D600 – Statistical Data Mining

Performance Assessment #3 – Principal Component Analysis

Bernard Connelly

Master of Science, Data Analytics, Western Governors University

Dr. Keiona Middleton

February 07, 2025

D600 – Statistical Data Mining: Principal Component Analysis

Purpose of Analysis

B1. Research Question

How effectively do the quantitative variables in the dataset provided predict the price of a home by using principal component analysis?

B2. Goal of Analysis

This analysis aims to parse through the quantitative variables provided and identify which principal components will be the best predictors of price so the organization can maximize its profits. For a company whose goal is to flip houses, identifying the characteristics of a home that yield the most significant price increase will allow them to sustain their business model the most effectively. By breaking the variables into their principal components and running a multiple linear regression model, we will learn how effectively the variables affect the price and how the least amount of principal components are adequate.

Reasoning for PCA

C1. Use of PCA to Prepare Data

Principal Component Analysis is a dimensionality reduction technique employed to transform a complex series of independent variables into a smaller set of principal components that capture the maximum variance in a dataset. Doing so allows the creation of a linear regression model, which is easier to visualize since it requires fewer variables, removes irrelevant variables to reduce noise, and increases the stability overall. In a linear regression model, utilizing too many variables alone will make the model too complex and reduce the overall predictive power. By using PCA, all variables can be used simultaneously. Still, only the most relevant principal components will be maintained, which will cause a variety of benefits, including reduced multicollinearity and making the model more straightforward to interpret.

C2. Summarize One Assumption of PCA

One central assumption of principal component analysis is that all of the variables in the dataset are standardized, having a mean of 0 and variance of 1. Real-world datasets have a variety of descriptive statistic values for each independent variable. To ensure standardized data, the dataset is scaled appropriately before applying PCA so all values align. In the model created for this assignment, the Python class StandardScaler from the sklearn was utilized to accomplish this task.

Summarize Data Preparation Process

D1. Identify continuous variables required

For this assignment, categorical and true binary variables were omitted from the analysis. The only variables included are whole-number integers or continuous numerical values. As such, the variables included in analysis were square_footage, num_bathrooms, num_bedrooms, backyard_space, crime_rate, school_rating, age_of_home, distance_to_city_center, employment_rate, property_tax_rate, renovation_quality, local_amenities, transport_access, previous_sale_price, and windows.

D2. Standaradize Dataset

As previously mentioned, StandardScaler from sklearn was utilized to standardize the dataset:

```
#D2 -  Standardize the continuous dataset variables identified in part D1. Include a copy of the cleaned dataset.
#Selecting the variables to use in PCA and standardizing the dataset
pca_df = df[['square_footage', 'num_bathrooms', 'num_bedrooms', 'backyard_space', 'crime_rate',
             'school_rating', 'age_of_home', 'distance_to_city_center', 'employment_rate',
             'property_tax_rate', 'renovation_quality', 'local_amenities', 'transport_access', 'previous_sale_price', 'windows']]

df_scaled = StandardScaler().fit_transform(pca_df)
df_scaled #Confirming Scaling worked
```

```
array([[-1.1322771 , -1.16948139,  0.97021289, ..., -0.7338697 ,
        -0.55295614, -0.36988525],
       [ 0.99392575, -1.16948139, -0.9869889 , ...,  0.43306635,
        -1.26287674,  0.08115503],
       [-1.17129301, -0.13224734, -0.00838801, ...,  1.27755954,
        -1.27171985,  1.99807623],
       ...,
       [-0.05302469, -0.13224734,  1.94881378, ..., -1.54765405,
         0.51712453, -0.36988525],
       [ 2.74489976, -1.16948139,  0.97021289, ...,  0.06967837,
         0.85132615, -0.70816546],
       [ 1.18461543,  1.94222075,  0.97021289, ..., -0.40630871,
         1.1184774 ,  0.30667517]])
```

The data was then transformed into a data frame and described to ensure the mean was zero and the standard deviation was 1 for each variable.

```
#Applying PCA and converting back to a DataFrame for future manipulation
pca_df_scaled = pd.DataFrame(df_scaled,
                             columns=pca_df.columns)
pca_df_scaled.describe() #Confirming conversion worked
## All variables noted as continuous, with a mean close to 0 and a std close to 1
```

	square_footage	num_bathrooms	num_bedrooms	backyard_space	crime_rate	school_rating
count	7000.000000	7.000000e+03	7.000000e+03	7.000000e+03	7.000000e+03	7.000000e+03
mean	0.000000	6.496391e-17	1.624098e-16	-1.624098e-17	2.030122e-16	-6.171571e-16
std	1.000071	1.000071e+00	1.000071e+00	1.000071e+00	1.000071e+00	1.000071e+00
min	-1.171293	-1.169481e+00	-1.965590e+00	-1.826027e+00	-1.730810e+00	-3.560846e+00
25%	-0.911152	-6.508644e-01	-9.869889e-01	-7.520797e-01	-7.676522e-01	-6.848062e-01
50%	-0.123544	-1.322473e-01	-8.388008e-03	-5.552578e-02	-4.667069e-02	3.552791e-02
75%	0.688636	9.049867e-01	9.702129e-01	6.877491e-01	6.904005e-01	7.505654e-01
max	4.286005	4.016689e+00	3.906016e+00	4.000810e+00	3.800691e+00	1.619204e+00

