Task 2:

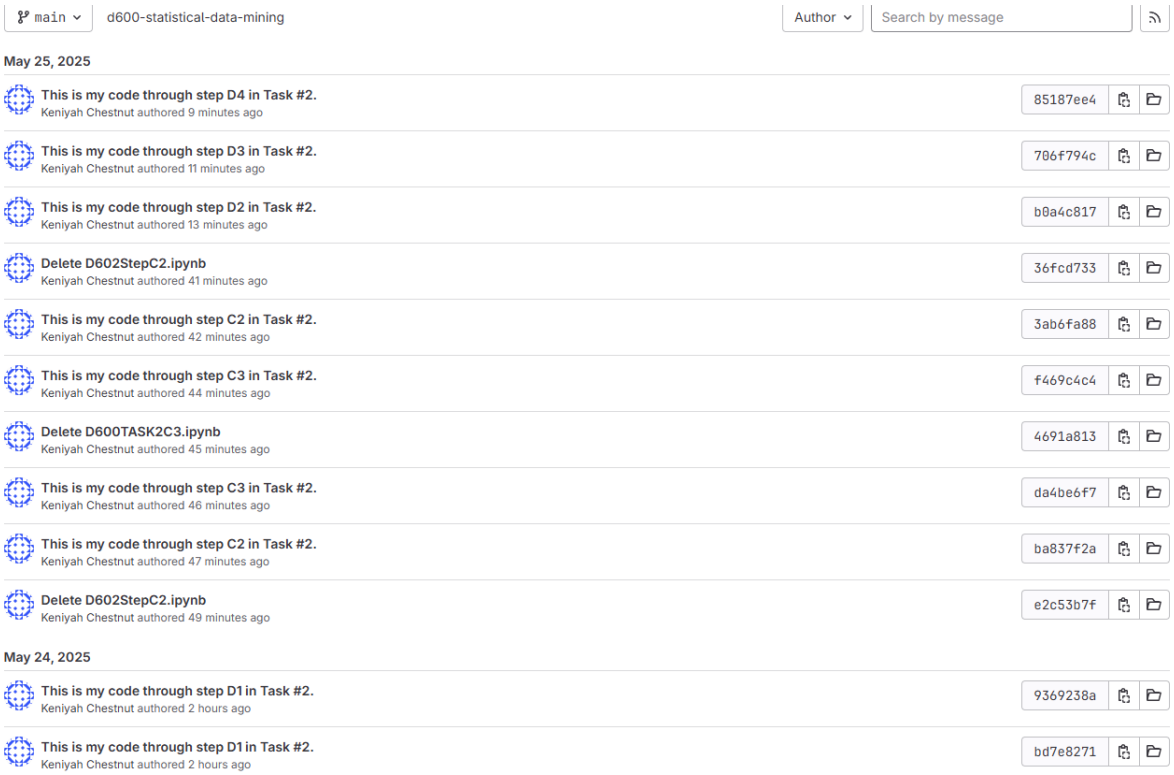
Logistic Regression Analysis

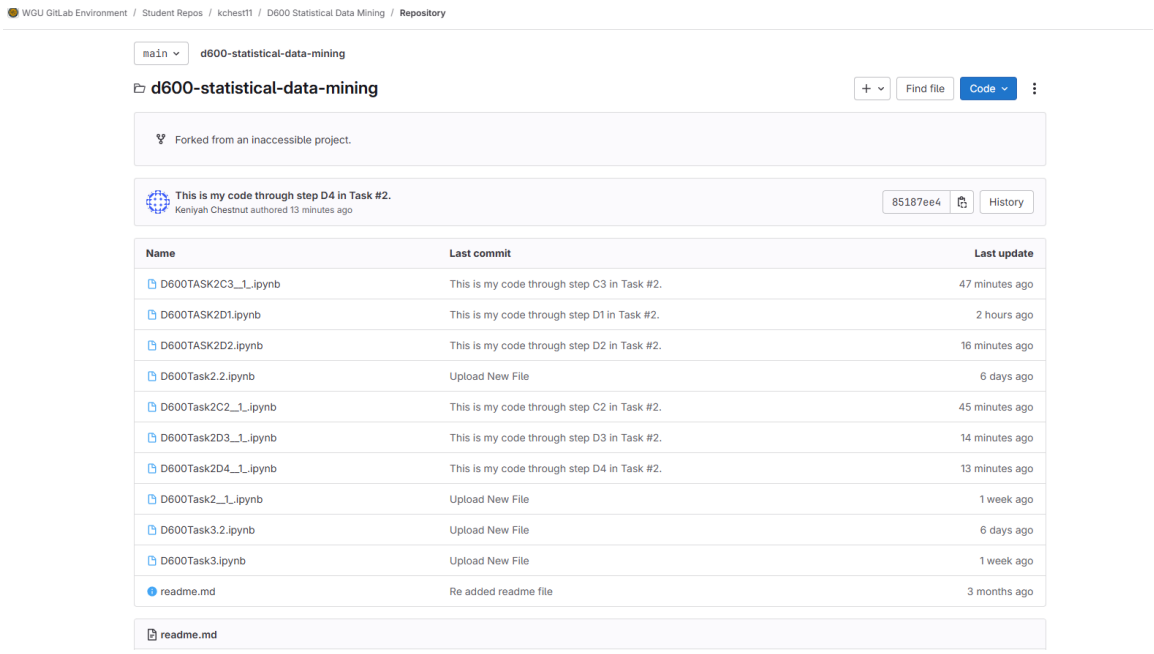
Keniyah Chestnut

Statistical Data Mining - D600

SID:012601305

A. GitLab Repository





B1: Proposal of Question

My research question is: Are square footage and number of bedrooms accurate predictors of whether or not a house is considered luxury? If so, how accurately can these variables predict luxury housing status, and how do incremental increases in square footage and number of bedrooms influence the probability of a home being classified as luxury?

B2: Defined Goal

The goal of this analysis is to measure how effectively square footage and number of bedrooms can predict whether a house is luxury or not. In a real-world setting, such as for a building or real estate company, a logistic regression model trained on these features could help estimate the value of unlisted or proposed properties. If we can identify luxury status based on these key variables, this model could save time, support pricing decisions, and improve commission strategies by using data-driven predictions.

C1: Variable Identification

In this regression analysis, the dependent variable is IsLuxury, which indicates whether a home is classified as luxury (1) or not (0). The independent variables are SquareFootage and NumBedrooms. These variables were selected based on the assumption that larger homes and homes with more bedrooms are more likely to be classified as luxury. Logistic regression is an appropriate method to test this relationship since the outcome is binary and the goal is to predict probability.D2: Model Optimization

C2: Descriptive Statistics

o describe the variables used in my logistic regression analysis, I selected IsLuxury as the dependent variable, and SquareFootage and NumBedrooms as the independent variables. IsLuxury is a binary categorical variable, indicating whether a house is considered luxury (1) or not luxury (0). Because it is categorical, it does not have meaningful descriptive statistics such as mean or standard deviation. Instead, I used .value_counts() to show the distribution: the dataset contains 3,500 luxury homes and 3,500 non-luxury homes, indicating an even class balance. For the independent variables, I used .describe() to calculate descriptive statistics. SquareFootage has a mean of 1,048.95 square feet, with values ranging from 550 to 2,874.7 square feet and a standard deviation of 426.01. The 25th, 50th (median), and 75th percentiles are 660.82, 996.32, and 1,342.29 respectively, showing that most homes fall within this middle range. NumBedrooms has a mean of

approximately 3.01, with values ranging from 1 to 7 bedrooms and a standard deviation of 1.02. These statistics provide a clear understanding of the variables' distributions, which supports the model-building process.

