Ministry of Communications and Information Technology

Digital Egypt Pioneers

Graduation Project

Sales Forecasting and Optimization

Team members

Marwan Mustafa Muhammad Muhammad
Yomna Sherif Hassan
Moaz Ahmed Mohamed Krar
Hussam El Deen Ahmed Ramadan Saad
Hadeer Abdelwahed Elkady

Under supervision

Eng/ Ahmed El-safty

House Sales Price Prediction Using Machine Learning

1.Introduction

In this project, we aim to predict house prices using a structured data science approach, leveraging machine learning techniques to forecast outcomes and extract valuable business insights.

We used the "House Sales in King County, USA" dataset from Kaggle, which includes over 21,000 entries and 21 features related to houses sold between May 2014 and May 2015.

2. Dataset Overview

The dataset contains 21,613 rows and 21 columns of residential home sales data from King County, Washington, covering the period between May 2014 and May 2015. The key features include:

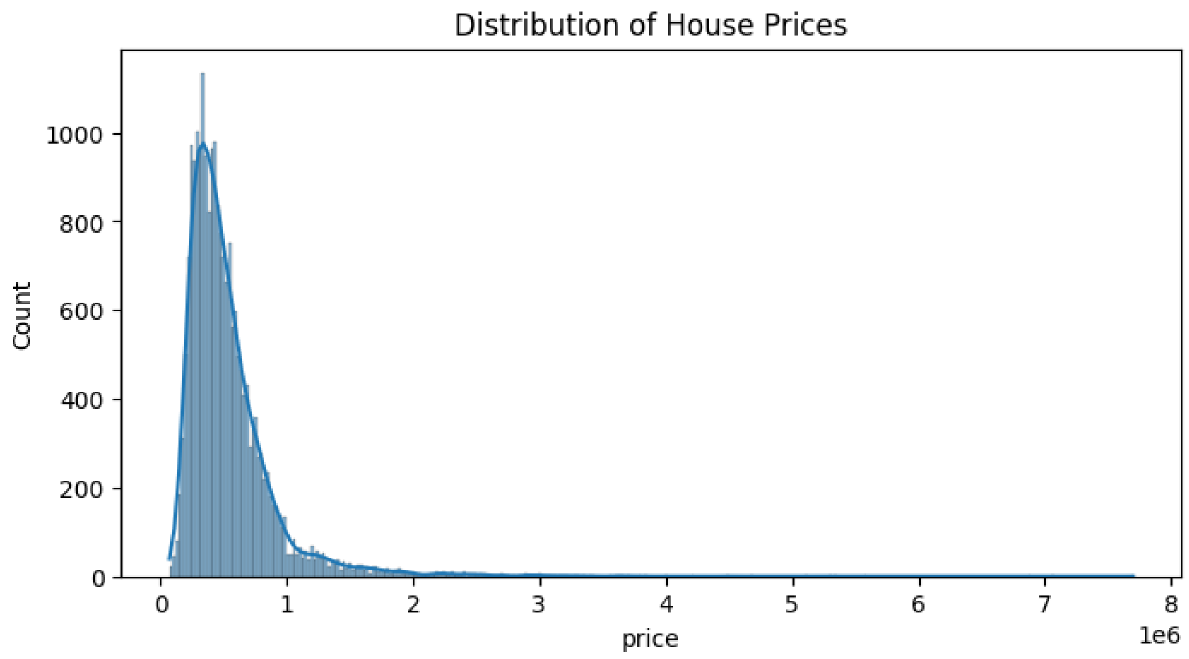
- **price**: Sale price of the house (target variable)
- **bedrooms, bathrooms**: Number of bedrooms and bathrooms
- **sqft_living, sqft_lot**: Square footage of living space and lot
- **floors, waterfront, view, condition, grade**: Descriptive categorical variables
- **sqft_above, sqft_basement**: Living space above ground and in basement
- **yr_built, yr_renovated**: Year built and renovated
- **lat, long**: Geographic coordinates
- **sqft_living15, sqft_lot15**: Average living area and lot size of the 15 nearest neighbors

3. Data Preprocessing and Cleaning

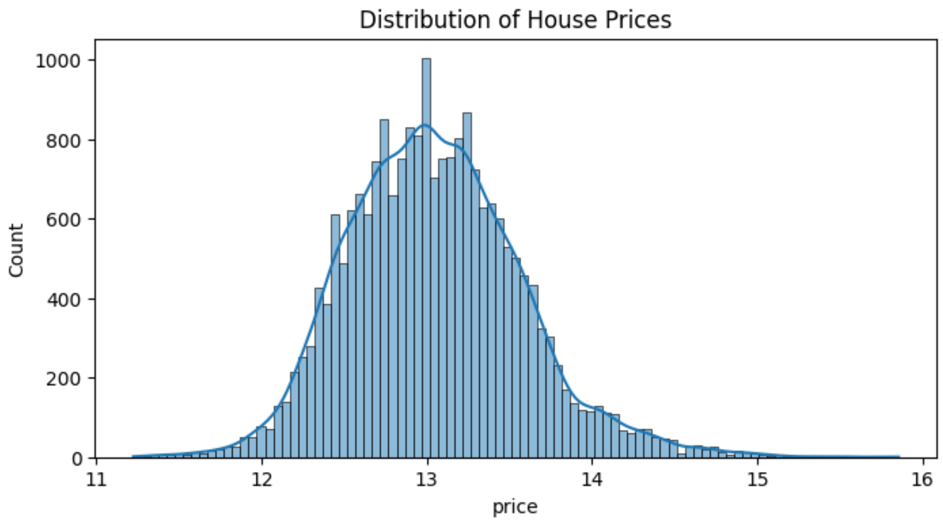
- **Dropped Unnecessary Columns**: The column id was dropped as it does not contribute to the prediction.
- **Date Conversion**: The date column was converted from string format to datetime.
- **Log Transformation**: The target variable price was log-transformed to normalize its distribution.
- **Outlier Removal**: Using the IQR method, 333 outliers were removed to ensure better model performance.

4. Exploratory Data Analysis (EDA)

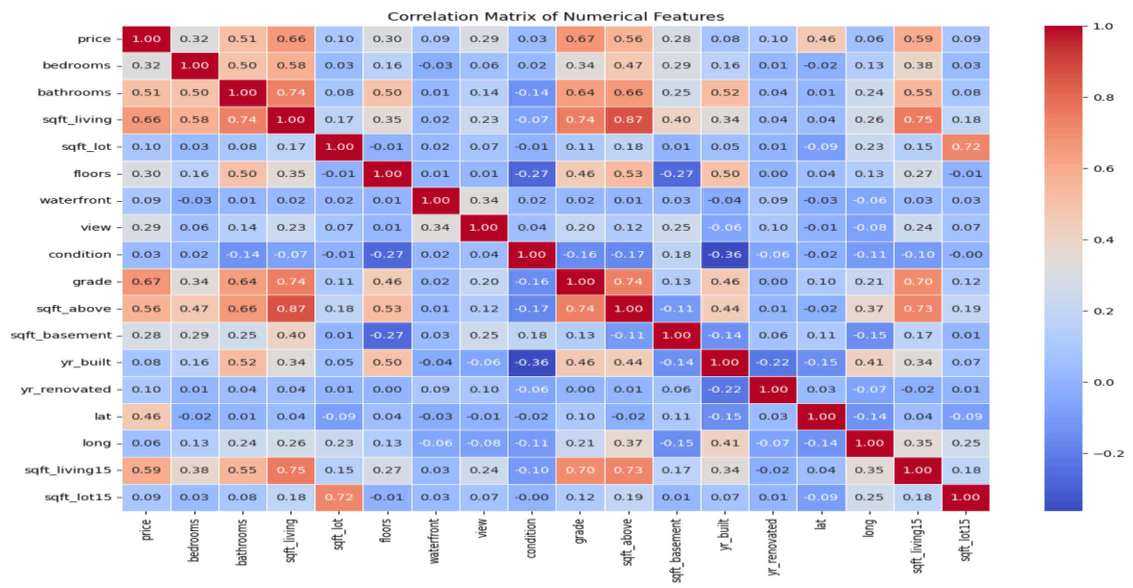
- Price Distribution Before Log Transformation This histogram shows the original distribution of housing prices, which is heavily right-skewed, indicating the presence of extremely high-value properties.



