Statistical Data Mining

Task 2

Nathan Hefner

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 logistic regression in the initial model.

The research question I will propose is: what do properties need to fulfill the requirements to be classified as a luxury?

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 this data analysis will be to find out if adding a garage and fireplace will turn a house into a luxury, along with the size of the property, the backyard space, the number of bedrooms, and the tax rate of the property and see if each of these variables can predict if a house will be considered a luxury. After finding the answer, we could relay this information to builders so they know how to price the house if it is considered a luxury.

C. Summarize the data preparation process for logistic regression analysis by doing the following:

1. Identify the dependent and all independent variables that are required to answer the research question and justify your selection of variables.

Our independent variables will be fireplace, garage, property size, number of bedrooms, backyard space, and tax rate. Our dependent variable will be whether the property is Luxury or not. We chose this as our dependent variable because it relies on our other variables, and logistical regression requires the dependent variable to be a binary categorical data type. The other variables are our independent variables because they affect our dependent variable. I chose Fireplace and Garage for the independent variables because they are not necessary for every property, I also believe a higher property space, backyard space, tax rate, and amount of bedrooms are good indicators of whether they make a house a luxury or not.

2. Describe the dependent variable and all independent variables from part C1 using descriptive statistics (counts, means, modes, ranges, min/max), including a screenshot of the descriptive statistics output for each of these variables.

Below is my code for descriptive analysis of my variables.

```
columns = ['IsLuxury', 'SquareFootage', 'BackyardSpace', 'NumBedrooms', 'PropertyTaxRate']
stats = df[columns].describe()
print(stats)
```

```
         IsLuxury  SquareFootage  BackyardSpace  NumBedrooms  PropertyTaxRate
count  7000.00000    7000.000000    7000.000000  7000.000000      7000.000000
mean      0.50400    1048.947459     511.507029     3.008571         1.500437
std       0.50002     426.010482     279.926549     1.021940         0.498591
min       0.00000     550.000000       0.390000     1.000000         0.010000
25%       0.00000     660.815000     300.995000     2.000000         1.160000
50%       1.00000     996.320000     495.965000     3.000000         1.490000
75%       1.00000    1342.292500     704.012500     4.000000         1.840000
max       1.00000    2874.700000    1631.360000     7.000000         3.360000
```

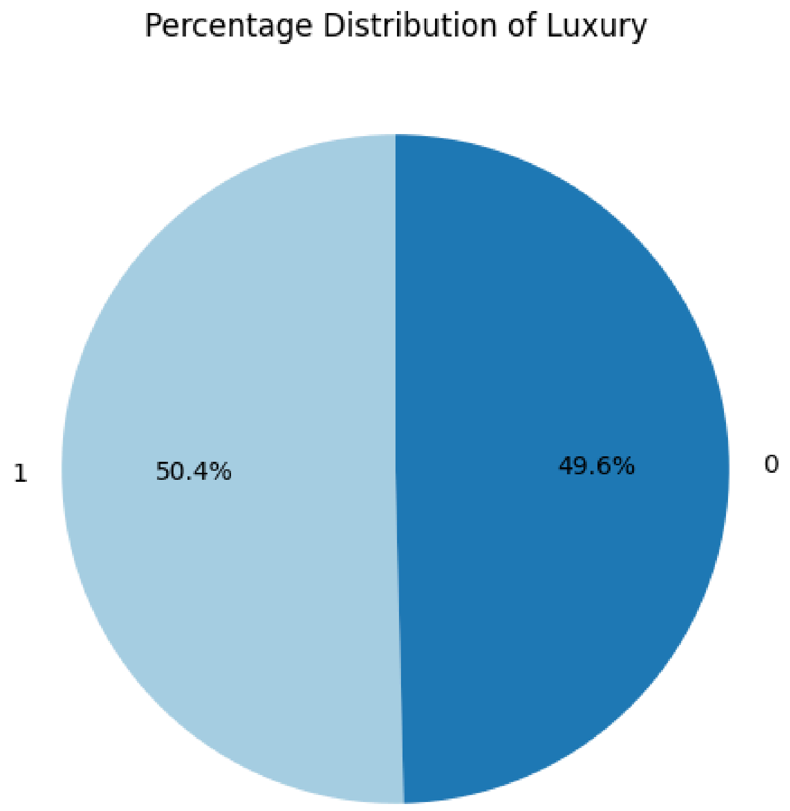
For the Garage, Fireplace, and IsLuxury variables, since they are binary categorical data types, I coded a percentage frequency table to help better describe them. For IsLuxury 1 means true, and 0 means false.

```
percentages_garage = df['Garage'].value_counts(normalize=True).reset_index()
percentages_garage.columns = ['Garage', 'Percentage']
percentages_garage['Percentage'] *= 100

percentages_fireplace = df['Fireplace'].value_counts(normalize=True).reset_index()
percentages_fireplace.columns = ['Fireplace', 'Percentage']
percentages_fireplace['Percentage'] *= 100

percentages_luxury = df['IsLuxury'].value_counts(normalize=True).reset_index()
percentages_luxury.columns = ['IsLuxury', 'Percentage']
percentages_luxury['Percentage'] *= 100

print("Garage Percentages:")
print(percentages_garage)

print("Fireplace Percentages:")
print(percentages_fireplace)

print("Luxury Percentages:")
print(percentages_luxury)
```

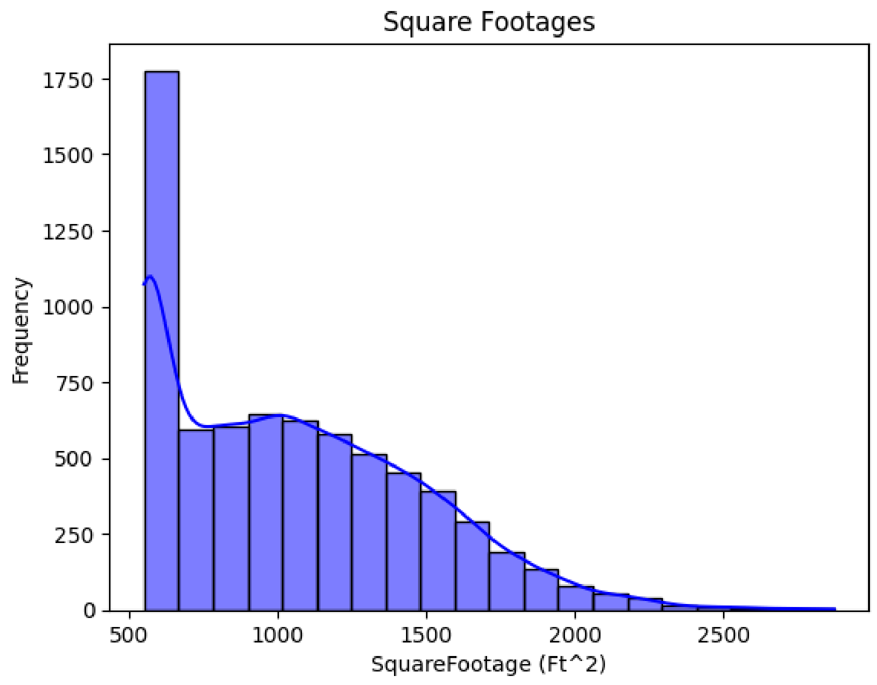
```
Garage Percentages:
  Garage  Percentage
0     No   64.114286
1    Yes   35.885714
Fireplace Percentages:
  Fireplace  Percentage
0        No   73.885714
1       Yes   26.114286
Luxury Percentages:
   IsLuxury  Percentage
0         1        50.4
1         0        49.6
```

3. Generate univariate and bivariate visualizations of the distributions of the dependent and independent variables from part C1, including the dependent variable in the bivariate visualizations.

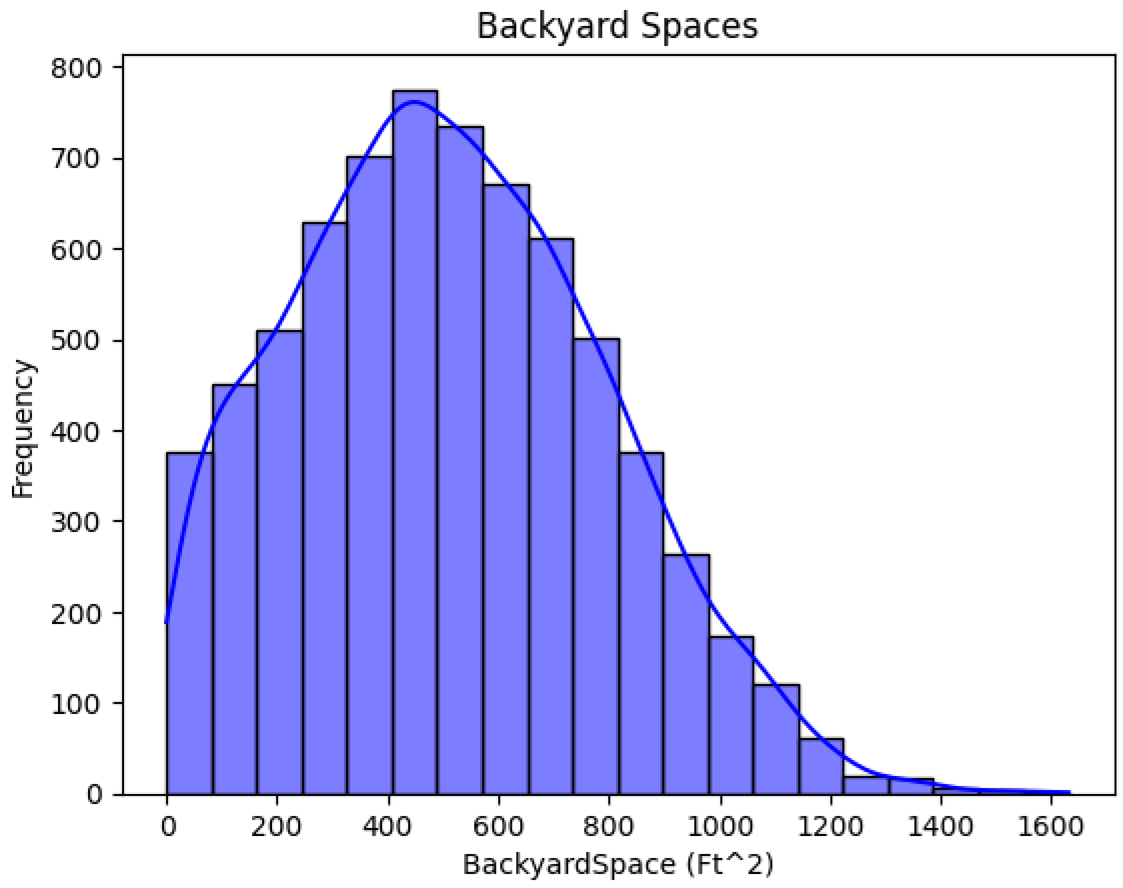
Below is my univariate visualization for the IsLuxury variable.