	age_of_home	distance_to_city_center	employment_rate	property_tax_rate	renovation_quality	local_amenities
count	7.000000e+03	7.000000e+03	7.000000e+03	7.000000e+03	7.000000e+03	7.000000e+03
mean	-4.060244e-17	3.248195e-16	2.931496e-15	-3.491810e-16	-2.273737e-16	-2.760966e-16
std	1.000071e+00	1.000071e+00	1.000071e+00	1.000071e+00	1.000071e+00	1.000071e+00
min	-1.472336e+00	-1.453356e+00	-4.808251e+00	-2.989512e+00	-2.534329e+00	-2.232942e+00
25%	-8.195139e-01	-8.023732e-01	-6.861982e-01	-6.828474e-01	-6.818077e-01	-7.279036e-01
50%	-1.314469e-01	-1.538854e-01	6.629278e-02	-2.093478e-02	8.446918e-03	3.966581e-02
75%	6.430807e-01	6.443015e-01	8.210035e-01	6.810937e-01	6.834753e-01	7.959474e-01
max	4.150208e+00	3.969075e+00	1.373718e+00	3.729903e+00	2.535997e+00	1.529653e+00

D3. Descriptive Statistics

The following code was utilized to accurately describe the statistics of the independent variable, price, and all predictor variables in the dataset.

```
print("Independent Variables:")
display(pca_df.describe().iloc[:, :8])
display(pca_df.describe().iloc[:, 8:])

print("\n\nDependent Variable:")
display(df["price"].describe())
```

This yielded the following results:

Independent Variables:

	square_footage	num_bathrooms	num_bedrooms	backyard_space	crime_rate	school_rating	age_of_home	distance_to_city_center
count	7000.000000	7000.000000	7000.000000	7000.000000	7000.000000	7000.000000	7000.000000	7000.000000
mean	1048.947459	2.127500	3.008571	511.507029	31.226194	6.942923	46.797046	17.475337
std	426.010482	0.964171	1.021940	279.926549	18.025327	1.888148	31.779701	12.024985
min	550.000000	1.000000	1.000000	0.390000	0.030000	0.220000	0.010000	0.000000
25%	660.815000	1.500000	2.000000	300.995000	17.390000	5.650000	20.755000	7.827500
50%	996.320000	2.000000	3.000000	495.965000	30.385000	7.010000	42.620000	15.625000
75%	1342.292500	3.000000	4.000000	704.012500	43.670000	8.360000	67.232500	25.222500
max	2874.700000	6.000000	7.000000	1631.360000	99.730000	10.000000	178.680000	65.200000

	employment_rate	property_tax_rate	renovation_quality	local_amenities	transport_access	previous_sale_price	windows
count	7000.000000	7000.000000	7000.000000	7000.000000	7000.000000	7.000000e+03	7000.000000
mean	93.711349	1.500437	5.003357	5.934579	5.983860	2.845346e+05	16.280286
std	4.505359	0.498591	1.970428	2.657930	1.953974	1.856946e+05	8.869021
min	72.050000	0.010000	0.010000	0.000000	0.010000	2.200000e+01	0.000000
25%	90.620000	1.160000	3.660000	4.000000	4.680000	1.420138e+05	11.000000
50%	94.010000	1.490000	5.020000	6.040000	6.000000	2.621825e+05	15.000000
75%	97.410000	1.840000	6.350000	8.050000	7.350000	3.961210e+05	20.000000
max	99.900000	3.360000	10.000000	10.000000	10.000000	1.296606e+06	63.000000

```
Dependent Variable:
count    7.000000e+03
mean     3.072815e+05
std      1.501734e+05
min      8.500000e+04
25%      1.921075e+05
50%      2.793225e+05
75%      3.918778e+05
max      1.046675e+06
Name: price, dtype: float64
```