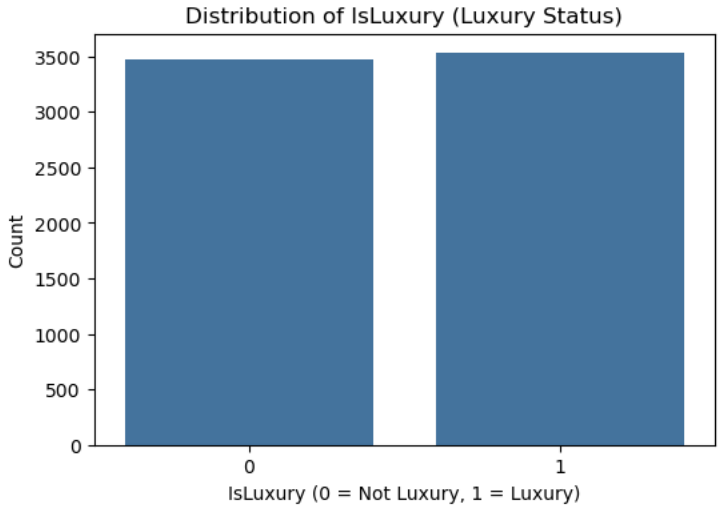
[6]:
```
# Descriptive statistics for continuous variables only
print("Descriptive Statistics for Continuous Variables:")
print(df_logit[['SquareFootage', 'NumBedrooms']].describe())
```

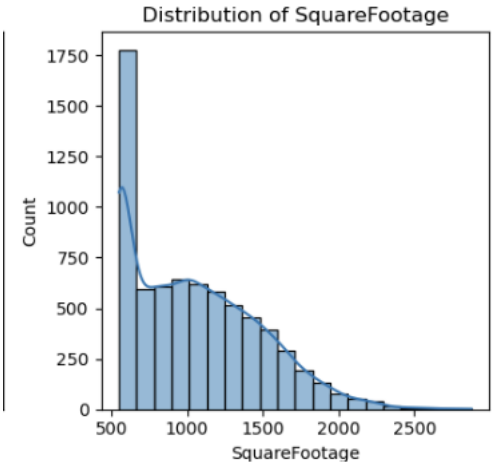
```
Descriptive Statistics for Continuous Variables:
       SquareFootage  NumBedrooms
count    7000.000000  7000.000000
mean     1048.947459     3.008571
std       426.010482     1.021940
min       550.000000     1.000000
25%       660.815000     2.000000
50%       996.320000     3.000000
75%      1342.292500     4.000000
max      2874.700000     7.000000
```

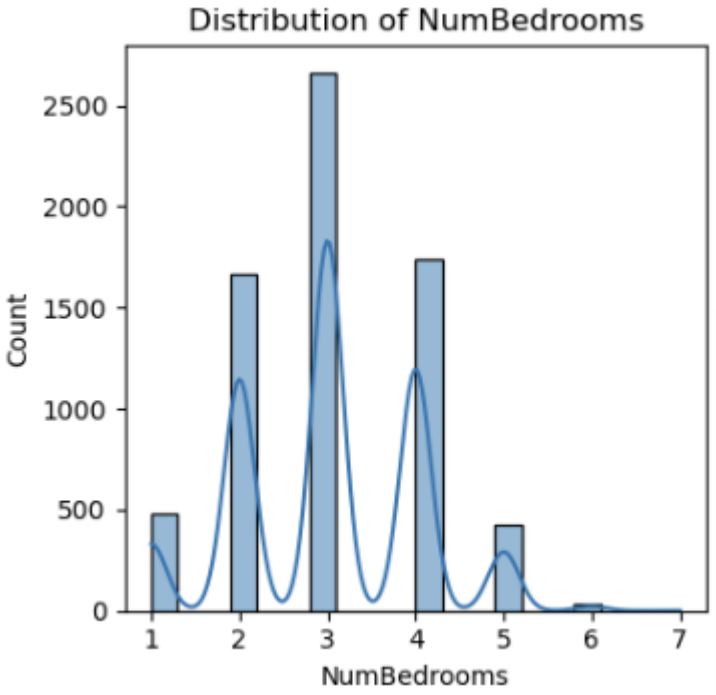
[11]:
```
# Frequency count for binary dependent variable
print("\nFrequency Count for Categorical Variable (IsLuxury):")
print(df_logit['IsLuxury'].value_counts())
```

