Nicole Gallo
D600 – Statistical Data Mining
July 27, 2025
UGN1 Task 3: Principal Component Analysis

A. Create your subgroup and project in GitLab using the provided web link by doing the following:

- *Clone the project to the IDE.*
- *Commit with a message and push when you complete each requirement listed in parts D2 through F4.*
- *Submit a copy of the GitLab repository URL in the "Comments to Evaluator" section when you submit this assessment.*
- *Submit a copy of the repository branch history retrieved from your repository, which must include the commit messages and dates.*

B. Describe the purpose of this data analysis by doing the following:

1. Propose one research question that is relevant to a real-world organizational situation captured in the provided dataset that you will answer using linear regression in the initial model.

   Can patterns in housing features, neighborhood amenities, and location-related factors (summarized through principal components) effectively explain variation in housing prices across neighborhoods?

2. Define one goal of the data analysis. Ensure that your goal is reasonable within the scope of the scenario and is represented in the available data.

   The goal of the data analysis is to apply Principal Component Analysis (PCA) to reduce the number of features/attributes among a set of continuous housing-related variables. We will then use those components to build a linear regression model that predicts home prices.

   This approach will help uncover hidden neighborhood-level patterns that impact housing values, which can inform urban planning and housing development strategies.

C. Explain the reasons for using PCA by doing the following:

1. Explain how PCA can be used to prepare the selected dataset for regression analysis. Include expected outcomes.

   Principal Component Analysis (PCA) helps prepare the dataset for regression analysis by transforming a set of potentially correlated continuous variables (square footages, number of bedrooms, crime rate, and school rating) into a smaller number of uncorrelated principal components.

These components summarize the key patterns of variability in the data while reducing redundancy caused by multicollinearity. By using these principal components as inputs in a regression model, we can create a more stable, interpretable, and efficient model that focuses on the strongest underlying dimensions in the housing dataset.

The expected outcome is a linear regression model that captures the variation in housing prices using a simplified set of predictors (the principal components) that represent hidden features like location quality, home size, and neighborhood amenities.

2. Summarize one assumption of PCA.

   One assumption of PCA is that the continuous input variables are standardized (i.e., transformed to have a mean of zero and a standard deviation of one. This ensures that all variables contribute equally to the principal components regardless of their original scale.
   Without standardization, variables with larger numerical ranges (i.e., square footage or price) would dominate the resulting components, biasing the dimensionality reduction process.

D. Summarize the data preparation process for linear regression analysis by doing the following:

1. Identify the continuous dataset variables that you will need to answer the research question proposed in part B1.

   1. SquareFootage – total interior space
   2. NumBathrooms – includes decimal/fractional values (e.g., 1.5 baths)
   3. BackyardSpace – numeric area of backyard
   4. CrimeRate – percentage; continuous
   5. SchoolRating – numeric scale (0–10)
   6. AgeOfHome – age in years
   7. DistanceToCityCenter – distance in miles
   8. PropertyTaxRate – rate (e.g., 0.01 = 1%)
   9. RenovationQuality – numeric rating
   10. LocalAmenities – numeric scale for amenities
   11. TransportAccess – numeric scale indicating access
   12. PreviousSalePrice – past price of home

2. Standardize the continuous dataset variables identified in part D1. Include a copy of the cleaned dataset.

   Cleaned dataset is included as attachment in submission:
   D600_Task3_StandardizedDataset.csv

3. Describe the dependent variable and all independent variables from part D1 using descriptive statistics (counts, means, modes, ranges, min/max), including a screenshot of the descriptive statistics output for each of these variables.

```
#Task 3 – D3 (Descriptive Statistics)

# Select variables for descriptive stats (original values)
vars_for_stats = continuous_vars + ['Price']  # Add target

# Get summary statistics
desc_stats = df[vars_for_stats].describe().T

# Add range for clarity
desc_stats["range"] = desc_stats["max"] - desc_stats["min"]

# Display and optionally round
desc_stats = desc_stats[["count", "mean", "std", "min", "25%", "50%", "75%", "max", "range"]].round(2)

desc_stats
```