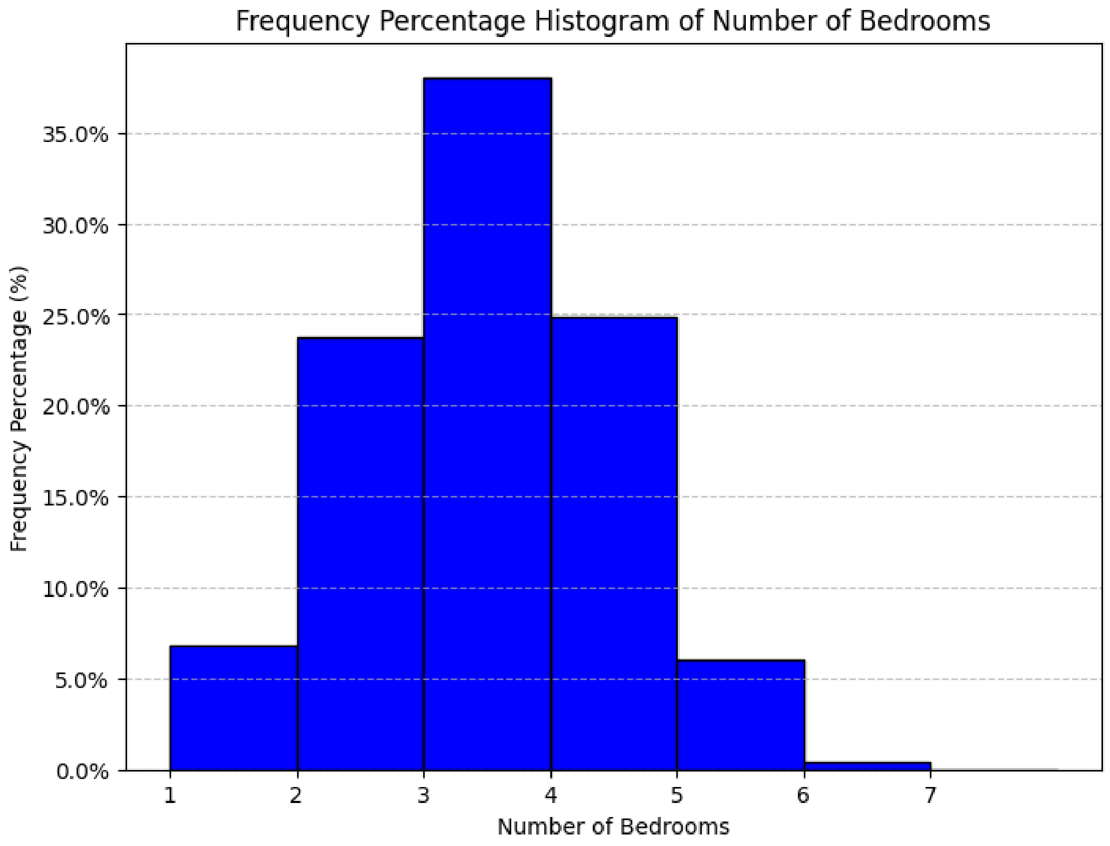
Below is my univariate visualization for the Square Footage variable.



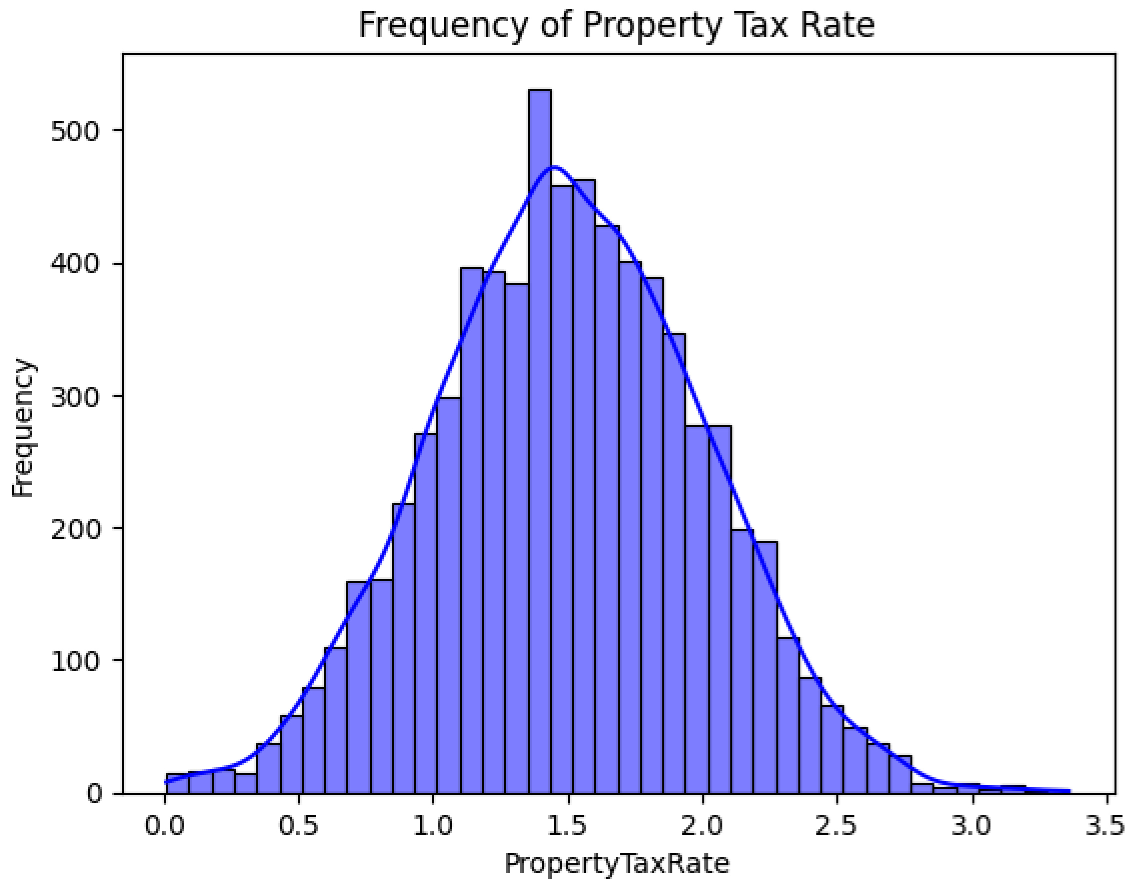
Below is my univariate visualization for the Backyard Space variable.



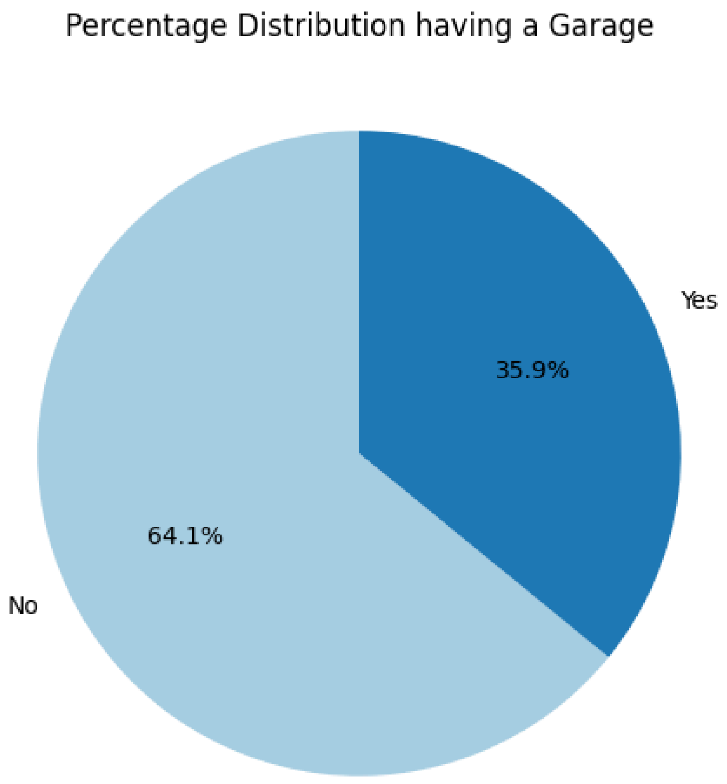
Below is my univariate visualization for the Number of Bedrooms variable.



