HOUSE PRICE PREDICTION USING MACHINE LEARNING ALGORITHMS

DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE OF

Master

Of

Computer Science & Applications

Under the Supervision of

Prof. Tamanna Siddiqui

Submitted by

Mohd Anas Shariq

(21CAMSA113, GH3614)

DEPARTMENT OF COMPUTER SCIENCE

ALIGARH MUSLIM UNIVERSITY

ALIGARH (INDIA)

2023

CERTIFICATE

ACKNOWLEDGEMENT

It gives me great pleasure and deep satisfaction in presenting this report of my project work as a part of MCA IV semester, arranged in order to gain practical knowledge in the Machine learning based Development.

I would like to express my immense gratitude to all those who helped and guided me throughout the project. I am highly grateful to my project supervisor **Prof. Tamanna Siddiqui** maam for all the help and encouragement extended by her at every stage during the project. Her inspiring suggestions and timely guidance enabled me to gain a different perspective on various aspects of the project.

I would also like to thank all the staff of the department for their timely support. My gratitude also extends to my friends and seniors who provided me with the moral support and guided me towards the completion of this wonderful project.

Mohd Anas Shariq

ABSTRACT

House Price Prediction Using Machine Learning Algorithms

In this project, I explore how predictive modeling can be applied in housing sale price prediction by analyzing the housing dataset and use machine learning models. Actually, I try three different models, namely, linear regression, lasso regression, random forest and . Additionally, as the data have 79 explanatory variables with many missing values, I spend much time dealing with the data. I do explorary data analysis, feature enginnering before model fitting. And then using rmse and R-squared to measure the model performance. After I try three different models, I get some results. As for the first model - linear regression, it doesn't meet the assumption of equality of the variances. Therefore, we can't use the linear model as the candidate of our final model. Then I try lasso regression, but the RMSE and R-squared looks not so good. Then I try Random forest. The R squared in this model of training set is very good. So, Finally I try the random forest. All of the results of this random forest model seem very good. Therefore, I will use this random forest model as my final model to predict the housing price. The random forest model also shows which variables have important effects on sale price.

Table of Contents

Certification……..…. **(i)**
Acknowledgement……..... **(ii)**
Abstract……..……(iii)

CHAPTER 1**: INTRODUCTION**

1: Background of problem ……..……..…….. .2
2: About ……..……...…..4
3 Objective ……....……..... .5
4: Overview ……..……..…….…7
5: Major Contribution……..…….……8

CHAPTER 2**: RELATED WORKS**

1: Related Work in past……. ……..……...10
2: Research Gaps…….. ……...…...11
3: Scientific Novelty and Need of the work……...……..….....12

CHAPTER 3**:CONTRIBUTIONS**

1: Techniques Used ……..….....…….....…….15
2: Steps of Model .....…...…….…….…….....25
3: Preparing data for modeling …….…….…….... 27
4: Splitting the data …….…….…….... 29
5: Model Evaluation and Performance Metrics.……...…….... 31
6: Proposed Model/Framework.....……...…….…….…33
7: Implementation.....……...……..…..……...…..…….....35

CHAPTER 4**: RESULTS AND COMPARISON**

CHAPTER 5**:CONCLUSION**

CHAPTER 6**:FUTURE EXTENSION**

References.....60

Appendix UML Diagram , code , Screenshots……..…….....62

CHAPTER 1

INTRODUCTION

1: Background of problem:

The prediction of house prices using machine learning algorithms has become a popular and extensively researched area in the field of real estate. It addresses the need for accurate and objective estimates of property values, which play a crucial role in various real estate activities such as buying, selling, investing, and mortgage lending.

The background of this problem stems from the inherent complexity of housing markets. House prices are influenced by a multitude of factors, including location, size, amenities, economic conditions, and market trends. This complexity makes it challenging to determine the precise value of a property based solely on subjective assessments or traditional methods.

In the past, real estate professionals heavily relied on their experience and market knowledge to estimate house prices. However, these approaches are often limited by human bias, inconsistencies, and the inability to consider a large number of variables simultaneously. Moreover, the real estate market is dynamic, and property values can fluctuate rapidly, making it difficult to keep up with the latest trends using traditional methods alone.

Machine learning algorithms offer a data-driven and objective approach to address these challenges. These algorithms leverage the power of computational analysis and statistical modeling to learn patterns and relationships from large amounts of historical data. By training on a diverse set of features such as the number of bedrooms, bathrooms, square footage, location, amenities, and previous sale prices, machine learning models can identify complex patterns that are not readily apparent to human observers.

One of the key advantages of machine learning algorithms is their ability to handle non-linear relationships and interactions between variables. Traditional methods like linear regression may struggle to capture these complexities, but algorithms such as decision trees, random forests, support vector machines, and neural networks excel in modeling non-linearities. These algorithms can analyze the historical data, identify the most important features for predicting house prices, and create accurate models that generalize well to unseen data.

