Predictive Modelling Report

By Alex Westgarth

Executive Summary

Samantha is a Lego enthusiast that has enlisted the help of a Data Scientist to help determine the price of a Lego set using details of the set. A dataset containing over 12,000 sets with details was processed using a linear regression model to produce a formula that outputs a price given the number of pieces within the set. The finalised model performs well at normal piece counts, such as 500 pieces for approximately $60 and 1000 pieces for approximately $100. The model is less accurate at very low and very high piece counts.

Table of Contents

1. Introduction

Samantha is a Lego enthusiast that wishes to be able to predict the price of a Lego set using details about the set. They have enlisted the assistance of a Data Scientist to provide a solution to their problem.

The dataset used in the completion of this report is one retrieved from Kaggle (Mattieterzolo 2018) which itself was scrapped from Lego.com. It features over 12,000 individual sets and several details of each, which are explained in detail in Table 1.

Column	**Sample record**	**Interpretation of columns**
ages	19	The age group the Lego product caters to
list_price	29.99	Price of the Lego product
num_reviews	2	Number of reviews on the Lego product
piece_count	277	Number of Lego pieces in the product

play_star_rating	4	Rating by players at Playstar (0 to 5)
review_difficulty	0	The difficulty level of the product
star_rating	4.5	General rating for the product
theme_name	0	Theme ID
val_star_rating	4	Another source of rating for the same products
country	20	Number of countries the product is sold in

Table 1: The metadata of the dataset used in the report. It shows the name of the column as it appears in the dataset, an example record and a brief description of the column. (Mattieterzolo 2018)

This report will follow a 7-step adjusted version of the Cross Industry Standard Process for Data Mining (CRISP-DM) Data Science life cycle. The key change is the division of the "Modelling" phase into a "Data Modelling" and "Model Building" phase. Figure 1 and Table 2 explain the individual steps in detail.

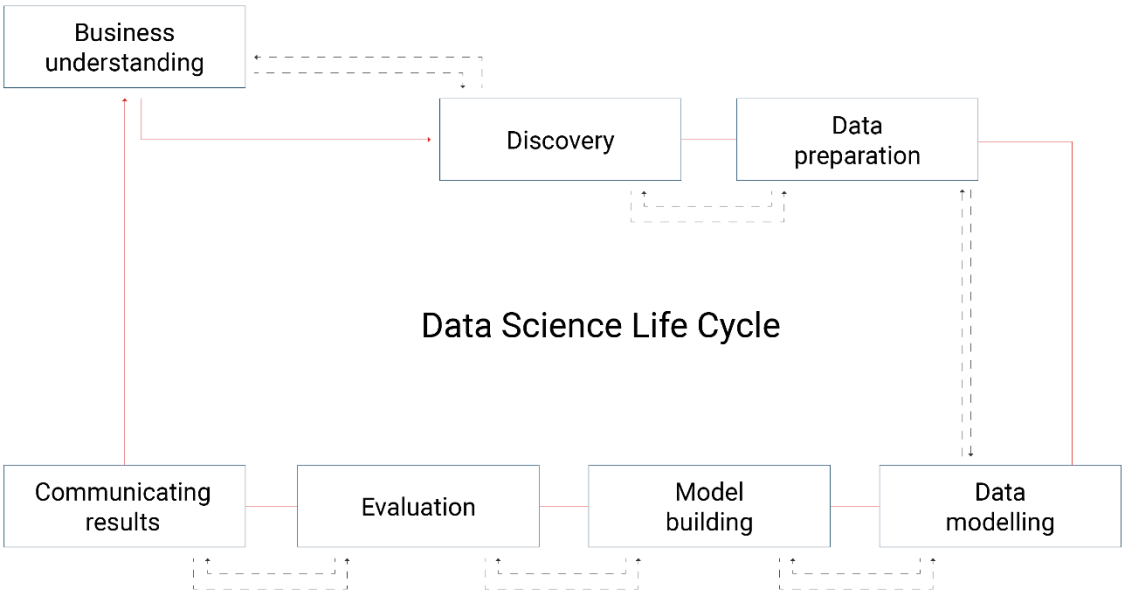


Figure 1: A flow diagram of the 7-phase CRISP-DM Data Science Life Cycle. It includes the typical pathways one takes between phases. (Aghaie 2022)

Phase	**Description**	**Business Application**
Business Understanding	This phase requires us to investigate the problem and determine what information we require prior to commencement. It is also the point where we develop the proper	We should have a basic understanding of Lego and how the price might be determined for a set.

	business problem for driving the process. Returning to this phase may be necessary if the discovery phase unveils crucial information that alters the original understanding.	We'll have to model a dataset against a model to produce a simple formula that can be used by the client. The business problem was clearly laid out already by the client. To provide a predictor for price of Lego sets based on details of the set.
Discovery	This phase requires us to seek out relevant data to the business problem. Sometimes work performed here requires a revisit of the previous phase based on new information. This phase can also be revisited if the data preparation phase unveils a need for additional data.	The data was provided to us in advance. The data was scrapped from Lego.com and posted onto Kaggle. No additional data is required.
Data Preparation	This phase requires us to prepare the acquired data to ensure it is fit to perform the task and answer the business problem. Preparation includes exploration, pre-processing, and conditioning. Preparing the data may reveal that additional data is required and thus, a return to the discovery phase. This phase is often returned to as the data modelling may require certain additional preparation steps.	Examination of the data showed some level of pre-processing prior to acquisition as there were no null values and the categorical values had already been converted to ordinal categorical values.
Data Modelling	This phase requires us to now assess the data given and problem presented to determine the appropriate model and then process the data to fit the model's requirements. Modelling the data may reveal certain additional steps of preparation, causing a return to the data preparation phase. This phase works closely with the model building phase and is often switching between the two.	A linear regression model was chosen based on the correlation of the data present. The independent variable was selected and then it and the target variable were conditioned to ensure proper function.
Model Building	This phase is where we apply the data to the model and produce an output. This phase has a close relationship with the data modelling phase as the model building proceeds. This causes a regular switching between the two phases. Depending on the evaluation phase, additional modelling may be required.	The model was trained against a sub-section of the dataset and tested against the remaining data in the dataset. The model's outputs were then checked against expectations.
Evaluation	This phase is where we ask ourselves, did we answer the business problem?	The model performed within expectations and was deemed sufficient to provide to the client.