Principal Component Analysis

E1. Determine the Matrix of all Principal Components

The matrix of principal components can be seen below:

```python
'''Part 3: Principal Component Analysis'''

#E1 - Determine the matrix of all the principal components
##Performing PCA
pca = PCA(n_components=None)
pca.fit(df_scaled)

#Creating Loadings MAtrix
pca_loadings_matrix = pd.DataFrame(
    pca.components_.T,
    index=pca_df.columns,
    columns=[f'PC{i+1}' for i in range(pca.n_components_)])
print("\nPCA Loadings Matrix:")
print(pca_loadings_matrix.round(4))
```

```
PCA Loadings Matrix:
                           PC1     PC2     PC3     PC4     PC5     PC6  \
square_footage          0.3248  0.1567  0.2474 -0.0723  0.1742 -0.0904
num_bathrooms           0.2816  0.1732  0.2267 -0.1027  0.1656 -0.1340
num_bedrooms            0.2906  0.1115 -0.0007  0.3183 -0.3360  0.1606
backyard_space          0.0950 -0.0283 -0.0973 -0.2789  0.6522 -0.2992
crime_rate             -0.1127  0.0311  0.5990  0.0305  0.0452  0.0943
school_rating           0.3826  0.0933 -0.2274  0.2083 -0.1075  0.0590
age_of_home            -0.1434  0.0784  0.1554  0.5043  0.2710 -0.2451
distance_to_city_center -0.2003 -0.0252  0.0668  0.4481  0.2654 -0.0443
employment_rate         0.1296 -0.0161 -0.5808  0.2606  0.2405 -0.0185
property_tax_rate      -0.1435 -0.0393  0.1455  0.4497 -0.0424 -0.1501
renovation_quality      0.4136  0.0536 -0.0005  0.1627  0.0842 -0.0447
local_amenities         0.1973 -0.6643  0.1024  0.0404 -0.0052 -0.0336
transport_access        0.2006 -0.6611  0.0998  0.0429 -0.0016 -0.0288
previous_sale_price     0.4630  0.1788  0.2403  0.0404 -0.0060 -0.0020
windows                 0.0065 -0.0278  0.0454  0.0640  0.4239  0.8702

                           PC7     PC8     PC9    PC10    PC11    PC12  \
square_footage         -0.1512  0.2295  0.1824  0.5640  0.3045  0.1530
num_bathrooms          -0.0718  0.1016  0.1836 -0.7595 -0.1462  0.1115
num_bedrooms            0.3446 -0.2946 -0.4400  0.0208 -0.0890  0.3084
backyard_space          0.4909 -0.2280 -0.2856  0.0769 -0.0347 -0.0638
crime_rate             -0.0958 -0.6490  0.2112  0.1449 -0.2741 -0.1809
school_rating           0.0002 -0.0261 -0.0806 -0.0091  0.0159 -0.6094
age_of_home            -0.2666  0.3368 -0.3579  0.1155 -0.4848 -0.0132
distance_to_city_center -0.1937 -0.2481 -0.1586 -0.2359  0.7080  0.0144
employment_rate        -0.2190 -0.3344  0.4254  0.0812 -0.2283  0.3418
property_tax_rate       0.6548  0.2174  0.4926  0.0069  0.0331  0.0087
renovation_quality     -0.0404 -0.0152  0.1603  0.0099  0.0291 -0.4456
local_amenities        -0.0522  0.0264 -0.0219 -0.0030 -0.0188  0.0450
transport_access       -0.0401  0.0209 -0.0280 -0.0340 -0.0254  0.0353
previous_sale_price    -0.0187  0.0209 -0.0546  0.0011  0.0825  0.3766
windows                 0.1066  0.2011  0.0042 -0.0282 -0.0573  0.0018

                          PC13    PC14    PC15
square_footage          0.0270 -0.1941 -0.4285
num_bathrooms          -0.0076 -0.1500 -0.3177
num_bedrooms           -0.0071  0.0802 -0.3926
backyard_space         -0.0044 -0.0341  0.0328
crime_rate              0.0051 -0.1067  0.0211
school_rating           0.0182 -0.5875  0.1020
age_of_home             0.0019 -0.0016  0.0123
distance_to_city_center -0.0017 -0.0153 -0.0083
employment_rate         0.0056 -0.0546 -0.0144
property_tax_rate       0.0013 -0.0778  0.0654
renovation_quality     -0.0350  0.7486 -0.0434
local_amenities        -0.7028 -0.0759 -0.0239
transport_access        0.7096 -0.0155 -0.0189
previous_sale_price    -0.0006 -0.0020  0.7361
windows                -0.0051  0.0009 -0.0070
```

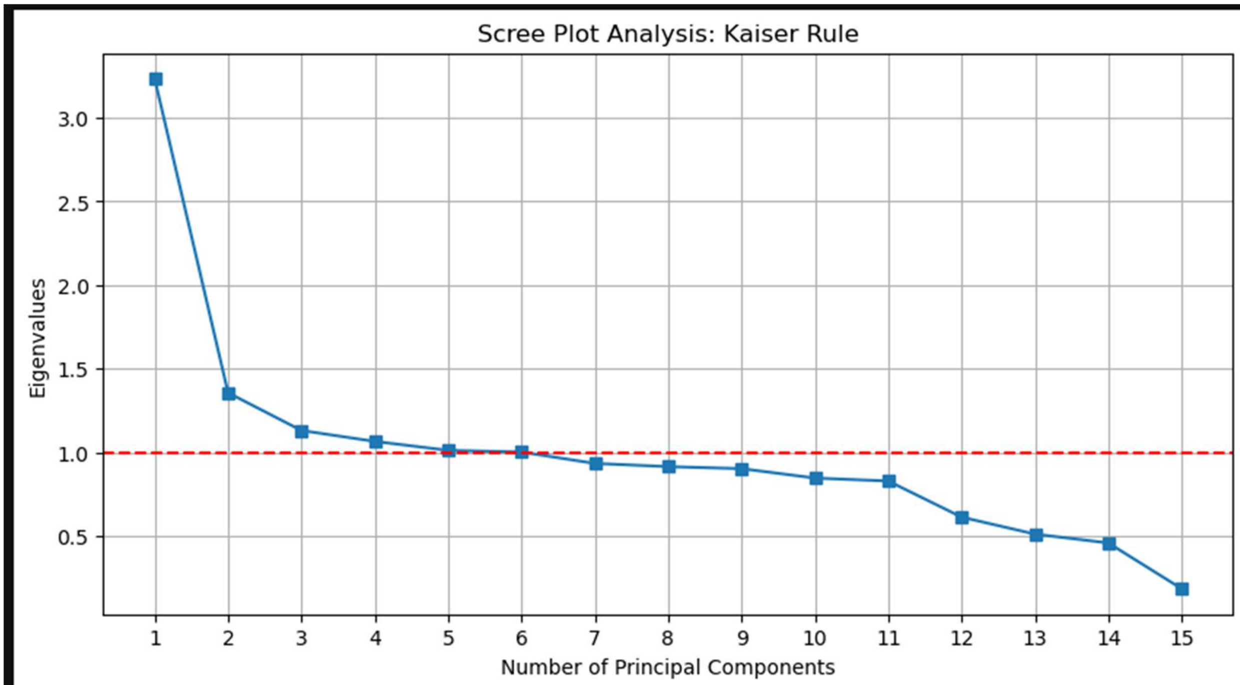
15 Principal Components were identified in the original matrix.