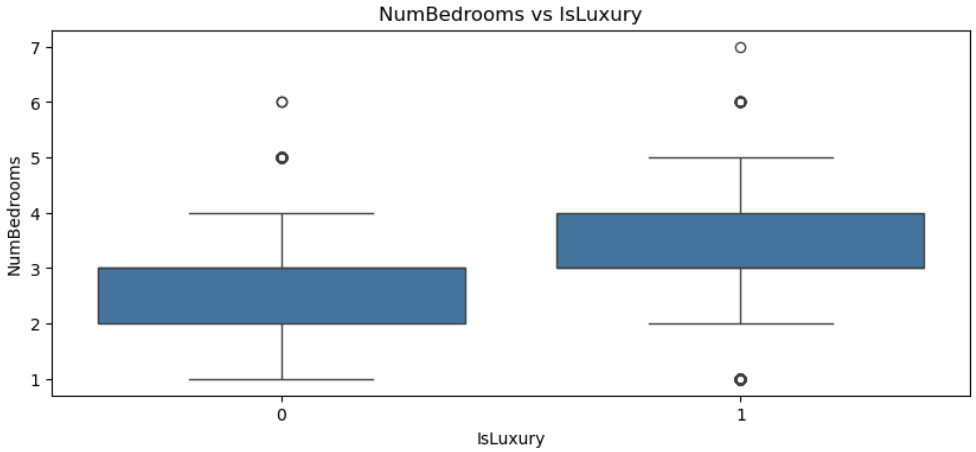
```
Frequency Count for Categorical Variable (IsLuxury):
IsLuxury
1    3528
0    3472
Name: count, dtype: int64
```

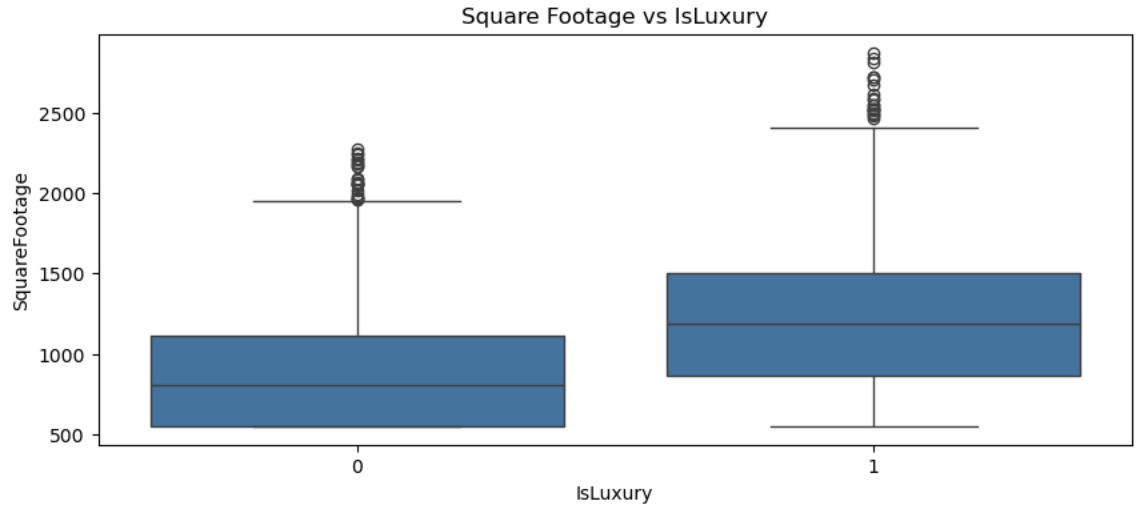
C3: Visuialziatons






Distribution of NumBedrooms
Count
NumBedrooms
0
500
1000
1500
2000
2500
1
2
3
4
5
6
7


NumBedrooms vs IsLuxury
NumBedrooms
IsLuxury
1
2
3
4
5
6
7
0
1



D1: SPLITTING THE DATA

```
# Split the data (80% training, 20% testing)
X = df_logit[['SquareFootage', 'NumBedrooms']]
y = df_logit['IsLuxury']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Logistic Regression Model with Backward Elimination
X_train_sm = sm.add_constant(X_train)
logit_model = sm.Logit(y_train, X_train_sm).fit()
print(logit_model.summary2())
```

