Housing Price Prediction

Project Report of the House Price Prediction in Seattle from 2014 to 2015.

1. Objective of the Predictory Model

The objective of this predictive model is to accurately forecast housing prices based on various features, such as living area, grade, geographic location, and others. The aim is to:

- ✓ Provide reliable predictions for housing prices under and above 1 million.
- ✓ Assist stakeholders, including real estate professionals and potential buyers, in making informed decisions based on data-driven insights.
- ✓ Leverage machine learning techniques to identify key factors influencing prices and optimize predictions across different market segments

2. Key Insights from Exploratory Data Analysis (EDA)

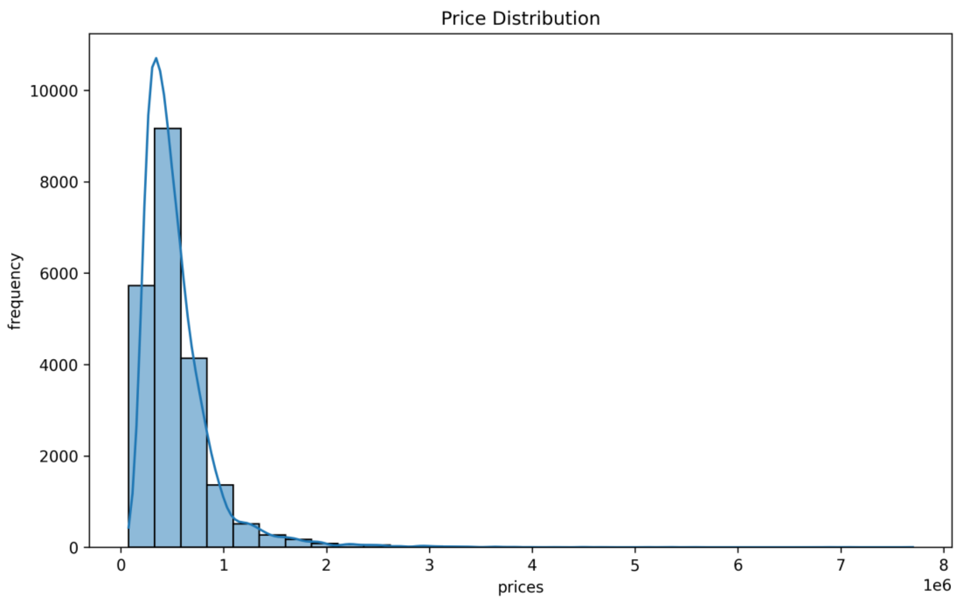
Data Source: The study is based on data collected from a dataset on Kaggle, which provides data of sufficient quality to analyze this issue.

Original dataset size: 21,613 rows and 21 columns.

This dataset provides a snapshot of real estate transactions in Seattle from 2014 to 2015. It includes key property details such as the number of bedrooms, bathrooms, living space size, lot size, and transaction price. Additionally, it incorporates information about property features like waterfront views, renovation history, and construction quality. The dataset offers insights into Seattle's real estate market trends and serves as a valuable resource for data analysis and machine learning applications.

Let´s continue with the most relevant data insights:

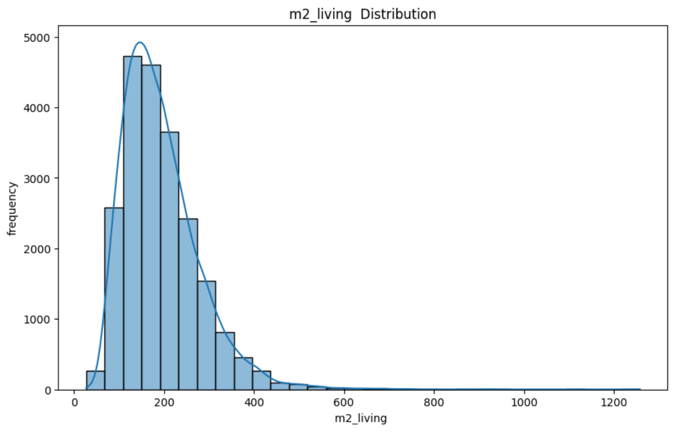
The **target** variable (housing prices) exhibits a right-skewed distribution, with most prices concentrated below 1 million, and significant outliers at higher price ranges.