E2. Identify the total number of PCs to maintain using the Kaiser Rule

To identify the number of variables to maintain, the Kaiser Rule was applied through a Scree Plot, mapping the eigenvalues for each principal component. The variance components, along with the eigenvalues, were calculated first.

```
eigenvalues = pca.explained_variance_
explained_variance = pca.explained_variance_ratio_
cumulative_variance = np.cumsum(explained_variance)
```

Then, these were used to create a Scree Plot and list an array of eigenvalues to aid in decision-making:




```
Eigenvalue Array
[3.23356586 1.35687976 1.13135077 1.066051   1.01267494 1.00246233
 0.93441445 0.91526809 0.90316045 0.84702412 0.82931247 0.61346435
 0.51193033 0.45908462 0.18549962]
```

The plot identified six principal components to maintain in the analysis.

E3 & E4. Identify Variance for each PC & Summarize Results of PCA

A combination of eigenvalues, the explained variance ratio, and cumulative variance were utilized to decide which variables to operate in the linear regression model. These were first plotted via the following code.

```
#Creating a table for Principal Component breakdown
variance_df = pd.DataFrame({
    "Principal Component": [f"PC{i+1}" for i in range(len(eigenvalues))],
    "Eigenvalue": eigenvalues,
    "Explained Variance Ratio": explained_variance,
    "Cumulative Variance": cumulative_variance
})

display(variance_df.set_index("Principal Component"))
```

Which yielded the following table for decision-making:

Principal Component	Eigenvalue	Explained Variance Ratio	Cumulative Variance
PC1	3.233566	0.215540	0.215540
PC2	1.356880	0.090446	0.305986
PC3	1.131351	0.075413	0.381399
PC4	1.066051	0.071060	0.452459
PC5	1.012675	0.067502	0.519961
PC6	1.002462	0.066821	0.586782
PC7	0.934414	0.062285	0.649067
PC8	0.915268	0.061009	0.710076
PC9	0.903160	0.060202	0.770278
PC10	0.847024	0.056460	0.826739
PC11	0.829312	0.055280	0.882018
PC12	0.613464	0.040892	0.922910
PC13	0.511930	0.034124	0.957034
PC14	0.459085	0.030601	0.987635
PC15	0.185500	0.012365	1.000000

The Kaiser Rule suggested using the first 6 PCs in the model; however, this only accounts for 58% of the variance in the model per the cumulative variance. To achieve the 90% cumulative variance, a typical selection benchmark, 12 Principal Components would have to be selected. These many dimensions run the risk of overfitting the model. As a result, I opted to include eight principal components in the linear regression model as this was a compromise and hit around 71% cumulative variance for the model overall, which could reduce the likelihood of overfitting or underfitting with the model. If the additional Principal Components were not necessary, the linear regression will identify them as statistically insignificant and remove them in the optimized model.

```
#Selecting the top 8 values and preparing for Multiple Linear Regression
pca_final_values = principal_components[:, :8]
pca_final_df = pd.DataFrame(pca_final_values, columns=[f'PC{i+1}' for i in range(8)]) #Convert to DF
display(pca_final_df.describe()) #Confirming DF populated
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
count	7000.000000	7000.000000	7.000000e+03	7.000000e+03	7.000000e+03	7.000000e+03	7.000000e+03	7.000000e+03
mean	0.000000	0.000000	-3.248195e-17	1.015061e-17	-1.827110e-17	2.588406e-17	1.218073e-17	3.045183e-18
std	1.798212	1.164852	1.063650e+00	1.032497e+00	1.006318e+00	1.001230e+00	9.666512e-01	9.566964e-01
min	-5.061368	-3.334616	-3.152703e+00	-3.495327e+00	-2.971960e+00	-2.459060e+00	-3.445826e+00	-3.183276e+00
25%	-1.289576	-0.840727	-7.543630e-01	-7.232384e-01	-6.990078e-01	-6.569039e-01	-6.689527e-01	-6.468180e-01
50%	-0.071973	-0.013946	-4.278736e-02	-3.211289e-02	-3.448977e-02	-1.004106e-01	-8.424973e-03	1.517953e-02
75%	1.245753	0.795492	7.155824e-01	6.958897e-01	6.631889e-01	5.097943e-01	6.410534e-01	6.602764e-01
max	6.549247	4.085575	4.000304e+00	4.157794e+00	4.925448e+00	5.298417e+00	3.614625e+00	3.464042e+00

Multiple Linear Regression

F1. Split the data into Training & Test datasets

Sticking with the guidelines set out in the first PA of this course, I utilized an 80/20 training to test split in the dataset.

