King County House Prices Predictive Modeling

This Project Done By

Zahra Abdelkareem Al-Hourani

Jameel Hisham Jameel Hamad

Obaida Majed Hassouneh

Data Science Programme Provided By

Orange

July, 2023

Abstract

 The King County House Prices Predictive Model project aims to develop an accurate and reliable model for predicting the sale prices of houses in King County, Washington. The project utilizes a comprehensive dataset that encompasses various attributes of residential properties, including location, size, number of bedrooms, bathrooms, and other pertinent features.

The primary objective of this project is to assist homeowners, real estate agents, and potential buyers in the King County area by providing an effective tool for estimating house prices. By leveraging advanced machine learning techniques and predictive modeling algorithms, the developed model offers valuable insights into the factors that influence house prices within this particular region.

The project follows a systematic approach, encompassing data preprocessing, feature engineering, model selection, training, and evaluation. Several regression algorithms, such as linear regression, support vector regression, or gradient boosting, are explored and compared to identify the most accurate and robust model for the King County housing market.

Evaluation of predictive models was performed using appropriate metrics, such as mean absolute error or R-squared, to assess the models' performance and generalization capabilities. In addition, a feature importance analysis was conducted to identify the key factors affecting home prices in the area.

The anticipated outcome is a predictive model that offers accurate estimates of house prices, enabling informed decision-making in the King County housing market.

1 Introduction

1.1 Overview

The King County House Prices Prediction project focuses on developing an accurate model for estimating house prices in King County, Washington. The project recognizes the significant impact of house features on property valuation. By analyzing a comprehensive dataset of property attributes, including location, size, number of bedrooms, bathrooms, and other relevant factors, the project aims to identify the key features that influence house prices. Through the application of advanced machine learning techniques and predictive modeling algorithms, the project seeks to establish relationships between these features and the corresponding price outcomes. This analysis will provide valuable insights into how specific house features contribute to variations in property values, ultimately assisting homeowners, real estate professionals, and potential buyers in understanding the pricing dynamics of the local housing market.

1.2 Objective of Analysis

The objective of this project is to develop an accurate and reliable predictive model that estimates house prices within the King County region of Washington state. The project aims to provide valuable insights and predictions for homeowners, real estate professionals, and potential buyers in the area. By leveraging advanced machine learning techniques and analyzing a comprehensive dataset of property attributes, the project seeks to identify the key factors that influence house prices. The ultimate goal is to assist stakeholders in making informed decisions,

2 Challenges

2.1 Outliers

- We found an outlier in the 'bedrooms' column with a value of 33

    - But, With 1 bathroom   and a sale price of 640,000$ it is likely that this house has 3 bedrooms, and the 33 was a data entry error.

- While analyzing the data, we noticed abnormal values in the price column.

    - Our approach includes excluding these outliers by considering only the data within one standard deviation. However, after completing the analysis and prediction, We noticed that the results weren't satisfactory.
    - We decided to include these outliers in our analysis and prediction model.

★   In the King County dataset, outliers in the context of housing prices can provide valuable insights into the real estate market. These outliers represent properties with significantly higher or lower prices compared to the majority of properties in the area. High-priced outliers may indicate luxury properties or unique architectural designs with exceptional amenities, commanding a premium in the market. On the other hand, low-priced outliers may suggest distressed properties, such as foreclosures or properties in need of significant repairs. Additionally, outliers in housing prices could also be attributed to data entry errors or other anomalies in the dataset. Exploring and analyzing these outliers can help identify interesting market trends, understand the factors influencing property values, and potentially uncover specific niches within the housing market in King County.

3 Dataset

3.1 Description

For this project, I used the King County dataset which is available publicly at Kaggle data repository. This dataset contains real data about House Prices from 2014 – 2015. The dataset consists of 21 columns and 21,613 rows, (see Table 3.1) which contains various information about the dataset.

Table 3.1 : The detailed attributes of "King County Dataset"

| Column Name	Data Type	Data Definition
id	Numeric	unique Identified for house
date	Date	Sale Date
price	Numeric	House Price
bedrooms	Numeric	Number of Bedrooms
bathrooms	Numeric	Number of Bathrooms
sqft_living	Numeric	Square Footage of the House Area
sqft_lot	Numeric	Square Footage of the Outdoor Area
floors	Numeric	Foors (Levels) in House
waterfront	Numeric	A House that have a View of a Water
view	Numeric	A House that have a Special View
condition	Numeric	How Good the Condition is (Overall)
grade	Numeric	Overall Grade Given ti the housing unit, Based on King County Grading System
sqft_above	Numeric	Square Footage of the Area above the Basement
sqft_basement	Numeric	Square Footage of the Basement Area
yr_built	Numeric	Built Year
yr_renovated	Numeric	Year of Rnovation
zipcode	Numeric	Identufy Specific Geographic Region
lat	Numeric	Latitude Coordenate
long	Numeric	Longitude Coordenate
sqft_living15	Numeric	Square Footage of the Interior Housing Space for the nearest 15 neighbor
sqft_lot15	Numeric	Square Footage of the Outdoor Area for the nearest 15 neighbor

4 Methodologies

4.1 Data Wrangling

- Redundant Values

    - We note that there are about 177 entries with the same id. The reason is likely because the house sold twice. We decided to keep these entries.

- Dropped Highly Correlated Columns to avoid multicollinearity

    -'id'

    -'date'

    -'sqft_living15'

    -'sqft_lot15'

    -'sqft_above'

- We Made changes to some of the attributes like

    - we have changed the data type of the 'date' column from numeric to date time.

    - we extracted the year from the date column and stored it in a new column called ' Year_of_sale'.

    - we have changed the value in the 'yr_renovated' to become 'Renovated' with zero and one's values.