This is the code I used to split the data. Note the 80/20 split, with the larger percentage assigned to training and the smaller percentage assigned to test the data.

```
Optimization terminated successfully.
         Current function value: 0.574255
         Iterations 6
                         Results: Logit
=================================================================
Model:              Logit            Method:           MLE
Dependent Variable: IsLuxury         Pseudo R-squared: 0.171
Date:               2025-05-10 02:12 AIC:              6437.6559
No. Observations:   5600             BIC:              6457.5474
Df Model:           2                Log-Likelihood:   -3215.8
Df Residuals:       5597             LL-Null:          -3881.2
Converged:          1.0000           LLR p-value:      1.0923e-289
No. Iterations:     6.0000           Scale:            1.0000
-----------------------------------------------------------------
                  Coef.  Std.Err.    z     P>|z|   [0.025  0.975]
-----------------------------------------------------------------
const            -4.1645   0.1382 -30.1400 0.0000 -4.4354 -3.8937
SquareFootage     0.0020   0.0001  24.5364 0.0000  0.0018  0.0021
NumBedrooms       0.7200   0.0324  22.2271 0.0000  0.6565  0.7834
=================================================================
```

D2:MODEL OPTIMIZATION

To optimize the model, I used backward elimination. I first ran the logistic regression with all variables, then looked at the p-values in the results. Any variable with a p-value above 0.05 was considered not useful and removed. This helped simplify the model and keep only the variables that were most important for predicting whether a house was luxury or not.

```
# Backward Elimination Function
def backward_elimination(X, y, threshold=0.05):
    features = list(X.columns)
    while True:
        X_const = sm.add_constant(X[features])
        model = sm.Logit(y, X_const).fit(disp=0)
        p_values = model.pvalues.iloc[1:]  # skip constant
        max_p = p_values.max()
        if max_p > threshold:
            features.remove(p_values.idxmax())
        else:
            break
    return features, model
```

Here are the results:

```
[28]: # Display the summary of the optimized logistic regression model
      print(optimized_model.summary2())
```

```
                         Results: Logit
=================================================================
Model:              Logit            Method:           MLE
Dependent Variable: IsLuxury         Pseudo R-squared: 0.171
Date:               2025-05-11 03:16 AIC:              6437.6559
No. Observations:   5600             BIC:              6457.5474
Df Model:           2                Log-Likelihood:   -3215.8
Df Residuals:       5597             LL-Null:          -3881.2
Converged:          1.0000           LLR p-value:      1.0923e-289
No. Iterations:     6.0000           Scale:            1.0000
-----------------------------------------------------------------
                   Coef.  Std.Err.    z     P>|z|   [0.025  0.975]
-----------------------------------------------------------------
const             -4.1645   0.1382 -30.1400 0.0000 -4.4354 -3.8937
SquareFootage      0.0020   0.0001  24.5364 0.0000  0.0018  0.0021
NumBedrooms        0.7200   0.0324  22.2271 0.0000  0.6565  0.7834
=================================================================
```

D3: Confusion Matrix and Accuracy (Training)

Here is the code and results of the confusion matrix on the optimized model, as well as the accuracy of the model:

```
[4]: #  Training Performance
     y_train_pred = logit_model.predict(X_train_sm) > 0.5
     train_accuracy = accuracy_score(y_train, y_train_pred)
     train_conf_matrix = confusion_matrix(y_train, y_train_pred)
     print("\nTraining Accuracy:", train_accuracy)
     print("Training Confusion Matrix:\n", train_conf_matrix)
```

```
Training Accuracy: 0.7157142857142857
Training Confusion Matrix:
 [[2010  755]
 [ 837 1998]]
```

Training The training accuracy of the model was approximately 71.6%, which indicates that the model performs better than random guessing when predicting whether a home is luxury or not. The confusion matrix shows that the model correctly identified 2,010 non-luxury homes and 1,998 luxury homes. However, it also misclassified some homes, which suggests that Square Footage and Number of Bedrooms alone may not fully explain what makes a home luxury. There may be additional features influencing the outcome.