- Price Distribution After Log Transformation After applying a logarithmic transformation to the price, the distribution becomes more normal-like, which helps improve model performance.



- **Correlations**:
  We generated a correlation heatmap to examine the relationships between variables
  <u>Key Findings:</u>
  - grade, sqft_living, and lat are the most positively correlated with price.
  - Some variables like yr_renovated had low correlation and many zero values, indicating less influence.

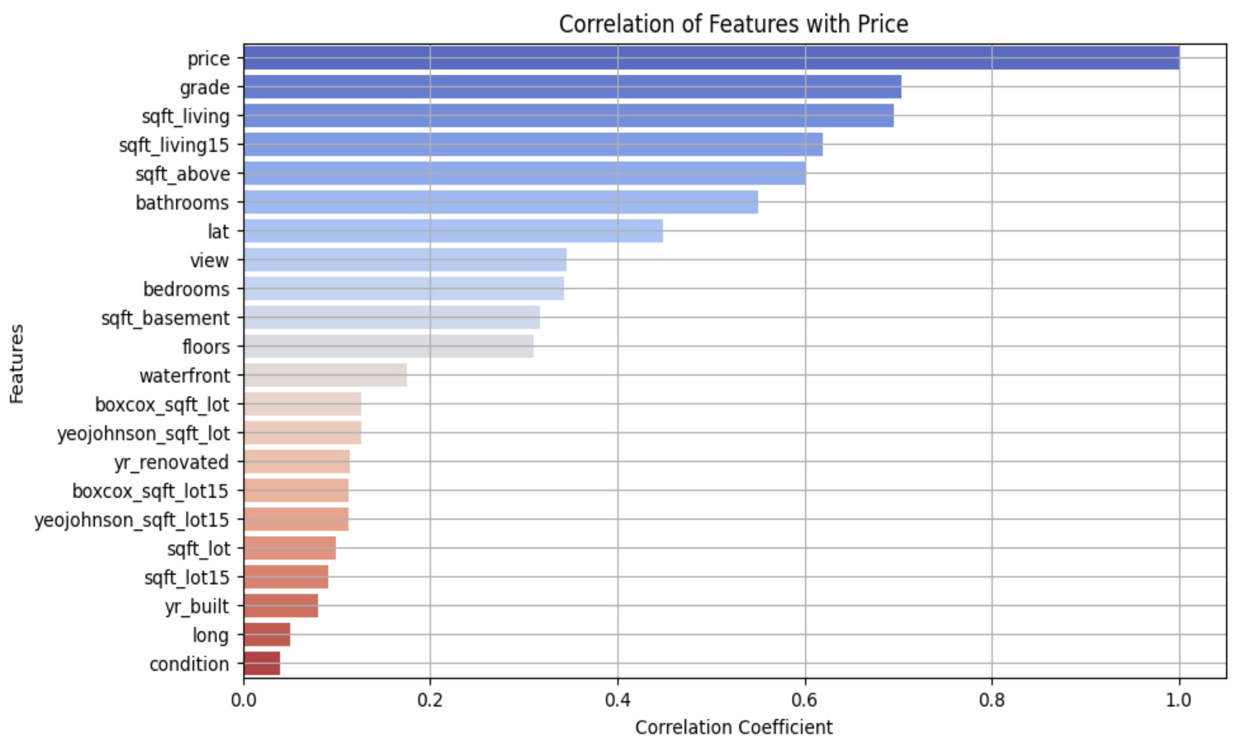


	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	lat	long	sqft_living15	sqft_lot15
price	1.00	0.32	0.51	0.66	0.10	0.30	0.09	0.29	0.03	0.67	0.56	0.28	0.08	0.10	0.46	0.06	0.59	0.09
bedrooms	0.32	1.00	0.50	0.58	0.03	0.16	-0.03	0.06	0.02	0.34	0.47	0.29	0.16	0.01	-0.02	0.13	0.38	0.03
bathrooms	0.51	0.50	1.00	0.74	0.08	0.50	0.01	0.14	-0.14	0.64	0.66	0.25	0.52	0.04	0.01	0.24	0.55	0.08
sqft_living	0.66	0.58	0.74	1.00	0.17	0.35	0.02	0.23	-0.07	0.74	0.87	0.40	0.34	0.04	0.04	0.26	0.75	0.18
sqft_lot	0.10	0.03	0.08	0.17	1.00	-0.01	0.02	0.07	-0.01	0.11	0.18	0.01	0.05	0.01	-0.09	0.23	0.15	0.72
floors	0.30	0.16	0.50	0.35	-0.01	1.00	0.01	0.01	-0.27	0.46	0.53	-0.27	0.50	0.00	0.04	0.13	0.27	-0.01
waterfront	0.09	-0.03	0.01	0.02	0.02	0.01	1.00	0.34	0.02	0.02	0.01	0.03	-0.04	0.09	-0.03	-0.06	0.03	0.03
view	0.29	0.06	0.14	0.23	0.07	0.01	0.34	1.00	0.04	0.20	0.12	0.25	-0.06	0.10	-0.01	-0.08	0.24	0.07
condition	0.03	0.02	-0.14	-0.07	-0.01	-0.27	0.02	0.04	1.00	-0.16	-0.17	0.18	-0.36	-0.06	-0.02	-0.11	-0.10	-0.00
grade	0.67	0.34	0.64	0.74	0.11	0.46	0.02	0.20	-0.16	1.00	0.74	0.13	0.46	0.00	0.10	0.21	0.70	0.12
sqft_above	0.56	0.47	0.66	0.87	0.18	0.53	0.01	0.12	-0.17	0.74	1.00	-0.11	0.44	0.01	-0.02	0.37	0.73	0.19
sqft_basement	0.28	0.29	0.25	0.40	0.01	-0.27	0.03	0.25	0.18	0.13	-0.11	1.00	-0.14	0.06	0.11	-0.15	0.17	0.01
yr_built	0.08	0.16	0.52	0.34	0.05	0.50	-0.04	-0.06	-0.36	0.46	0.44	-0.14	1.00	-0.22	-0.15	0.41	0.34	0.07
yr_renovated	0.10	0.01	0.04	0.04	0.01	0.00	0.09	0.10	-0.06	0.00	0.01	0.06	-0.22	1.00	0.03	-0.07	-0.02	0.01
lat	0.46	-0.02	0.01	0.04	-0.09	0.04	-0.03	-0.01	-0.02	0.10	-0.02	0.11	-0.15	0.03	1.00	-0.14	0.04	-0.09
long	0.06	0.13	0.24	0.26	0.23	0.13	-0.06	-0.08	-0.11	0.21	0.37	-0.15	0.41	-0.07	-0.14	1.00	0.35	0.25
sqft_living15	0.59	0.38	0.55	0.75	0.15	0.27	0.03	0.24	-0.10	0.70	0.73	0.17	0.34	-0.02	0.04	0.35	1.00	0.18
sqft_lot15	0.09	0.03	0.08	0.18	0.72	-0.01	0.03	0.07	-0.00	0.12	0.19	0.01	0.07	0.01	-0.09	0.25	0.18	1.00