    - we have changed the value in the 'basement' and 'view' columns to zero and one's

- Dealing with Zero values

    - There is zero values in 'bedrooms' and 'bathroom' columns

    - We discovered that houses with 0 bedrooms and 0 bathrooms are typically associated with specific types of properties, including studios, lofts, or open concept living spaces. where there are no bedrooms or separate enclosed bathroom areas.

4.2 Data Exploratory

This is not very big data and we do not have too many features. Thus, we have a chance to plot most of them and reach some useful analytical results. Drawing charts and examining the data before applying a model is a very good practice because we may detect some possible outliers or decide to do normalization. This is not a must

but getting to know the data is always good. Then, I started with the histograms of the dataframe.

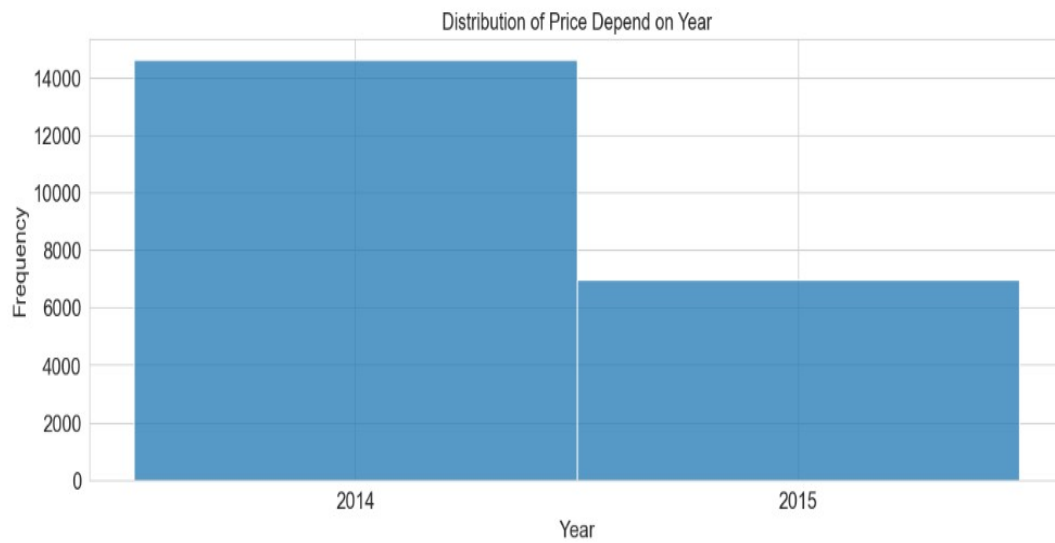The following diagrams illustrate the overall process of the proposed work
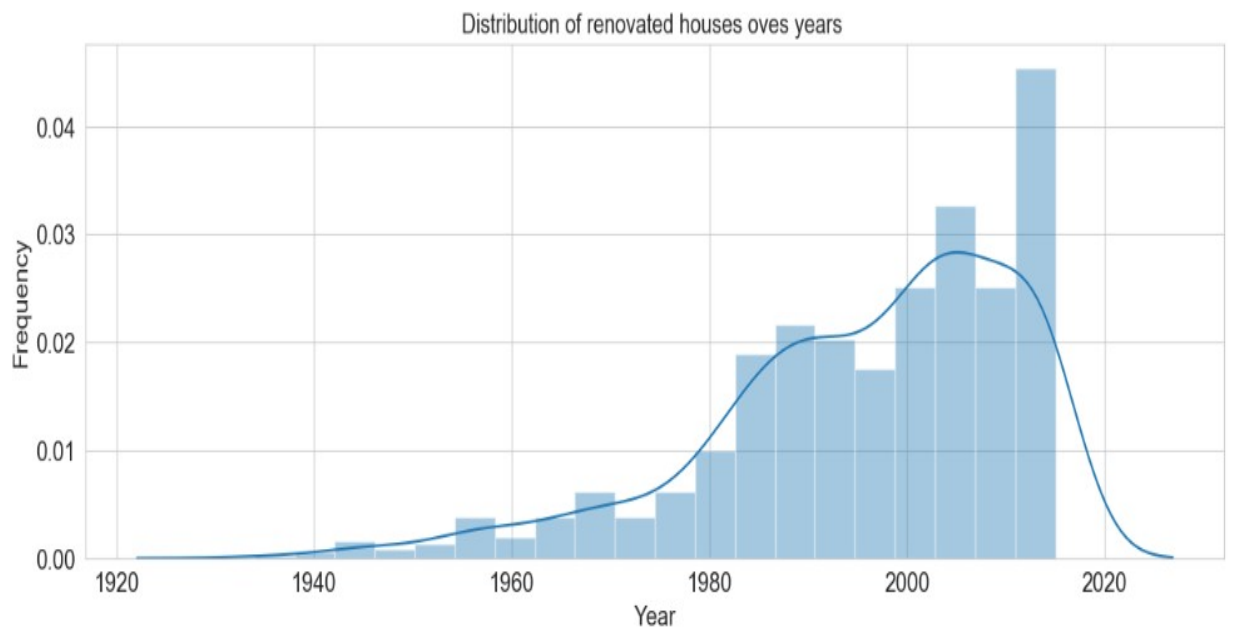
4.2.1: Distribution of Renovated Houses over Years
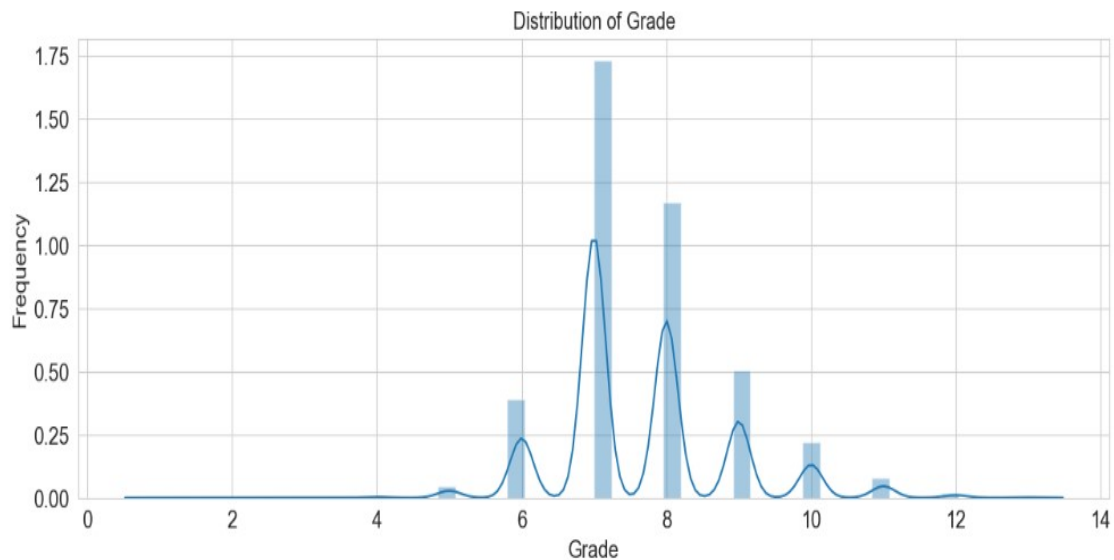
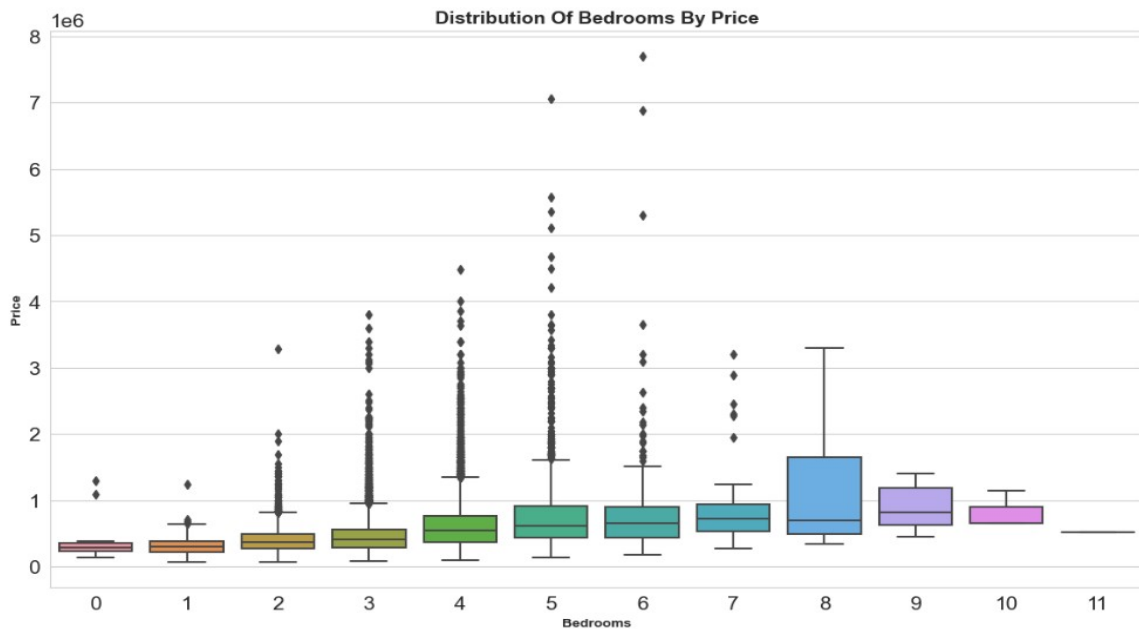4.2.2: Distribution of Houses Grades