	If the result is insufficient, returning to the previous phases is likely to attempt to gain a preferrable result.	
Communication Results	This final phase is where we collate all the work performed into a finalised report to then be presented to the client. This phase and the evaluation phase have much overlapping and so the two are regularly interchangeable.	This very report is the application of this step. Additionally, a copy of the programming notebook used to process all the data and output the values will be provided.

Table 2: A detailed explanation of the 7-phase CRISP-DM Data Science Life Cycle. In includes the name of the phase, a description of the phase and how that applies to this specific business problem. (Aghaie 2022)

2. Methodology

2.1. Libraries and Functions

First, all libraries and functions appropriate to the completion of the analysis are imported. A list of them and their justification can be seen in table 3 below.

Library/Function	Purpose/Justification
Pandas	Containing the DataFrame Object, a powerful tool in holding, manipulating, and otherwise working with tabulated data. Most libraries and functions are designed to work well with Pandas DataFrames.
Numpy	Numerical Python for short, numpy holds many powerful and useful mathematical functionality which make the transformation of the data simple.
Seaborn	A more simplified and visually appealing graphing library. It performs faster at plotting certain graphs than matplotlib but has some missing functionality.
Matplotlib	A more rudimentary graphing library. Used in conjunction with seaborn to plot multiple figures easier and to ensure the simplest option is available for each graph type.
Train Test Split	A function from sklearn's model_selection library, it splits two Pandas Series into training and testing segments. This allows the user to use a single dataset to both train and test a model without having to manually split the data into separate dataframes.
Linear Regression	A function from sklearn's linear model library, it is one of the simplest models available. It accepts one independent continuous variable and a target continuous variable and produces a linear equation for predictive purposes.
Chi Square	A function from scipy's stats library, it allows the user to perform the chi squared test with minimum effort. Its use was to determine the normality of data.

Table 3: A list of the libraries and functions imported into the notebook for use of the data analysis. It also contains a brief explanation of the library or function's purpose.

2.2. Loading Data

The dataset provided was first analysed using a raw text editor to examine the formatting of the file. As it was a plain CSV file with no special conditions, it was then loaded into a Pandas DataFrame Object. It was then examined for any surface level details, as well as null values as seen in figure 2 below.

```
# Display the various surface details of the data
print("Head of DataFrame \n")
df.head()
print("\nTail of DataFrame \n")
df.tail()
print("\nShape of DataFrame \n")
df.shape
print("\nInfo of DataFrame \n")
df.info()
print("\nNull values present in DataFrame \n")
df.isnull().sum()
print("\nDescription of DataFrame")
df.describe(include = "all")
```

Figure 2: The primary investigation into the loaded dataset.

Two key findings were identified from this investigation.

1. There are no null values present.
2. All categorical data has been converted into ordinal encoding already.

These two tasks are commonly required when dealing with new datasets. Given they have already been completed, it is safe to assume that the data has already been treated prior to being provided to us.

2.3. Target variable information

The target variable of the business problem within the dataset is list_price. This is the price of the set, in dollars. This variable is a continuous numerical value. To derive a prediction for a continuous numerical value, a linear regression model is most suitable. A linear regression model requires one independent variable to model from and has 4 key assumptions about the data (Statology 2022).

1. Linearity – The relationship between the independent variable and the target variable is linear.
2. Constant Variance (Homoscedasticity) – The variance of the residuals (errors) is the same for any value of the independent variable.
3. Independence – The independent variables are independent of each other.
4. Normality – For any fixed value of the independent variable, the target is normally distributed.

A multilinear regression model would use multiple independent variables. For now, the report assumes there will only be one viable option.

2.4. Investigation of independent variables

The quickest method for determining the most optimal independent variable as well as checking assumption 3, is through a correlation matrix. The output of the correlation matrix can be seen in figure 3 below.

