

MCA

Semester – IV

Research

Project – Interim Report

|  |  |
| --- | --- |
| **Project** | House price prediction |
| **Group** | **Group 7 (DS)** |
| **Date of Submission** | 18-07-2023 |



**A study on “House Price Prediction “**

## Research Project submitted to Jain Online (Deemed-to-be University)

## In partial fulfillment of the requirements for the award of:

**Master of Computer Application**

*Submitted by:*

|  |  |
| --- | --- |
| **Name** | **USN** |
| Mohammed Asif Ali | 211VMTR00788 |
| Sonam Jain | 211VMTR01377 |
| Neha Verma | 211VMTR00887 |
| Saravanan M | 211VMTR01243 |
| Kartik Tiwari | 211VMTR00617 |

*Under the guidance of:*

Mr. Jainesh Garg

(Faculty-JAIN Online)

Jain Online (Deemed-to-be University)

Bangalore

**2022-23**

**DECLARATION**

I, *Group 7(DS)* hereby declare that the Research Project Report titled *“House Price Prediction” has been* prepared by me under the guidance of the *Jainesh Garg.* I declare that this Project work is towards the partial fulfillment of the University Regulations for the award of the degree of Master of Computer Application by Jain University, Bengaluru. I have undergone a project for a period of Eight Weeks. I further declare that this Project is based on the original study undertaken by me and has not been submitted for the award of any degree/diploma from any other University / Institution.

Place: Bangalore

Date: 18-07-2023

**Table of Contents**

|  |  |
| --- | --- |
| **Title** | **Page Nos.** |
| Executive Summary | 5 |
| List of Tables | 6 |
| List of Graphs | 6 |
| Introduction of the business problem | 7 |
| Data cleaning and data Pre processing | 8-13 |
| Exploratory Data Analysis | 10-15 |
| Insights from EDA -skeweness | 14-23 |
| Model building and interpretation | 25-30 |
| Model tuning and business implications | 30-32 |
| Appendix | 33 |

**EXECUTIVE SUMMARY**

The house price prediction model for the King County dataset is a statistical model designed to estimate the prices of houses in King County, Washington. The model utilizes various features such as the number of bedrooms, bathrooms, square footage, location, and other relevant factors to predict the selling price of a house. This paper focuses on formulating a feasible method for house price prediction. A dataset containing features and house price of King County in the US is used. The research studies the house price in King County, US, during a 2-year period from 2014 to 2015.

During the data preprocessing, extreme values are winsorized and highly correlated features are removed. This paper employs different technical models, including Linear regressor, Decision tree regressor, Catboost, LightGBM, XGBoost and regressions to identify the important influencer of the house price. The model that has low RMSE, achieves a high R-squared score and adjusted R-squared score, especially in the test set, and acquires a high score in cross validation is considered a better model. This paper finds out that Catboost performs the best among all models and can be used for house price prediction. The best model will be selected through training and testing, which will allow us to have the most accurate result.

|  |  |  |
| --- | --- | --- |
| **List of Tables** | | |
| **Table No.** | **Table Title** | **Page No.** |
| 1.1 | Raw unprocessed data | 8 |
| 1.2 | Zipcode data | 9 |
| 6.1 | Model evaluation summary | 29 |

|  |  |  |
| --- | --- | --- |
| **List of Graphs** | | |
| **Graph No.** | **Graph Title** | **Page No.** |
| 1.1 | Before outlier treatment | 11 |
| 1.2 | After outlier treatment | 12 |
| 2.1 | Distribution of price | 14 |
| 2.2 | Univariate analysis of continuous variables | 15 |
| 2.3 | Before log transformation | 16 & 17 |
| 2.4 | After log transformation | 17 & 18 |
|  |  |  |
| 2.5 | Univariate analysis of categorical variables | 18 |
| 2.6 | Bivariate analysis of continuous variables | 19 |
| 2.7 | Heatmap correlation | 20 |
| 2.8 | Coast vs price | 21 |
|  |  |  |
| 2.9 | Furnished vs price | 22 |
| 2.10 | condition vs price | 22 |
| 2.11 | City vs price | 23 |
| 6.1 | Feature importance of Cat boost model | 32 |

**Introduction and Background**

**Introduction:**

The business problem at hand is predicting house prices accurately. This problem is of significant importance in the real estate industry, as accurate price predictions enable buyers, sellers, and real estate agents to make informed decisions. By developing a reliable house price prediction model, various stakeholders can benefit from improved decision-making, such as setting appropriate listing prices, estimating property values, identifying investment opportunities, and negotiating fair deals.

**Defining problem statement:**

The problem at hand is to develop a house price prediction model that accurately estimates the selling price of residential properties. Given a set of input features describing a house, such as its location, size, number of rooms, amenities, and other relevant factors, the goal is to create a predictive model that can provide an accurate estimation of the house's selling price.