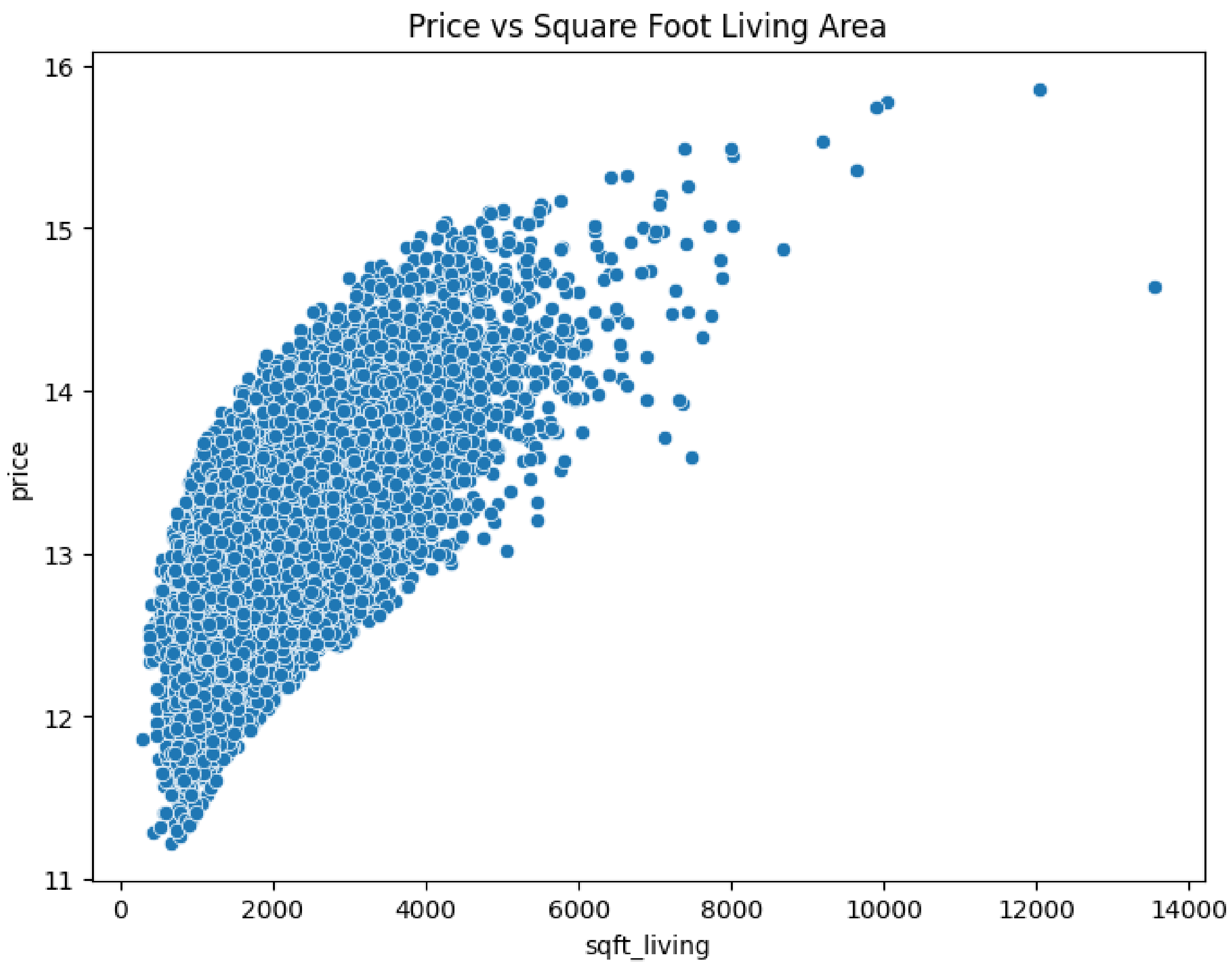
- **Barplot to visualize the correlation of features with the price**

  Key Findings:

- **grade**, **sqft_living**, and **sqft_living15** are the most positively correlated features with the price, indicating that improvements in the house's grade and living space lead to a significant increase in price.
- **bathrooms**, **lat**, and **view** also show positive correlations, although to a lesser extent, suggesting they contribute moderately to the price.
- Features like **waterfront**, **yr_renovated**, and **boxcox_sqft_lot15** have lower correlations with price, with some features showing weak correlations, indicating less impact on the price.

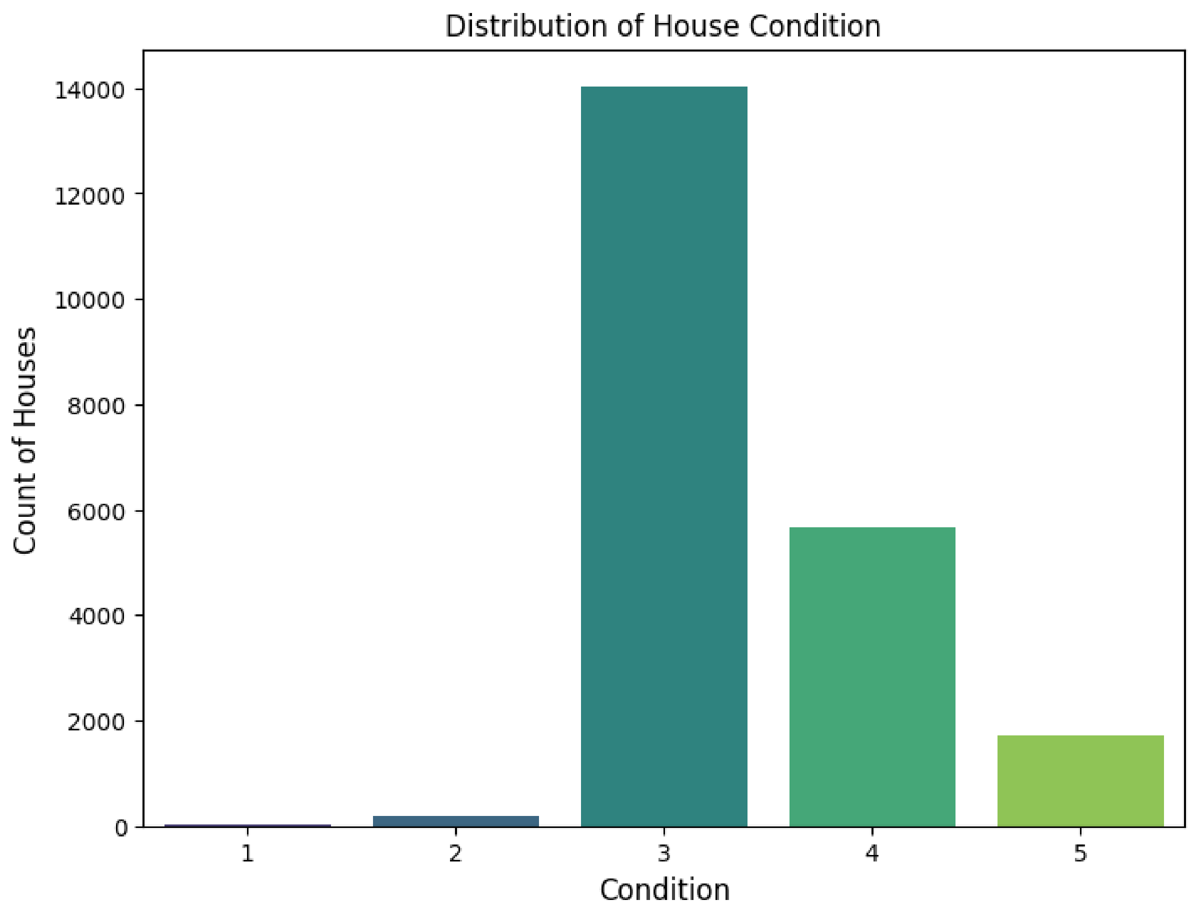


- **Scatter Plot**: Sqft Living vs. Price A clear upward trend indicates that larger houses tend to have higher prices, though the relationship isn't perfectly linear.



