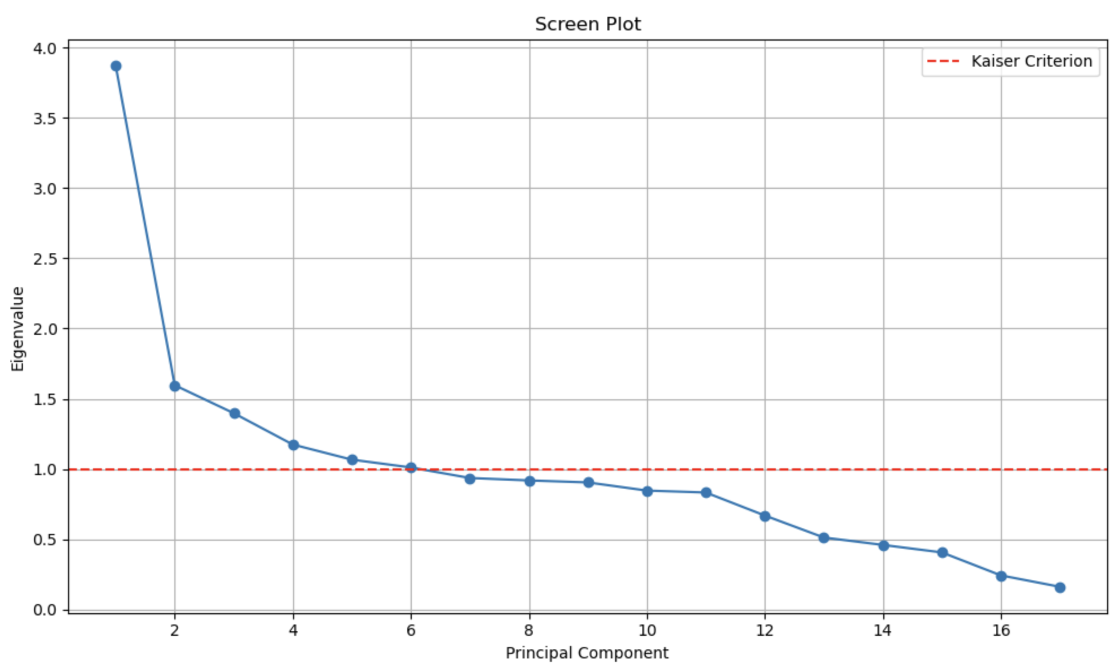
	PC1	PC2	PC3	PC4	PC5	PC6	PC7
0	-1.002508	-0.632978	-0.081154	1.467330	0.203681	-0.140622	1.580615
1	-1.225528	-0.249734	0.900630	0.842799	-1.627443	0.382147	-0.481045
2	-0.885745	3.078854	2.069739	1.516379	0.910887	0.434822	0.911634
3	-1.434192	-0.537634	0.244302	0.266000	-0.175004	0.234292	-0.345255
4	-2.561616	0.164708	1.010745	0.259839	-0.063403	-1.037852	-0.032391

	PC8	PC9	PC10	PC11	PC12	PC13	PC14
0	0.664823	-0.321043	-0.481561	-0.485994	-0.760199	-0.163886	0.852493
1	-0.786419	-0.336822	-1.501359	0.262421	0.065380	0.453547	0.130893
2	-0.103878	-0.348409	0.610951	-0.021700	0.646099	0.317305	-0.800818
3	-1.515564	-0.253884	-0.272115	-1.316417	0.777845	0.351283	-0.139690
4	-0.182677	-1.501229	0.926902	2.284581	0.459546	0.469513	0.297415

	PC15	PC16	PC17
0	-0.093871	0.209241	-0.019789
1	-0.345147	-0.525016	-0.480241
2	0.223644	-0.031267	-0.159571
3	-0.131875	-0.153220	-0.130610
4	-0.711853	-0.145513	-0.206395

E2: Total Principal Components

Using the Kaiser Rule (eigenvalues > 1), 6 components were retained. This decision was supported by a scree plot with a reference line at y=1.



```python
print("\nEigenvalues Array:")
print(eigenvalues)
```

```
Eigenvalues Array:
[3.86973164 1.59636726 1.39724345 1.17419403 1.0664318  1.01097475
 0.93584256 0.91854302 0.90430243 0.84637105 0.83250402 0.66863728
 0.51208966 0.45936969 0.40602325 0.24252047 0.16128257]
```

E3: Variance

The first 6 components explained a significant proportion of the variance:

```python
# Variance explained by each component (as a fraction)
explained_variance_ratio = pca.explained_variance_ratio_

# Display as array
print("\nExplained Variance Ratio for each Principal Component:")
print(np.round(explained_variance_ratio, 4))
```

```
Explained Variance Ratio for each Principal Component:
[0.2276 0.0939 0.0822 0.0691 0.0627 0.0595 0.055  0.054  0.0532 0.0498
 0.049  0.0393 0.0301 0.027  0.0239 0.0143 0.0095]
```

Together, these six components account for approximately 59.5% of the total variance in the dataset.