**Needs of study project:**

By addressing the house price prediction problem, we aim to provide value to both individual buyers and sellers as well as real estate professionals by empowering them with accurate price estimates and actionable insights based on data-driven predictions.

**Understanding business:**

The housing market is influenced by a multitude of factors, including location, property size, number of bedrooms and bathrooms, amenities, proximity to schools and transportation, and economic indicators such as interest rates and market trends. Analyzing and understanding these factors can be complex and time-consuming, making it essential to employ advanced techniques such as machine learning to build an effective predictive model.

The objective is to leverage historical data on house attributes and their corresponding prices to develop a robust prediction model that can estimate the price of a house accurately. By utilizing this model, prospective buyers can make informed decisions about affordability, and sellers can set competitive prices to attract potential buyers. Real estate professionals can also leverage this model to gain insights into market trends and assist clients in making well-informed investment decisions

**Data Cleaning And Preprocessing:**

**Importing libraries:** The required libraries such as pandas, numpy, matplotlib, and seaborn are imported for data analysis and visualization.

**Importing libraries for model building and model processing:**

* Importing necessary libraries that are required for building models.
* Models to be imported: LinearRegression, Ridge, DecisionTreeRegressor, XGBRegressor, LGBM Regressor and CatBoostRegressor
* Encoding techniques: LabelEncoder,MinMaxScaler,StandardScaler
* Model prerequisites: train\_test\_split,GridSearchCV
* Testing models: r2\_score,mean\_squared\_error,mean\_absolute\_error,accuracy\_score

**Reading the raw house dataset CSV:**

`raw\_data = pd.read\_csv('innerCity.csv')`: Reads the CSV file named 'innerCity.csv' and assigns it to the variable `raw\_data`. The data is loaded as a Pandas data frame.

**Printing the raw\_data:**

`raw\_data`: Prints the `raw\_data` data frame, displaying its contents.

**Raw unprocessed data**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| cid | dayhours | price | room\_bed | … | lot\_measure15 | furnished | total\_area |
| 0 | 3876100940 | 20150427T000000 | 600000 | … | 2020 | 8660 | 12490 |
| 1 | 3145600250 | 20150317T000000 | 190000 | … | 1660 | 4100 | 3771 |
| 2 | 7129303070 | 20140820T000000 | 735000 | … | 2620 | 2433 | 5455 |
| 3 | 7338220280 | 20141010T000000 | 257000 | … | 2030 | 3794 | 5461 |
| 4 | 7950300670 | 20150218T000000 | 450000 | … | 1120 | 5100 | 5710 |

**Table 1.1**

**Reading the zip\_codes CSV:**

`zip\_codes = pd.read\_csv('zip\_codes\_ml.csv')`: Reads the CSV file named 'zip\_codes\_ml.csv' and assigns it to the variable `zip\_codes`. The data is loaded as a Pandas data frame.

**Printing the zip\_codes data frame:**

`zip\_codes`: Prints the `zip\_codes` data frame, displaying its contents.

**Zip codes dataset**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| zip | decommissioned | primary\_city | acceptable\_cities |  | latitude | | | | longitude |
| 98001 | STANDARD | 0 | Auburn | | |  | … | US | 47.3 |
| 98002 | STANDARD | 0 | Auburn | | |  | … | US | 47.31 |
| 98003 | STANDARD | 0 | Federal Way | … | US | | | | 47.3 |
| 98004 | STANDARD | 0 | Bellevue | | |  | … | US | 47.61 |
| 98005 | STANDARD | 0 | Bellevue | | |  | … | US | 47.62 |

**Table 1.2**

**Removing unwanted variables:**

* Removing unwanted features cid and yr\_renovated from our dataset as they have no meaningful relationship with the price variable.

**Null values and inappropriate values treatment:**

* Removing and replacing the null values in our dataset from all the columns wherever we have null values .
* Replacing the null values with the mean or median according to the type of the feature.
* Dropping the records for some features where the values are null.
* Some of the variables includes
* room\_bed 108
* room\_bath 108
* living\_measure 17
* lot\_measure 42
* ceil 42
* coast 1
* sight 57
* condition 57
* quality 1
* ceil\_measure 1
* basement 1
* yr\_built 1
* living\_measure15 166
* lot\_measure15 29
* furnished 29
* total\_area 29