4.2.3: Distribution of Prices

4.2.4: Distribution of Prices Depend on Year

Overview of house features

How does every feature affect the house price?

To determine distribution of bedrooms, floors, or bathrooms vs price, we preferred to use boxplot because we have numerical data but they are not continuous as 1,2,... bedrooms, 2.5, 3,... floors
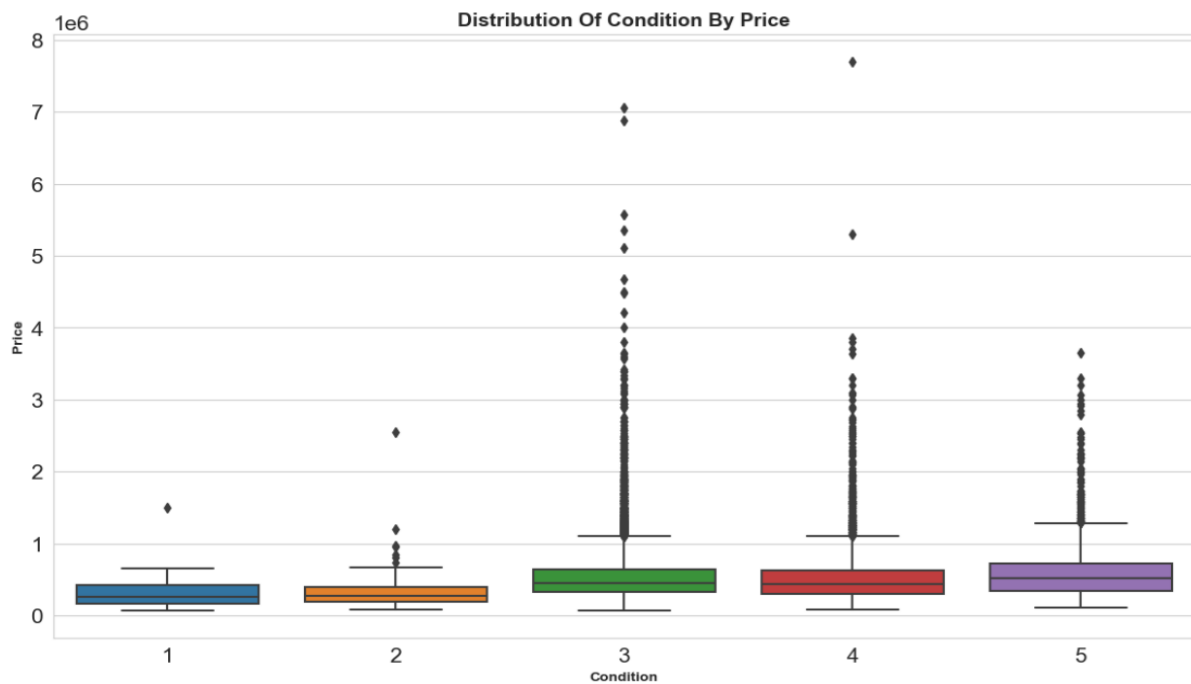