- **Countplot to visualize the distribution of house conditions.**

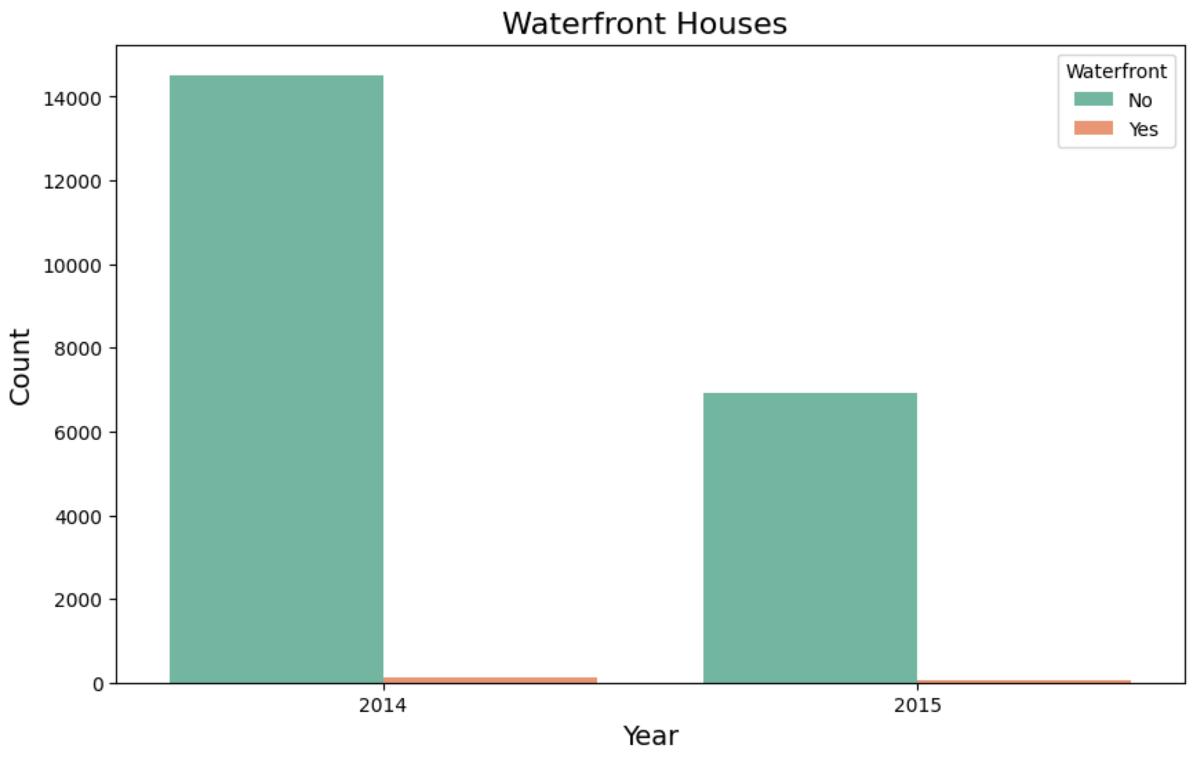
  Key Findings:
- The plot shows the frequency of each house condition category in the dataset.
- Most houses fall under condition 3 and condition 4, suggesting that the majority of the houses in this dataset are in average to good condition.
- Fewer houses fall into condition 1 (poor condition) and condition 5 (excellent condition), indicating these are less common in the dataset.



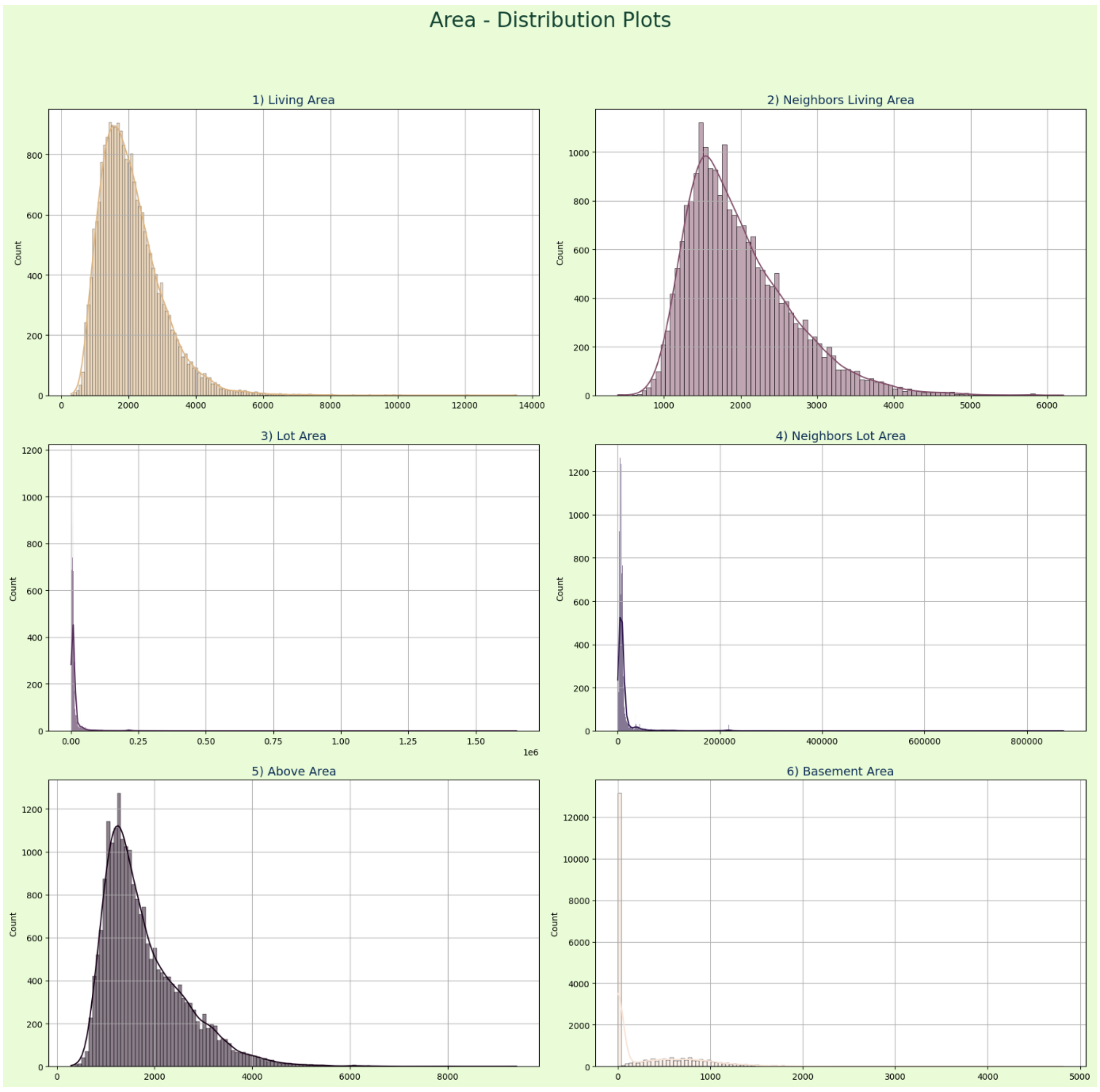
- **Countplot to show the distribution of waterfront houses by year.**

  Key Findings:

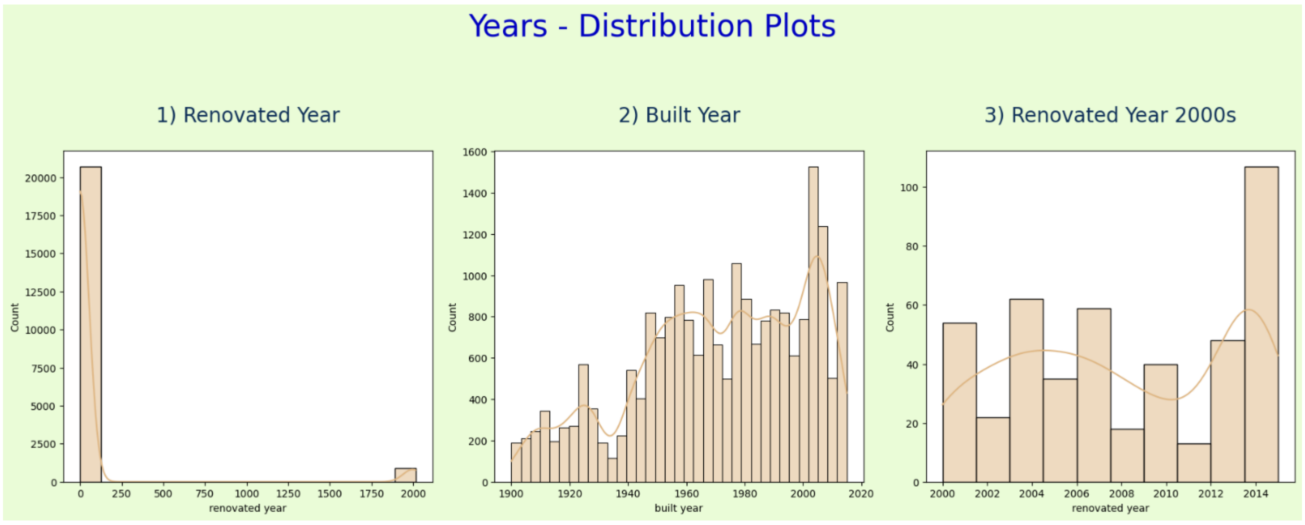
- The plot shows the distribution of houses with and without waterfronts, categorized by year.
- From the plot, we observe that houses with waterfronts have become more common in recent years compared to earlier years.
- Houses without waterfronts were more prevalent in the earlier years of the data.