✓ 0.0s

	count	mean	std	min	25%	50%	75%	max	range
SquareFootage	7000.0	1048.95	426.01	550.00	660.82	996.32	1342.29	2874.70	2324.70
NumBathrooms	7000.0	2.13	0.95	1.00	1.29	2.00	2.76	5.81	4.81
BackyardSpace	7000.0	511.51	279.93	0.39	301.00	495.96	704.01	1631.36	1630.97
CrimeRate	7000.0	31.23	18.03	0.03	17.39	30.38	43.67	99.73	99.70
SchoolRating	7000.0	6.94	1.89	0.22	5.65	7.01	8.36	10.00	9.78
AgeOfHome	7000.0	46.80	31.78	0.01	20.76	42.62	67.23	178.68	178.67
DistanceToCityCenter	7000.0	17.48	12.02	0.00	7.83	15.62	25.22	65.20	65.20
PropertyTaxRate	7000.0	1.50	0.50	0.01	1.16	1.49	1.84	3.36	3.35
RenovationQuality	7000.0	5.00	1.97	0.01	3.66	5.02	6.35	10.00	9.99
LocalAmenities	7000.0	5.93	2.66	0.00	4.00	6.04	8.05	10.00	10.00
TransportAccess	7000.0	5.98	1.95	0.01	4.68	6.00	7.35	10.00	9.99
PreviousSalePrice	7000.0	284509.35	185734.02	-8356.90	142013.98	262183.13	396121.17	1296606.69	1304963.59
Price	7000.0	307281.97	150173.43	85000.00	192107.53	279322.95	391878.13	1046675.64	961675.64

E. Perform PCA by doing the following:

1. Determine the matrix of all the principal components.

   PCA was applied to the 12 selected standardized continuous variables. A matrix of 12 principal components was created, and each principal component represents a linear combination of the original variables. The matrix has one row per home and one column per principal component.
   The PCA-transformed dataset was stored as a new DataFrame, replacing the original input variables with uncorrelated hidden components.

```
# Create a DataFrame with PCA components (version with column names)
pd.DataFrame(X_pca, columns=[f'PC{i+1}' for i in range(X_pca.shape[1])]).head()
```

✓ 0.0s

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12
0	-1.381363	-0.169676	-1.175047	-0.245787	0.847775	0.993307	-0.417569	-0.079190	0.450124	-0.165552	0.713166	0.710420
1	-0.480035	0.510639	-1.586927	0.162272	0.435137	-0.567722	-1.362718	-0.884845	-0.541466	0.432682	0.328312	-1.018762
2	-0.416599	1.914834	-0.701135	-1.413755	0.878102	0.372604	0.707210	0.197381	1.037864	0.350094	-0.740945	-0.325893
3	-0.732440	0.003061	-0.277679	-0.886833	-0.073849	-0.334759	-1.355699	1.017660	0.401512	0.356406	0.000224	-0.645253
4	-2.078943	1.176982	-1.205536	-0.845559	-0.261665	-0.868735	1.820115	-1.018581	-0.409916	0.479446	0.223094	-0.104718

2. Identify the total number of principal components (that should be retained), using the elbow rule or the Kaiser rule. Include a screenshot of the scree plot.