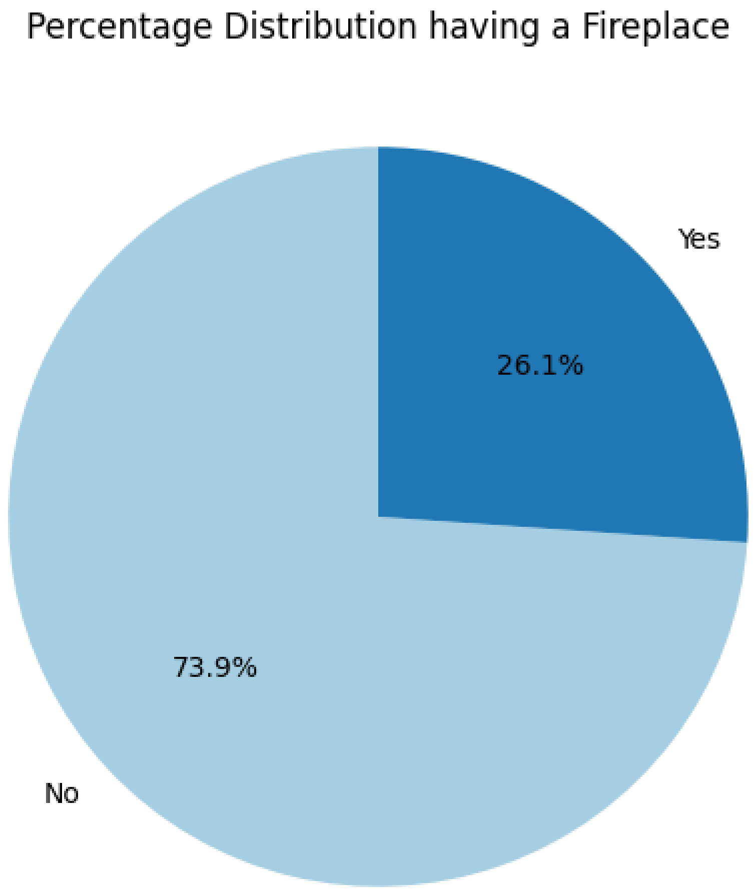
Below is my univariate visualization for the Property Tax Rate variable.



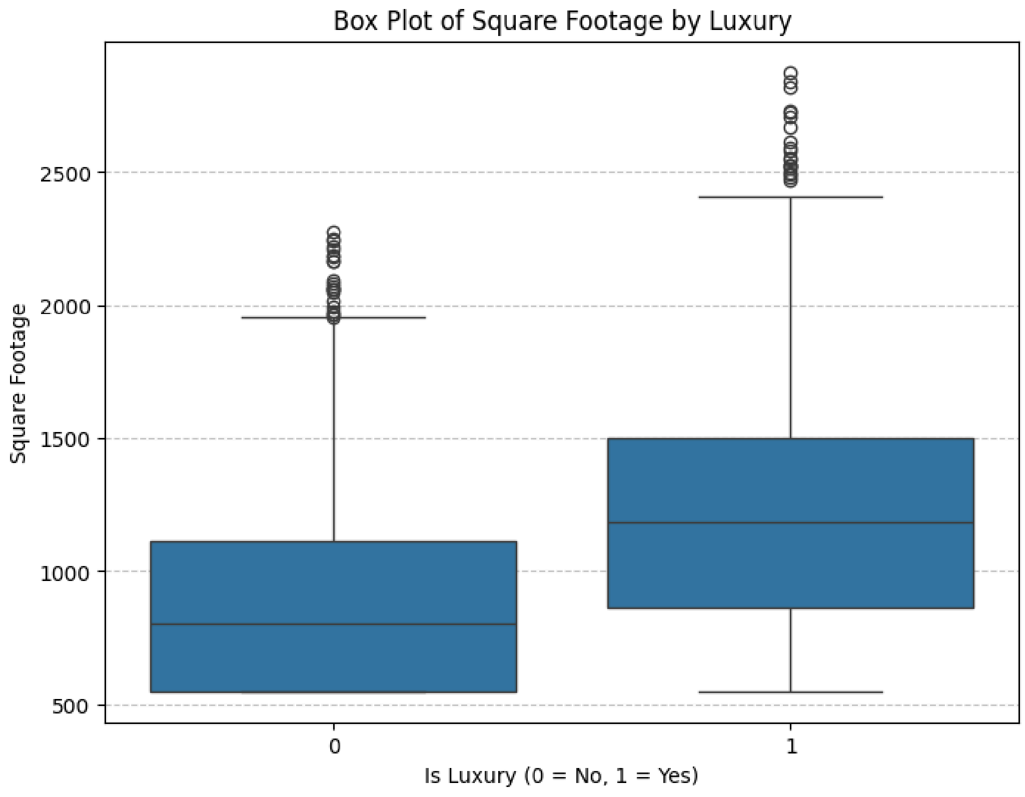
Below is my univariate visualization for the Garage variable.



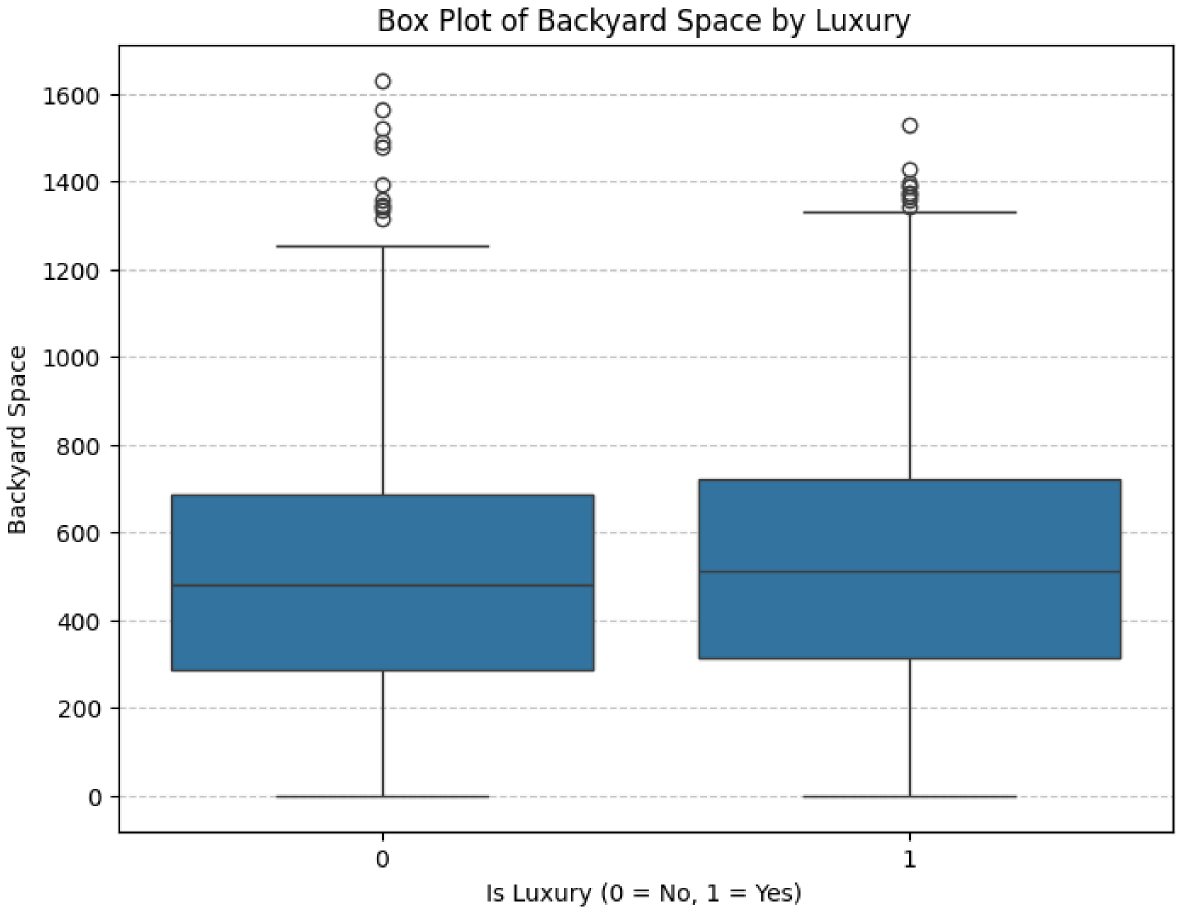
Below is my univariate visualization for the Fireplace variable.