- **Distribution of Area-related Features**
  This plot displays the distribution of various area-related features, including living space, lot size, and basement area. Each subplot represents the distribution of a specific area variable, such as sqft_living, sqft_living15, sqft_lot, and so on. The histograms show how these areas are spread across the dataset, while the overlaid kernel density estimate (KDE) curve provides insight into the overall shape of the distribution. The colors used for each plot help differentiate between the features visually.

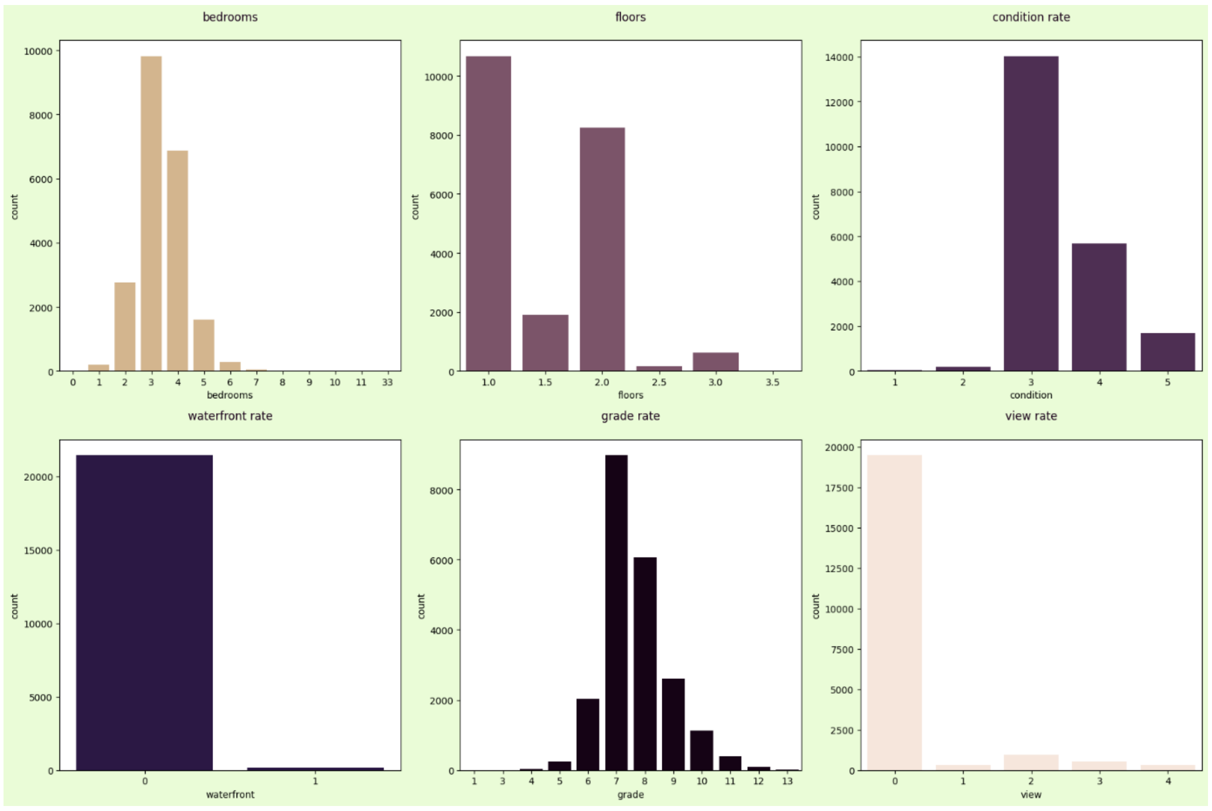


- **Distribution of Years (Year Built and Year Renovated)**
- This plot displays the distribution of the years houses were built and renovated. From the analysis of the years, we observe that most of the houses have not been renovated, as the year of renovation is recorded as 0, indicating no renovation. However, the houses that were renovated were mostly restored in 2015, which suggests that any renovations took place relatively recently (considering the dataset was collected in 2016).
- Regarding the construction history, the houses span a long period (from 1900 to 2015), with the largest number of houses being built between 2000 and 2015. However, there are also many houses that can be considered relatively old.



- **Distribution of Ordinal Categorical Features**
  The plot shows the distribution of various ordinal categorical features such as the number of bedrooms, number of floors, condition, waterfront presence, grade, and view rate.
- **Bedrooms**: The most common number of bedrooms is 3.
- **Floors**: Houses generally have 1 or 2 floors.
- **Condition Rate**: The condition of the houses is predominantly average or above average.

- **Waterfront**: 99% of the houses are not located on the waterfront.
- **Grade**: The structural design and quality of the houses is most frequently rated 7 out of 13.
- **View Rate**: Surprisingly, a large portion of the houses has a view rate of 0 out of 4.
- **Sale Year**: The ratio of houses sold in 2015 compared to 2014 is approximately 1:2.



5. Feature Selection and Encoding

In this section of the report, we will discuss the steps taken to build and evaluate different models to predict house prices based on the available data.

5.1. Model Selection

We selected a variety of models to predict house prices based on the available features. The following models were chosen:

Linear Regression: A basic model to understand the linear relationship between the variables and the target (price).

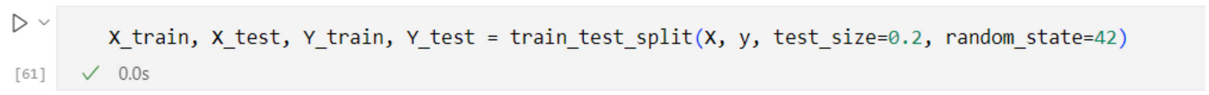
Polynomial Regression: To explore a non-linear relationship between the features and price.

Decision Tree Regression: To capture more complex relationships between features.

XGBoost Regression: An advanced model using boosting techniques to improve prediction accuracy.

5.2. Data Splitting

The dataset was split into **training set** and **test set** with an 80% split for training and 20% for testing. This ensures that the models are trained on one portion of the data and tested on a separate portion to avoid **overfitting**.



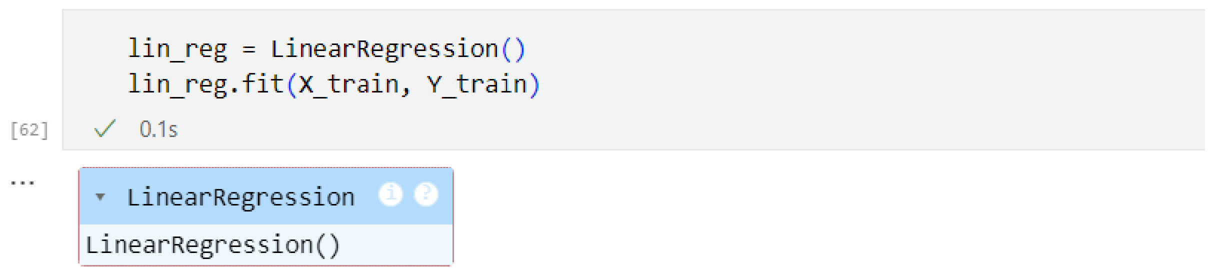
```
[61]  X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2, random_state=42)
      ✓ 0.0s
```

5.3. Model Training and Evaluation

Each model was trained using the **training data**, and **cross-validation** was applied to assess the performance across multiple splits of the data.

- We used the **fit()** method to train the models and identified the best performing one for price prediction.
- Each model was evaluated using R^2, MAE, and MSE. Below, you can find the performance metrics and images representing the results of each model.