```
#Splitting the Dataset into Test and Training
y = df.price
X = pca_final_df

#Splitting the Dataset into a Test and Training dataset with an 80/20 split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
y_train = y_train.to_frame() #Converting to dataframe for ease in exporting
y_test = y_test.to_frame()
print(f"Training data: {X_train.shape}, Testing data: {X_test.shape}")
print(f"Training labels: {y_train.shape}, Testing labels: {y_test.shape}")
```

```
Training data: (5600, 8), Testing data: (1400, 8)
Training labels: (5600, 1), Testing labels: (1400, 1)
```

The dataset was then combined and exported per instructions.

```
#Combining the datasets for exporting
train_data = pd.concat([X_train, y_train], axis=1)
test_data = pd.concat([X_test, y_test], axis=1)
```

```
#Exporting the Test & Train Datasets to share
train_data.to_csv("training_data", index=False)
test_data.to_csv("test_data", index=False)
```

F2. Perform Linear Regression Model & Optimize on Training Dataset

The initial multiple linear regression model yielded several insights, most critically an Adjusted R^2 value of only .639 and 4 Principal Components whose p-values were above the .05 threshold for statistical significance.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.639
Model:                            OLS   Adj. R-squared:                  0.639
Method:                 Least Squares   F-statistic:                     1238.
Date:                Mon, 03 Feb 2025   Prob (F-statistic):               0.00
Time:                        23:21:05   Log-Likelihood:                -71884.
No. Observations:                5600   AIC:                         1.438e+05
Df Residuals:                    5591   BIC:                         1.438e+05
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
PC1         6.232e+04    671.806     92.766      0.000     6.1e+04    6.36e+04
PC2         2.397e+04   1040.603     23.039      0.000    2.19e+04     2.6e+04
PC3         3.054e+04   1136.937     26.858      0.000    2.83e+04    3.28e+04
PC4         5229.2088   1182.774      4.421      0.000    2910.513    7547.905
PC5        -1769.2148   1214.552     -1.457      0.145   -4150.208     611.779
PC6         -587.5374   1212.379     -0.485      0.628   -2964.272    1789.197
PC7        -1469.7242   1262.380     -1.164      0.244   -3944.480    1005.031
PC8         2000.8418   1266.679      1.580      0.114    -482.342    4484.025
const       3.081e+05   1215.745    253.420      0.000    3.06e+05     3.1e+05
==============================================================================
Omnibus:                      326.441   Durbin-Watson:                   1.983
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              415.985
Skew:                           0.562   Prob(JB):                     4.68e-91
Kurtosis:                       3.721   Cond. No.                         1.90
==============================================================================

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

Backward Stepwise Elimination was created via a loop to remove each principal component, which was statistically insignificant via the following.

```python
#Defining the Backwards Stepwise Elimination Loop
def backward_elimination(X, y, significance_level=0.05):
    X = X.assign(const=1)
    iteration = 1  #Track loop iterations
    while True:
        model = sm.OLS(y, X).fit()
        p_values = model.pvalues
        print(f"\nIteration {iteration} - P-values:\n", p_values)#Print current iteration and p-values


        max_p_value = p_values.max() #To identify highest p-value
        if max_p_value > significance_level:
            worst_feature = p_values.idxmax()
            print(f"Removing {worst_feature} (p-value: {max_p_value:.4f})")
            if worst_feature in X.columns:
                X.drop(columns=[worst_feature], inplace=True)
        else:
            break

        iteration += 1  #Increase iteration count

    print("\nFinal Model Summary:")
    return X, model.summary()

#Run the backward elimination
X_final, model_summary = backward_elimination(X_train, y_train)
print(model_summary)
```

With these results:

```
Iteration 1 - P-values:
 PC1       0.000000e+00
PC2      2.813727e-112
PC3      1.489747e-149
PC4       1.000404e-05
PC5       1.452606e-01
PC6       6.279683e-01
PC7       2.443730e-01
PC8       1.142559e-01
const     0.000000e+00
dtype: float64
Removing PC6 (p-value: 0.6280)

Iteration 2 - P-values:
 PC1       0.000000e+00
PC2      2.960429e-112
PC3      1.479677e-149
PC4       1.005036e-05
PC5       1.458492e-01
PC7       2.460231e-01
PC8       1.133403e-01
const     0.000000e+00
dtype: float64
Removing PC7 (p-value: 0.2460)

Iteration 3 - P-values:
 PC1       0.000000e+00
PC2      3.994701e-112
PC3      1.652587e-149
PC4       1.005009e-05
PC5       1.475339e-01
PC8       1.095401e-01
const     0.000000e+00
dtype: float64
Removing PC5 (p-value: 0.1475)