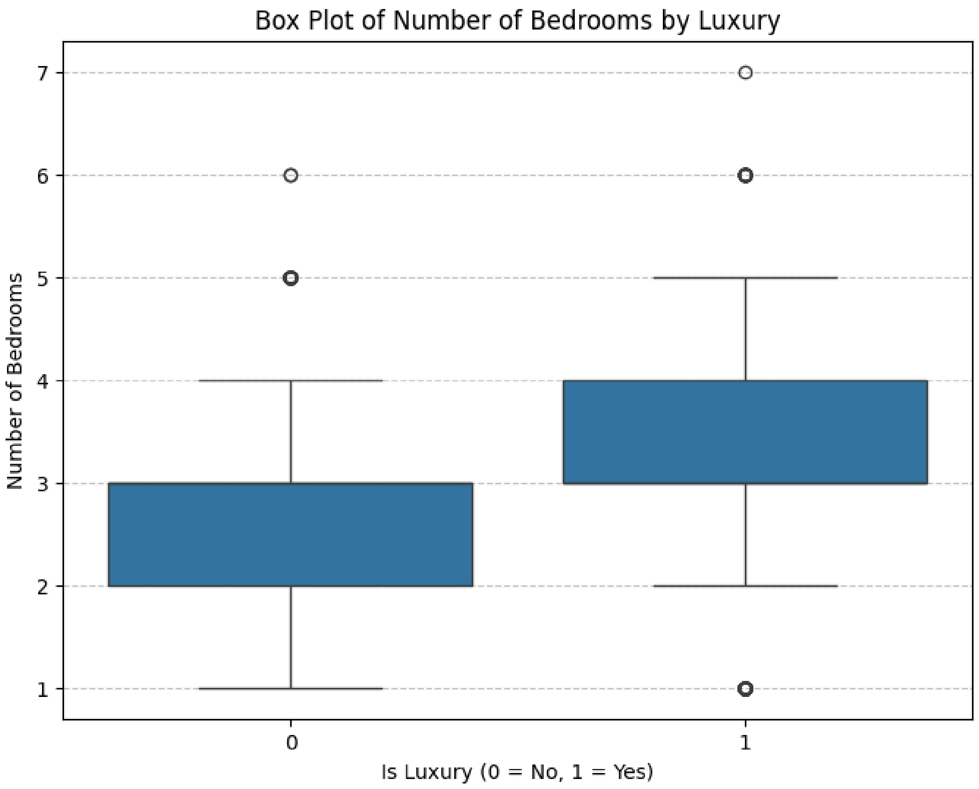
Below is my bivariate graph for Square Footage versus Luxury.



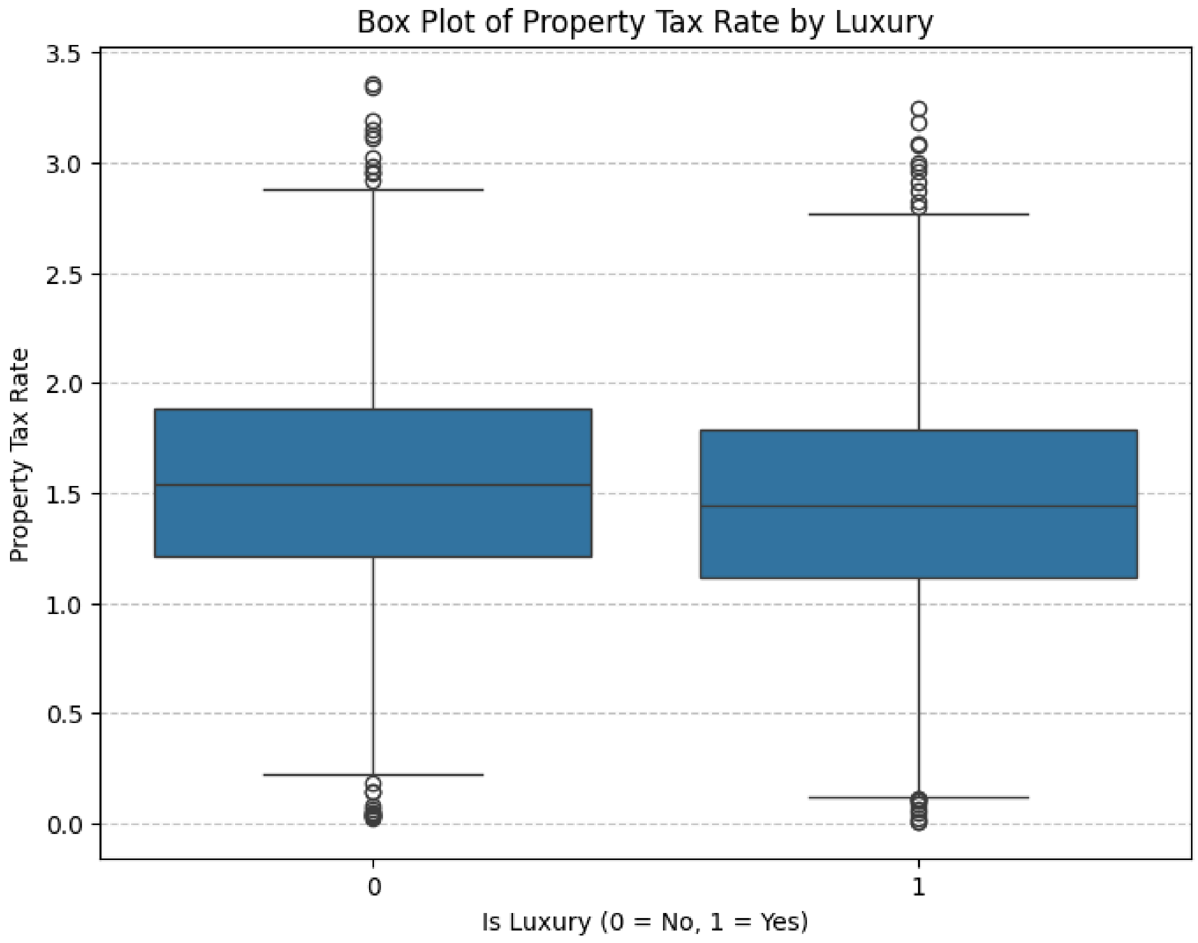
Below is my bivariate graph for Backyard versus Luxury.



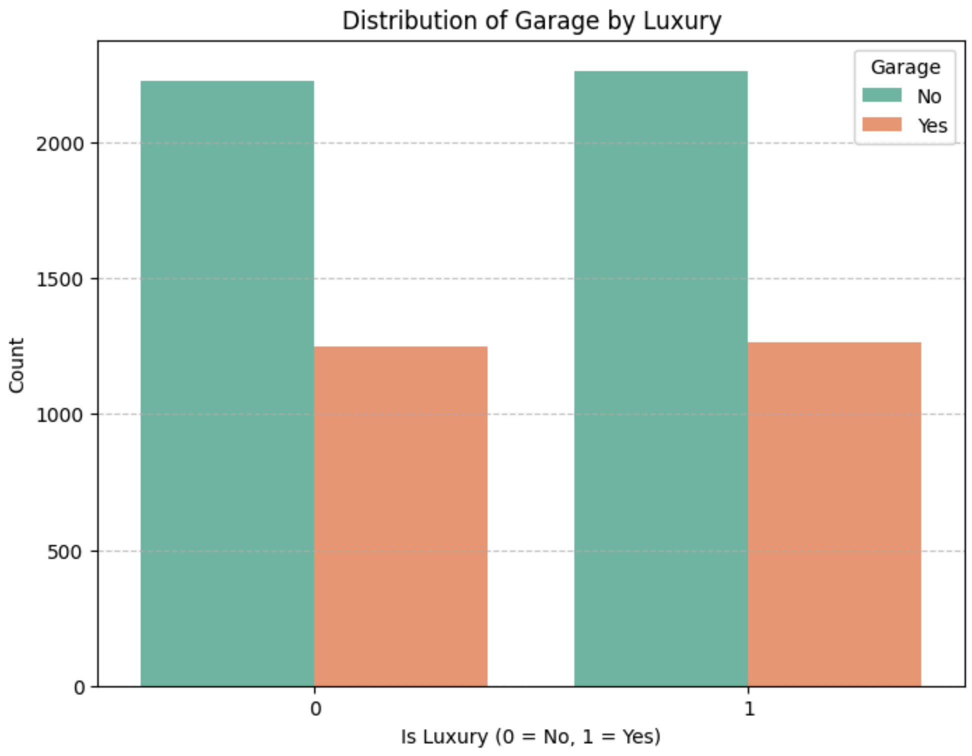
Below is my bivariate graph for Bedrooms versus Luxury.