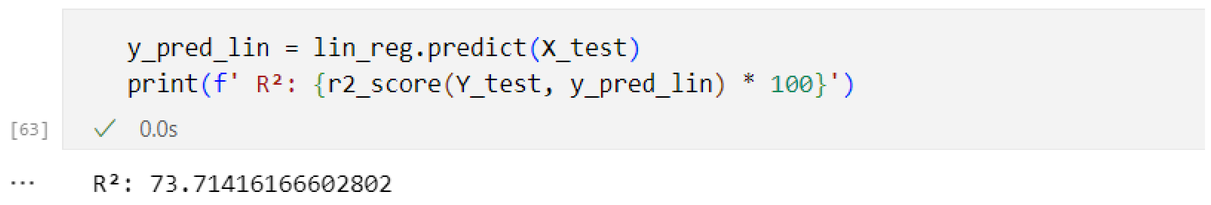
Linear Regression:



```
lin_reg = LinearRegression()
lin_reg.fit(X_train, Y_train)
```

```
[62]  ✓  0.1s
```

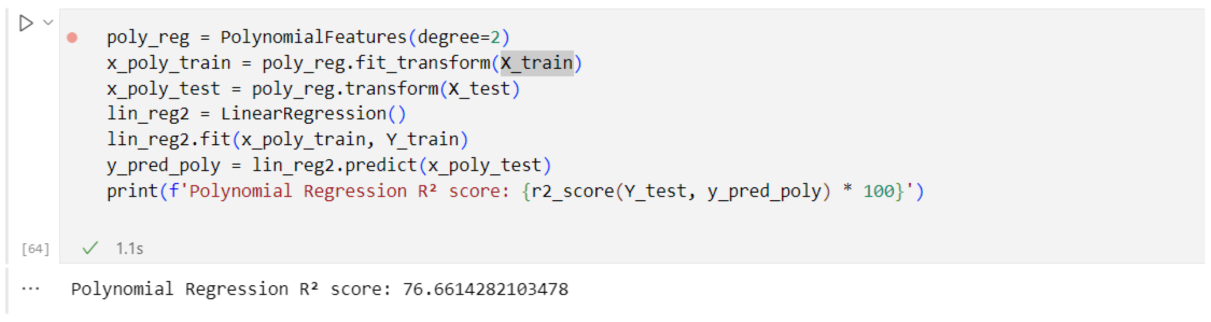
```
...  ▾ LinearRegression
     LinearRegression()
```



```
y_pred_lin = lin_reg.predict(X_test)
print(f' R²: {r2_score(Y_test, y_pred_lin) * 100}')
```

```
[63]  ✓  0.0s
...   R²: 73.71416166602802
```

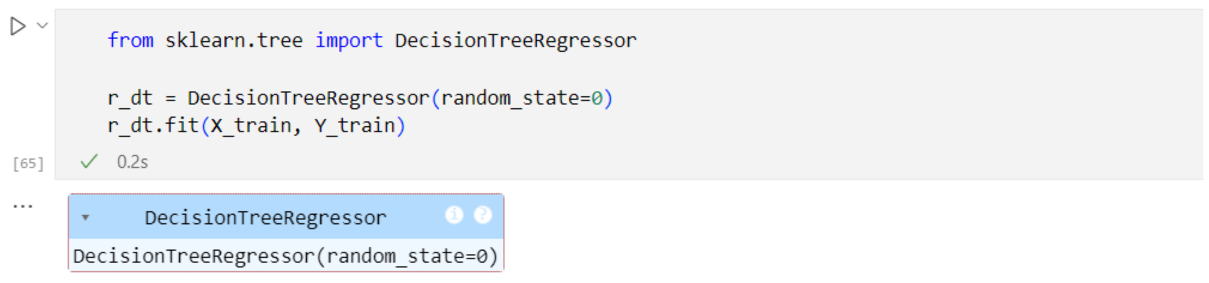
Polynomial Regression:



```
poly_reg = PolynomialFeatures(degree=2)
x_poly_train = poly_reg.fit_transform(X_train)
x_poly_test = poly_reg.transform(X_test)
lin_reg2 = LinearRegression()
lin_reg2.fit(x_poly_train, Y_train)
y_pred_poly = lin_reg2.predict(x_poly_test)
print(f'Polynomial Regression R² score: {r2_score(Y_test, y_pred_poly) * 100}')
```

```
[64]  ✓  1.1s
...   Polynomial Regression R² score: 76.6614282103478
```

Decision Tree Regression:

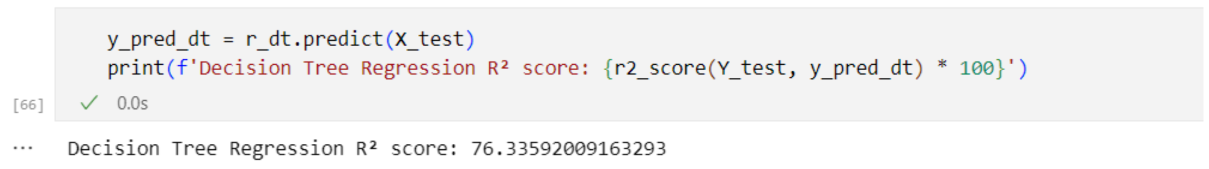


```
from sklearn.tree import DecisionTreeRegressor

r_dt = DecisionTreeRegressor(random_state=0)
r_dt.fit(X_train, Y_train)
```

```
[65]  ✓ 0.2s
```

```
DecisionTreeRegressor
DecisionTreeRegressor(random_state=0)
```

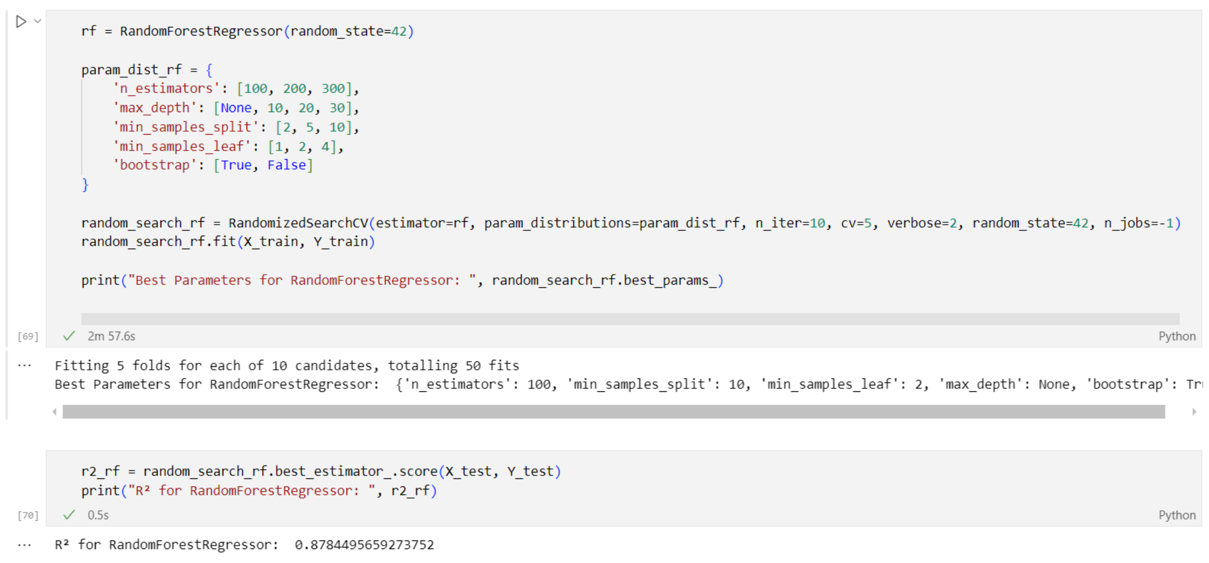


```
y_pred_dt = r_dt.predict(X_test)
print(f'Decision Tree Regression R² score: {r2_score(Y_test, y_pred_dt) * 100}')
```

```
[66]  ✓ 0.0s
```

```
Decision Tree Regression R² score: 76.33592009163293
```

Random Forest after Tunning:



```
rf = RandomForestRegressor(random_state=42)

param_dist_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

random_search_rf = RandomizedSearchCV(estimator=rf, param_distributions=param_dist_rf, n_iter=10, cv=5, verbose=2, random_state=42, n_jobs=-1)
random_search_rf.fit(X_train, Y_train)

print("Best Parameters for RandomForestRegressor: ", random_search_rf.best_params_)
```

```
[69]  ✓ 2m 57.6s                                                                Python
```