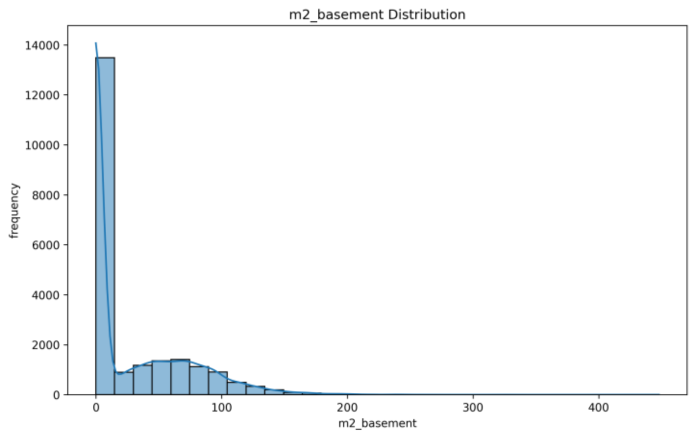


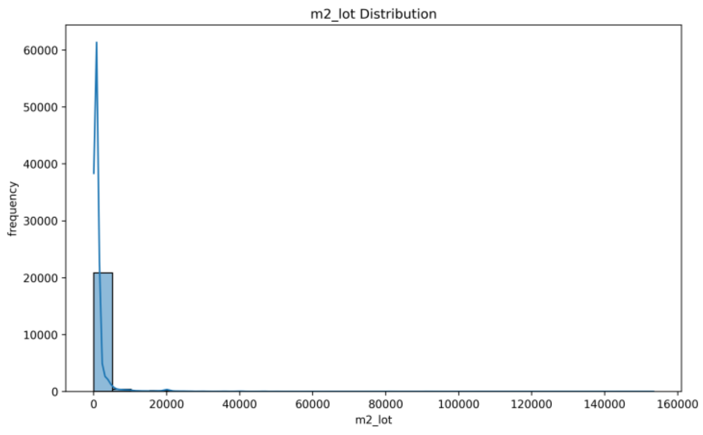
We will apply the following solution:

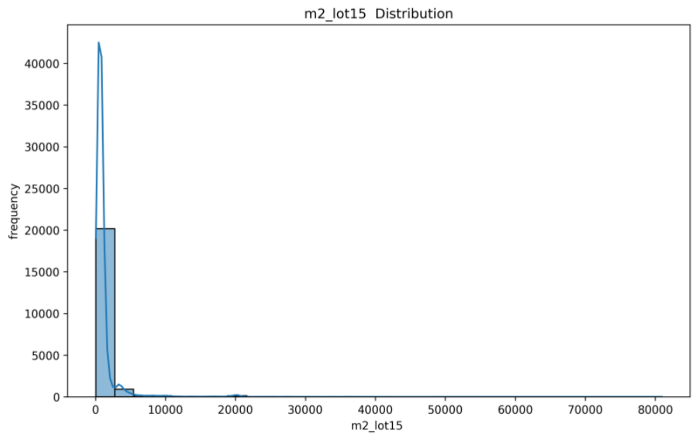
We are going to create three predictive models. The first model will be a classifier that predicts whether the property is valued at over one million $ (i.e., whether it is classified as luxury or not). Based on the predicted class, other models will be applied—this time, a regression model to estimate not luxury houses prices and a classifier model to predict a range Price for luxury houses.

Related to the **features**, the biggest issue is again the right-skewed distribution. Find bellow the most significant:









Logarithmic transformations will be necessary on those variables to improve linear relationships with the target.