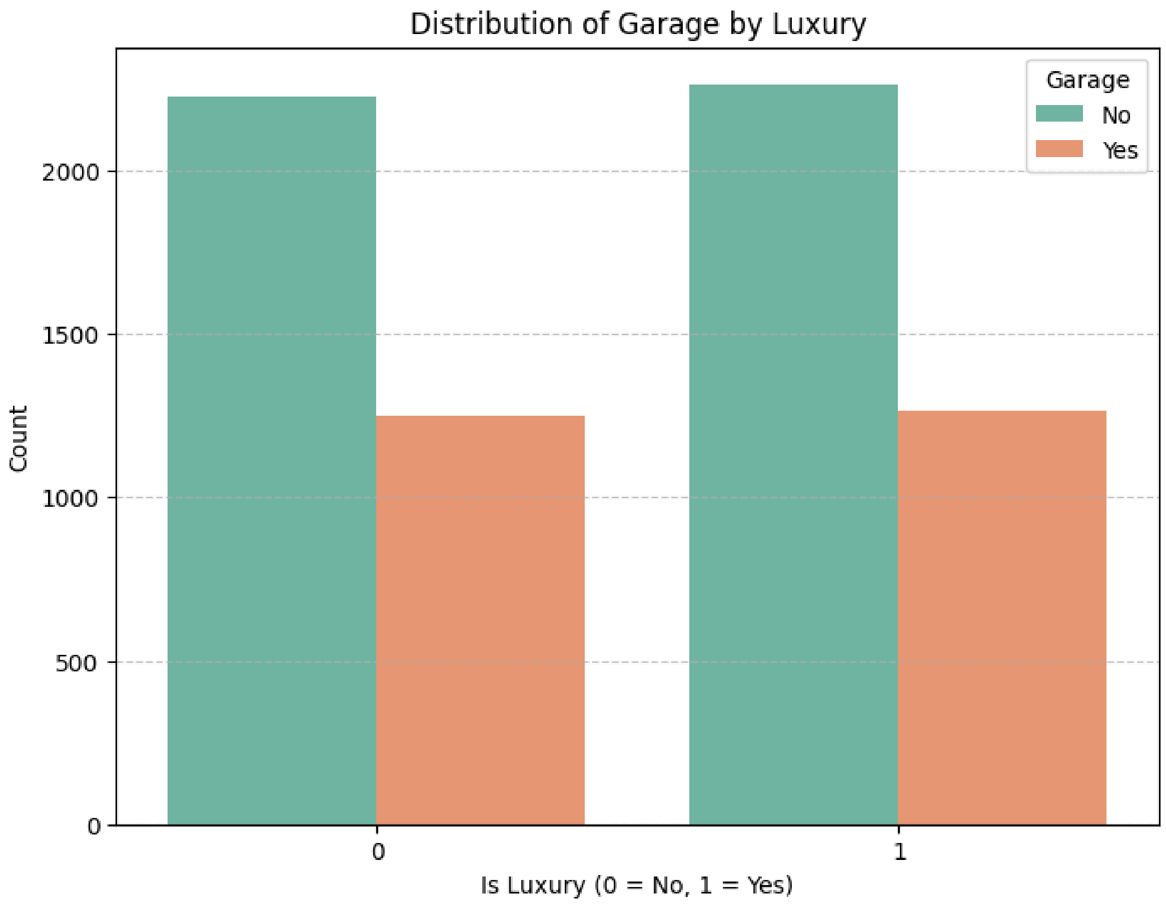
Below is my bivariate graph for Tax Rate versus Luxury.



Below is my bivariate graph for Garage versus Luxury.



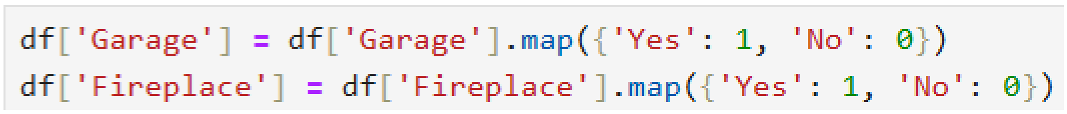
Below is my bivariate graph for Fireplace versus Luxury.



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

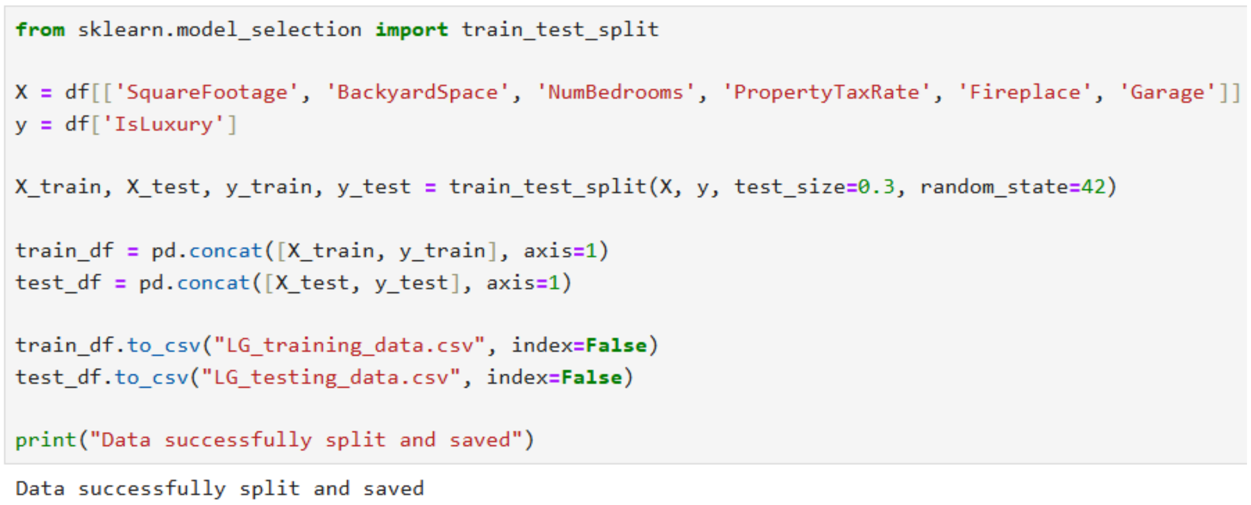
1. Split the data into two datasets, with a larger percentage assigned to the training dataset and a smaller percentage assigned to the test dataset. Provide the file(s).

Before splitting the data, I encoded my categorical columns and turned them into binary answers. Below is the code.



```
df['Garage'] = df['Garage'].map({'Yes': 1, 'No': 0})
df['Fireplace'] = df['Fireplace'].map({'Yes': 1, 'No': 0})
```

Below is my code for splitting the data. I did a 70/30 split.

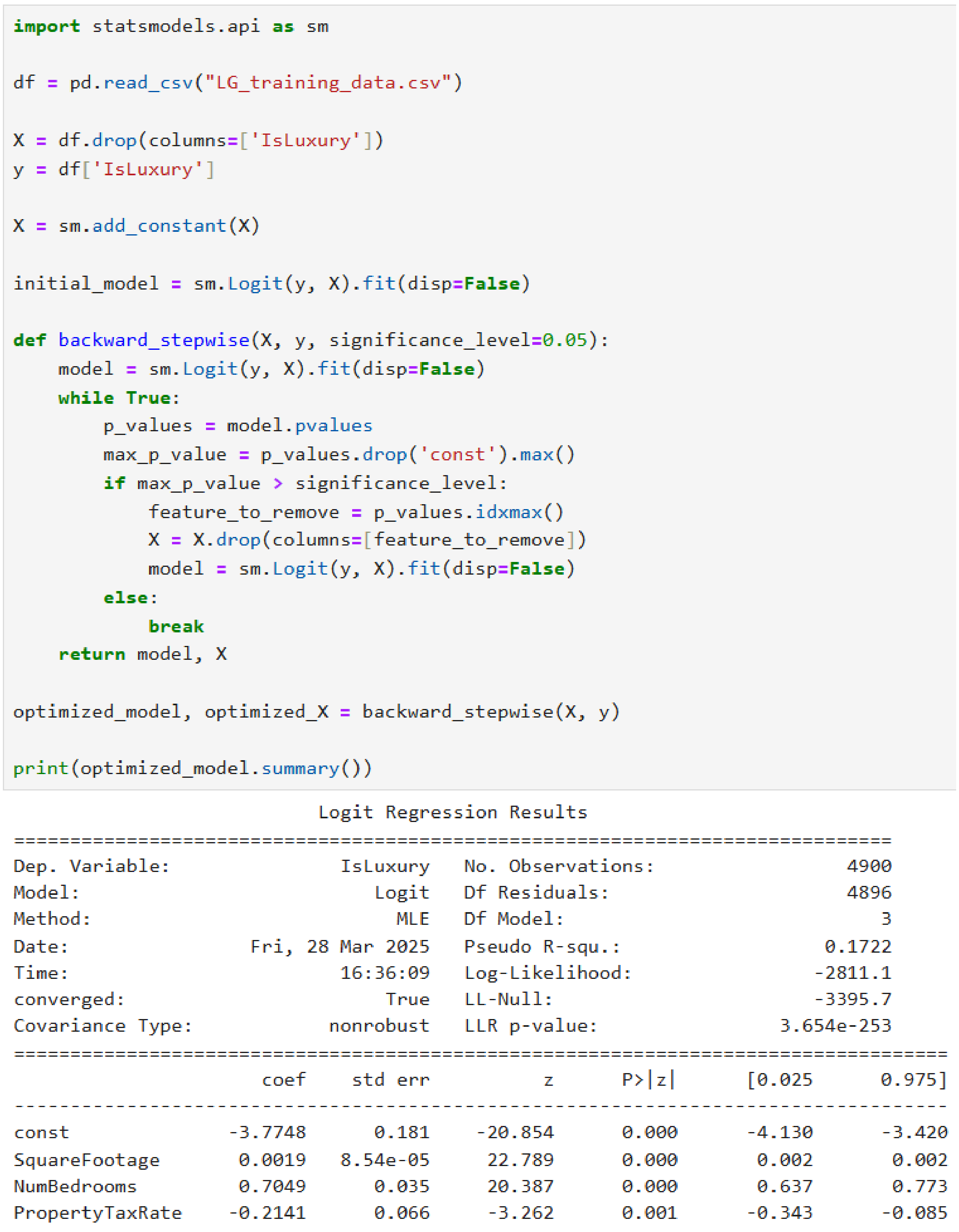


```
from sklearn.model_selection import train_test_split

X = df[['SquareFootage', 'BackyardSpace', 'NumBedrooms', 'PropertyTaxRate', 'Fireplace', 'Garage']]
y = df['IsLuxury']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

train_df = pd.concat([X_train, y_train], axis=1)
test_df = pd.concat([X_test, y_test], axis=1)

train_df.to_csv("LG_training_data.csv", index=False)
test_df.to_csv("LG_testing_data.csv", index=False)

print("Data successfully split and saved")
```

```
Data successfully split and saved
```

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.