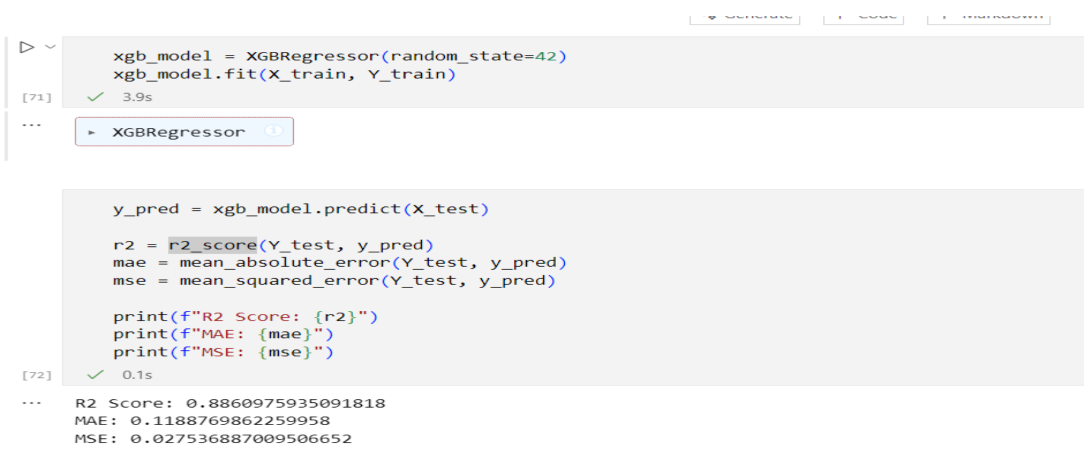
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Best Parameters for RandomForestRegressor:  {'n_estimators': 100, 'min_samples_split': 10, 'min_samples_leaf': 2, 'max_depth': None, 'bootstrap': Tr
```

```
r2_rf = random_search_rf.best_estimator_.score(X_test, Y_test)
print("R² for RandomForestRegressor: ", r2_rf)
```

```
[70]  ✓ 0.5s                                                                    Python
```

```
R² for RandomForestRegressor:  0.8784495659273752
```

XGBoost Regression:



```
xgb_model = XGBRegressor(random_state=42)
xgb_model.fit(X_train, Y_train)
```

```
[71]  ✓ 3.9s
```

```
XGBRegressor
```

```
y_pred = xgb_model.predict(X_test)

r2 = r2_score(Y_test, y_pred)
mae = mean_absolute_error(Y_test, y_pred)
mse = mean_squared_error(Y_test, y_pred)