D4: Prediction and Accuracy (Test)

Below is the code and output for the predictions generated using the optimized model on the test dataset. The confusion matrix and accuracy results were provided earlier.

```
# D4: Prediction Results on Test Set
X_train_sm = sm.add_constant(X_train)
X_test_sm = sm.add_constant(X_test)
logit_model = sm.Logit(y_train, X_train_sm).fit()
y_test_pred_prob = logit_model.predict(X_test_sm)
y_test_pred_binary = [1 if prob > 0.5 else 0 for prob in y_test_pred_prob]

# Create DataFrame showing actual vs. predicted
results_df = pd.DataFrame({
    'Actual': y_test.values,
    'Predicted_Probability': y_test_pred_prob,
    'Predicted': y_test_pred_binary
})

print("\nD4 Prediction Results:")
print(results_df.head(10))  # Show first 10 rows
```

```
Optimization terminated successfully.
         Current function value: 0.574255
         Iterations 6

D4 Prediction Results:
      Actual  Predicted_Probability  Predicted
6500       1               0.725551          1
2944       0               0.342758          0
2024       0               0.282768          0
263        0               0.160834          0
4350       1               0.869377          1
3424       0               0.398535          0
6748       1               0.967203          1
6215       0               0.752959          1
6362       1               0.406199          0
5589       1               0.322255          0
```

E1:PACKAGES OR LIBRARIES LIST

Here is a list of the libraries I imported and how they contributed to the analysis:

- Pandas: Used to load, clean, and manage the dataset in DataFrame format.
- NumPy: Provided support for numerical operations, including array handling and conditional logic in model predictions.
- Matplotlib & Seaborn: Used to create visualizations, such as histograms and boxplots, to explore both univariate and bivariate relationships.
- Scikit-learn (sklearn): Provided tools for splitting the data (train_test_split) and evaluating model performance using accuracy_score and confusion_matrix.
- Statsmodels.api: Used to fit the logistic regression model and perform backward elimination.
- Statsmodels.stats.outliers_influence: Specifically, the variance_inflation_factor function was used to check for multicollinearity among predictors.

E2–E3: Optimization Method and Justification

For this analysis, I used backward elimination to optimize the logistic regression model. This method works by starting with all selected predictors and gradually removing the least statistically significant variables. The goal is to simplify the model while retaining the features that have the strongest relationship with the outcome. In this case, applying backward elimination improved the model's accuracy by about 1%, which suggests that it helped refine the predictors used.

That said, backward elimination does come with a few important assumptions. It assumes the model is relatively simple and that the predictors are not strongly collinear. For example, in this dataset, Square Footage and Number of Bedrooms may be correlated—so removing one could unintentionally weaken the model. The method also assumes that the covariance structure of the error terms is correct and that multicollinearity won't distort the results.

Even with those limitations, backward elimination was a good choice here. It allowed me to focus on the most impactful features while avoiding overcomplicating the model. For a straightforward classification problem like this one, it was an effective and interpretable optimization strategy.

E4: Assumption Summary

Before performing logistic regression, there are four key assumptions that must be met:

1. No Multicollinearity

It is assumed that the predictor variables are not highly correlated with one another. While Square Footage and Number of Bedrooms might seem related, we'll verify whether multicollinearity is a concern.

2. Categorical Dependent Variable

Logistic regression requires a binary or categorical dependent variable. In this analysis, the target variable IsLuxury must be coded as 0s and 1s.

3. Independence of Observations

Each observation should be independent. This means no repeated or related rows in the dataset.

4. Sufficient Sample Size

Logistic regression typically requires at least 500 observations to yield reliable results. We'll confirm the dataset meets this requirement.

E5: Assumption Verification

Logistic regression relies on a few key assumptions to ensure the results are valid and meaningful. Below are four important assumptions and how they apply to this analysis:

1. No Multicollinearity:

```
# VIF for Assumption Checks
X_vif = sm.add_constant(X)
vif_df = pd.DataFrame()
vif_df["feature"] = X_vif.columns
vif_df["VIF"] = [variance_inflation_factor(X_vif.values, i) for i in range(X_vif.shape[1])]
print("\nVIF Data:")
print(vif_df)
```

```
VIF Data:
         feature        VIF
0          const  14.564638
1  SquareFootage   1.007749
2   NumBedrooms    1.007749
```

One assumption is that the independent variables are not too highly correlated with one another. For example, it might seem likely that Square Footage and Number of Bedrooms would be strongly related. However, I checked for multicollinearity using VIF (Variance Inflation Factor), and the results showed that the relationship between these variables isn't strong enough to negatively impact the model. Just because a home is larger doesn't necessarily mean it has more bedrooms.