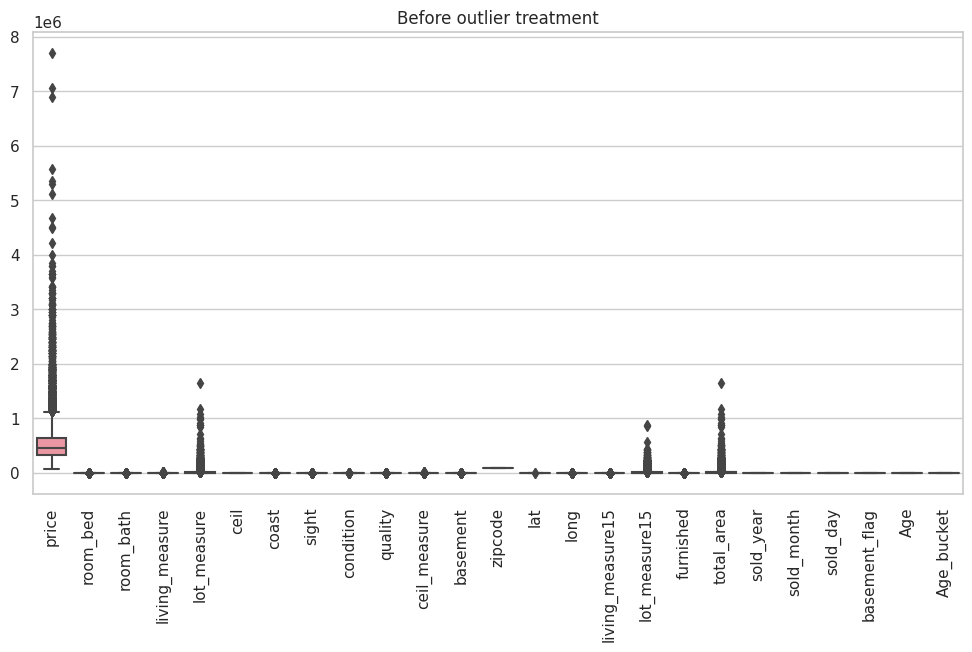
**Feature generation and transformation:**

* Transforming the feature dayhours into sold\_year, sold\_month, sold\_day and sold\_date based on the respective values of the dayhours.
* Generate a feature called Age by taking the difference between 2015 and the yr\_built column values.
* From the Age column generated create a new column Age\_bucket with the age decades and it’s weightage ranges from 1 to 5.
* Adding a column basement\_flag which shows if the basement values is more than 0 it shows 1 and if not it shows 0.
* Creating a new column city based on the zipcode values in our dataset by using the zipcode\_ml dataset that we have.

**Outlier treatment:**

* The code then creates two boxplots, one before outlier treatment and another after outlier treatment, to visualize the presence of outliers in the dataset.

**Before outlier treatment:**

****

**Graph 1.1**

* `outliers = processing\_data.quantile(0.97)`: Calculates the quantile range to exclude outliers.
* The code then filters the 'processing\_data' DataFrame to remove outliers in the 'price', 'lot\_measure', 'lot\_measure15', and 'room\_bed' columns using the calculated quantile range.
* Another boxplot is created to visualize the data after outlier treatment.

**After outlier treatment:**

A graph showing the amount of water in the water

Description automatically generated

**Graph 1.2**

* Finally, the 'processed\_data' DataFrame is assigned the filtered 'processing\_data' DataFrame.

**Label Encoding:**

* The code first creates an instance of the `LabelEncoder` class.
* It defines a list called `label\_Features` that contains the names of the features to be label encoded.
* It then loops through each feature in `label\_Features` and applies label encoding using the `fit\_transform()` method of `LabelEncoder` on the corresponding feature in the `processed\_data` DataFrame.
* The label-encoded features are stored back into the `processed\_data` DataFrame, and the encoded features are displayed.
* One-Hot Encoding:The code creates a new DataFrame `X` by dropping the 'price' column from `processed\_data`.
* It uses the `pd.get\_dummies()` function to perform one-hot encoding on the 'zipcode' and 'city' columns of the `X` DataFrame. This converts categorical variables into binary columns representing each category.
* Target Variable:The code assigns the 'price' column from `processed\_data` to the variable `y`. This represents the target variable or the variable to be predicted.

**Scaling Features:**

* The code creates an instance of the `MinMaxScaler` class, which is used for scaling features.
* Train-Test Split:The code splits the data into training and testing sets using the `train\_test\_split()` function from scikit-learn. It assigns 80% of the data to the training set (`X\_train` and `y\_train`) and 20% to the testing set (`X\_test` and `y\_test`).
* Scaling Features (continued):The code applies feature scaling to the training and testing sets using the `fit\_transform()` method of the `MinMaxScaler` instance. It scales the features of `X\_train` and `X\_test` to a specified range (typically between 0 and 1) to ensure that all features contribute equally to the model training.
* The scaled features are stored in `x\_train` and `x\_test`, respectively.
* scales the features using `MinMaxScaler`, and splits the data into training and testing sets. The resulting scaled features and corresponding target variables are ready to be used for future model tuning and analysis.

This code performs operations such as data type conversion, generating new features, handling missing values, merging DataFrames, outlier treatment, and assigns the resulting processed DataFrame to the 'processed\_data' variable.