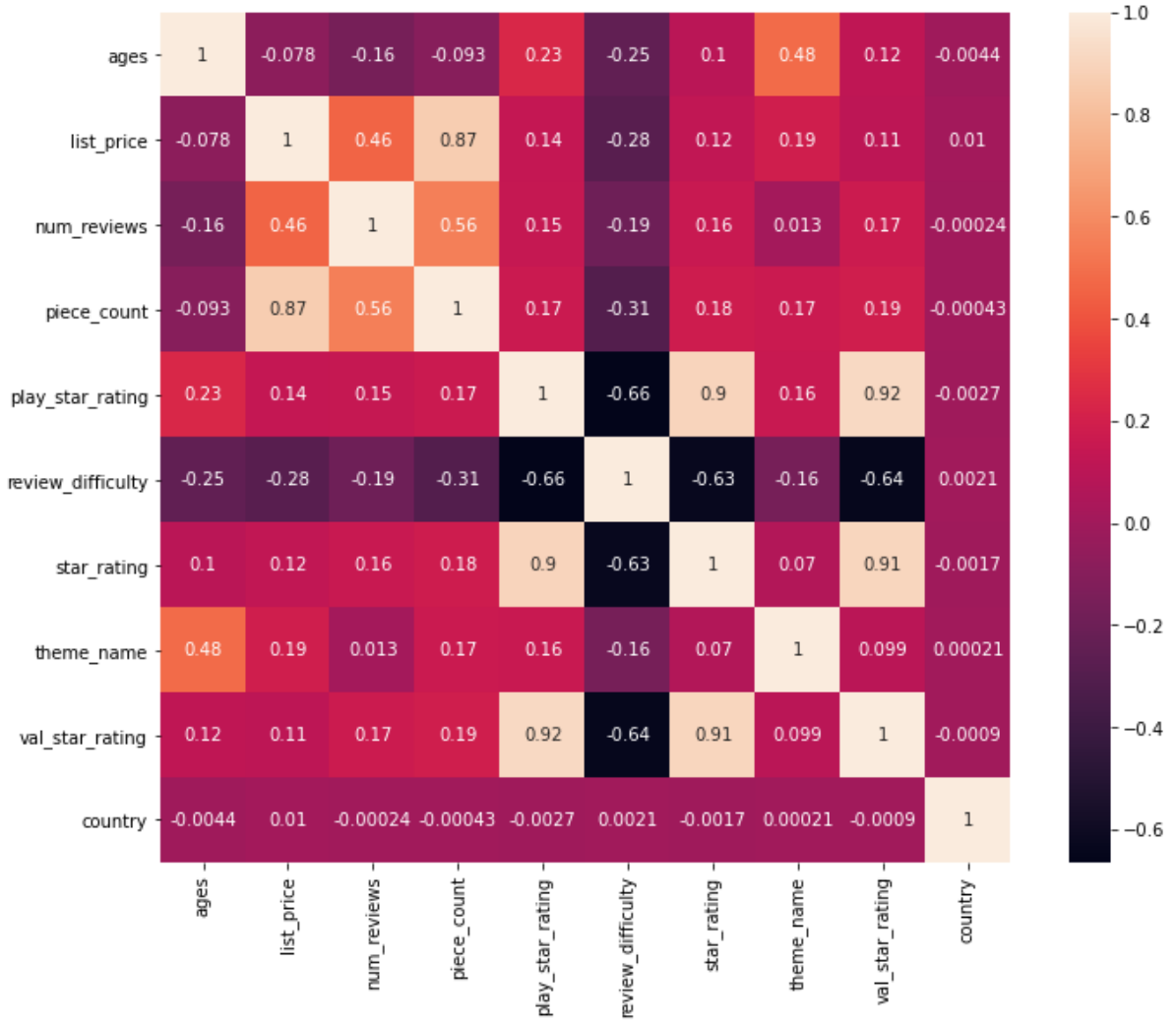


Figure 3: A correlation matrix plotted via a heatmap for the dataset. The columns and rows each represent a data column. Each square has a numerical value and color. The closer to 1 the value is, the more correlated the row-column pair are. The top left to bottom right diagonal will show a correlation value of 1, as it compares a column with itself.

Checking the list_price row reveals only one variable with a correlation value higher than 0.5, piece_count. Piece count is itself not strongly correlated with any other variable, having at most a 0.56 relation with num_reviews. From this we can conclude that our desired independent variable will be piece_count and it satisfies assumption 3. The three rating variables all share high correlation with each other. This would mean that if we wished to use any of these variables, we would be restricted to only using one to avoid conflicting with assumption 3. A pairplot graph was also produced to check for any other possible trends not highlighted by the correlation matrix as seen in figure 4 below.