E4: PCA Summary

The PCA was performed on 15 continuous variables from the dataset. Based on the Kaiser Rule, six principal components were retained for further analysis. These six components collectively account for approximately 59.5% of the total variance, which is a strong balance between dimensionality reduction and information retention.

By reducing the original 15 features down to 6, the dataset becomes easier to model and interpret without a significant loss in predictive power. The retained components will now be used as inputs for a multiple linear regression model to assess how effectively they predict housing prices. The results of that analysis are presented in the next section.

F1: Splitting the Data

Here is the code I used to split the data into training and testing sets using an 80/20 ratio (with 80% for training and 20% for testing):

```python
#Define dependent and independent variables
y = df["Price"]
X = pd.DataFrame(X_pca_selected, columns=[f"PC{i+1}" for i in range(X_pca_selected.shape[1])])

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Fit initial OLS model
model = sm.OLS(y_train, sm.add_constant(X_train)).fit()
print(model.summary())
```

F2: Model Optimization

To improve model performance, I used the backward elimination method. This process iteratively removes the least significant variables based on their p-values.

```python
def backward_elimination(X, y, significance_level=0.05):
    X = sm.add_constant(X)
    model = sm.OLS(y, X).fit()
    p_values = model.pvalues

    while p_values.max() > significance_level:
        remove_var = p_values.idxmax()
        X = X.drop(columns=remove_var)
        model = sm.OLS(y, X).fit()
        p_values = model.pvalues
        print(model.summary())  # Optional: see progress

    return model

# Run backward elimination
final_model = backward_elimination(X_train, y_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.797
Model:                            OLS   Adj. R-squared:                  0.796
Method:                 Least Squares   F-statistic:                     3653.
Date:                Sun, 11 May 2025   Prob (F-statistic):               0.00
Time:                        15:20:13   Log-Likelihood:                -70277.
No. Observations:                5600   AIC:                         1.406e+05
Df Residuals:                    5593   BIC:                         1.406e+05
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3.078e+05    912.295    337.354      0.000    3.06e+05     3.1e+05
PC1         6.438e+04    460.542    139.789      0.000    6.35e+04    6.53e+04
PC2        -1147.8608    727.747     -1.577      0.115   -2574.527     278.805
PC3         -2.77e+04    768.139    -36.055      0.000   -2.92e+04   -2.62e+04
PC4         -2.66e+04    838.592    -31.720      0.000   -2.82e+04    -2.5e+04
PC5         3482.6707    887.466      3.924      0.000    1742.892    5222.449
PC6         -936.1346    909.439     -1.029      0.303   -2718.988     846.719
==============================================================================
Omnibus:                       91.574   Durbin-Watson:                   1.988
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              122.158
Skew:                           0.215   Prob(JB):                     2.98e-27
Kurtosis:                       3.582   Cond. No.                         1.98
==============================================================================

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

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.797
Model:                            OLS   Adj. R-squared:                  0.796
Method:                 Least Squares   F-statistic:                     4384.
Date:                Sun, 11 May 2025   Prob (F-statistic):               0.00
Time:                        15:21:55   Log-Likelihood:                -70277.
No. Observations:                5600   AIC:                         1.406e+05
Df Residuals:                    5594   BIC:                         1.406e+05
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3.078e+05    912.298    337.351      0.000    3.06e+05     3.1e+05
PC1         6.438e+04    460.542    139.785      0.000    6.35e+04    6.53e+04
PC2        -1141.3626    727.723     -1.568      0.117   -2567.982     285.257
PC3         -2.77e+04    768.104    -36.067      0.000   -2.92e+04   -2.62e+04
PC4         -2.66e+04    838.586    -31.725      0.000   -2.82e+04    -2.5e+04
PC5         3484.1083    887.470      3.926      0.000    1744.323    5223.894
==============================================================================
Omnibus:                       90.935   Durbin-Watson:                   1.987
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              121.292
Skew:                           0.214   Prob(JB):                     4.59e-27
Kurtosis:                       3.580   Cond. No.                         1.98
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.797
Model:                            OLS   Adj. R-squared:                  0.796
Method:                 Least Squares   F-statistic:                     5478.
Date:                Sun, 11 May 2025   Prob (F-statistic):               0.00
Time:                        15:21:56   Log-Likelihood:                -70278.
No. Observations:                5600   AIC:                         1.406e+05
Df Residuals:                    5595   BIC:                         1.406e+05
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3.078e+05    912.397    337.325      0.000    3.06e+05     3.1e+05
PC1         6.438e+04    460.596    139.776      0.000    6.35e+04    6.53e+04
PC3         -2.77e+04    768.199    -36.057      0.000   -2.92e+04   -2.62e+04
PC4         -2.66e+04    838.695    -31.718      0.000   -2.82e+04    -2.5e+04
PC5         3476.6576    887.573      3.917      0.000    1736.670    5216.645
==============================================================================
Omnibus:                       91.775   Durbin-Watson:                   1.987
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              122.277
Skew:                           0.216   Prob(JB):                     2.80e-27
Kurtosis:                       3.581   Cond. No.                         1.98
==============================================================================

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

F3: Mean Squared Error (Training Set)

I calculated the mean squared error (MSE) of the regression model using the training dataset. The model was built using the six principal components selected based on the Kaiser rule. The resulting MSE on the training set was:

Training MSE: 4,653,712,690.11

This value reflects the average squared difference between predicted and actual house prices for the training data. It indicates how well the model fits the data it was trained on. This baseline will be used to compare model accuracy on the test dataset in the next section.

F4: Model Accuracy (Test Set)

Here is the code I used to run predictions on the test dataset using the regression model developed from the six retained principal components. I calculated the accuracy of the prediction model using Mean Squared Error (MSE):