print(f"R2 Score: {r2}")
print(f"MAE: {mae}")
print(f"MSE: {mse}")
```

```
[72]  ✓ 0.1s
```

```
R2 Score: 0.8860975935091818
MAE: 0.1188769862259958
MSE: 0.027536887009506652
```

5.4. Model Comparison and Conclusion

The XGBoost model showed the highest performance and is the most suitable for this dataset. We will proceed with XGBoost as the final model for house price prediction.

5.5. Hyperparameter Tuning

Some models, such as XGBoost, require hyperparameter tuning to improve performance. We used RandomizedSearchCV to find the optimal values for hyperparameters like:

- learning_rate
- n_estimators
- max_depth

5.6. Model Evaluation

The models were evaluated using the following metrics:

- R^2 (Coefficient of Determination): Measures how well the model explains the variance in the data.
- MAE (Mean Absolute Error): Calculates the average difference between predicted and actual values.
- MSE (Mean Squared Error): Measures the average squared difference between predicted and actual values.

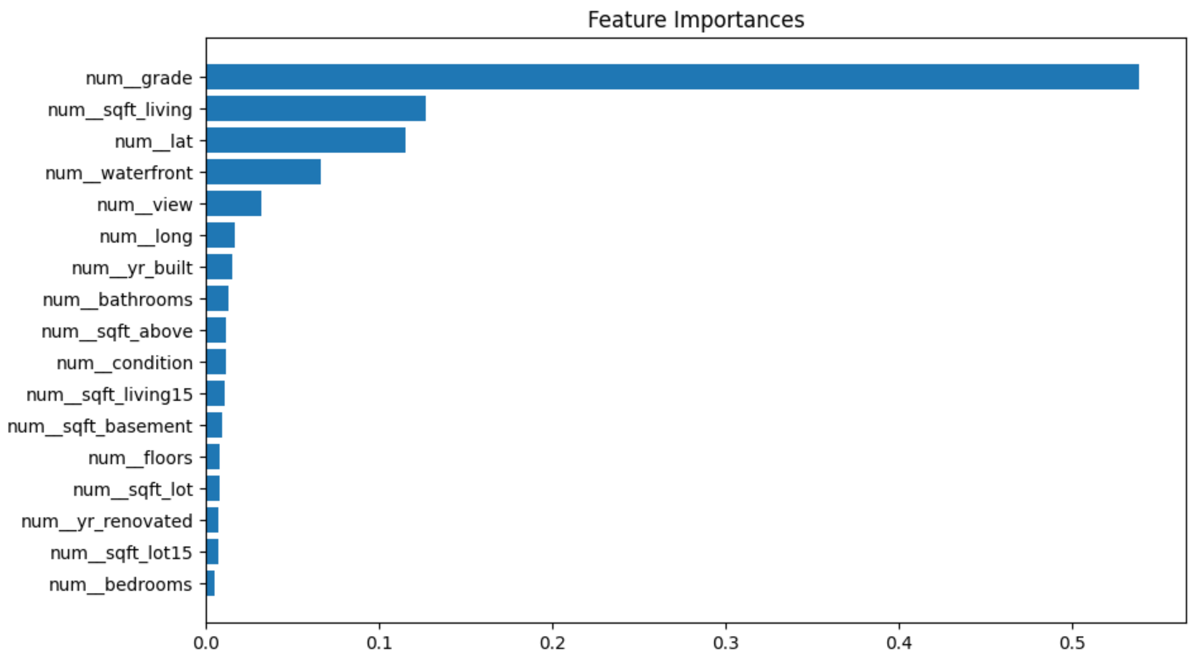
The results indicated that XGBoost Regression performed the best, achieving $R^2 = 0.9030$, MAE = 0.11817, and MSE = 0.0276.

5.7. Feature Importance

We used Feature Importance to analyze the features that have the greatest impact on house prices. Key findings include:

- Grade: The overall grade of the house.
- Sqft Living: The total living area of the house.
- Latitude: The geographic location of the house.

The results showed that these features have a significant impact on the house price, while others like yr_renovated had minimal effect.



5.8. Conclusion of Modeling

After evaluating the various models, we found that **XGBoost** was the best model for predicting house prices in this project. **Grade**, **Sqft Living**, and **Latitude** were the most important features influencing house prices, indicating that higher-rated houses, larger living spaces, and better locations command higher prices.

6. Deployment

In this section, we will discuss how the final model can be deployed in a real-world scenario to predict house prices based on input features from new homes.

6.1. Deployment Overview

After building and training the model using the available data, it can be deployed via an interactive web interface, allowing users to input features of new houses they want to estimate the price for. For example, a user might enter the number of bedrooms, bathrooms, living space area, and the year the house was built. The model then processes these inputs and predicts the house price.

6.2. How the Application Works

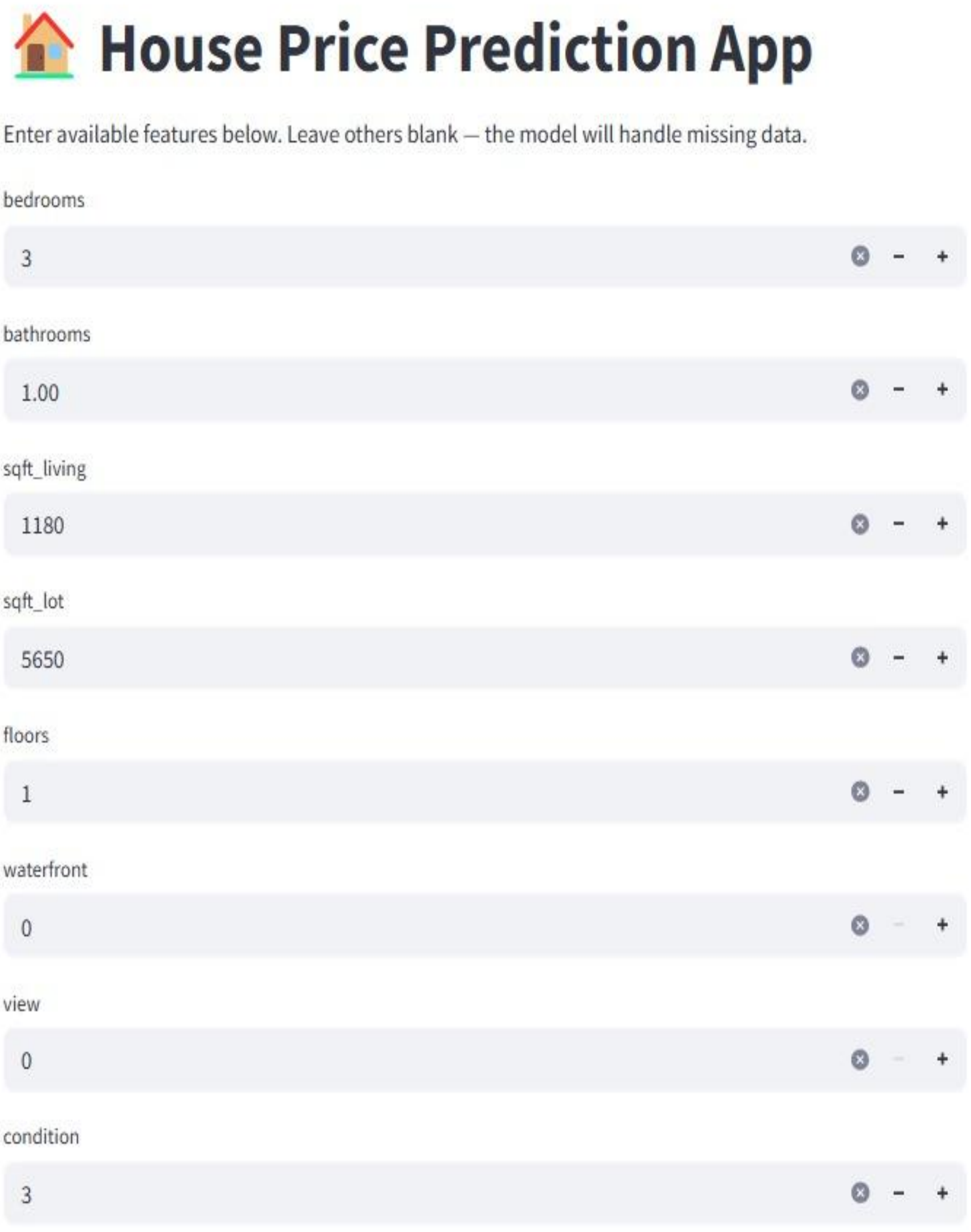
- **User Interface**: The user interface is designed to allow users to easily enter data about the house, such as the number of rooms, area, house condition, and other relevant features.
- **Price Prediction**: Once the data is entered, the model uses the provided inputs to predict the house price. The predicted price is displayed immediately on the application interface.
- **Handling Missing Inputs**: If some features are not provided, the model handles the missing inputs appropriately based on the techniques used during training.

6.3. Application Features

- **Ease of Use**: The application is easy to use, allowing users to quickly input information and receive an accurate estimate.
- **Prediction Accuracy**: The model offers accurate price predictions based on the data it has been trained on, utilizing a wide range of features that influence house prices.

6.4. Conclusion

By deploying the model as an interactive tool, users can easily obtain valuable insights into house prices based on specific features, providing a practical solution for buyers, sellers, and real estate professionals alike.



🏠 House Price Prediction App

Enter available features below. Leave others blank — the model will handle missing data.

bedrooms
3

bathrooms
1.00

sqft_living
1180

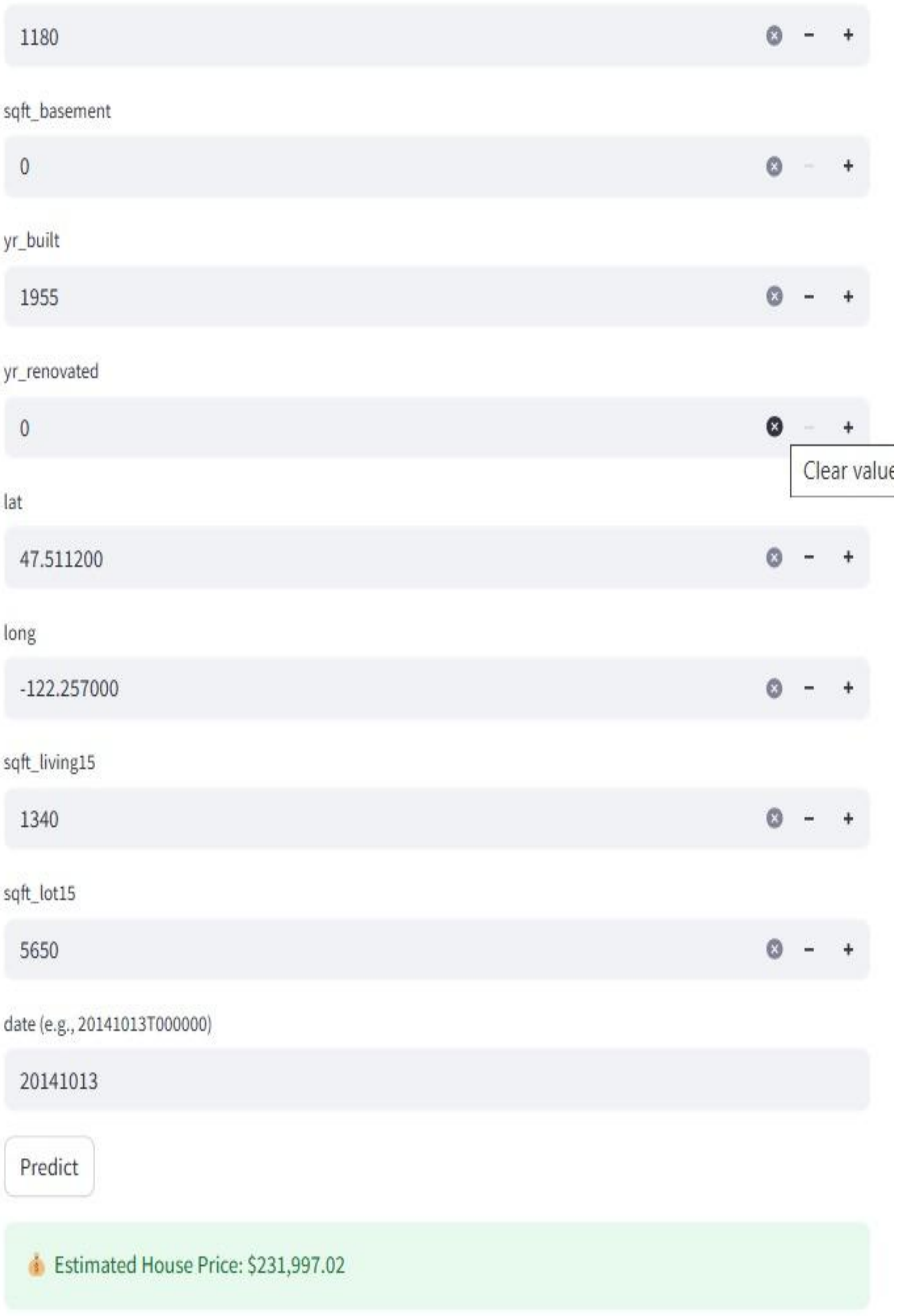
sqft_lot
5650

floors
1

waterfront
0

view
0

condition
3


1180

sqft_basement

0

yr_built

1955

yr_renovated

0

Clear valu

lat

47.511200

long

-122.257000

sqft_living15

1340

sqft_lot15

5650

date (e.g., 20141013T000000)

20141013

Predict

💰 Estimated House Price: $231,997.02

"If there's anything unclear, Ask GPT or Google it!"

Thank you