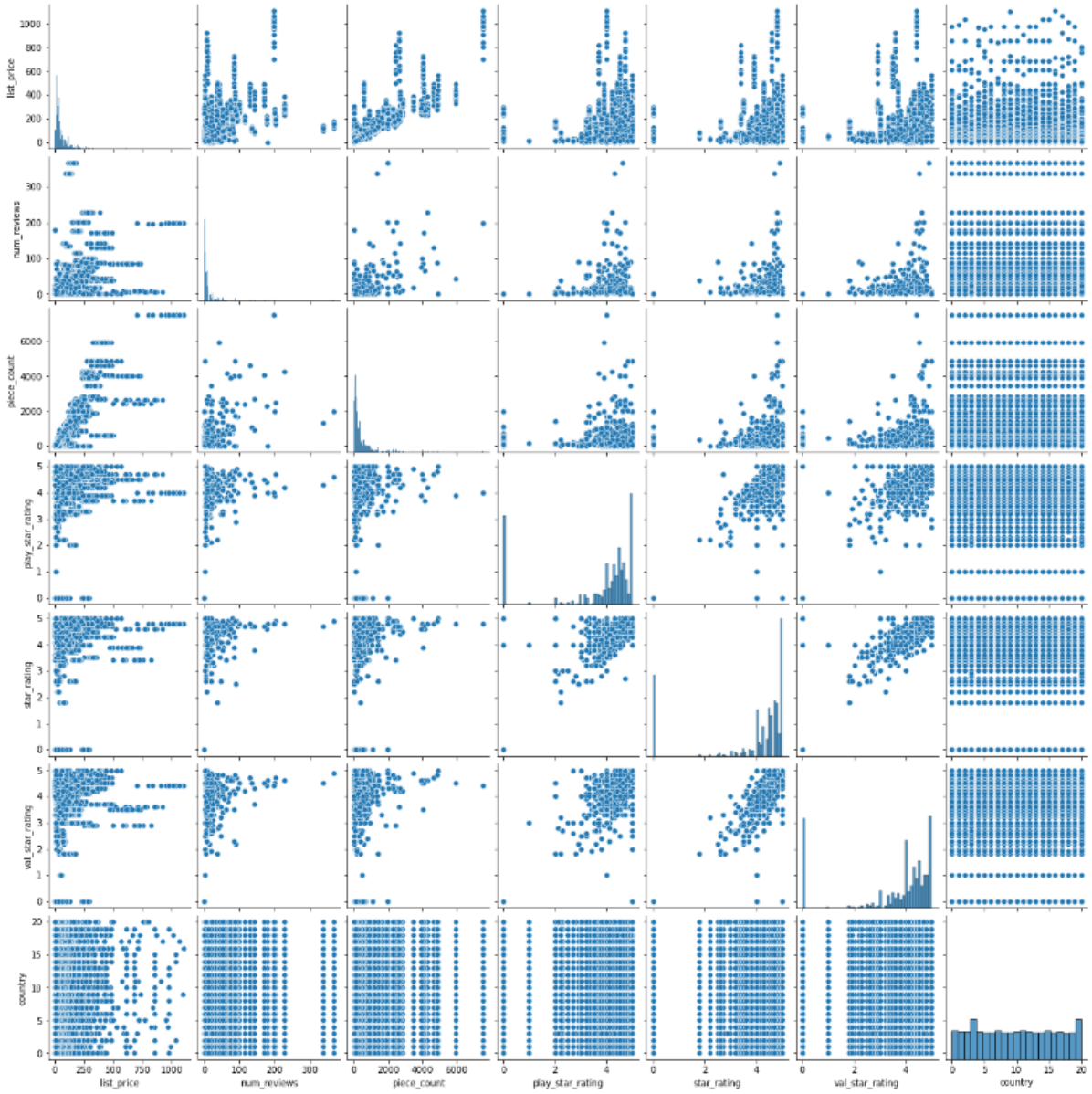


Figure 4: A Seaborn Pairplot of the numerical columns of the dataset. Each row and column represents a data column. Each box represents a plot between the row and column, with the main diagonal being a histogram of the column.

The pair plot shows that the piece_count and list_price relation is somewhat linear in nature, which helps complete assumption 1. Additionally, the correlations between ratings appear to be grouped in nature, rather than linear, quadratic, or similar.

2.5. Clean and Transformation of Data

Now that we have a satisfactory set of independent and target variables it is time to ensure they are appropriately cleaned and transformed for use within the model. First a boxplot and histogram were created for both variables with the use of the helper function as seen in figure 5 and 6 below.

```python
def box_dist_plot(col):
    """
    This function will produce a boxplot of a specified column of a dataframe
    col = column to be plotted
    """
    f, (ax_box, ax_dist) = plt.subplots(2, figsize=(10,6), sharex=True, gridspec_kw={"height_ratios": (.15, .85)})
    sns.boxplot(x=col, data=df, ax=ax_box)
    sns.histplot(df[col], kde=False, ax=ax_dist)
    ax_dist.set_ylabel("Counts")
    ax_box.set_xlabel(col)
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_whisker = Q1 - 1.5*IQR
    upper_whisker = Q3 + 1.5*IQR
    print(col, ": Lower Whisker:",lower_whisker, "Upper Whisker:",upper_whisker)
```

Figure 5: The box plot and histogram plotting helper function. It takes a single variable, a dataframe column and outputs a completed boxplot and histogram.

```
piece_count : Lower Whisker: -573.5 Upper Whisker: 1214.5
list_price : Lower Whisker: -55.31330000000001 Upper Whisker: 145.4955
```

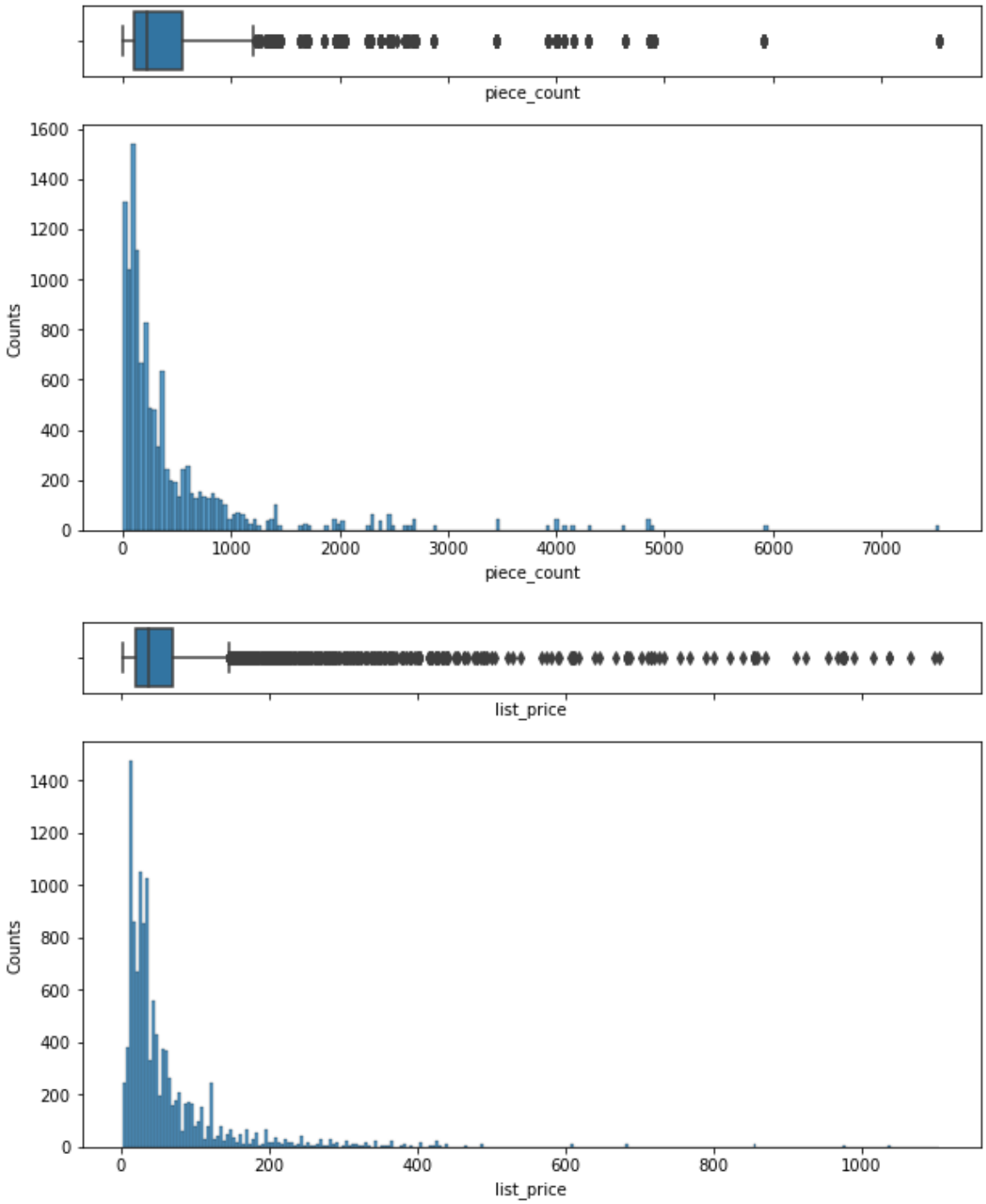


Figure 6: A boxplot and histogram plot for list_price and piece_count.