Iteration 4 - P-values:
 PC1       0.000000e+00
PC2      3.810944e-112
PC3      1.392853e-149
PC4       9.963506e-06
PC8       1.108763e-01
const     0.000000e+00
dtype: float64
Removing PC8 (p-value: 0.1109)
```

```
Iteration 5 - P-values:
 PC1       0.000000e+00
PC2      4.018027e-112
PC3      1.528861e-149
PC4       9.718893e-06
const     0.000000e+00
dtype: float64
```

```
Final Model Summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.639
Model:                            OLS   Adj. R-squared:                  0.638
Method:                 Least Squares   F-statistic:                     2473.
Date:                Mon, 03 Feb 2025   Prob (F-statistic):               0.00
Time:                        23:23:49   Log-Likelihood:                -71887.
No. Observations:                5600   AIC:                         1.438e+05
Df Residuals:                    5595   BIC:                         1.438e+05
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
PC1         6.231e+04    671.892     92.735      0.000     6.1e+04    6.36e+04
PC2         2.396e+04   1040.639     23.021      0.000    2.19e+04     2.6e+04
PC3         3.054e+04   1137.124     26.856      0.000    2.83e+04    3.28e+04
PC4         5237.5928   1182.994      4.427      0.000    2918.465    7556.721
const       3.081e+05   1215.908    253.367      0.000    3.06e+05     3.1e+05
==============================================================================
Omnibus:                      325.843   Durbin-Watson:                   1.981
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              414.563
Skew:                           0.562   Prob(JB):                     9.52e-91
Kurtosis:                       3.717   Cond. No.                         1.81
==============================================================================

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

The final result was a model that retained four principal components with an adjusted R^2 value of .638. Further analysis through the test dataset indicated the PC4 had a p-value of greater than .9 when using the test data – a classic sign of overfitting the model and confirmation that the model was not generalizing well to new data. To rectify this, PC4 was removed from the training set as a means of optimization, and the model was re-run to confirm there was little to no change in the relevant variables – once confirmed, this constituted the final model for linear regression.

```
##After running the full model on the Test dataset, PC4 was identified as not statistically significant
X_train_reduced = X_train[["PC1", "PC2", "PC3"]].assign(const=1)
final_model_reduced = sm.OLS(y_train, X_train_reduced).fit()
print(final_model_reduced.summary())
##Dropping PC4 did not affect R² values in any meaningful way
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.637
Model:                            OLS   Adj. R-squared:                  0.637
Method:                 Least Squares   F-statistic:                     3280.
Date:                Mon, 03 Feb 2025   Prob (F-statistic):               0.00
Time:                        20:34:44   Log-Likelihood:                -71896.
No. Observations:                5600   AIC:                         1.438e+05
Df Residuals:                    5596   BIC:                         1.438e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
PC1         6.232e+04    673.004     92.595      0.000     6.1e+04    6.36e+04
PC2         2.395e+04   1042.366     22.975      0.000    2.19e+04     2.6e+04
PC3         3.054e+04   1139.011     26.817      0.000    2.83e+04    3.28e+04
const        3.08e+05   1217.878    252.918      0.000    3.06e+05     3.1e+05
==============================================================================
Omnibus:                      321.997   Durbin-Watson:                   1.982
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              408.749
Skew:                           0.558   Prob(JB):                     1.74e-89
Kurtosis:                       3.711   Cond. No.                         1.81
==============================================================================

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

The change in all relevant metrics from dropping PC4 was minimal compared to including all four previously identified Principal Components.

F3. Provide Mean Squared Error

The Mean Squared Error of the training dataset was identified as 8299174234.112436 via the following code:

```
final_model_reduced = sm.OLS(y_train, X_train_reduced).fit()
y_pred = final_model_reduced.predict(X_train_reduced)
mse_train = mean_squared_error(y_train, y_pred)
print ("Mean Squared Error (MSE) - Training Dataset:", mse_train)
```

```
Mean Squared Error (MSE) - Training Dataset: 8299174234.112436
```

F4. Run Prediction on Test Dataset

```
#Running Optimized Model on Test Dataset
X_test_reduced = X_test[["PC1", "PC2", "PC3"]].assign(const=1)
test_model = sm.OLS(y_test, X_test_reduced).fit()
print(test_model.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.619
Model:                            OLS   Adj. R-squared:                  0.618
Method:                 Least Squares   F-statistic:                     756.7
Date:                Mon, 03 Feb 2025   Prob (F-statistic):          4.91e-292
Time:                        23:44:01   Log-Likelihood:                -17954.
No. Observations:                1400   AIC:                         3.592e+04
Df Residuals:                    1396   BIC:                         3.594e+04
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
PC1         6.137e+04   1372.095     44.726      0.000    5.87e+04    6.41e+04
PC2          2.25e+04   2090.287     10.765      0.000    1.84e+04    2.66e+04
PC3         3.018e+04   2310.257     13.065      0.000    2.57e+04    3.47e+04
const       3.043e+05   2405.132    126.515      0.000       3e+05    3.09e+05
==============================================================================
Omnibus:                       63.664   Durbin-Watson:                   1.924
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               71.549
Skew:                           0.529   Prob(JB):                     2.91e-16
Kurtosis:                       3.327   Cond. No.                         1.76
==============================================================================