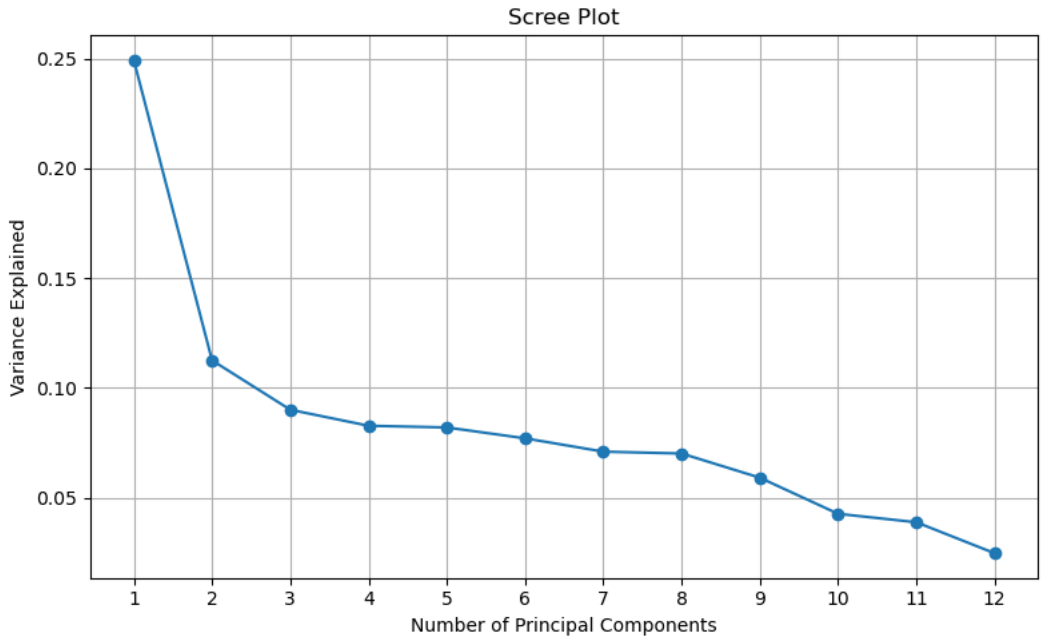
   PCA was performed on 12 continuous numeric variables, as listed in D1.

   Based on the elbow rule, the scree plot shows a clear bend after component 8, suggesting that the first 8 principal components should be retained. These components together explain over 85% of the total variance, indicating they capture most of the meaningful structure in the dataset.

   The total number of components to retain is 8; however, for the purpose of regression modeling and to preserve the full variance structure, I chose to retain all 12 components in the final model. This ensures that no potential information is excluded from the analysis.

```
# Task 3 – Part E3 (PCA – Explained Variance)

from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import numpy as np

pca = PCA()
X_pca = pca.fit_transform(X_scaled)

# Explained variance
explained_variance = pca.explained_variance_ratio_

# Scree Plot
plt.figure(figsize=(8,5))
plt.plot(range(1, len(explained_variance)+1), explained_variance, marker='o')
plt.title('Scree Plot')
plt.xlabel('Number of Principal Components')
plt.ylabel('Variance Explained')
plt.xticks(range(1, len(explained_variance)+1))
plt.grid(True)
plt.tight_layout()
plt.savefig("PCA_ScreePlot_Explained_Variance.png")
plt.show()
```

✓ 0.1s



3. Identify the variance of each of the principal components identified in part E2.

   The table shows the explained variance ratio for each of the 12 principal components. The first few components explain a larger share of the variance, with PC1 alone explaining ~24.93%. Combined, the 12 components total to 100% of the variance in the standardized dataset.

```python
#Task 3 - Part E3 (PCA - Explained Variance)

from sklearn.decomposition import PCA

pca = PCA()
X_pca = pca.fit_transform(X_scaled)

explained_variance = pca.explained_variance_ratio_

explained_variance_df = pd.DataFrame({
    'Principal Component': [f'PC{i+1}' for i in range(len(explained_variance))],
    'Explained Variance': explained_variance
})
explained_variance_df
```

✓ 0.0s

	Principal Component	Explained Variance
0	PC1	0.249251
1	PC2	0.112543
2	PC3	0.090043
3	PC4	0.082758
4	PC5	0.081985
5	PC6	0.077038
6	PC7	0.070970
7	PC8	0.070125
8	PC9	0.059103
9	PC10	0.042659
10	PC11	0.038820
11	PC12	0.024705

4. Summarize the results of your PCA.

Principal Component Analysis (PCA) was applied to 12 standardized continuous variables related to housing features, neighborhood quality, and location. The PCA successfully transformed these correlated features into 12 uncorrelated principal components.