4.2.5: Distribution of Bedrooms By Price

- Referring to this box plot, We can conclude that there is a positive relationship between price and 'bedrooms' while its increase, So we decided to apply some statistics to get in-depth insight

- We calculate the mean living square-footage and look at bedroom count for houses within 0.1 standard deviation of this mean.
  - We conclude that :
    - We are looking at 1811 houses with living space between 1990 sq_ft and 2170 sq_ft.
    - for the houses of space between 1990 sq_ft and 2170 sq_ft, the optimal bedroom count is 8.
    - Adding an additional bedroom does not necessarily affect the house price.
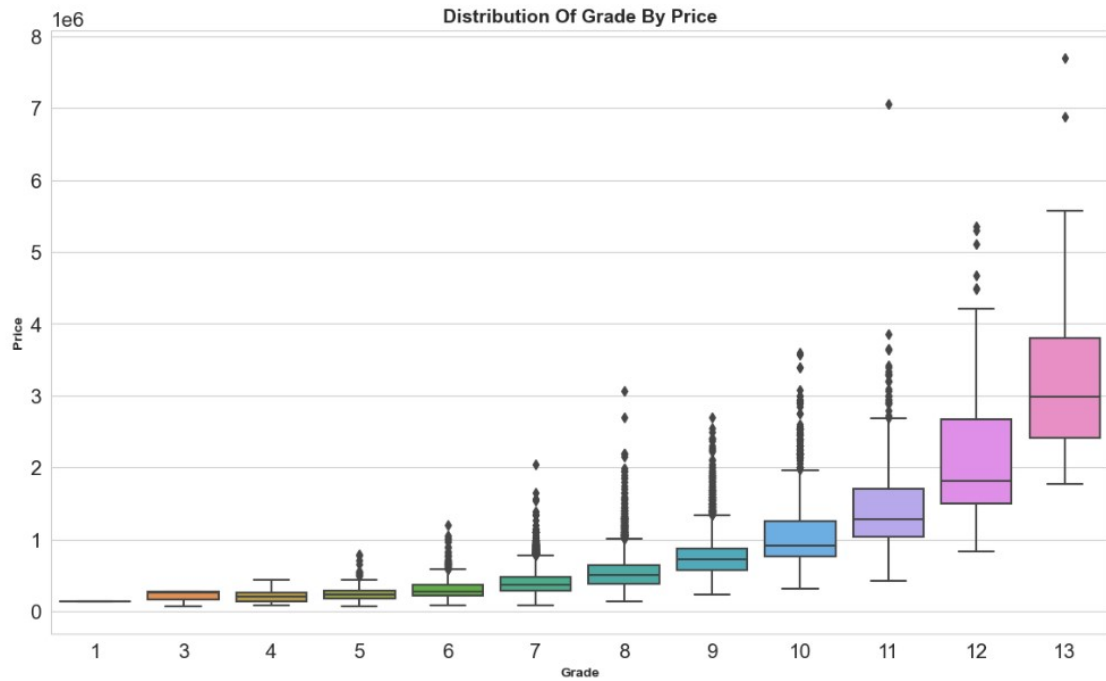
*4.2.6: Distribution of Grade By Price*

- Referring to the box plot the median house price increases with grade. Which indicates that these features are positively correlated.

4.2.7: Distribution of Condition By Price

- Referring to this box plot, We can conclude that there is a positive relationship between 'price' and 'condition', So condition feature affect the price

Let's see the relationship between price and some variables such as sqft_living, sqft_lot, and sqft_ living vs price with hue grade through the scatter plots
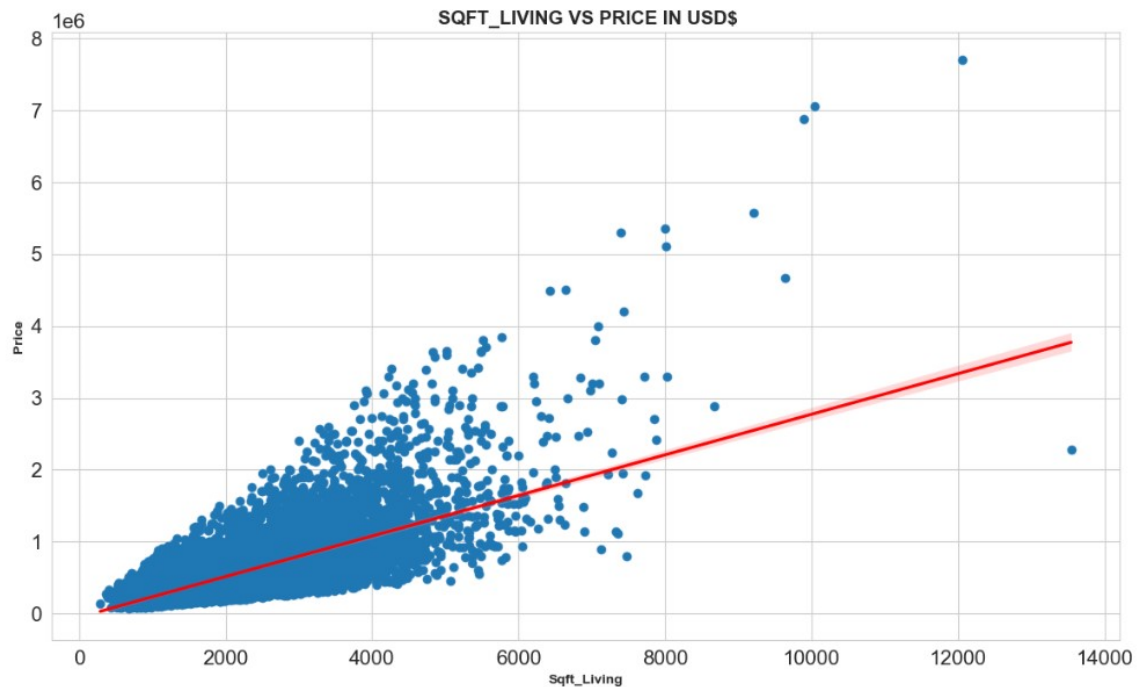
4.2.8: Distribution of sqft_living vs price