The data distribution is far from normalised, as per assumption 4. To correct this, a log transformation was applied to both columns and replotted in figure 7.

```
piece_count_log : Lower Whisker: 1.9883535759745437 Upper Whisker: 8.88530664938478
list_price_log : Lower Whisker: 1.1112245806597358 Upper Whisker: 6.13524476159993
```

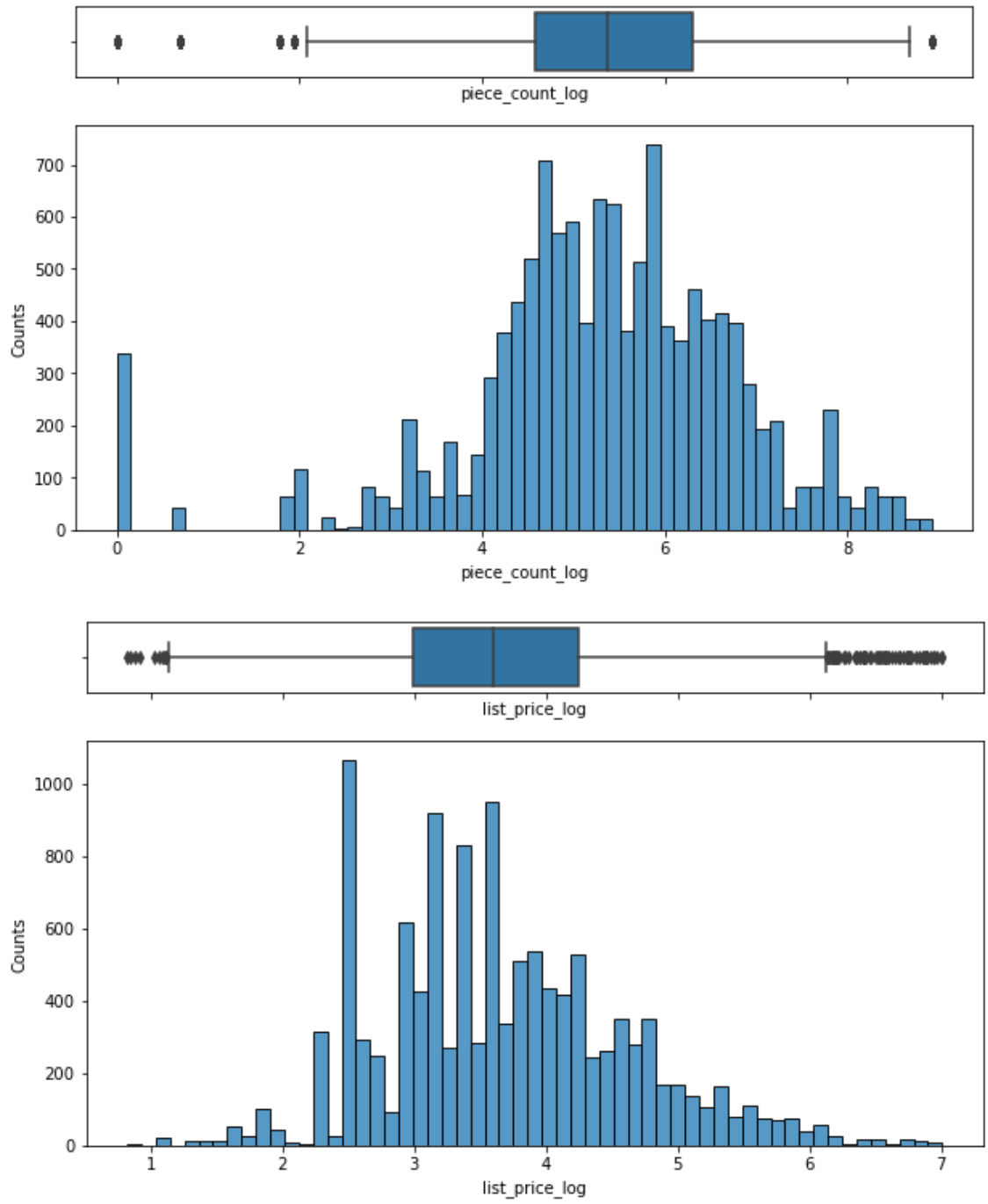


Figure 7: A boxplot and histogram of the log transformed independent and target variables.

The data now looks normalised. Before we proceed with removing outlier, a Chi Squared test was performed to confirm the normalisation hypothesis as seen in figure 8 below.

```
# Perform the Chi Squared test.
stat, p = chisquare(df["piece_count_log"])
print("Stat =%.3f, p=%.3f\n" % (stat,p))

stat, p = chisquare(df["piece_count"])
print("Stat =%.3f, p=%.3f\n" % (stat,p))

stat, p = chisquare(df["list_price_log"])
print("Stat =%.3f, p=%.3f\n" % (stat,p))

stat, p = chisquare(df["list_price"])
print("Stat =%.3f, p=%.3f\n" % (stat,p))
```

```
Stat =5609.082, p=1.000

Stat =16926913.253, p=0.000

Stat =2984.855, p=1.000

Stat =1592282.995, p=0.000
```

Figure 8: The Chi Squared test performed on both the original and log transformed independent and target variable columns.

The p value for the normalisation hypothesis jumps from 0.000 to 1.000 after log transformation which means the data is normally distributed and thus validating assumption 4.