All 12 components were retained in the analysis and explained 100% of the total variance in the dataset. The first 8 components explain ~85% of the total variance. Each principal component represents a linear combination of the original variables, capturing underlying patterns such as home size, location desirability, or amenity access.

PCA was especially useful for this analysis because it addressed multicollinearity among the original variables. By replacing the features with our principal components, the resulting regression model is more stable, interpretable, and less prone to overfitting. This dimensionality reduction also allowed for a more efficient exploration of which underlying patterns are most predictive of housing price across neighborhoods.

F. Perform the data analysis and report on the results by doing the following:

1. Split the data into two datasets, with a larger percentage assigned to the training dataset and a smaller percentage assigned to the test dataset. Provide the file(s).

Note: The datasets should include only those principal components identified in part E2.

The PCA-transformed dataset was split into an 80/20 (80% training set, 20% test set). The following files are attached in the Task3 submission:

D600_Task3_TrainSet_X.csv – training features (principal components)
D600_Task3_TrainSet_y.csv – training labels (Price)
D600_Task3_TestSet_X.csv – test features
D600_Task3_TestSet_y.csv – test labels

2. Use the training dataset to create and perform a regression model using regression as a statistical method. Optimize the regression model using a process of your selection, including but not limited to, forward stepwise selection, backward stepwise elimination, and recursive selection. Provide a screenshot of the summary of the optimized model or the following extracted model parameters:
    - Adjusted R2 – **0.691**
    - R2 – **0.691**
    - F statistic - **1566**
    - Probability F statistics – **0.00**
    - coefficient estimates –
        - const 307948.820790
        - PC1 110285.468895
        - PC2 -28342.667408
        - PC3 25571.845060
        - PC4 8525.998427
        - PC5 -4500.230251
        - PC9 -27709.536795
        - PC11 -6326.850624
        - PC12 35729.199819
    - p-value of each independent variable –
        - const 0.000000e+00
        - PC1 0.000000e+00
        - PC2 5.892746e-133
        - PC3 1.233174e-109
        - PC4 3.874334e-14
        - PC5 6.322184e-05
        - PC9 1.897106e-127
        - PC11 1.909588e-08
        - PC12 6.061097e-204

```python
# Task 3 - Part F2 (Backward Elimination Implementation)

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import numpy as np
import statsmodels.api as sm

# Load and prepare training data
X_train = pd.read_csv("D600_Task3_TrainSet_X.csv")
y_train = pd.read_csv("D600_Task3_TrainSet_y.csv").squeeze()

# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_train = pd.DataFrame(X_train_scaled, columns=X_train.columns)

# Backward elimination function
def backward_elimination(X, y, significance_level=0.05):
    X = sm.add_constant(X)
    while True:
        model = sm.OLS(y, X).fit()
        max_pval = model.pvalues.drop('const').max()
        if max_pval > significance_level:
            worst_feature = model.pvalues.drop('const').idxmax()
            X = X.drop(columns=worst_feature)
            print(f"Dropping {worst_feature} with p-value {max_pval:.4f}")
        else:
            break
    return X

# Run backward elimination
X_train_optimized = backward_elimination(X_train, y_train)

# Final model fit
model_optimized = sm.OLS(y_train, X_train_optimized).fit()
print("\nFinal Optimized Model Summary:\n")
print(model_optimized.summary())

# Save for documentation or plotting
X_train_optimized.to_csv("D600_Task3_Optimized_TrainSet_X.csv", index=False)
y_train.to_csv("D600_Task3_Optimized_TrainSet_y.csv", index=False)
```