- strong relationship between 'sqft_living' and 'price' which means The more living space, the higher the price
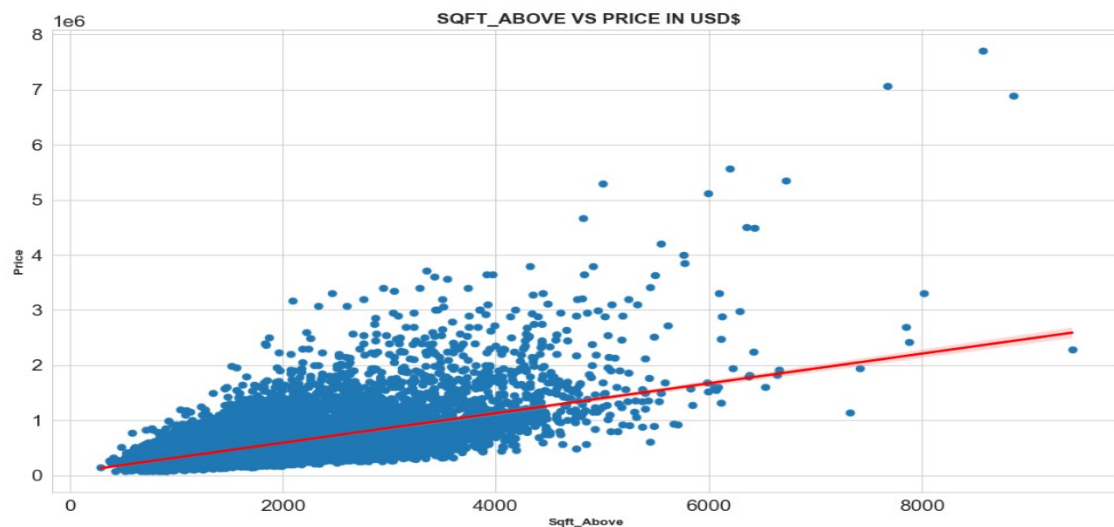
4.2.9: Distribution of sqft_above vs price

- strong relationship between 'sqft_above' and 'price' which means The more above space, the higher the price

12

◆   Note that we won't keep all the sqft-type features for our model due to them likely being correlated. This will be explored further in the multicollinearity section below. Note that sqft_living and sqft_above look to be most appropriate due to demonstrating greater linearity with respect to our target feature, whereas sqft_lot is the least linear


4.2.10: Distribution of sqft_living vs price price with hue grade

- Referring to the scatter plot, there is a clear relationship between grade and living area
- The lower grade houses also have a small square footage
- Once a house has a living area of at least 2000 sqft, it is most likely to be average or above-average in terms of grade.

◆ Correlation is an important aspect to consider when working with features in a dataset. When there is a very high correlation between two features, it is often not advisable to keep both of them as it can lead to overfitting. For example, if overfitting is observed, one may choose to remove features like 'sqt_above' or 'sqt_living' due to their high correlation. It is possible to estimate this relationship by examining the definitions in the dataset, but it is crucial to verify the correlation matrix to ensure accuracy.