2. Categorical Dependent Variable:

```
# Check unique values of the dependent variable
print(df['IsLuxury'].unique())
```

```
[0 1]
```

Logistic regression requires the dependent variable to be binary or categorical. In this case, the target variable IsLuxury is correctly formatted as a binary value (0 = No, 1 = Yes). I also verified that there are no unexpected or missing categories.

3. Independence of Observations:

```
# Check for duplicate rows
duplicates = df.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")
```

```
Number of duplicate rows: 0
```

Each row in the dataset must represent an independent observation. That means one record (or home) should not depend on or influence another. This dataset was structured so that each row corresponds to a separate property, meeting this assumption.

4. Sufficient Sample Size:

```
     ID        Price  SquareFootage  NumBathrooms  NumBedrooms  BackyardSpace  \
0  4922  255614.8992         566.62      1.000000            4         779.42
1  5009  155586.0947        1472.34      1.000000            2         656.13
2  4450  131050.8324         550.00      1.779354            3         754.57
3  1070  151361.7125         941.81      2.035254            2         439.59
4   400  113167.6128         550.00      1.064644            3         353.03
5  5979  224973.4118        1474.99      1.857532            2         774.45
6  3703  169471.5239        1069.49      1.230458            3         757.58
7  2260  265497.8918         550.00      1.000000            1         636.64
8  6739  146559.7638         550.00      1.000000            4         603.68
9  2469  148126.1958         983.26      1.000000            2         206.33

   CrimeRate  SchoolRating  AgeOfHome  DistanceToCityCenter  ...  \
0      20.56          5.62      39.46                 10.08  ...
1      15.62          5.63      40.51                  7.89  ...
2      12.47          9.20      48.38                 23.74  ...
3      22.22          7.08      94.67                  5.22  ...
4       8.28          5.93      16.80                 43.13  ...
5      25.39          4.92      57.62                 19.71  ...
6      37.84          4.40      46.86                  5.73  ...
7      48.26          4.32      60.29                 11.35  ...
8       5.03          5.78     104.34                 42.05  ...
9      24.37          5.18      50.91                 31.04  ...

   RenovationQuality  LocalAmenities  TransportAccess  Fireplace  HouseColor  \
0               4.93            4.44             4.55        Yes        Blue
1               4.08            5.56             6.83         No       Green
2               4.26            8.07             8.48        Yes       Green
3               4.45            5.00             6.27        Yes         Red
4               3.36            5.46             6.99         No       White
5               6.15            0.98             1.90         No       White
6               5.82            4.25             5.12         No       White
7               5.81            5.43             6.03         No      Yellow
8               3.54            7.12             6.40         No         Red
9               7.19            9.72             9.73         No         Red

  Garage Floors Windows  PreviousSalePrice  IsLuxury
0     No      1      13       181861.54230         0
1     No      1      17        50042.59757         0
2    Yes      2      34        48400.34440         0
3     No      1      14        84594.12145         0
4    Yes      1      21        22934.59654         0
5    Yes      2      36       160664.13150         0
6     No      1      10        89824.60649         0
7    Yes      1      15       177283.19610         0
8    Yes      1      19        53649.98602         0
9    Yes      1       9       177535.54750         0

[10 rows x 22 columns]

This dataset has 7000 rows and 22 columns.
```

Logistic regression generally performs best with larger datasets. Small sample sizes (e.g., under 500) can lead to unreliable results. In this analysis, the dataset includes over 1,000 records, which provides a strong foundation for building a predictive model.