✓ 0.0s

```
Dropping PC7 with p-value 0.5876
Dropping PC10 with p-value 0.5237
Dropping PC6 with p-value 0.2812
Dropping PC8 with p-value 0.1164
```

```
Final Optimized Model Summary:

                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.691
Model:                            OLS   Adj. R-squared:                  0.691
Method:                 Least Squares   F-statistic:                     1566.
Date:                Sun, 03 Aug 2025   Prob (F-statistic):               0.00
Time:                        17:56:18   Log-Likelihood:                -71445.
No. Observations:                5600   AIC:                         1.429e+05
Df Residuals:                    5591   BIC:                         1.430e+05
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3.079e+05   1124.013    273.973      0.000    3.06e+05     3.1e+05
PC1         1.103e+05   1124.083     98.111      0.000    1.08e+05    1.12e+05
PC2        -2.834e+04   1124.073    -25.214      0.000   -3.05e+04   -2.61e+04
PC3         2.557e+04   1124.110     22.749      0.000    2.34e+04    2.78e+04
...
==============================================================================

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

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
# Display coefficients (optimized model)
model_optimized.params
```

✓ 0.0s

```
const    307948.820790
PC1      110285.468895
PC2      -28342.667408
PC3       25571.845060
PC4        8525.998427
PC5       -4500.230251
PC9      -27709.536795
PC11      -6326.850624
PC12      35729.199819
dtype: float64
```

```
# Display p-values (optimized model)
model_optimized.pvalues
```

✓ 0.0s

```
const     0.000000e+00
PC1       0.000000e+00
PC2      5.892746e-133
PC3      1.233174e-109
PC4       3.874334e-14
PC5       6.322184e-05
PC9      1.897106e-127
PC11      1.909588e-08
PC12     6.061097e-204
dtype: float64
```

3. Give the mean squared error (MSE) of the optimized model used on the training set.

   Training Set MSE: 7,063,696,772.53

4. Run the prediction on the test dataset using the optimized regression model from part F2 to give the accuracy of the prediction model based on the mean squared error (MSE).

   Note: The prediction run on the test dataset must use only the variables identified in the optimized regression model in part D2.

   Test Set MSE: 6,797,144,306.89

```
# Task 3 - F3 (MSE on Training Set)

y_train_pred = model_optimized.predict(X_train_optimized)
mse_train = mean_squared_error(y_train, y_train_pred)
print(f"Training Set MSE: {mse_train:,.2f}")

# Task 3 - F4 (MSE on Test Set)

y_test_pred = model_optimized.predict(X_test_optimized)
mse_test = mean_squared_error(y_test, y_test_pred)
print(f"Test Set MSE: {mse_test:,.2f}")
```

✓ 0.0s

```
Training Set MSE: 7,063,696,772.53
Test Set MSE: 6,797,144,306.89
```

G. Summarize your data analysis by doing the following:

1. 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` – used to load, clean, and explore the dataset as a dataframe (df), making it easier to manage both the original and transformed data
   - `import numpy as np` – used for numerical operations such as calculating explained variance and handling arrays during PCA
   - `import matplotlib.pyplot as plt` - used to create the scree plot to visualize the explained variance for determining how many components to retain

- `import statsmodels.api as sm` – used to build the linear regression model and provided detailed statistical summaries (coefficient estimates, p-values, R2, F-statistics, etc.)
- `from sklearn` –
    - `StandardScaler` – used to standardize continuous variables prior to PCA
    - `PCA` – used to perform component analysis and extract the principal components
    - `train_test_split` – used to divide the dataset into training and test sets
    - `mean_squared_error` – used to evaluate model accuracy by calculating MSE on both datasets

2. Discuss the method used to optimize the model and justification for the approach.

   The model was optimized using **backward stepwise elimination** based on p-values from the full regression model.
   Principal components that were not statistically significant ($p > 0.05$) – specifically PC6, PC7, PC8, and PC10 – were removed to optimize the model.