**Exploratory data analysis:**

**Univariate Analysis:**

**1.Target variable analysis of its distribution:**

A graph of a distribution of a number of items

Description automatically generated

**Graph 2.1**

sns.histplot(data=processed\_data, x='price', kde=True): Plots a histogram of the 'price' variable to visualize its distribution. The `kde=True` parameter adds a kernel density estimate line to the plot. The analysis suggests that the 'price' variable is right-skewed.

**2. Univariate analysis of continuous variables:**

A group of blue and white graphs

Description automatically generated

**Graph 2.2**

Several histogram plots are created for different continuous variables using `sns.histplot(). The variables plotted include 'living\_measure', 'lot\_measure', 'sight', 'quality', 'ceil\_measure', 'basement', 'living\_measure15', 'lot\_measure15', and 'Age'. These plots visualize the distribution of each variable before any transformation.

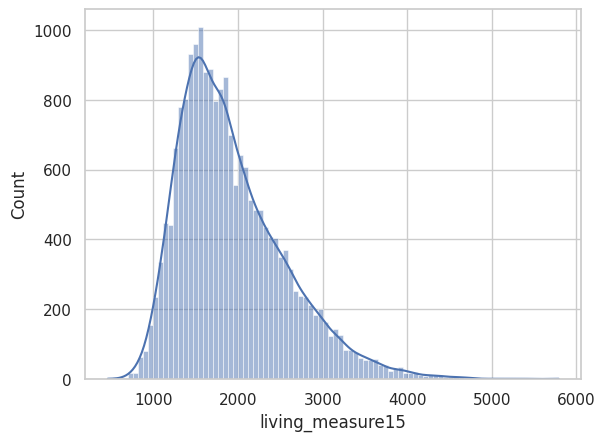
**3. Log transformation of variables:**

The code performs a log transformation on the'living\_measure', 'ceil\_measure', and 'living\_measure15' variables using `np.log1p()`. This transformation is applied to reduce the skewness and make the variables more normally distributed.

Before log transformation:

A graph of a living scale

Description automatically generated

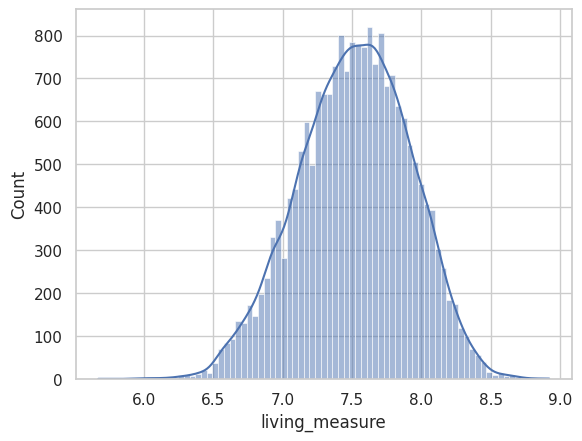


A graph of a graph

Description automatically generated

**Graph 2.3**

**After the log transformation**, the histograms for these variables are plotted again using `sns.histplot()` to show the improved distribution.