However, it is important to note that the presence of high correlation does not always necessitate the removal of one of the features. Consider the case of 'bathrooms' and 'sqt_living', which exhibit high correlation. Despite the correlation, the relationship between them may not be the same as the relationship between 'sqt_living' and 'sqt_above'. Thus, it is essential to evaluate the context and the specific information conveyed by each feature before making decisions about feature removal.
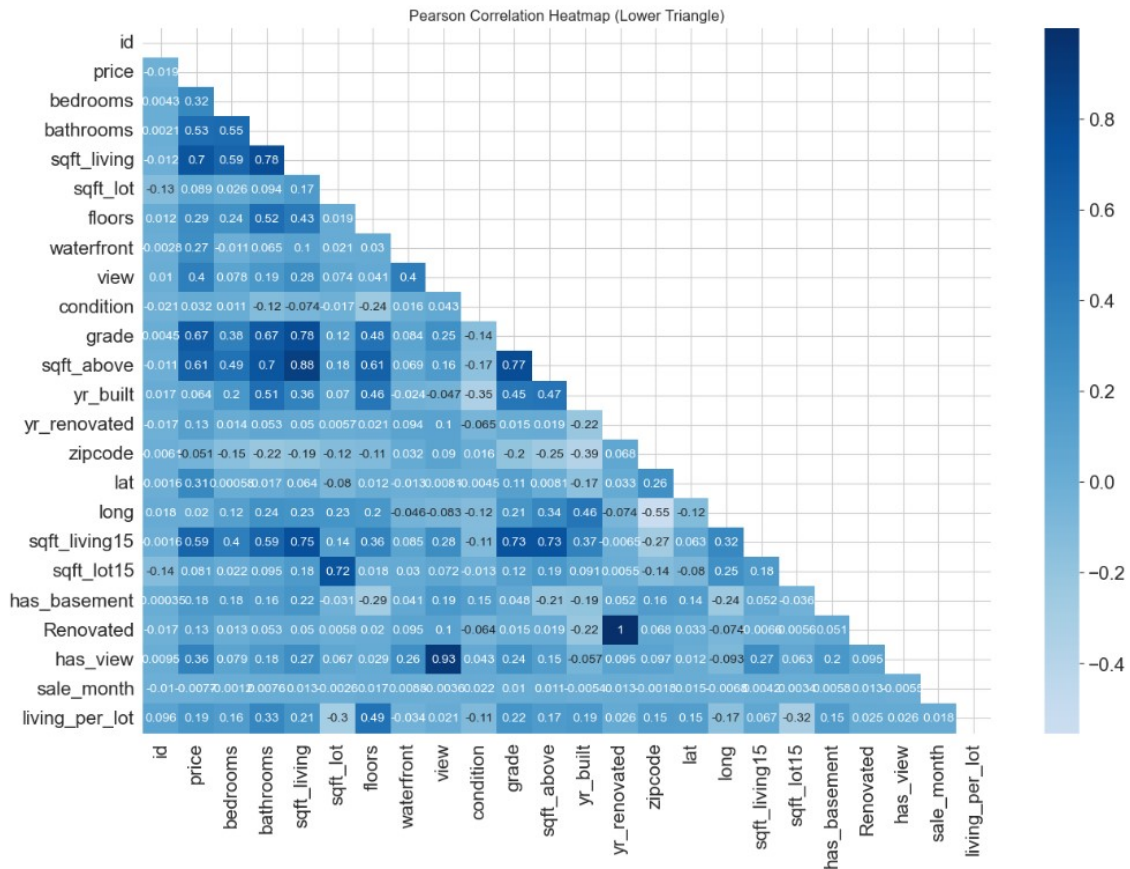
3.3.11: Correlation Matrix

- Correlation between sqft_living and price = 0.70 it is a very strong relationship which means the house with a large living square footage has a higher price

- Correlation between sqft_above and price = 0.59 it is a strong relationship which means the house with a large above square footage has a higher price.

'sqft_above' is the square footage of the house apart from the basement, and as we know that most of the houses didn't have a basement. We decided to drop this feature'

4.3 Data Splitting

The process of data splitting is an essential step to ensure reliable model evaluation. The dataset was divided into separate training and testing sets to assess the performance of the predictive models accurately.Additionally, the target variable was split into y_train and y_test. The test size was set to 0.25, representing 25% of the data, and a random state of 0 was specified for reproducibility.

This data splitting methodology ensures that a portion of the dataset remains unseen during model training, allowing for an unbiased evaluation of the trained models on the testing set.

5 Feature Selecting

In this project, the feature selection process played a vital role in improving model performance and enhancing interpretability. To identify the most relevant features, an exhaustive search approach was employed. Exhaustive search involved systematically evaluating all possible feature combinations to find the optimal subset. The evaluation metrics used during this process were carefully selected based on the project's objectives, ensuring that the chosen subset maximized the desired performance metric, such as accuracy or mean squared error. While exhaustive search can be computationally demanding, strategies such as parallel computing were implemented to manage computational resources effectively. The results obtained from the exhaustive search provided valuable insights into the subset of features that exhibited the strongest relationship with the target variable, enabling robust analysis and informed decision-making. It is important to acknowledge that while exhaustive search ensures optimal feature selection, it may come with limitations such as increased computational time and the need for careful consideration of model complexity. However, these challenges were mitigated through diligent management and validation of the results.

- **the results of exhaustive search was:**
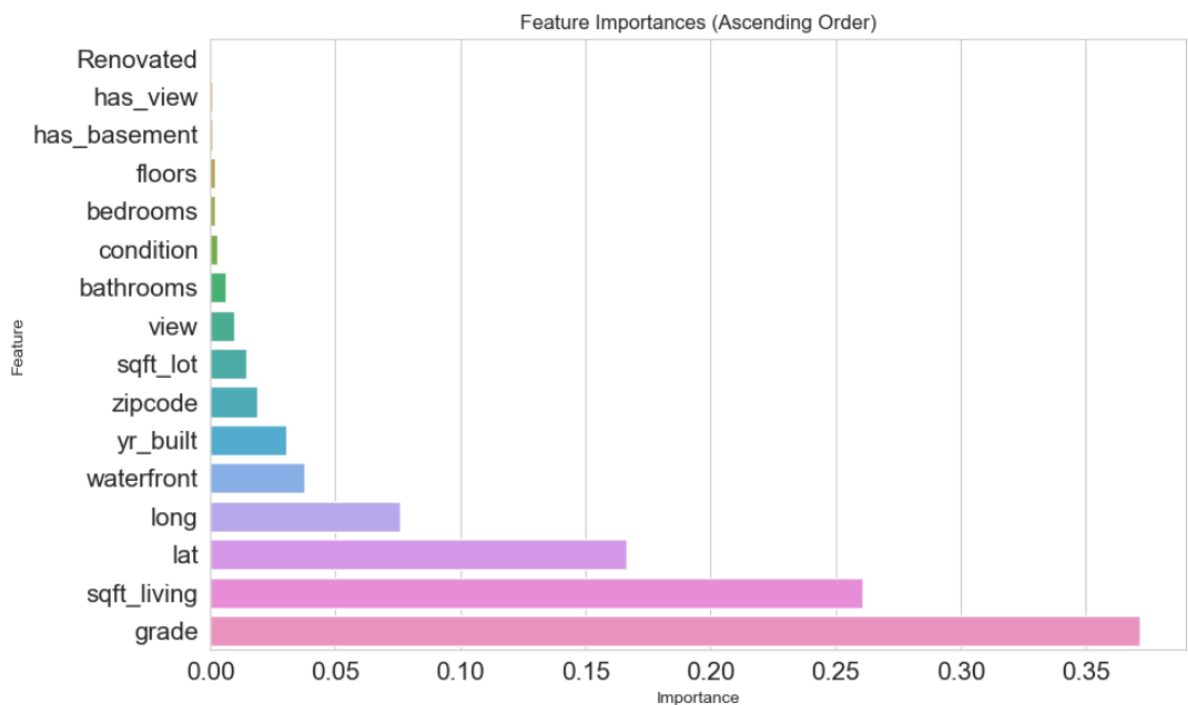
5.1: Selected Features

6 Predictive Modeling

We are examining the prediction accuracy of proposed models and seeing the best model of them with the best accuracy.