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

Running the test model determined all Principal Components were statistically significant with p-values of 0, and the adjusted R^2 determined the model could predict 61.8% of the price value in the model. Additionally, the MSE was calculated for the test dataset and determined to be 8059401217.70743.

```
#Calculating Mean Squared Error for the Test Dataset
model = sm.OLS(y_test, X_test.assign(const=1)).fit()
y_pred = test_model.predict(X_test)
mse_test = mean_squared_error(y_test, y_pred)
print ("Mean Squared Error (MSE) - Test Dataset:", mse_test)

Mean Squared Error (MSE) - Test Dataset: 8059401217.70743
```

For a final review of the results, the RMSE was calculated for both datasets and compared next to the mean cost of a house to determine the model's overall predictive power and accuracy.

```
#Validating results by calculating Root Mean Squared Error (RMSE)
training_rmse = np.sqrt(mse_train)
test_rmse = np.sqrt(mse_test)

print("Training RMSE:", training_rmse)
print("Test RMSE:", test_rmse)
print("Mean housing Price:", df["price"].mean())
##Test RMSE within 1,217 (1.4%) of Training RMSE

Training RMSE: 91099.80369963722
Test RMSE: 89774.1678753272
Mean housing Price: 307281.5217142857
```

Summary of Analysis

G1. List Packages Utilized in Python

```
#Importing Relevant Packages
##G1 - List the packages or libraries you have chosen for Python or R and justify how each item on the list supports the analysis.
import pandas as pd #For Dataframes
import numpy as np #For general functionality
from IPython.display import display #To format outputs
import matplotlib.pyplot as plt #Used for Data Visualizations
import statsmodels.api as sm #Used to create the Linear Regression Model
from sklearn.model_selection import train_test_split #Used to split the datasets
from sklearn.decomposition import PCA #To perform Principal Component Analysis
from sklearn.preprocessing import StandardScaler #To scale the dataset
from sklearn.metrics import mean_squared_error #For calculating MSE
```

Numpy was imported for general functionality of mathematics and statistics to use .cumsum in explained variance and .sqrt for RMSE. Pandas was included for use of dataframes which would be required. The display function from the IPython.display module was imported to make cleaner outputs and more formatted views for various cells in the code. The matplotlib package was used for multiple visualizations, including the Kaiser Rule. The sm alias from the statsmodels.api module was used as a part of the multiple linear regression model after PCA was performed. The test_train_split function was included to satisfy the splitting of the dataset into a test and train portions per rubric F1. The various modules from the sklearn package were imported to accomplish several portions of principal component analysis and results. The PCA function was to perform principal component analysis, StandardScaler helped prepare the dataset for PCA by scaling it appropriately, and mean_squared_error was utilized to satisfy the rubric of F3 requiring that detail.

G2. Discuss Optimization Method

Backward stepwise elimination was utilized to optimize the model. This was the logical selection as Principal Component Analysis utilizes all quantitative variables in the dataset, so the model begins with many principal components. By eliminating a PC one at a time and then re-running the model to check if any remaining PCs have a p-value that denotes them as insignificant, we ensure only those that are statistically insignificant are removed. This prevents the underfitting of the model by eliminating too many variables at once. The reduced complexity also makes the model easier to understand and visualize when explaining it to someone outside of the analytics world, a critical detail when explaining results to a non-technical audience.

G3. Discuss Verification of Assumptions to Create Optimized Model

The two assumptions verified to optimize the model were ensuring no multicollinearity and a sufficient number of observations to complete the analysis. To verify the multiple linear regression model's assumptions, the dataset must be checked for multicollinearity. This was tested using a correlation matrix.

```
#Creating Correlation Matrix to confirm no Multicollinearity
print("Correlation Matrix of Final Training Dataset:\n")
display(X_train_reduced.corr())
print("\nCorrelation Matrix of Final Test Dataset:\n")
display(X_test_reduced.corr())

print("\nShape of Training & Test Dataset:")
display(f'X_train Shape: {X_train_reduced.shape}')
display(f'y_train Shape: {y_train.shape}')
display(f'X_test Shape: {X_test_reduced.shape}')
display(f'y_test Shape: {y_test.shape}')
```

```
Correlation Matrix of Final Training Dataset:
```

	PC1	PC2	PC3	const
PC1	1.000000	0.002582	0.000433	NaN
PC2	0.002582	1.000000	0.004951	NaN
PC3	0.000433	0.004951	1.000000	NaN
const	NaN	NaN	NaN	NaN

```
Correlation Matrix of Final Test Dataset:
```

	PC1	PC2	PC3	const
PC1	1.000000	-0.010185	-0.001197	NaN
PC2	-0.010185	1.000000	-0.021698	NaN
PC3	-0.001197	-0.021698	1.000000	NaN
const	NaN	NaN	NaN	NaN

```
Shape of Training & Test Dataset:
'X_train Shape: (5600, 4)'
'y_train Shape: (5600, 1)'
'X_test Shape: (1400, 4)'
'y_test Shape: (1400, 1)'
```

The correlation matrix demonstrated Pearson correlation coefficients for multicollinearity, which were nowhere near a positive or negative correlation for any of the principal components. All coefficients were well under .1 or -.1, and none approached any range, denoting multicollinearity.