 A graph of a normal distribution

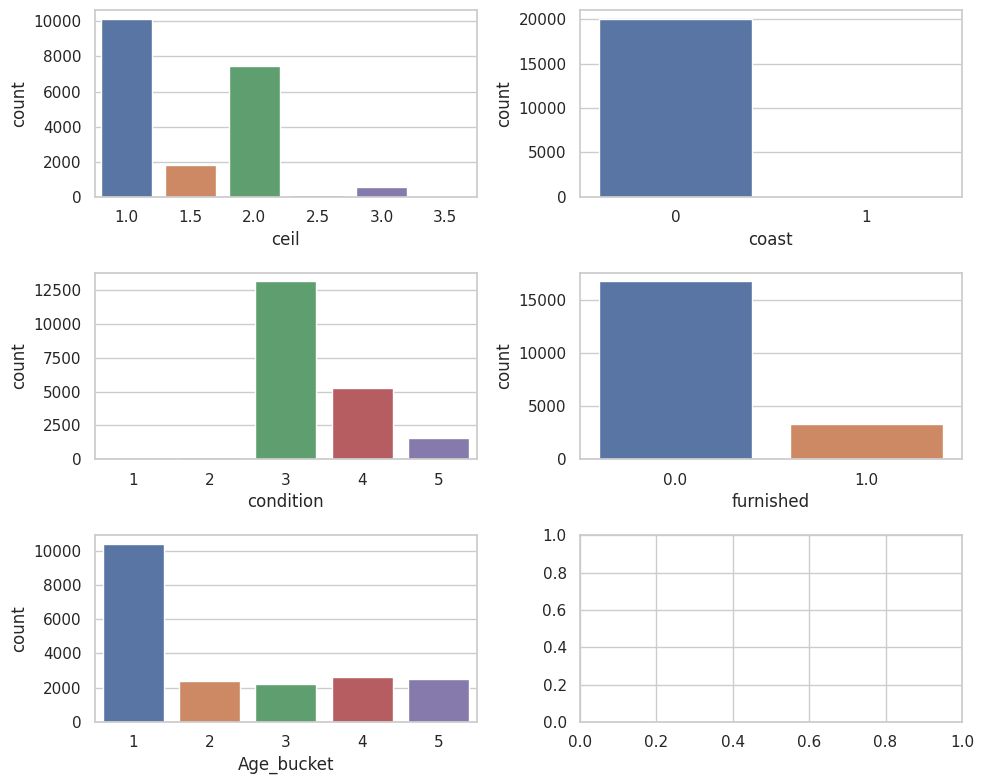
Description automatically generated

A graph of a normal distribution

Description automatically generated

**Graph 2.4**

4. **Univariate analysis of categorical variables:**



**Graph 2.5**

Similar to the continuous variables, count plots are created for categorical variables using `sns.countplot()`. The variables plotted include 'ceil', 'coast', 'condition', 'furnished', and 'Age\_bucket'. These plots display the frequency distribution of each category.

The code provides insights into the distributions of variables in the dataset. It reveals the skewness of continuous variables and performs log transformations to improve their distribution. Additionally, it displays the frequencydistribution of categories in categorical variables, offering an overview of the dataset's composition.

**Bivariate analysis:**

1.**Bivariate analysis of numerical features:**

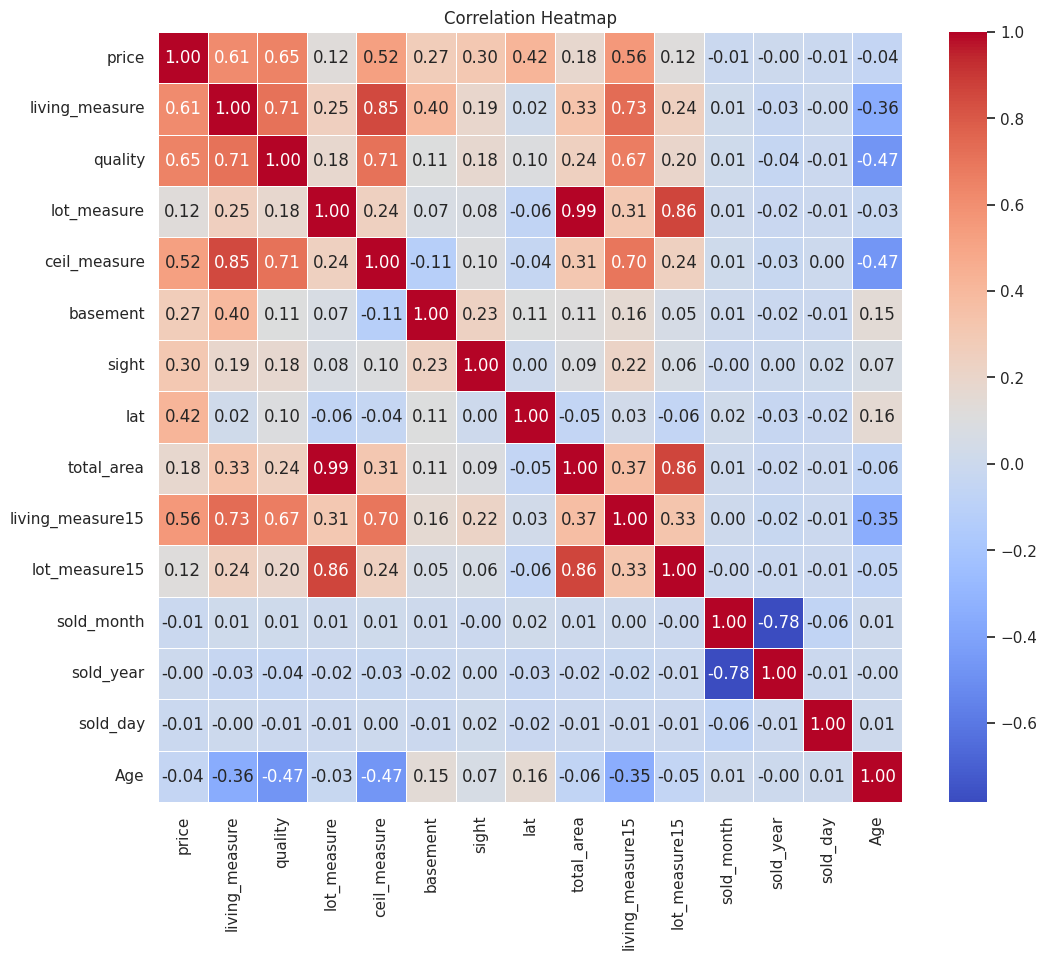
A group of blue graphs

Description automatically generated

**Graph 2.6**

Scatter plots are created for each numerical variable (e.g., 'living\_measure', 'lot\_measure', 'sight', etc.) against the 'price' variable using `sns.regplot()`. These plots visualize the relationship between each numerical variable and the target variable 'price'. They help identify any linear relationships or trends between the variables.