To deal with the remaining outliers a helper function named OutlierRemover was utilized. The details of the helper function can be seen in figure 9 below.

```
def OutlierRemover(col):
    """
    Removes any outliers in a specified column as defined by box plots.
    col = column to have outliers removed.
    """
    global df
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3-Q1
    lower_whisker = Q1 - 1.5*IQR
    upper_whisker = Q3 + 1.5*IQR
    df = df[(df[col]>lower_whisker)&(df[col]<upper_whisker)]
```

Figure 9: The OutlierRemover helper function. It takes a single DataFrame column as input and removes the outlier data points from the original dataset.

We then check the data again in figure 10 below to see how the outlier remover performed.

```
piece_count_log : Lower Whisker: 2.1864770051098943 Upper Whisker: 8.7664325919
03571
list_price_log : Lower Whisker: 1.3048774961042624 Upper Whisker: 6.01905301233
3215
```

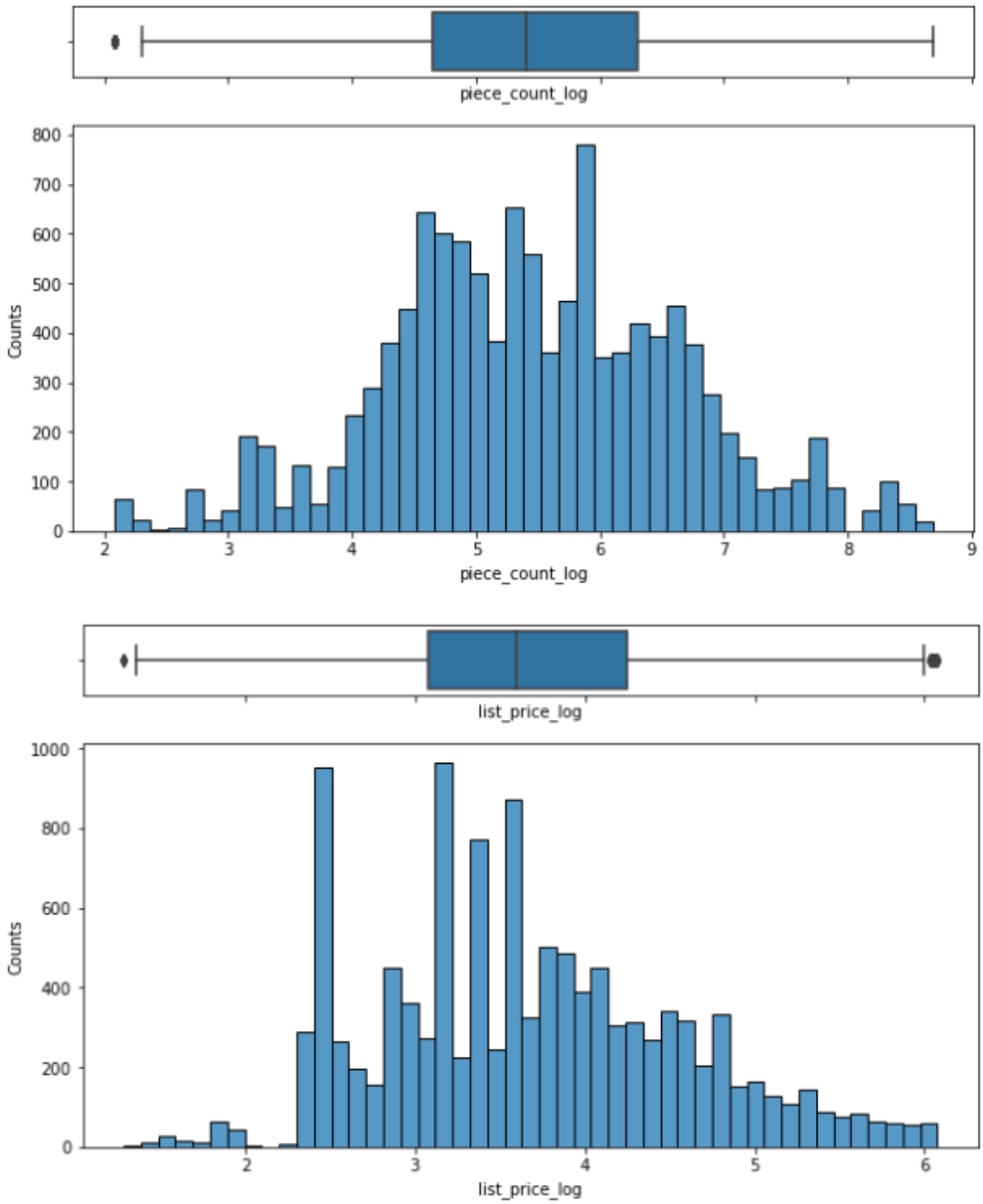


Figure 10: A boxplot and histogram of the outlier removed log transformed independent and target variables.

There are still a few outliers present but the data should now be sufficient.

Now the variables have been cleaned and transformed, they are ready for use within the model.

2.6. Modelling Data

To model the data and perform the final checks, two helper functions were created as seen in figures 11 and 12 below.

```
def homoscedasticity_test(model, X_test, y_test):
    '''
    Function for testing the homoscedasticity of residuals in a linear regression model.
    It plots residuals vs. fitted values.

    model - fitted model
    X_test - the test portion of the independant variable
    y_test - the test portion of the target variable
    '''
    # Predict values.
    fitted_vals = model.predict(X_test)
    # Calculate residuals.
    resids = (y_test - fitted_vals)
    # Plot.
    fig, ax = plt.subplots(1,1)

    sns.regplot(x=fitted_vals, y=resids, lowess=True, ax=ax, line_kws={'color': 'red'})
    ax.set_title('Residuals vs Fitted', fontsize=16)
    ax.set(xlabel='Fitted Values', ylabel='Residuals')
```