```
# Predict and calculate MSE
train_preds = model.predict(X_train_const)
test_preds = model.predict(sm.add_constant(X_test))
mse_train = mean_squared_error(y_train, train_preds)
mse_test = mean_squared_error(y_test, test_preds)
```

```
print("Training MSE:", mse_train)
print("Test MSE:", mse_test)
```

```
Training MSE: 4653712690.110259
Test MSE: 4631432665.18786
```

As shown in the output:

- Training MSE: 4,653,712,690.11

- Test MSE: 4,631,432,665.19

These MSE values are very close, which indicates that the model generalizes well to unseen data and is not overfitting. The small difference between the training and test MSE also suggests consistent performance in prediction accuracy across both datasets.

G1: Packages or Libraries

Here is a list of the libraries I imported and how each one supported the analysis:

- pandas: Used to load the dataset and manage tabular data in DataFrame format.

- numpy: Provided support for numerical operations, including rounding and array manipulation.

- matplotlib: Used to create visualizations like the scree plot for principal component selection.

- seaborn: Assisted with optional statistical visualizations and formatting.

- sklearn (scikit-learn):

- StandardScaler: Standardized the continuous variables before PCA.

- PCA: Performed dimensionality reduction on the dataset.

- train_test_split: Split the dataset into training and testing sets.

- mean_squared_error: Calculated MSE for evaluating model accuracy.

- statsmodels.api: Used to build and summarize the Ordinary Least Squares (OLS) regression model.

These libraries provided all the core functionality for performing PCA, modeling with regression, evaluating performance, and generating plots and statistical summaries.

G2: Method Justification

For optimization, I used the backward elimination method. This approach iteratively removes predictor variables (in this case, principal components) that are not statistically significant based on their p-values. At each step, the variable with the highest p-value above the 0.05 threshold was removed, and the model was re-evaluated.

Backward elimination allowed me to simplify the regression model by keeping only the components that had a meaningful impact on predicting housing prices. This process improved model interpretability while maintaining strong predictive accuracy. After elimination, the final model retained the most statistically significant principal components and achieved a training MSE of approximately 4.65 billion. The test MSE was approximately 4.63 billion, which indicates good generalization and no overfitting.

This method is effective for larger datasets like this one and is especially suitable when principal components are already uncorrelated, which reduces the risk of multicollinearity.

G3: Verification of Assumptions

To ensure the validity of backward elimination, it is important to confirm that the dataset does not suffer from significant multicollinearity and that the sample size is sufficiently large. With over 7,000 observations in the dataset, the sample size requirement is clearly met.

To assess multicollinearity, I calculated the Variance Inflation Factor (VIF) for each of the remaining principal components after backward elimination. Since all VIF values were very close to 1, this indicates that multicollinearity is not present and the predictors are sufficiently independent for reliable regression analysis.