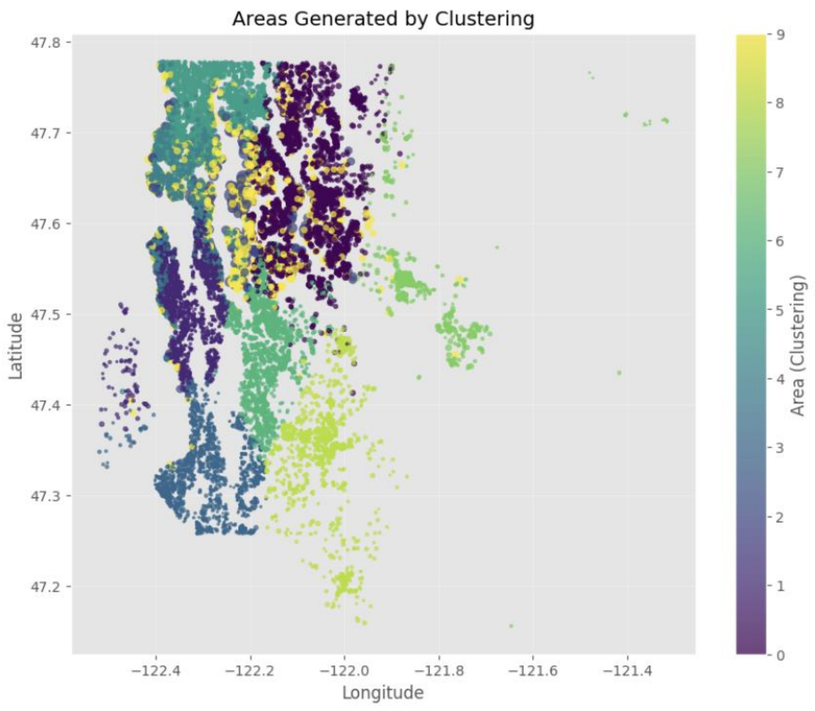
3. Features Engineering

New features will be created as results:

M2_basement issue: The vast majority of homes (60.7%) do not have a basement, which results in a zero for this variable. This needs to be addressed to avoid confusing the prediction model. I will choose to create an additional binary variable: "basement" to confirm the existence of that space. (0: No Basement, 1: With Basement).

yr_renovated issue: The vast majority of homes (95.8%) have not been renovated, which results in a zero for this variable. This needs to be addressed to avoid confusing the prediction model. I will choose to create an additional binary variable 'renovated' (0: Not Renovated, 1: Renovated).

Treatment of housing *coordinates*: Location is always a key factor in determining housing prices. We will use a clustering algorithm (KMeans) to group the homes into separate areas.



From the 'date' variable, more precise data about the time of the transaction can be obtained, such as *year, month, day, day_of_week, hour, minute,* and *second*.

In addition to applying a one-hot encoder to the *zip_code* variable since we only have 70 unique values, we will group by postal code and impute the mean of the houses corresponding to that postal code. As a result, we will obtain a new variable called *avg_mean_by_zip_code*.

4. Features Importance Analysis

This analysis has been done using **features_importances** tool available implementing a Random Forest Classifier. Find bellow the results:

```
17                area    0.383152
2             m2_living   0.094519
8                 grade   0.083110
18   avg_price_by_zipcode 0.075881
9              m2_above   0.064792
1             bathrooms   0.051859
13          m2_living15   0.050865
6                  view   0.025971
11             yr_built   0.025890
14             m2_lot15   0.020401
3                m2_lot   0.020230
10          m2_basement   0.019255
23                 week   0.012846
21                  day   0.012533
0              bedrooms   0.009528
20                month   0.008766
22          day_of_week   0.007726
5            waterfront   0.006713
15             basement   0.006238
7             condition   0.006127
4                floors   0.005221
12         yr_renovated   0.004677
19                 year   0.001913
16            renovated   0.001787
24                 hour   0.000000
25               minute   0.000000
26               second   0.000000
```

Which variables do we keep?

We are going to select features with importance greater than 0.05.

Each of these features contributes at least 5% to the total reduction of impurity in the model. In other words, these features have been responsible for at least 5% of the improvements in the model's quality.