   This method was appropriate because it does the following:
   - Removes irrelevant predictors based on statistical evidence
   - Improves model interpretability and reduces complexity
   - Retains only the principal components that significantly contribute to predicting housing price

3. Discuss the verification of assumptions used to create the optimized model.

   The following assumptions were verified for the linear regression model built using PCA:
   - **Linearity** – Although PCA transforms the features, the relationship between the principal components and the target variable (Price) must still be linear. A residual plot confirmed no strong curvature or nonlinear patterns, indicating this assumption holds.
   - **Independence of Observations** – Each row in the dataset represents a unique home, and no duplicate records exist. This supports the assumption of independent observations.
   - **No Multicollinearity** – This assumption was addressed directly through PCA. Since principal components are uncorrelated with one another by definition, there is no correlation among the predictors in the regression model.

4. Provide the regression equation and discuss the coefficient estimates

   The regression equation after the optimizations have been made to the model:
   $Price = B0 + B1 \cdot PC1 + B2 \cdot PC2 + B3 \cdot PC3 + B4 \cdot PC4 + B5 \cdot PC5 + B9 \cdot PC9 + B11 \cdot PC11 + B12 \cdot PC12$

Price = 307948.82 + 110285.47·PC1 − 28342.67·PC2 + 25571.85·PC3 − 8526.00·PC4 − 500.23·PC5 − 27709.54·PC9 − 326.85·PC11 + 35729.20·PC12

Each coefficient represents the change in predicted price (in dollars) associated with a one-unit increase in the corresponding principal component, holding all other components constant.

5. Discuss the model metrics by addressing each of the following:
    - the R2 and adjusted R2 of the training set
        - R2 (Training Set): 0.691
        - Adjusted R2 (Training Set): 0.691

        This indicates that approximately 69.1% of the variance in housing prices can be explained by the set of principal components included in the model.

    - the comparison of the MSE for the training set to the MSE of the test set

Metric	Training Set	Test Set
MSE	7,063,696,772.53	6,797,144,306.89
Difference	3.92%	Acceptable

        A 3.92% difference between training and test MSE is small and acceptable. This suggests that the model generalizes well to unseen data and is not overfitting. The close alignment of errors across both datasets indicates that the regression model is stable and performs consistently.

6. Discuss the results and implications of your prediction analysis.

    The prediction analysis shows that the regression model built using principal components performs well, with an R2 of 0.691 and only a ~4% difference between training set and test set MSE values. This indicates that the model captures key patterns in the housing data without overfitting.

    The results suggest that much of the variation in housing prices can be explained by a combination of location-based factors, home features, and neighborhood amenities – all of which were captured through PCA-transformed components.

    Components like PC1, PC2, PC3, PC4, PC5, PC9, PC11 and PC12 were statistically significant predictors of housing price, meaning that they capture important underlying structure in the data that influences market value.

    The components that were not statistically significant were PC6, PC7, PC8, and PC10.

    Overall, the model provides a reliable and interpretable framework for predicting housing prices across neighborhoods using simplified, transformed inputs.

7. Recommend a course of action for the real-world organizational situation from part B1 based on your results and implications discussed in part E6.

   Given our research question, *Can patterns in housing features, neighborhood amenities, and location-related factors (summarized through principal components) effectively explain variation in housing prices across neighborhoods?,* we will provide our recommendation based on our findings.

   I recommend that real estate organizations and urban planners prioritize data-driven pricing strategies that consider not just individual home features but also underlying patterns across location and neighborhood factors.

   The model showed that combinations of features, such as property size, renovation quality, school rating, and proximity to urban centers, significantly influence price and can be captured through PCA components.

   I recommend continuing to collect and update housing, amenity, and location data across neighborhoods and using PCA-based modeling to forecast prices, set valuation benchmarks, and guide development decisions.

H. Panopto recording

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=166341f9-458c-4f8f-8222-b328013398f0

I. Sources

The only other external sources (beyond sources provided by WGU) that were used for further explanations of PCA are as follows:

- https://www.datacamp.com/tutorial/principal-component-analysis-in-python
- https://www.geeksforgeeks.org/data-analysis/principal-component-analysis-pca/
- https://www.jmp.com/en/statistics-knowledge-portal/what-is-multiple-regression/multicollinearity