2**. Heatmap of correlation:**



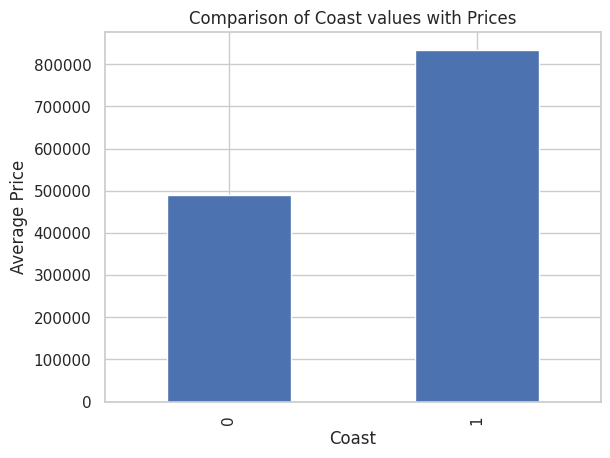
**Graph 2.7**

A heatmap is created using `sns.heatmap()` to visualize the correlation matrix of selected variables. The variables included are 'price', 'living\_measure', 'quality', 'lot\_measure', 'ceil\_measure', 'basement', 'sight', 'lat', 'total\_area', 'living\_measure15', 'lot\_measure15', 'sold\_month', 'sold\_year', 'sold\_day', and 'Age'. The heatmap provides an overview of the pairwise correlations between these variables, with higher values indicating stronger correlations.

**3. Bivariate analysis of categorical variables:**

Various plots are created to explore the relationship between categorical variables and the 'price' variable:

sns.barplot(): Compares the **'coast'** variable (coastal region or not) with the average **'price'**. It displays the average price for each category.



**Graph 2.8**

sns.barplot(): Compares the **'furnished'** variable (furnished or not) with the average **'price'**. It shows the average price for each category.

A graph with blue squares

Description automatically generated

**Graph 2.9**

sns.barplot(): Compares the **'condition'** variable (condition of the house) with the average **'price'**. It displays the average price for each condition category.

A graph with numbers and a bar chart

Description automatically generated

**Graph 2.10**

plt.scatter(): Compares the **'city'** variable with the average **'price'**. It plots the average price for each city.

A graph with blue dots

Description automatically generated

**Graph 2.11**

The code provides insights into the relationships between variables in the dataset. It helps identify any significant associations between numerical variables and the target variable 'price'. Additionally, it explores how categorical variables relate to the 'price' variable, providing information on the average prices across different categories.

**Business insights from EDA**

* 1. **Data skewness and the measures in the context of business**
* Based on the Exploratory Data Analysis (EDA) that has been done we can clearly the dataset that has price as a target variable is right skewed and has a long tail pointing to the right side as per Graph 1.
* This means that most houses have prices that are relatively lower or closer to the median, but there are a few houses with exceptionally high prices, creating a long tail on the right side of the distribution.

In the business context, right-skewed data can present challenges and opportunities:

**Challenges:**

* **Model Performance:** Many traditional machine learning models assume that the target variable (here ‘price’) follows a normal distribution. Right-skewed data might violate this assumption, leading to biased predictions and reduced model accuracy.
* **Risk of Overestimation:** The presence of a few very high-priced houses can potentially lead to overestimation of house prices, making your model less reliable for average properties.

**Opportunities:**

* **Data Insights:** Analyzing the right-skewed distribution can provide valuable insights into the luxury segment of the real estate market. Understanding the factors that contribute to these high-priced properties can help you better serve high-end clients and identify potential opportunities for business growth.
* **Feature Engineering:** Identifying the factors that influence the high prices can lead to feature engineering opportunities. You might discover specific amenities, locations, or property characteristics that significantly impact house prices, enabling you to create more informative features for the model.

**1.2 Other business insights:**

* We can notice that some features that are driving and leads to high correlation with the target variable like living\_measure, quality, location, ceil\_measure, living\_measure15, etc as mentioned in the Graph 1.
* We can predict in the future that if any or all of the features has high values in our data which in turn helps us to derive that the house price must be high, and contrarily if these have low values the house price decreases accordingly.

**Model building and interpretation**

**A machine learning model** is defined as a mathematical representation of the output of the training process. The learning algorithm discovers patterns within the training data, and it outputs an ML model which captures these patterns and makes predictions on new data.

* 1. **Building various models**
  2. **Testing models**
  3. **Interpretation of models**
  4. **Building various models:**

Let’s dive deeper into the models that have been employed and implemented in this project.