Selected features: 'area', 'm2_living', 'grade', 'avg_price_by_zipcode', 'm2_above', 'bathrooms' and 'm2_living15'

5. Methodology for Dataset Classificatory Model

As saw previously, this project addresses the problem of housing price prediction using a hybrid approach. Given that prices are imbalanced relative to a threshold of 1 million, the dataset has been spitted into two classes using a **classifier model**:

- Class 0: Houses priced fewer than 1 million.
- Class 1: Houses priced at or above 1 million.

5.1 Dataset Preparation:

✓ Predictor variables and target variable were identified. (Selected features: 'area', 'm2_living', 'grade', 'avg_price_by_zipcode', 'm2_above', 'bathrooms' and 'm2_living15').

✓ Data has been split into training and test sets, ensuring representativeness for both classes.

✓ One Hot Enconder for categorical variables "*area*". (Just as a reminder, we identified 10 different clusters)

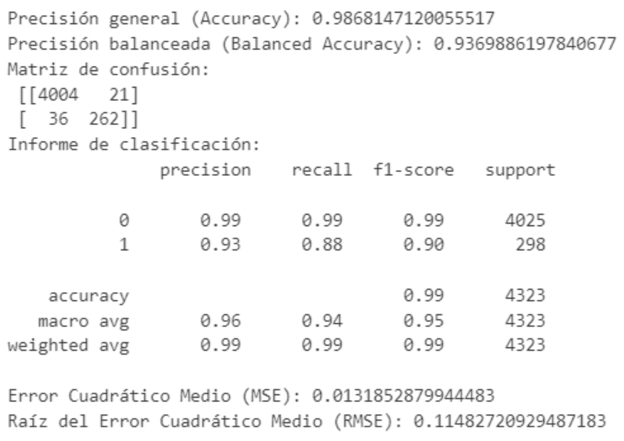
5.2 Training Classification Models:

✓ Several regression models were used with and without RandomUnderSampler and SMOTE techniques.

- **Random Forest Classifier**
- **XGBoost Classifier**

5.3 Model Evaluation and Results:

Best Performing Model: XGBoost Classifier without smoothing techniques.



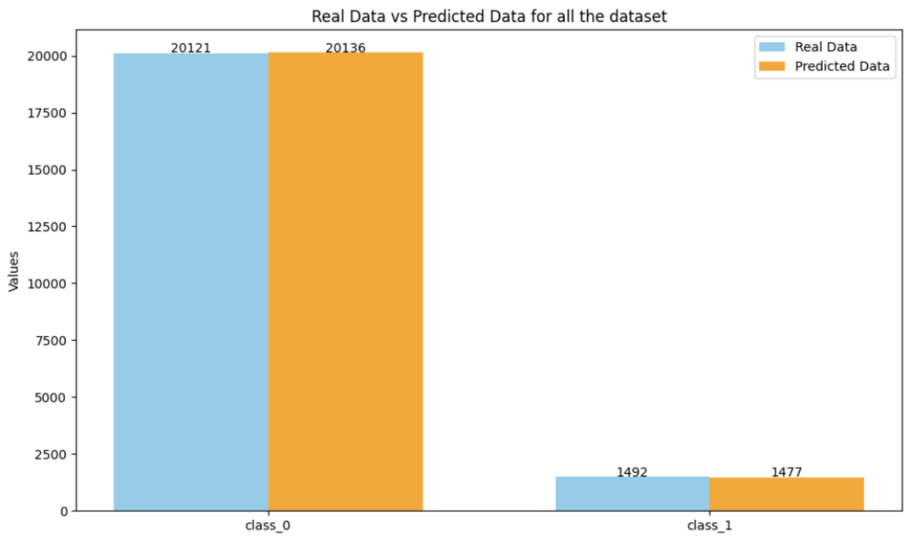
```
Precisión general (Accuracy): 0.9868147120055517
Precisión balanceada (Balanced Accuracy): 0.9369886197840677
Matriz de confusión:
 [[4004   21]
 [  36  262]]
Informe de clasificación:
               precision    recall  f1-score   support

           0       0.99      0.99      0.99      4025
           1       0.93      0.88      0.90       298

    accuracy                           0.99      4323
   macro avg       0.96      0.94      0.95      4323
weighted avg       0.99      0.99      0.99      4323

Error Cuadrático Medio (MSE): 0.0131852879944483
Raíz del Error Cuadrático Medio (RMSE): 0.11482720929487183
```

Once we find the best model, I use it to predict the classification for the entire dataset into class_0 and class_1. These are the results for the entire dataset:



Subsequently, specific models will be developed for each class, optimized according to their particular characteristics.