6.1 Choosing the appropriate model:

- Selecting an appropriate model for the house prices dataset depends on the data characteristics, exploratory data analysis, and specific requirements of the task. This process helps narrow down the available models to ensure the chosen model is suitable for the given objectives. Considering the type of model that best fits the data and problem at hand involves exploring various options, such as linear models and tree-based models. Furthermore, evaluating the model's performance and complexity is crucial, as it aids in striking a balance between the model's predictive accuracy and its complexity. Finding this balance is essential to ensure the selected model achieves the desired performance while avoiding excessive complexity.

- The Grid Search technique was utilized to select the best hyperparameters for the predictive model. Grid Search is a systematic approach that exhaustively searches through a predefined set of hyperparameter combinations to identify the optimal configuration that yields the highest model performance.

```python
# Create an instance of CatBoost Regressor
Cat_Boost = CatBoostRegressor()

# Fit the CatBoost model to the training data
Cat_Boost.fit(X_train , y_train)

# Create an instance of GridSearchCV
Grid_Parameters = {'n_estimators': np.arange(50, 200, 10),
    'max_depth':np.arange(5,15),
    'learning_rate': [0.1, 0.2, 0.3]}

# Create an instance of GridSearchCV
Grid_Search = GridSearchCV(Cat_Boost, Grid_Parameters , cv=5, scoring='neg_mean_squared_error')

# This line fits the GridSearchCV on trained data
Grid_Search.fit(X_train , y_train)

#This line retrieves the best hyperparameters
Best_Parameter = Grid_Search.best_params_

# Calculate model preformane
Best_Score = Grid_Search.best_score_

print(Best_Parameter)
```

6.1.1: Apply Grid Search

-In this project, the CatBoost model was employed for predictive modeling. CatBoost is a gradient boosting algorithm designed to handle categorical variables efficiently. To apply the CatBoost model, the data was preprocessed, including cleaning and encoding categorical features, handling missing values, and scaling numerical variables. The model was trained using the prepared data, with hyperparameters tuned to optimize performance. Cross-validation techniques, such as k-fold validation, were employed to assess the model's performance and avoid overfitting.

The result of print statment

- By performing GridSearchCV we got these results

```
{'learning_rate': 0.2, 'max_depth': 6, 'n_estimators': 190}
```

```python
# Create a CatBoost Regressor
Cat_Boost = CatBoostRegressor(n_estimators = 190, learning_rate = 0.2 , max_depth = 6)

# Fit the CatBoost Regressor model to the training data
Cat_Boost.fit(X_train, y_train)

# Apply the model on training dataset
y_pred = Cat_Boost.predict(X_train)

# Apply the model on testing dataset
y_test_pred = Cat_Boost.predict(X_test)

# Calculate model preformane
Cat_Boost_Accuracy = metrics.r2_score(y_test, y_test_pred)
Cat_Boost_RMSE = np.sqrt(metrics.mean_squared_error(y_train, y_pred))
```

6.1.2: Apply CatBoost Regressor

6.2 Model Evaluation

Upon evaluation, the CatBoost model demonstrated exceptional predictive accuracy, outperforming other algorithms tested in the project. The model exhibited strong performance across various evaluation metrics, including R-Squared , and RMSE.

```python
Regression_Models = pd.DataFrame({
    'Regression Model': ['CatBoostRegressor', 'RandomForstRegressor' ,'DecisionTreeRegressor','LinearRegression'], 'RMSE' : [Cat_
    'R^2': [Cat_Boost_Accuracy*100, Random_Forst_Accuracy*100,Decision_Tree_Regressor_Accuracy*100,Linear_Regression_Accuracy*100
Regression_Models.sort_values(by='R^2', ascending=False)
```

	Regression Model	RMSE	R^2
0	CatBoostRegressor	83142.080607	90.231686
1	RandomForstRegressor	81793.414654	88.584926
2	DecisionTreeRegressor	117957.740442	80.290568
3	LinearRegression	202678.735192	69.076718

6.2.1: CatBoost Result in Comparison with other Regression Models

6.3 Model saving:

In this example, Cat_Boost represents the trained CatBoost model object, and save_model is the method that saves the model weights and other relevant information to the specified file. After executing this line of code, the trained CatBoost model will be saved to the "model_weights.cbm" file.

- Saving the model in this way allows you to reuse the trained model later for prediction on new data without having to retrain it from scratch.

```
In [126]: # Saving the weights and other necessary information of the CatBoost model to the model_weights.cbm file

Cat_Boost.save_model("model_weights.cbm")
```

6.3.1: CatBoost Result Saving

6.4 Mode Deployment:

- Deploying a CatBoost model using 'Gradio' is a straightforward and efficient way to create an interactive and user-friendly interface for real-time predictions.

- Gradio is a Python library that simplifies the process of building web interfaces for machine learning models. By leveraging Gradio, the CatBoost model can be integrated into a web application effortlessly.

- With Gradio, you can define input and output interfaces, such as text input fields, to capture user input and display the model's predictions in real-time. This allows users to interact with the CatBoost model directly through a web browser without the need for coding or technical expertise.