Below is my code for the model and optimizing using the Backward Stepwise Elimination optimization method. Below are the results.



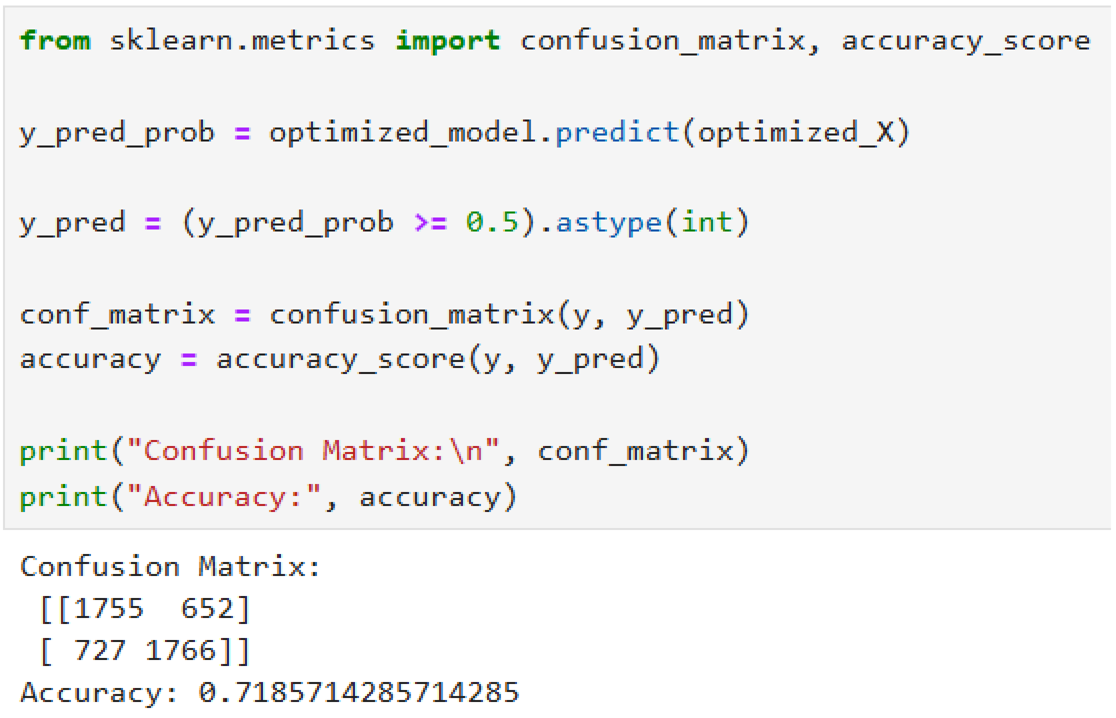
```python
import statsmodels.api as sm

df = pd.read_csv("LG_training_data.csv")

X = df.drop(columns=['IsLuxury'])
y = df['IsLuxury']

X = sm.add_constant(X)

initial_model = sm.Logit(y, X).fit(disp=False)

def backward_stepwise(X, y, significance_level=0.05):
    model = sm.Logit(y, X).fit(disp=False)
    while True:
        p_values = model.pvalues
        max_p_value = p_values.drop('const').max()
        if max_p_value > significance_level:
            feature_to_remove = p_values.idxmax()
            X = X.drop(columns=[feature_to_remove])
            model = sm.Logit(y, X).fit(disp=False)
        else:
            break
    return model, X

optimized_model, optimized_X = backward_stepwise(X, y)

print(optimized_model.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:               IsLuxury   No. Observations:                 4900
Model:                          Logit   Df Residuals:                     4896
Method:                           MLE   Df Model:                            3
Date:                Fri, 28 Mar 2025   Pseudo R-squ.:                  0.1722
Time:                        16:36:09   Log-Likelihood:                -2811.1
converged:                       True   LL-Null:                       -3395.7
Covariance Type:            nonrobust   LLR p-value:                 3.654e-253
===================================================================================
                      coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------
const              -3.7748      0.181    -20.854      0.000      -4.130      -3.420
SquareFootage       0.0019   8.54e-05     22.789      0.000       0.002       0.002
NumBedrooms         0.7049      0.035     20.387      0.000       0.637       0.773
PropertyTaxRate    -0.2141      0.066     -3.262      0.001      -0.343      -0.085
```

We set the value to 0.05 henceforth, the model will drop any variables that it deems insignificant, and in this case, it dropped the Garage, Fireplace, and Backyard Space variables.

3. Give the confusion matrix and accuracy of the optimized model used on the training set.

Below is my code for calculating the accuracy of the confusion matrix.



```python
from sklearn.metrics import confusion_matrix, accuracy_score

y_pred_prob = optimized_model.predict(optimized_X)

y_pred = (y_pred_prob >= 0.5).astype(int)

conf_matrix = confusion_matrix(y, y_pred)
accuracy = accuracy_score(y, y_pred)

print("Confusion Matrix:\n", conf_matrix)
print("Accuracy:", accuracy)
```

```
Confusion Matrix:
 [[1755  652]
 [ 727 1766]]
Accuracy: 0.7185714285714285
```

This is what the matric represents:

$$[TN\ FP]$$
$$[FN\ TP]$$

TN is True negatives which means amount of correctly predicted 0s
FP is False positives which means amount of incorrectly predicted 1s
FN is False negatives which means amount of incorrectly predicted 0s
TP is True positives which means amount of correctly predicted 1s

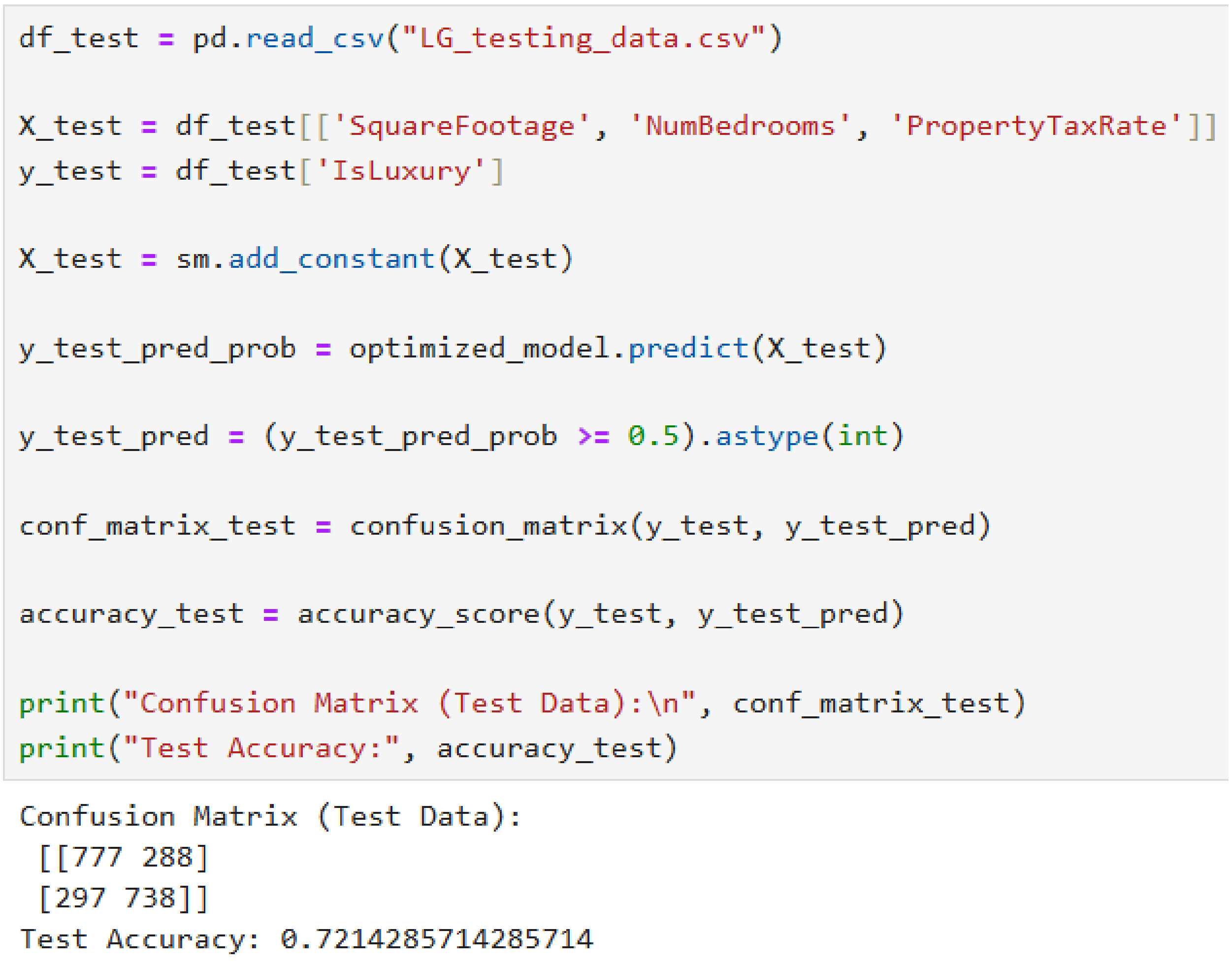
The equation to calculate Accuracy is

$$Accuracy = \frac{TP + TN}{Total}$$

Our accuracy is near 72%, which is well-regarded, but it also means our model could use some fine tweaking.