6. Methodology for Class 0

A specific model will be created to predict the Price of Class 0 houses (under 1 million $)

6.1 Dataset Preparation:

- ✓ Predictor variables and the target variable were identified. (Selected features: '*area*', '*m2_living*', '*grade*', '*avg_price_by_zipcode*', '*m2_above*', '*bathrooms*' *and* '*m2_living15*').
- ✓ Data was split into training and test sets, ensuring representativeness for both classes.
- ✓ Logarithmic transformations were applied only for m2_living variable to reduce skewness.

6.2 Training Regression Models:

- ✓ Pipelines were designed with appropriate preprocessing for each model: Standardization of numerical variables using StandardScaler.
- ✓ Several regression models were used:

- **Ridge**
- **Lasso**
- **Random Forest**
- **XGBoost**
- **Support Vector Regression**

- ✓ Hyperparameters were optimized using GridSearchCV, with the main metric being Root Mean Squared Error (RMSE) in order to maximize accuracy.

6.3 Model Evaluation:

- ✓ Metrics used to evaluate each model were:

- **MSE**: Mean Squared Error.
- **RMSE**: Root Mean Squared Error.
- **MAE**: Mean Absolute Error.
- **R²**: Coefficient of Determination.
- **MedAE**: Median Absolute Error.

6.4 Results and Conclusions:

Performance metrics for each model are presented below:

Model	RMSE	MAE	R2	MedAE
Ridge	81706.0519	62051.9917	0.8210	49074.5142
Lasso	81704.8000	62050.9970	0.8210	49042.2657
Random Forest	74225.1376	55143.4807	0.8523	42052.3018
XGBoost	**71360.6061**	**52190.4166**	**0.8635**	**39666.5625**
SVR	82408.4214	59176.3802	0.8179	44229.2697

The best-performing model was **XGBoost**, standing out for:

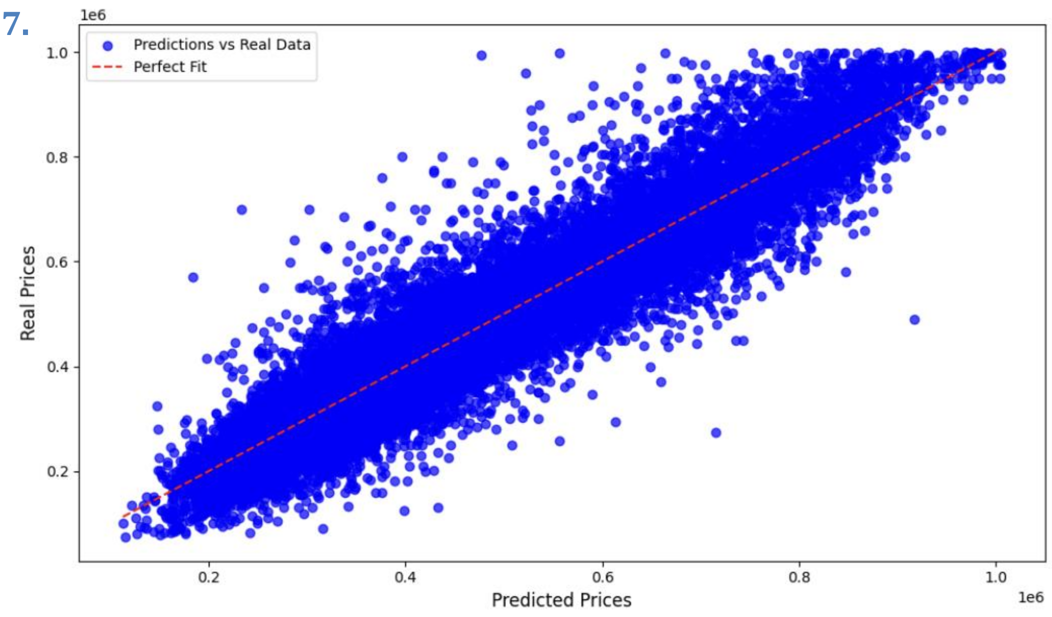
Lowest value for RMSE, MAE and MedAE. The relevant R² is the closest to 1.