- By combining the power of CatBoost with the convenience and versatility of Gradio, deploying the model becomes a seamless process, enabling users to access and benefit from the predictive capabilities of the CatBoost model with ease.

```
In [1]: import numpy as np
        import gradio as gr
        import numpy as np
        from catboost import CatBoostRegressor

        # Load the saved model
        Cat_Boost = CatBoostRegressor()

        Cat_Boost.load_model("model_weights.cbm")
        # Prepare the input data for prediction
        def predict(bedrooms, bathrooms, sqft_living, sqft_lot, floors, waterfront, view, condition, grade, yr_built, zipcode, lat, long,
            input_data = np.array([bedrooms, bathrooms, sqft_living, sqft_lot, floors, waterfront, view, condition, grade, yr_built, zip
            prediction = Cat_Boost.predict(input_data)    # Make the prediction using the loaded CatBoost model
            rounded_prediction = round(prediction[0], 2)  # Round the prediction to 2 decimal places
            return rounded_prediction

        # Define the input components for the Gradio interface
        input_components = [
            gr.inputs.Number(label="Number of Bedrooms"),
            gr.inputs.Number(label="Number of Bathrooms"),
            gr.inputs.Number(label="Sqft Living"),
            gr.inputs.Number(label="Sqft Lot"),
            gr.inputs.Number(label="Floors"),
            gr.inputs.Checkbox(label="Waterfront"),
            gr.inputs.Checkbox(label="View"),
            gr.inputs.Number(label="Condition"),
            gr.inputs.Number(label="Grade"),
            gr.inputs.Number(label="Year Built"),
            gr.inputs.Number(label="Zipcode"),
            gr.inputs.Number(label="Latitude"),
            gr.inputs.Number(label="Longitude"),
            gr.inputs.Checkbox(label="Has Basement"),
            gr.inputs.Checkbox(label="Renovated"),
            gr.inputs.Checkbox(label="Has View"),
        ]
        # Define the output component for the Gradio interface
        output_component = gr.outputs.Textbox(label="Prediction")
        # Create and launch the Gradio interface
        gr.Interface(fn=predict, inputs=input_components, outputs=output_component, title="House Price Predictor", button_text="Predict",
```
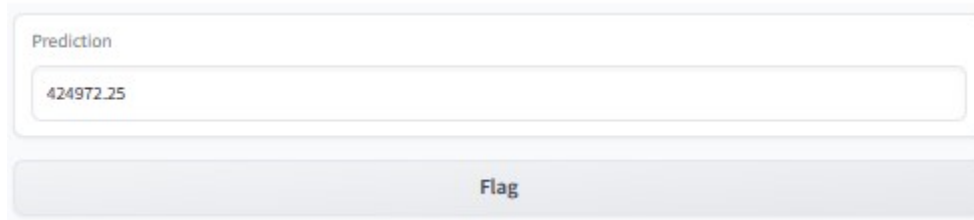
6.4.1: CatBoost Deployment

The GUI offers an intuitive and visually appealing interface that allows users to easily input the necessary information and receive immediate predictions for house prices. It eliminates the need for manual coding or technical expertise, making it accessible to a broader audience interested in estimating house prices accurately.

6.4.2: Prediction GUI Input Section

- The interface consists of input components where users can provide the details of a house, such as the number of bedrooms, bathrooms, square footage, floors, waterfront presence, view, condition, grade, year built, zip code, latitude, longitude, and information about basement, renovation, and view availability.
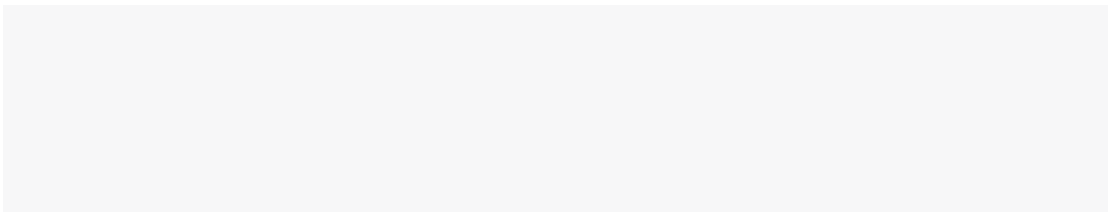
- Users can interact with the interface by filling in the input fields with the relevant information. Once the user clicks on the "Predict" button, the CatBoost model loaded from the saved weights is utilized to process the input data and generate a prediction.

6.4.3: Prediction GUI Output Section

- The output component of the interface is a textbox where the predicted house price is displayed. This prediction is computed based on the provided house characteristics and the model's learned patterns and correlations from the training data.

- Gradio provides a default flag button as part of its interface, which allows users to report any issues or problems they encounter while using the application. When users click on the flag button, it typically triggers a mechanism to collect feedback or report the issue to the application owner or administrator. But in our case by clicking flag the data and the prediction will be stored in csv file

6 Future Work

The following data would provide additional insights and improve our model's performance:

- **Commuting time** Time it takes from the house location to downtown Seattle could be a good indicator, with better connected properties potentially being valued higher.
- **Median Income per zip code** Understanding income distribution amongst zip codes would also be an indicator of which neighborhoods are more affluent and should be the focus of the campaign.
- **Longer time span** Having data beyond the one year of May 2014-May 2015 would let us examine whether there are any trends in location. For instance some neighborhoods may be experiencing a price increase due to recent infrastructure development. Which areas are up and coming?
- **School rankings** Proximity to a good school is often a key requirement for wealthy parents and likely to drive a house price up.

7 Acknowledgment

We would like to express our sincere gratitude to Orange Academy for providing us with this valuable opportunity. Additionally, we extend our heartfelt appreciation to our coaches, Ms. Aya Miqdadi and Mr. Imran Bashbsha, for their guidance and support in enhancing our skills.