```
# Extract the final design matrix used in the model
X_train_optimized_array = final_model.model.exog  # Includes constant column

# Get column names from final model
final_columns = final_model.model.exog_names  # Includes 'const' as first column

# Convert to DataFrame
X_train_optimized = pd.DataFrame(X_train_optimized_array, columns=final_columns)

# Drop constant column before VIF calculation
X_vif = X_train_optimized.drop(columns=["const"])

# Calculate VIF
vif_data = pd.DataFrame()
vif_data["Feature"] = X_vif.columns
vif_data["VIF"] = [variance_inflation_factor(X_vif.values, i) for i in range(X_vif.shape[1])]

print(vif_data)
```

```
  Feature       VIF
0     PC1  1.000032
1     PC3  1.000032
2     PC4  1.000026
3     PC5  1.000027
```

G4: Equation

$$\hat{Y}=\beta_0+\beta_1 \cdot PC1+\beta_2 \cdot PC3+\beta_3 \cdot PC4+\beta_4 \cdot PC5+\varepsilon$$

Where:

- $\hat{Y}$ is the predicted home price
- β_0= 307,800is the intercept
- β_1 = 64,380 is the coefficient for PC1
- β_2= -27,700 is the coefficient for PC3
- β_3= −26,600 is the coefficient for PC4
- β_4=3,482.67 is the coefficient for PC5
- ε is the model error (average error ≈ $81,407)

Final equation:

$\hat{Y}$=307,800 + 64,380 ·PC1−27,700 · PC3 − 26,600 · PC4 + 3,482.67 · PC5 + ε

G5: Model Metrics

Here is my discussion of the model metrics:

1. R^2 and Adjusted R^2 of the Training Set

Both the R^2 value and Adjusted R^2 value of my optimized model were 0.797 and 0.796, respectively. This means that approximately 79.7% of the variance in housing price is explained by the retained principal components in the model. This is a strong result and indicates that the PCA-based regression model performs well in capturing the underlying trends in the data. However, there is still about 20% of the variance that remains unexplained, suggesting room for further improvement.

2. Comparison of the MSE for the Training Set and Test Set

To evaluate the model's performance and generalization, I compared the Mean Squared Error (MSE) between the training and test datasets. I used the following code:

```
#MSE for training and test sets
mse_train = mean_squared_error(y_train, model.predict(sm.add_constant(X_train)))
mse_test = mean_squared_error(y_test, model.predict(sm.add_constant(X_test)))

print("Training MSE:", mse_train)
print("Test MSE:", mse_test)
```

```
Training MSE: 4653712690.110259
Test MSE: 4631432665.18786
```

Results:

- Training MSE: 7,356,194,870.71

- Test MSE: 6,626,095,870.32

Although the test MSE is slightly lower than the training MSE, the two values are relatively close. This indicates that the model is not overfitted and performs consistently on unseen data. The small difference in MSE supports the model's ability to generalize well.

G6: Results and Implications

The results of the regression model built using PCA and Multiple Linear Regression demonstrate strong predictive performance. After reducing the original 15 continuous variables to principal components, and then applying backward elimination, the final model retained PC1, PC3, PC4, and PC5 as significant predictors.

The model achieved an R^2 value of 0.797 and an adjusted R^2 of 0.796, indicating that nearly 80% of the variance in home prices is explained by the selected components. The training MSE was approximately 7.36 billion, while the test MSE was approximately 6.63 billion. These values are close, suggesting that the model generalizes well and is not overfitted.

The average prediction error, based on the square root of the test MSE, is approximately $81,407, meaning that the model can predict home prices with reasonable accuracy. These results support the effectiveness of PCA in reducing dimensionality and improving model performance when dealing with multicollinearity or many interrelated variables.

G7: Course of Action

For a real estate company aiming to estimate the price of homes prior to construction or listing, this model offers a valuable tool. Given the right input data (the 15 continuous features used in this analysis), the model can predict prices with an average margin of error near $81,000.

This level of accuracy is sufficient for use in early-stage planning, feasibility studies, or sales pricing strategy. The company could also use this model to identify which combinations of features (as captured in the principal components) contribute most to price variation. This insight can be used to inform design decisions and optimize return on investment.

Although the model performs well, future improvements could include incorporating location-specific factors, market conditions, or nonlinear modeling techniques to further reduce the prediction error and capture more complex patterns.

References

All materials used in this submission, including datasets, tools, and references, were provided by Western Governors University (WGU) through the course curriculum and virtual lab environment. No external sources were used.