- ✓ Dividing the dataset into classes allowed for better precision in addressing the prediction problem.
- ✓ Optimization via GridSearchCV significantly improved model performance.
- ✓ The selected model (XGBoost) is suitable for predicting housing prices based on the most predictive features.

- ✓ **Random Forest model** also shows a significant improvement, with a good R² of 0.8523 and a lower RMSE compared to the earlier models.
  Ridge and Lasso models have similar performance but are slightly less accurate than **Random Forest** and **XGBoost**.
  SVR model has the lowest performance, with a slightly lower R² and higher RMSE and MAE, indicating that its predictions are less accurate.

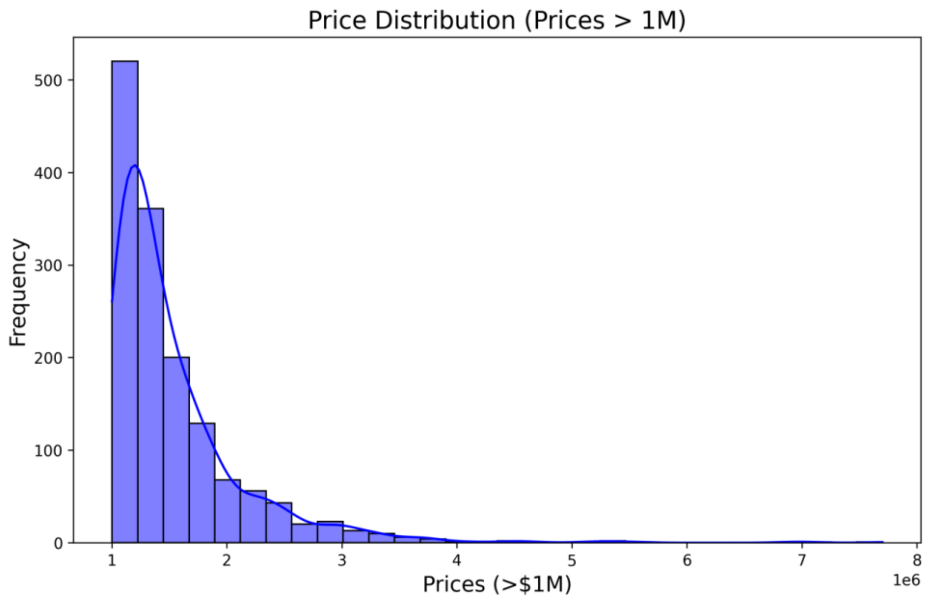
Once we find the best regressor model, I use it to predict the classification for the entire dataset for class_0. These are the results for the entire dataset:

7.



7. Methodology for Class 1

Let´s have a look at the target distribution filtered by “more than 1 Million $”



Since target distribution remains right-skewed (long tail) and we do not have enough data to create a reliable prediction model, I will choose to categorize luxury home prices into three groups:

```
price_luxury_category
Above 1.5M     533
1M-1.2M        485
1.2M-1.5M      474
Name: count, dtype: int64
```

7.1 Dataset Preparation:

- ✓ Predictor variables and the target variable were identified.
- ✓ Data was split into training and test sets, ensuring representativeness for both classes.
- ✓ One Hot Enconder for categorical variables “area”. (Just as a reminder, we identified 10 different clusters)

7.2 Training Classifier Models:

- ✓ Pipelines were designed with appropriate preprocessing for each model: Standardization of numerical variables using StandardScaler.
- ✓ A Random Forest classifier model and a XGBoost Classifier have been trained.
- ✓ Hyperparameters were optimized using GridSearchCV, with the main metric being Accuracy.