Figure 11: A helper function designed to perform a homoscedasticity test. It inputs three variables, a trained model, an independent variable and a target variable that have been split into testing groups.

```
def model_data(X, y):
    """
    Performs the modelling process for a set independant variable and target variable.
    Contains train and test processes and uses the Linear Regression model.

    X = Independant column
    y = Target column
    """
    # Reshape the independant column to fit the modeller.
    X = X.values.reshape(-1,1)
    # Split the data into training and testing sets using 20% test size. Set random state.
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 1)
    # Create the model.
    linreg = LinearRegression()
    # Train the model against the training data.
    linreg.fit(X_train, y_train)
    # Output the alpha and beta values.
    print(linreg.coef_ , linreg.intercept_)
    # Map a prediction on the test values.
    prediction = linreg.predict(X_test)
    # Plot the data
    plt.figure(figsize=(12,6))
    plt.plot(X, y, "o")
    plt.plot(X_test, prediction, "r", linewidth=2)
    plt.xlabel("Piece Count")
    plt.ylabel("Listed Price")
    plt.title("Piece Count Vs Listed Price")
    plt.show()
    # Perform a homoscedasticity test
    homoscedasticity_test(linreg, X_test, y_test)
```

Figure 12: A helper function designed to perform the entirety of the modelling process, from reshaping to splitting to training and graphing. It also includes the previous homoscedasticity helper function at the end. It inputs an independent variable and a target variable.

The first helper function is to complete assumption 2, homoscedasticity. That helper function is executed at the end of the modelling helper function. The modelling helper function performs all the steps from start to finish.

1. Reshape the independent variable
2. Split the data into train and test groups
3. Make the model
4. Train the model
5. Output model coefficients
6. Predict test values
7. Plot predictions against actuals
8. Perform homoscedasticity

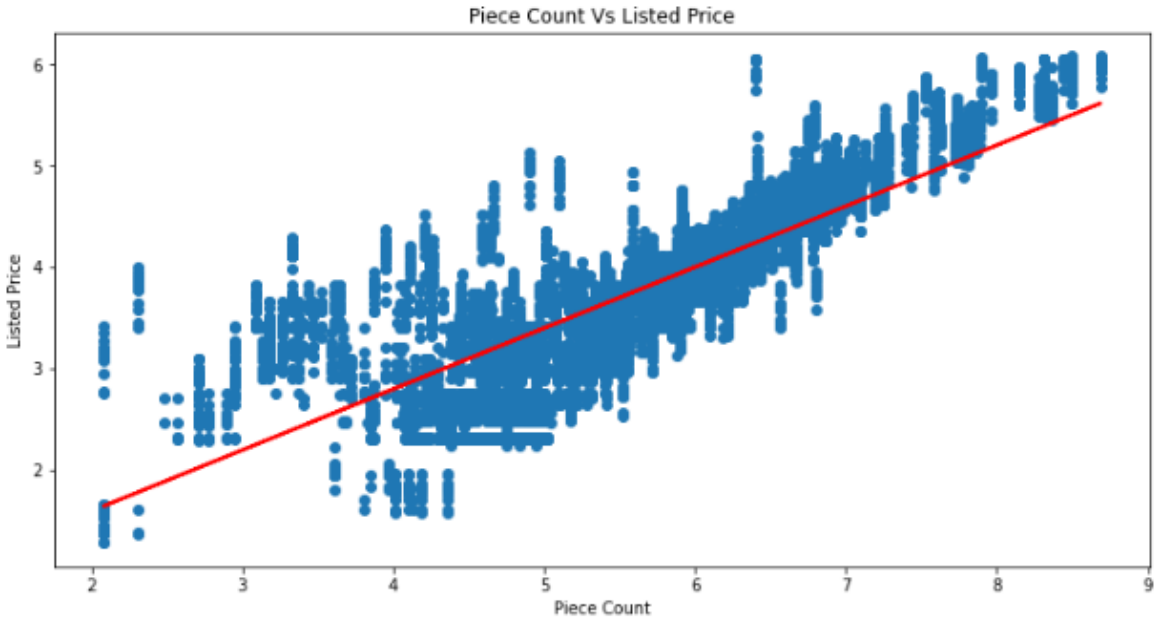
For thorough testing, both the log transformed data and the non-log transformed data were trained.

3. Results and Discussion

The output of the log transformed data are shown in figure 13 below.

```
# Test using the log transformed data
model_data(df["piece_count_log"], df["list_price_log"])
```

```
[0.60186404] 0.3851509445439323
```