To build an effective house price prediction model, a large and diverse dataset is crucial. Historical data that includes a wide range of properties with detailed information is necessary to train the model effectively. The dataset should ideally include features such as the size of the property, the number of rooms, the location (including factors like proximity to schools, parks, shopping centers, and transportation hubs), crime rates, income levels, and other relevant socio-economic indicators. By incorporating such information, the model can capture the influence of both internal and external factors on house prices.

Once the model is trained, it can be used to make predictions on new or unseen data. For example, given the characteristics of a house, the model can estimate its value with reasonable accuracy. This information is valuable for various stakeholders in the real estate market. Homebuyers can use the predicted prices to assess whether a property is priced appropriately, negotiate effectively, or make informed decisions about their investment. Sellers can benefit by pricing their properties competitively, ensuring a quicker sale. Real estate professionals can leverage these models to provide more accurate and objective property valuations and offer valuable insights to their clients.

It is worth noting that while machine learning algorithms can provide accurate predictions, they are not infallible. The accuracy of the predictions heavily relies on the quality and representativeness of the training data. Issues like data quality, missing values, outliers, and the presence of bias can impact the model's performance. Therefore, it is important to carefully preprocess and validate the data, perform feature engineering to extract relevant information, and continuously monitor and update the model to account for changing market dynamics.

In conclusion, the use of machine learning algorithms for house price prediction addresses the need for accurate, objective, and data-driven estimates in the real estate market. By leveraging historical data and advanced statistical techniques, these algorithms can provide valuable insights to buyers, sellers, and real estate professionals.

2 : About

Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence [1]. However, this dataset related to the thesis proves that may have more effects on the housing price than the number of bedrooms or floors. Also, I want to predict the reasonable housing price with these aspects of the houses by using this dataset.

This dataset contains 79 explanatory variables which related to almost every aspect .In the following steps, I will explore this dataset, do feature engineering, fit some machine learning models to predict the housing prices and find which aspects of the house influence the housing prices mostly.

Machine learning is closely related to computational statistics, which focus on using mathematical optimization to deliver methods, theory and application domains to solve medical,industry, social and business problems in the real world.

In my thesis, I will try three models: Linear Regression, Lasso Regression, Random Forest to predict the housing sale price. And finally, I will use the random forest model as my final model. This model also gives us which aspects have big effects on housing sale price.

3: Objective:

The project of house price prediction using machine learning algorithms addresses a significant need in the real estate industry and for potential homebuyers. The following points highlight the need for such a project:

1. Informed Decision Making: Buying or selling a house is a major financial decision for individuals and families. Accurate predictions of house prices can provide valuable insights to both buyers and sellers, enabling them to make informed decisions based on market trends and expected returns.

2. Market Transparency: The real estate market can be complex and opaque, making it challenging for individuals to gauge the fair value of a property. A machine learning-based house price prediction system can contribute to market transparency by providing objective estimates based on historical data, current market conditions, and relevant features.

3. Efficient Pricing Strategies: For sellers, accurately pricing a house is crucial to attract potential buyers and maximize returns. By leveraging machine learning algorithms, sellers can gain insights into the factors influencing house prices, allowing them to set competitive and realistic prices for their properties.

4. Risk Mitigation: Homebuyers and real estate investors face risks associated with overpaying for a property or experiencing unexpected price fluctuations. Predictive models can assist in mitigating these risks by estimating house prices and identifying potential undervalued or overpriced properties, aiding buyers in making prudent investment choices.

5. Improved Financial Planning: House price predictions can facilitate better financial planning for individuals and families. Knowing the expected value of a property allows homeowners to make informed decisions regarding mortgage options, property taxes, insurance, and long-term financial goals.

6. Market Analysis and Research: Real estate agents, developers, and analysts require accurate price forecasts to assess market trends, analyze investment opportunities, and make informed recommendations to clients. Machine learning algorithms can provide valuable tools for market analysis, allowing professionals to identify emerging patterns, understand customer preferences, and optimize business strategies.

7. Time and Cost Savings: Traditionally, determining the value of a property involves manual appraisal processes, market research, and expert opinions, which can be time-consuming and costly. By automating the prediction process using machine learning algorithms, the project aims to save time and resources for both buyers and sellers.

Overall, the project on house price prediction using machine learning algorithms serves a vital need in the real estate industry by providing accurate, data-driven estimates of house prices. It empowers stakeholders with valuable insights, enhances market transparency, facilitates informed decision making, and contributes to efficient pricing strategies, ultimately benefiting individuals, businesses, and the overall housing market.

4: Overview

The project of house price prediction using machine learning algorithms aims to develop a system that can accurately estimate the prices of residential properties. By analyzing historical data and incorporating various features such as location, size, amenities, and market trends, this project seeks to provide valuable insights for buyers, sellers, and real estate professionals. By leveraging advanced machine learning techniques, the project aims to facilitate informed decision making, efficient pricing strategies, and market analysis in the real estate industry. Ultimately, this project has the potential to enhance transparency, save time and resources, and empower stakeholders with accurate and data-driven house price predictions.