7.3 Model Evaluation and Results:

Random Forest Classifier

```
 f1_macro: 0.6533
 accuracy: 0.6522
 balanced_accuracy: 0.6521
 conf_matrix:
[[48 36  5]
 [ 8 65 13]
 [ 1 41 82]]
 report:
              precision    recall  f1-score   support

           0       0.84      0.54      0.66        89
           1       0.46      0.76      0.57        86
           2       0.82      0.66      0.73       124

    accuracy                           0.65       299
   macro avg       0.71      0.65      0.65       299
weighted avg       0.72      0.65      0.66       299
```

XGBoost Classifier

```
 f1_macro: 0.6547
 accuracy: 0.6555
 balanced_accuracy: 0.6558
 conf_matrix:
[[50 35  4]
 [ 9 64 13]
 [ 6 36 82]]
 report:
              precision    recall  f1-score   support

           0       0.77      0.56      0.65        89
           1       0.47      0.74      0.58        86
           2       0.83      0.66      0.74       124

    accuracy                           0.66       299
   macro avg       0.69      0.66      0.65       299
weighted avg       0.71      0.66      0.66       299
```

In this case, the best multiclass classifier model is again **XGBoost**, but by a very small margin.

7.4 Conclusions

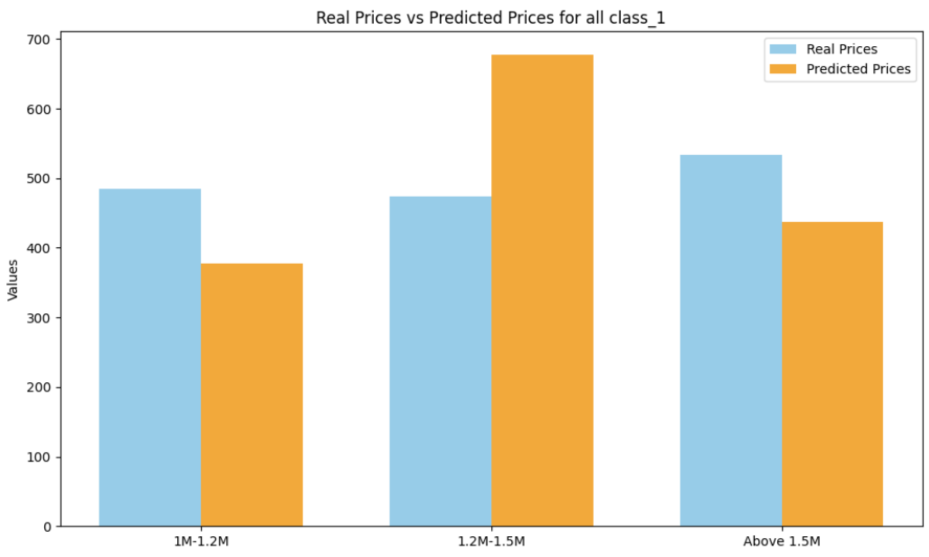
Overall Performance:

- ✓ The model has moderate performance with an accuracy of 65.55%.
- ✓ The F1-macro score of 0.6547 suggests there are classes where the model performs worse (class 1 and class 0).

Issues with Specific Classes:

- ✓ Class 0: Has a low recall (56%), meaning the model misses many instances of this class.
- ✓ Class 1: Has good recall (74%) but very low precision (47%), indicating many false predictions.
- ✓ Class 2: Shows a better balance between precision and recall.

Once we find the best regressor model, I use it to predict the classification for the entire dataset for class_1. These are the results for the entire dataset:



8. Next Steps:

- ✓ Additional Validation:
  Validate the model on an external dataset to evaluate its generalization capability.

- ✓ Feature Engineering Improvements:
  Consider creating new derived variables or interactions between existing variables.

- ✓ Model Deployment:
  Implement the model in an application for real-time predictions.

- ✓ Performance Monitoring:
  Evaluate the model's performance with new data and update it regularly.