**Multiple Linear Regression model:**

A close-up of a sign

Description automatically generated

* **Multiple linear regression** refers to a statistical technique that is used to predict the outcome of a variable based on the value of two or more features.

**Decision tree regression model:**

****

* The [**decision**](https://scikit-learn.org/stable/modules/tree.html#tree) **tree regression model** is used to fit a sine curve with addition noisy observation. As a result, it learns local linear regressions approximating the sine curve.
* We can see that if the maximum depth of the tree (controlled by the max\_depth = 4 parameter) is set too high, the decision trees learn too fine details of the training data and learn from the noise, i.e. they overfit.

**RIDGE REGRESSION:**

****

* **Ridge regression** is a model tuning method that is used to analyse any data that suffers from multicollinearity.
* This method performs L2 regularization.
* When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values being far away from the actual values.

**X Gradient Boosting model:**

****

* **Extreme Gradient boosting** refers to a class of ensemble machine learning algorithms that can be used for classification or regression predictive modeling problems.
* Ensembles are constructed from decision tree models. Trees are added one at a time to the ensemble and fit to correct the prediction errors made by prior models. This is a type of ensemble machine learning model referred to as boosting.

**Light GBM model:**

****

* **LightGBM** is a gradient-boosting framework based on decision trees to increase the efficiency of the model and reduces memory usage.
* It uses two novel techniques:
  + Gradient-based One Side Sampling(GOSS)
  + Exclusive Feature Bundling (EFB)

**Cat boost regression model:**

* **CatBoost** builds upon the theory of decision trees and gradient boosting.
* The main idea of boosting is to sequentially combine many weak models (a model performing slightly better than random chance) and thus through greedy search create a strong competitive predictive model.
* Because gradient boosting fits the decision trees sequentially, the fitted trees will learn from the mistakes of former trees and hence reduce the errors.
* This process of adding a new function to existing ones is continued until the selected loss function is no longer minimized.
  1. **Testing models:**

**Linear regression:**

Linear Regression R squared test score: **0.84**

Linear Regression Adjusted R squared score: **0.84**

Linear Regression RMSE: **95068.17250172315**

Cross Validation Scores (Linear Regression):

[**0.83701857 0.83262015 0.83793048 0.82813596 0.83705172**]

Average CV Score: **0.83**

**Decision Tree regression:**

Decision Tree Regression R squared test score: **0.68**

Decision Tree Regression Adjusted R squared test score: **0.67**

Decision Tree Regression RMSE: **133956.71166403254**

Cross Validation Scores (Decision Tree):

[**0.69309842 0.67977301 0.65877833 0.65607487 0.67744377**]

Average CV Score: **0.67**

**Ridge Regression:**

Ridge Regression R squared test score: **0.84**

Ridge Regression Adjusted R squared test score**: 0.83**

Ridge Regression RMSE: **95264.1423229366**

Cross Validation Scores (Ridge Regression):

[**0.8366125 0.83216099 0.83712537 0.83088806 0.83651926**]

Average CV Score: **0.83**

**XGB Regressor**

XGB Regression R squared test score: **0.88**

XGB Regression Adjusted R squared test score: **0.88**

XGB Regression RMSE: **81671.63245186348**

Cross Validation Scores (XGB Regressor):

[**0.88542147 0.87353581 0.87304241 0.86784802 0.87703363**]

Average CV Score: **0.88**

**Light GBM Regressor**

LGBM Regression R squared test score**: 0.89**

LGBM Regression Adjusted R squared test score: **0.88**

LGBM Regression RMSE: **79713.01163482816**

Cross Validation Scores (LGBM Regressor):

[**0.89190452 0.88174406 0.88295565 0.87603302 0.88330129**]

Average CV Score: **0.88**

**Cat Boost Regressor**

Cat Boost Regression R2 test score: **0.90**

Cat Boost Regression R2 test score: **0.90**

Cat Boost RMSE: **75374.57417365463**

Cross Validation Scores (Cat Boost Regressor):

[**0.90283051 0.88989255 0.89476846 0.88815738 0.89111944**]

Average CV Score: **0.89**

* + 1. **Interpretation of models:**

**Model evaluation summary:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Models** | **Train\_R\_Squared** | **Test\_R\_Squared** | **Train\_RMSE** | **Test\_RMSE** | **Train\_MAE** | **Test\_MAE** |
| Cat Boost | 0.954023 | 0.900144 | 50433.87 | 75374.55 | 36876.92 | 50752.74 |
| Light GBM | 0.923079 | 0.889184 | 65233.66 | 79403.64 | 45974.92 | 53921.83 |
| XGradientBooting | 0.954823 | 0.882641 | 49993.09 | 81714.21 | 35815.3 | 55227.1 |
| Linear | 0.838026 | 0.841149 | 94661.41 | 95067.94 | 67412.94 | 68231.57 |
| Ridge | 0.83795 | 0.84095 | 94683.66 | 95127.31 | 67410.21 | 68256.11 |
| Decision Tree | 0.681525 | 0.684607 | 132735.8 | 133956.7 | 94838.54 | 95966.93 |

**Table 6.1**

* After determining the candidate models as well as the evaluation indicators, we processed the data with Python.
* During the evaluation process, Adjusted 𝑅 2 and RMSE(Root Mean Squared Error) and Mean Absolute Error(MAE) are calculated not only for the training set but also for the test set, helping us to see clearly how the models are performing respectively in the two sets.
* K-fold cross validation is conducted on the whole dataset to holistically assess how well the model is performing.