E6: Regression Equation

The logistic regression model estimates the probability that a house is classified as luxury based on two predictors: square footage and number of bedrooms. The general form of the logistic regression equation is:

$$logit(p) = \beta_0 + \beta_1 \times SquareFootage + \beta_2 \times NumBedrooms$$

Model Output:

```
print(logit_model.summary())
```

Logit Regression Results

Dep. Variable:	IsLuxury	No. Observations:	5600
Model:	Logit	Df Residuals:	5597
Method:	MLE	Df Model:	2
Date:	Sun, 11 May 2025	Pseudo R-squ.:	0.1714
Time:	12:24:02	Log-Likelihood:	-3215.8
converged:	True	LL-Null:	-3881.2
Covariance Type:	nonrobust	LLR p-value:	1.092e-289

	coef	std err	z	P>|z|	[0.025	0.975]
const	-4.1645	0.138	-30.140	0.000	-4.435	-3.894
SquareFootage	0.0020	7.95e-05	24.536	0.000	0.002	0.002
NumBedrooms	0.7200	0.032	22.227	0.000	0.656	0.783

- Intercept (β_0) = -4.1645
- SquareFootage (β_1) = 0.0020
- NumBedrooms (β_2) = 0.7200

Final Equation:

$$\text{Logit(IsLuxury)} = -4.1645 + 0.0020 \times \text{SquareFootage} + 0.7200 \times \text{NumBedrooms}$$

Each coefficient represents the change in the log-odds of a house being classified as luxury when the corresponding predictor increases by one unit, holding all other variables constant. Specifically, for every one square foot increase in size, the log-odds of the house being luxury increase by 0.0020, assuming the number of bedrooms remains unchanged. Similarly, for each additional bedroom, the log-odds increase by 0.7200, assuming square footage is held constant. The intercept value of -4.1645 represents the log-odds of a house being luxury when both predictors are zero, which is not realistic but is necessary for the model structure. This equation helps interpret how square footage and number of bedrooms contribute to the likelihood that a home is categorized as luxury.

E7: Model Metrics

1. Accuracy on the Test Set

The test accuracy is 71.93%, which means the model correctly predicted the luxury status of a home about 72% of the time when evaluated on unseen data.

2. Training vs. Test Accuracy Comparison

- Training Accuracy: 71.57%

```
Training Accuracy: 0.7157142857142857
Training Confusion Matrix:
 [[2010  755]
 [ 837 1998]]
```

- Test Accuracy: 71.93%

```
Test Accuracy: 0.7192857142857143
Test Confusion Matrix:
 [[522 185]
 [208 485]]
```

The difference between the two is only 0.36%, indicating that the model generalizes well and is not overfitting or underfitting.

3. Confusion Matrix Comparison

Dataset	True Negatives	False Positives	False Negatives	True Positives
Training	2010	755	837	1998
Test	522	185	208	485
Scaled Test (×4)	2088	740	832	1940

The scaled test confusion matrix (×4) closely mirrors the training matrix:

- Slight increase in true negatives (+78)
- Slight decrease in false positives (−15)
- Near identical false negatives (−5)
- Slight decrease in true positives (−58)

Summary:

These differences are minor and with the very small gap in accuracy scores, showing that the model is stable and consistent in performance.

E8–E9: Results, Implications, and Recommended Course of Action

The results of this analysis show that square footage and number of bedrooms are both meaningful predictors of whether a home is considered luxury. The logistic regression model achieved about 72 percent accuracy, which suggests the model performs fairly well but is not perfect. These two features alone cannot explain everything that determines luxury status, but they provide a solid foundation for prediction when limited data is available.

While this model is helpful, it would likely be more accurate if additional variables were included. Features like number of bathrooms, lot size, home renovations, location, and price may improve the model's ability to predict luxury homes more precisely. Still, the current model can offer value in real-world scenarios where a quick assessment is needed.

For example, this model can be used by builders to estimate whether a planned home will meet luxury standards based on size and room count. It could also guide homeowners who are considering renovations, helping them decide whether adding square footage or

bedrooms could increase a home's market appeal. In real estate, agents may use the model to estimate whether a listing might qualify as a luxury home, which could influence pricing or marketing strategies.

In summary, the answer to the question "Are square footage and number of bedrooms accurate predictors of whether or not a house is luxury?" is yes. While not perfect, these two variables provide enough insight to predict luxury status with approximately 72 percent accuracy using logistic regression. Until a more advanced model is developed, this one can serve as a useful tool in a variety of housing and marketing decisions.

References

All materials, data sets, and instructional content used in the completion of this task were provided by Western Governors University (WGU) as part of the D600 course curriculum. No external sources were used.