To verify there was a sufficient amount of observations in the dataset to perform the analysis, the shape of the training and test datasets were checked and counted for rows (See code above). Regarding sample size, the dataset had 7000 observations across all variables included, and the shape of the split datasets confirmed this detail. Seven thousand observations are enough to ensure the sample size is sufficiently sized, as the typical benchmark for too few is less than 500 total records. Seven thousand more than sufficiently passes this threshold, and this assumption has also been verified.

G4. Provide Regression Equation & Coefficients

As the final model had principal components included in the linear regression analysis, the equation can be summarized as follows:

$$Y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \varepsilon$$

Y is the price of the homes (outcome variable)

b_0 is the intercept value, which is 308,000

b_1x_1 is the coefficient on PC1, which is 62,320

b_2x_2 is the coefficient on PC2, which is 23,950

b_3x_3 is the coefficient on PC3, which is 30,540

ε is the error term and can be estimated using the RMSE, which is 91,099

This makes the final regression equation for the model to be

$$Y=308{,}000+62{,}320(PC1)+23{,}950(PC2)+30{,}540(PC3)+ \varepsilon$$

Per the above, assuming all other details remain constant:

- PC1 had the strongest influence, where a 1 unit increase in PC1 is associated with a $62,320 increase in home price.

- PC2 had a moderate influence, where a 1 unit increase in PC2 is associated with a $23,950 increase in home price
- PC3 had a moderate influence, where a 1 unit increase in PC3 is associated with a $30,540 increase in home price
- The intercept does not directly correspond to a fundamental home feature, as PCA transforms all variables into standardized values. Its function is to serve as a baseline estimate.
- The RMSE identifies the model accuracy and suggests that the model's predictions will deviate by ±$91,099 from the actual prices of homes -indicating the model does capture a significant portion of variability, but there are still unexplained factors contributing.

G5. Discuss Model Metrics of R^2 and Comparison of MSE

The R-squared value and adjusted R-squared value for the final model were both 0.637. This indicates that the model was somewhat successful and can explain 63.7% of the variance in price based on the variables included in the Principal Component Analysis and Multiple Linear Regression. This also means the model did not adequately explain 36.3% of the variance.

To compare the MSE for the training and test set, the RMSE was calculated as it provides an easier-to-visualize number. These values were compared next to the mean cost of a home to provide additional perspective on the model's effectiveness.

```
#Comparing both MSE
print ("Mean Squared Error (MSE) - Training Dataset:", mse_train)
print ("Mean Squared Error (MSE) - Test Dataset:", mse_test)
```

```
Mean Squared Error (MSE) - Training Dataset: 8299174234.112436
Mean Squared Error (MSE) - Test Dataset: 8059401217.70743
```

```
#Validating results by calculating Root Mean Squared Error (RMSE)
training_rmse = np.sqrt(mse_train)
test_rmse = np.sqrt(mse_test)

print("Training RMSE:", training_rmse)
print("Test RMSE:", test_rmse)
print("Mean housing Price:", df["price"].mean())
##Test RMSE within 1,217 (1.4%) of Training RMSE
```

```
Training RMSE: 91099.80369963722
Test RMSE: 89774.1678753272
Mean housing Price: 307281.5217142857
```

The training set's MSE was 239,773,016.405, greater than the test MSE, translating to a 3% difference. This 3% is well within an acceptable difference between MSE and identifies that there is no overfitting or underfitting to the model and that it is generalizing well to the data.

G6. Discuss Results & Implications of Analysis

The analysis yielded improved predictive and explanatory power compared to the linear regression and logistic regression alternatives completed in the other tasks. Principal Component Analysis assisted in reducing multicollinearity and reduced dimensions compared to the alternatives. An adjusted R^2 value predicted 63.7% of the variance in the price of a house given the 15 variables selected in the analysis. Further, including 15 variables ensured that as much data was utilized in the analysis as possible while producing results that avoided over or underfitting. The RMSE indicates that the actual value of the dependent variable could fluctuate by $91,099. This is approximately 29.6% of the mean cost of the houses in the dataset, which is a relatively significant margin of error when attempting to predict prices along a thin margin.

G7. Recommendations of Analysis

The real estate company should employ the data model to provide them with loose guidance on which factors to focus upon when maximizing the profit of individual homes. With 36.3% of the variance still unexplained and nearly a 30% potential error in the home's value, they should strive to pull in additional variables and factors that may increase their profits in the longer term. These predictions are still more accurate than the alternative logistic and linear regression models offered separately, as the PCA component improved the model's overall effectiveness. I would also recommend they acquire additional metrics that may affect the cost of homes and pull those variables into their dataset to further refine the model in the future, and perhaps fold in categorical variables where possible using one-hot encoding to make the model more robust and further explain the price.

Sources

No other sources were used outside of WGU course materials and details from previous assessments written by the author.