**Model Tuning and business implication**

**Ensemble modelling:**

* Ensemble modeling is a machine learning technique that combines multiple individual models (learners) to create a more powerful and robust predictive model.
* The idea behind ensemble modeling is to leverage the diversity of multiple models to improve overall performance, generalization, and accuracy compared to using a single model.
* Ensemble modeling is based on the concept that different models may excel in different areas, and by combining their strengths, the resulting ensemble can outperform any of the individual models.
* In the project we’ve combined some models to build a stacking model which can predict things.
* Created the stacking ensemble using Catboost, Light gbm, linear regression as base model and Ridge regression as meta model.
* After performing ensemble modelling we’ve got the following observations

R squared score of stacked model: **0.90**

Adjusted R squared score of stacked model: **0.90**

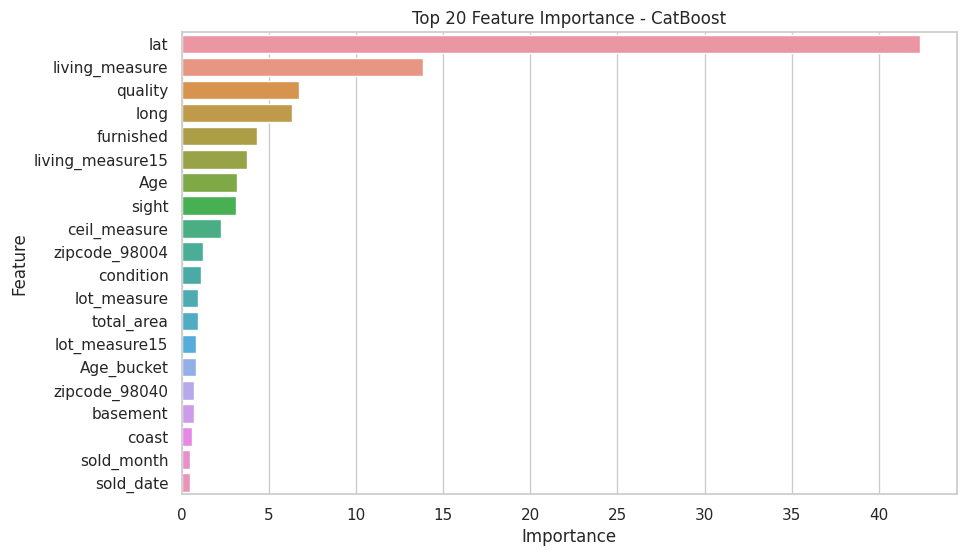
RMSE of stacked model: **75244.8741561014**

* Which is similar as of cat boost model’s scoring rate

**Interpretation of the most optimum model**

* From Table 1, it is not difficult to conclude that **Catboost Regressor** performs the best among all models.
* It has an RMSE of 75374.55 and becomes the only model that has an RMSE of less than 100,000.
* When it comes to the R 2 score, adjusted R 2 score, as well as the 5-Fold Cross Validation score, Catboost stands out from the candidate models as well.
* Catboost demonstrates a great capability of precise prediction and does not show any tendency of overfitting, therefore, there is no doubt that Catboost is selected as the final model used to predict house prices.

**Feature importance of CB model:**

****

**Graph 6.1**

* Here we can visualise the top 20 features that affects the Cat boost model.
* We can see that latitude, living\_measure, quality, longitude, living\_measure15, etc features plays important and effective role in predicting the house price for the cat boost model.

**Appendix**

**Attachments:**

**House\_price\_prediction.ipynb (python code file)**

**link ->** [https://colab.research.google.com/drive/1wLHffbyFe0K\_54P5OYpdICHZ4aMh8IsM#scrollTo=k8EgVxr7yjTz](https://colab.research.google.com/drive/1wLHffbyFe0K_54P5OYpdICHZ4aMh8IsM%23scrollTo=k8EgVxr7yjTz)

**innerCity.csv (housing data set) download link ->** <https://drive.google.com/file/d/1d3Az8SywHcMAg--nmynVgUwuTP1wVKAi/view?usp=drive_link>

**zip\_codes\_ml.csv (zip codes data set) download link ->**

<https://drive.google.com/file/d/1-cTlP3u_iiYK945kALQw7uBLr28HXJc5/view?usp=drive_link>