4. Run the prediction on the test dataset using the optimized regression model from part D2 to evaluate the performance of the prediction model on the test data based on the confusion matrix and accuracy. Provide a screenshot of the results.

Below is my code on the test data set’s confusion matrix and accuracy.



```
df_test = pd.read_csv("LG_testing_data.csv")

X_test = df_test[['SquareFootage', 'NumBedrooms', 'PropertyTaxRate']]
y_test = df_test['IsLuxury']

X_test = sm.add_constant(X_test)

y_test_pred_prob = optimized_model.predict(X_test)

y_test_pred = (y_test_pred_prob >= 0.5).astype(int)

conf_matrix_test = confusion_matrix(y_test, y_test_pred)

accuracy_test = accuracy_score(y_test, y_test_pred)

print("Confusion Matrix (Test Data):\n", conf_matrix_test)
print("Test Accuracy:", accuracy_test)
```

```
Confusion Matrix (Test Data):
 [[777 288]
 [297 738]]
Test Accuracy: 0.7214285714285714
```

The accuracy for the test dataset is very similar to the training dataset, which means the model has good generalization and is not overfitting or underfitting. However, with lower accuracy, our model could be incomplete and may need more features to increase complexity and further increase the accuracy value.

E. 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.

Here are all the Python packages I used and the reason for using them.

- Pandas - Allows us to import, read, and change datasets
- Sklearn.model_selection - Allows us to split data into separate training and test sets

- Statsmodels.api - Allows us to perform regression and extract the model parameters
- Sklearn.feature_selection - Allows us to optimize the model using Recursive Feature Elimination
- Sklearn.metrics - Allows us to calculate the confusion matrix and accuracy score
- Sklearn.linear_model - Allows us to use linear regression models
- Seaborn/Matplotlib.pyplot - Allows us to use graph and visual different data relationships

2. Discuss the method used to optimize the model. / 3. Justify the approach discussed in part E2 that was used to optimize the model.

We used the Backward Stepwise Elimination method to optimize the dataset, which helps remove the least significant features. This helps improve the model performance and prevent overfitting by eliminating unnecessary predictors. It relies on p-values to determine which insignificant predictors to remove. In our case any p-value greater than 0.05 will be dropped, thus simplifying the model by keeping the most significant features. We used BSE because our dataset is large enough, and variables can be removed without significantly harming the integrity of the dataset.

4. Summarize at least four assumptions of logistic regression.

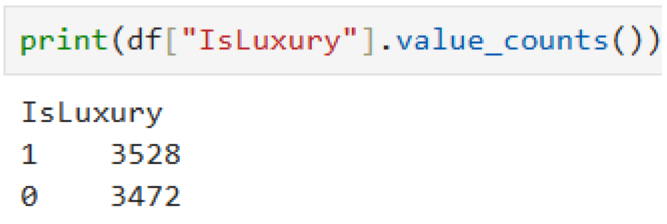
Here are four assumptions of logistic regression

1. The dependent variable must be categorical and binary.
2. There must be a sufficient sample size
3. There must be no multicollinearity between the variables
4. Each observation in the dataset should be independent of the others

5. Provide evidence that the assumptions from part E4 were verified by providing either a code snippet or a screenshot.

Below is the code and evidence to support the assumptions mentioned before.

1. The dependent variable must be categorical and binary. In our case, we chose IsLuxury variable because the values were 1 and 0, as shown below

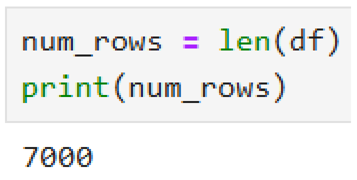


```
print(df["IsLuxury"].value_counts())
```

```
IsLuxury
1    3528
0    3472
```

This shows that the column is binary.

2. The data frame must be sufficient in size, or there might be overfitting from lack of data. We will show the data frame size below

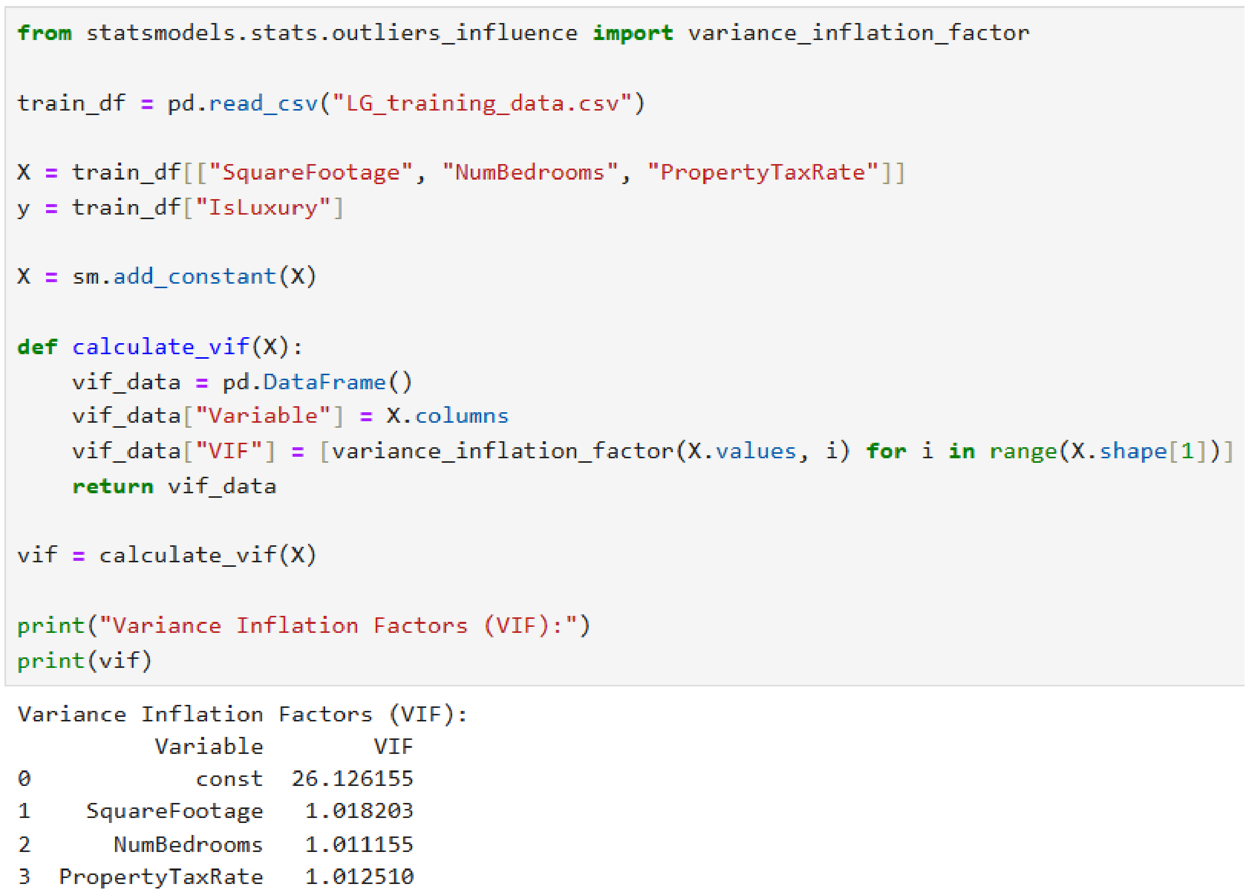


```python
num_rows = len(df)
print(num_rows)
```

```
7000
```

7000 rows is a sufficient amount of data

3. We will use the variance inflation factor command to check for multicollinearity between the variables. Any value less than 5 will be sufficient.

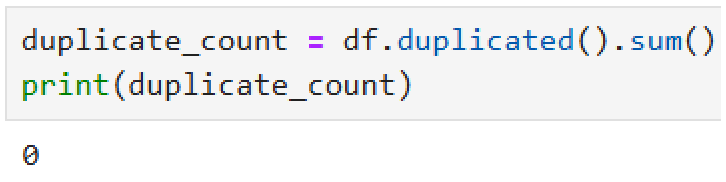


```python
from statsmodels.stats.outliers_influence import variance_inflation_factor

train_df = pd.read_csv("LG_training_data.csv")

X = train_df[["SquareFootage", "NumBedrooms", "PropertyTaxRate"]]
y = train_df["IsLuxury"]

X = sm.add_constant(X)

def calculate_vif(X):
    vif_data = pd.DataFrame()
    vif_data["Variable"] = X.columns
    vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
    return vif_data

vif = calculate_vif(X)

print("Variance Inflation Factors (VIF):")
print(vif)
```

```
Variance Inflation Factors (VIF):
          Variable        VIF
0            const  26.126155
1    SquareFootage   1.018203
2      NumBedrooms   1.011155
3  PropertyTaxRate   1.012510
```

The values being close to 1 show that there is no multicollinearity.

4. Each observation in the dataset has to be unique. The best way to prove this is to see if any rows of data are repeats or are redundant.



```python
duplicate_count = df.duplicated().sum()
print(duplicate_count)
```

```
0
```

With no duplicates, we can be sure that our datasets are independent.