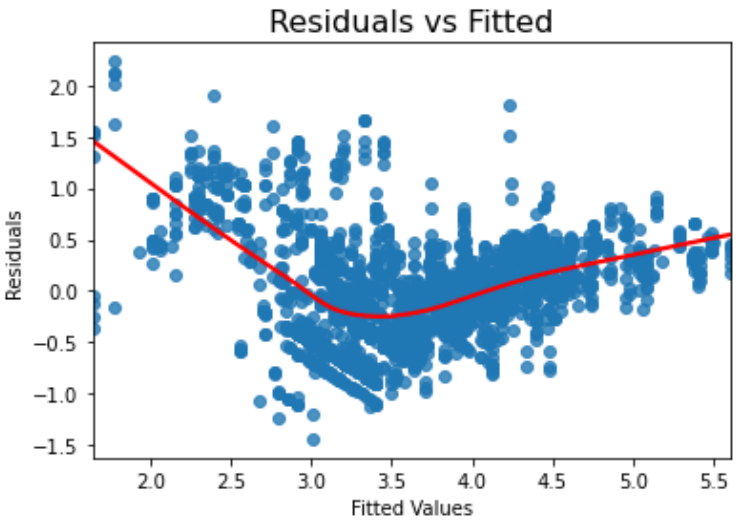
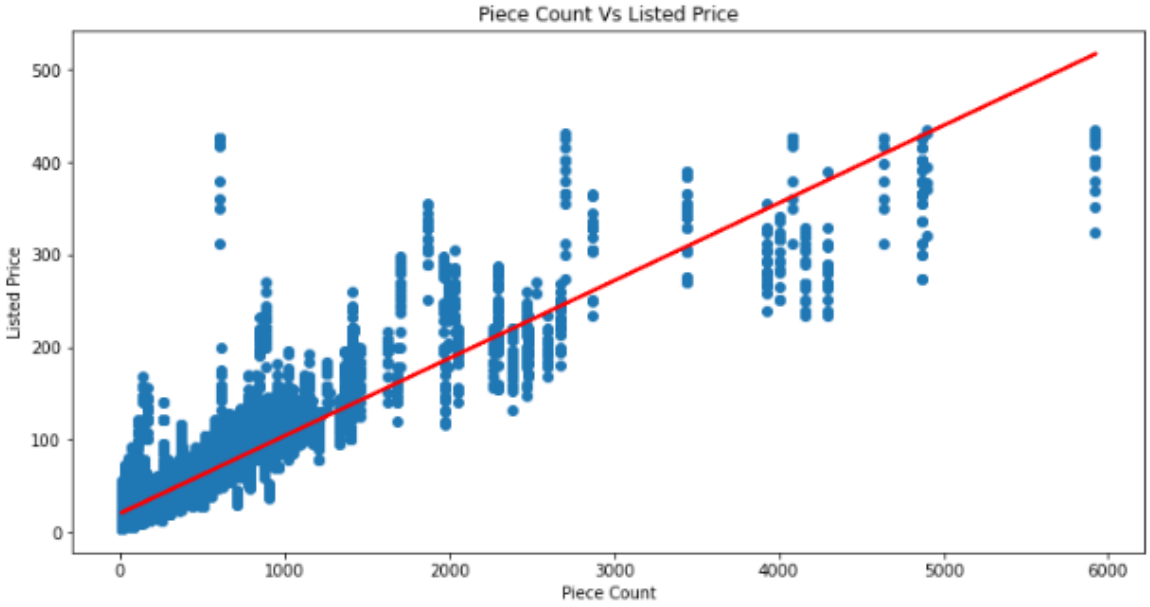


Figure 13: The output of the model helper function from the log-transformed data. The square bracket number is the gradient or alpha and the next number is the intercept or beta. The first graph shows the input data as a scatter plot and the predicted line as a red line through the data. The second graph plots the residuals, as well as the line of best fit.

The general shape of the model and the data does maintain the linearity of the relationship. The homoscedasticity test claims a large portion of the linearity is lost and by all accounts the data does not meet the assumption requirements. It claims that the model will be unreliable at low and high piece counts. To compare, the original non-transformed data was also modelled as seen in figure 14 below.

```
# Test using the original variables.
model_data(df["piece_count"], df["list_price"])
```

```
[0.08398582] 19.90020931575323
```



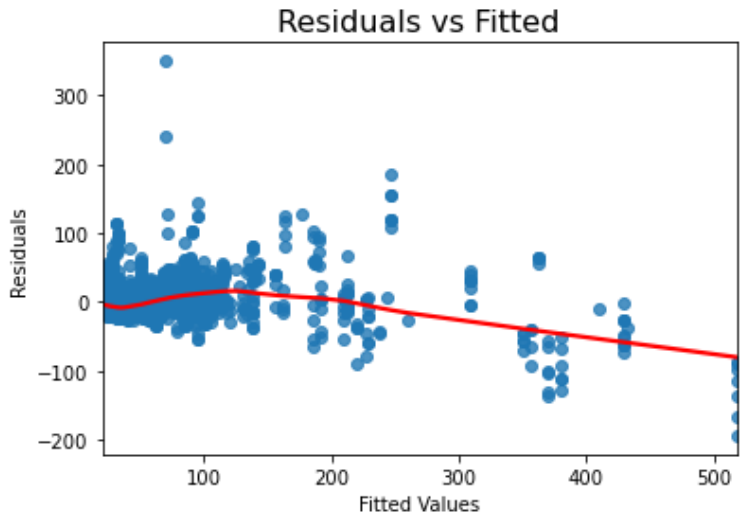


Figure 14: The output of the model helper function from the non-transformed data. The square bracket number is the gradient or alpha and the next number is the intercept or beta. The first graph shows the input data as a scatter plot and the predicted line as a red line through the data. The second graph plots the residuals, as well as the line of best fit.

The original data shows the linear relationship observed at the start of the investigation. The homoscedasticity test is more linear but still has the original issues in that the model is going to perform worse in low and high piece count assumptions.

When testing both models, a range of piece counts were used to get a better understanding of how the two models differ. The exact spread can be seen in table 4 below.

Piece Count	Log-Transformed Model Prediction	Non-Transformed Model Prediction
10	6	21
100	23	28

500	62	62
1000	94	104
5000	248	440

Table 4: A table of predictions of set prices based on piece count for both the log-transformed and original data trained models. All values have been rounded up to the nearest whole number. For the log-transformed model the input piece count had a log-transformation applied before calculation and the result had an exponential applied to undo the original log-transformation to achieve a sensible whole number.

The log-transformed model assumes a lower price range overall. The two models both differ wildly at the extreme ends of the piece counts, but agree on pricing for the middle range, where most of the data existed. Both options produce pros and cons, however the log-transformed data only violates one of the 4 assumptions (homoscedasticity) whereas the original data violates 2 (Normality and homoscedasticity). Because of this, we'll use the log transformed model for our client.

4. Conclusion

This report is confident that the client will receive a suitable prediction for the price of a Lego set using the piece count for budgeting purposes. The model was accomplished using a log-transformed data set and modelled against a linear regression model. The output formula would require a single input and would output a dollar value. The model would come with a warning to the client that it may be unreliable at very high and very low piece count values.

References

Mattieterzolo 2018, "Lego Sets", *Kaggle*, viewed 1st November 2022, <https://www.kaggle.com/datasets/mterzolo/lego-sets>

Aghaie R 2022, "1.7 The seven-phased life cycle", *COS60013 Data Science with Python and R*, Learning materials via Canvas, Swinburne Universirt of Technology, October 6th, viewed 13th October 2022.

Statology c. 2022, *Linear Regression Assumptions*, Statology, viewed 7th November 2022, <https://www.statology.org/linear-regression-assumptions/>