6. Provide the regression equation and discuss the coefficient estimates

Below is the regression equation

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_n X_n$$

- p is the probability of the outcome being 1
- β_0 is the intercept
- β_n is the coefficient value
- X_n is the independent variables

With our values calculated from the optimized dataset, we can plug the values in and find the equation for this dataset.

The constant coefficient, β_0, equals -39,110. This value will be our intercept.

The Square Footage coefficient, β_1, equals 0.0019. This means for every square foot added to the property, the log odds of the property being a luxury increases by 0.0019

The Number of Bedrooms coefficient, β_2, equals 0.7049. This means for every bedroom added to the property, the log odds of the property being a luxury increases by 0.7049

The Property Tax Rate coefficient, β_3, equals -0.2141. This means when the tax rateincreases, the log odds of the property being a luxury decreases by 0.2141

Subbing in variables where needed this will be our final equation

$$\log\left(\frac{p}{1-p}\right) = -3.7748 + (0.0019 \times \mathit{SquareFootage}) + (0.7049 \times \mathit{NumBedrooms}) - (0.2141 \times \mathit{PropertyTaxRate})$$

7. Discuss the model metrics by addressing each of the following:

- **the accuracy for the test set**
- **the comparison of the accuracy of the training set to the accuracy of the test set**
- **the comparison of the confusion matrix for the training set to the confusion matrix of the test set**

The accuracy for the test set is 0.71857, rounded to 71.86%, meaning the model correctly predicted the correct value for the IsLuxury column 72 percent of the time.

The accuracy for the training set is 0.72142, rounded to 72.14%. The difference between the two values is 0.28 With a slim difference, the model is near optimal, but there may be some slight underfitting, meaning the model may not be complex enough.

The confusion matrix for the training set is

```
[1755 652]
[727 1766]
```

Which means the model:

- Correctly predicted 1755 True Negatives
- Incorrectly predicted 652 False Positives
- Incorrectly predicted 727 False Negatives
- Correctly predicted 1766 True Positives

The confusion matrix for the testing set is

[777 288]
[297 738]

But since we did a 70/30 split we must scale this matrix accordingly by multiplying by 2.3 and rounding where needed. After doing so here is the scaled matrix

[1787 662]
[683 1697]

Which means the model:

- Correctly predicted 1787 True Negatives
- Incorrectly predicted 662 False Positives
- Incorrectly predicted 683 False Negatives
- Correctly predicted 1697 True Positives

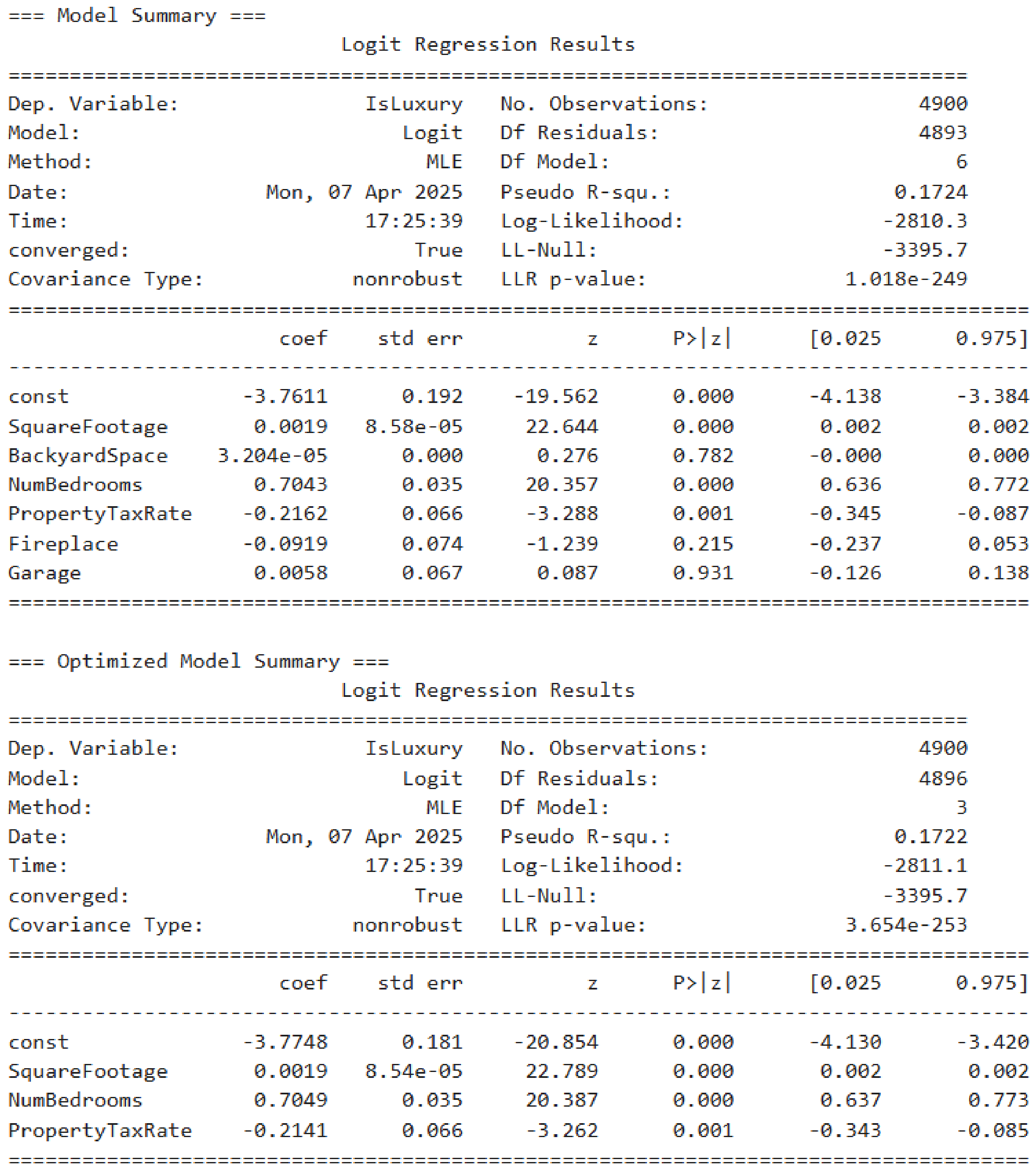
Which means when comparing them the differences are:

- 32 for correctly predicting True Negatives
- 10 for incorrectly predicting False Positives
- 44 for incorrectly predicting False Negatives
- 69 for correctly predicting True Positives

The difference of 0.28% is reflected in these values.

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

Below is my model summary, where we can observe the values for the Pseudo R-squared value for both the Test Set and the Optimized Model, both of which are 0.1724 and 0.1722, respectively.



```
=== Model Summary ===
                           Logit Regression Results
==============================================================================
Dep. Variable:              IsLuxury   No. Observations:                 4900
Model:                         Logit   Df Residuals:                     4893
Method:                          MLE   Df Model:                            6
Date:               Mon, 07 Apr 2025   Pseudo R-squ.:                 0.1724
Time:                       17:25:39   Log-Likelihood:               -2810.3
converged:                      True   LL-Null:                      -3395.7
Covariance Type:           nonrobust   LLR p-value:                1.018e-249
===================================================================================
                      coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------
const              -3.7611      0.192    -19.562      0.000      -4.138      -3.384
SquareFootage       0.0019   8.58e-05     22.644      0.000       0.002       0.002
BackyardSpace    3.204e-05      0.000      0.276      0.782      -0.000       0.000
NumBedrooms         0.7043      0.035     20.357      0.000       0.636       0.772
PropertyTaxRate    -0.2162      0.066     -3.288      0.001      -0.345      -0.087
Fireplace          -0.0919      0.074     -1.239      0.215      -0.237       0.053
Garage              0.0058      0.067      0.087      0.931      -0.126       0.138
===================================================================================

=== Optimized Model Summary ===
                           Logit Regression Results
==============================================================================
Dep. Variable:              IsLuxury   No. Observations:                 4900
Model:                         Logit   Df Residuals:                     4896
Method:                          MLE   Df Model:                            3
Date:               Mon, 07 Apr 2025   Pseudo R-squ.:                 0.1722
Time:                       17:25:39   Log-Likelihood:               -2811.1
converged:                      True   LL-Null:                      -3395.7
Covariance Type:           nonrobust   LLR p-value:                3.654e-253
===================================================================================
                      coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------
const              -3.7748      0.181    -20.854      0.000      -4.130      -3.420
SquareFootage       0.0019   8.54e-05     22.789      0.000       0.002       0.002
NumBedrooms         0.7049      0.035     20.387      0.000       0.637       0.773
PropertyTaxRate    -0.2141      0.066     -3.262      0.001      -0.343      -0.085
===================================================================================
```

With the Pseduo R Squared value being slightly higher than the optimized model value and being relatively low, we can say the model isn't the best fit, and optimizing it didn't help it. In fact, it reduced it slightly. Also, from our analysis, we learned that the higher the amount of bedrooms has a more significant impact on whether a house is considered a luxury or not, along with the tax rate having a negative impact on whether a house is a luxury or not.

9. Recommend a course of action for the real-world organizational situation from part B1 based on your results and implications discussed in part E8.

A course of action I recommend is to include more data points to further pinpoint what makes a house a luxury or not. I would also recommend that builders who want their houses classified as a luxury maximize the number of bedrooms while also building in a lower tax rate location.

Sources

No sources were used except for WGU Material