5: Major Contribution

The application of machine learning algorithms in house price prediction has made significant contributions to the real estate industry. One major contribution is the ability to handle large and complex datasets. Machine learning algorithms can efficiently process vast amounts of historical housing data, including various features such as location, size, amenities, and economic indicators. This allows for a comprehensive analysis of the factors influencing house prices and enables more accurate predictions.

Another significant contribution is the ability to capture non-linear relationships and interactions between variables. Traditional methods often assume linear relationships, which may not accurately represent the complexities of the housing market. Machine learning algorithms, such as decision trees, random forests, and neural networks, can model non-linearities, allowing for more accurate predictions by capturing intricate relationships between features.

Additionally, machine learning algorithms provide automated and objective predictions, reducing human bias and subjectivity. Real estate professionals traditionally relied on their expertise and market knowledge to estimate house prices, which can introduce inconsistencies and errors. Machine learning models, on the other hand, learn patterns and relationships directly from the data, leading to more consistent and unbiased predictions.

The integration of machine learning algorithms in house price prediction has also enabled the development of sophisticated tools and platforms for buyers, sellers, and real estate professionals. These tools provide users with instant access to estimated property values, market trends, and insights, empowering them to make informed decisions.

Overall, the major contribution of machine learning algorithms in house price prediction lies in their ability to handle complex datasets, capture non-linear relationships, reduce bias, and provide objective predictions. These advancements have greatly enhanced the accuracy and efficiency of estimating house prices, benefiting both individuals and professionals in the real estate industry.

CHAPTER 2

RELATED WORK

1. Related Works in past :

Over the last two decades there have been a large number of empirical studies analysing land prices. Kilpatrick showed the usefulness of time-series regression model which used economic data to provide forecast of Central Business District (CBD) land price in moving market. Wilson et al studied the residential property market accounts for a substantial proportion of UK economic activity. Valuers estimate property values based on current bid prices. In this project, the national housing transaction data was trained using Random forest , which forecasts future trend of the housing market. Mark and John developed a regression model with vacant land sales. The model explained up to 93% of the market values. Wang and Tian used the wavelet Neural Network (NN) to forecast the real estate price index. This kind of wavelet NN integrated the merit of the wavelet analysis and the tradition NN. It also compared the forecasting result with smoothing method and the NN forecast.

Zhangming forecasted the real estate price index by using the Back Propagation (BP) NN. The BPN used the sigmoid function. Tinghao used the Auto Regressive Integrated Moving Average (ARIMA) model and carried the demonstrative analysis on year data from 1998 to 2006. He used the established model to make the forecast to the real estate price index of 2007. A hedonic regression on the price of land suggested that de facto policy differences between political jurisdictions have had a significant effect on land prices between1970 and 1980. Steven and Albert used 46,467 residential properties spanning 1999 - 2005 and demonstrated that using matched pairs that relative to linear hedonic pricing models, ANN generate lower dollar pricing errors, had greater pricing precision out-of-sample, and extrapolate better from more volatile pricing environments. ANN is better suited to hedonic models that utilize large numbers of variables. Sampath kumar and Santhi studied the land price trend of Sowcarpet which is the central part. They developed statistical model using economic factors and predicted that the annual rise in land price would be of 17%. Urmila reported that the past trends were analysed to ascertain the rate of growth or decline and the trends are used in forecasting. Economic parameters might be introduced to formulate more realistic relationship. Some of the other techniques they Mansural Bhuiyan and Mohammad Al Hasan 2016 use is regression, deep learning to learn the nature of models from the previous results (the property/land which were sold off previously which are used as training data).There are different models used such as

linear model data using only one feature, multivariate model, using several features as its input and polynomial model using the input as cubed or squared and hence calculated the root mean squared error (RMS value) for the model.

2.Reasearch Gaps:

I found several key research gaps that present opportunities for further investigation. Firstly, while many studies focus on the application of traditional machine learning algorithms like regression and decision trees, there is a need for more exploration into the effectiveness of advanced algorithms such as ensemble methods (e.g., random forest, gradient boosting) or deep learning techniques (e.g., neural networks) in predicting house prices.

Additionally, research gaps exist in terms of the incorporation of alternative data sources. While most studies rely on traditional variables like location and property characteristics, there is potential for integrating non-traditional data such as social media sentiment, neighborhood crime rates, or environmental factors to enhance prediction accuracy.

Furthermore, there is a lack of comprehensive comparative analyses across different regions or housing markets. Housing dynamics can vary significantly between geographical areas, and investigating the applicability and performance of machine learning models across diverse contexts can provide valuable insights and establish best practices.

Moreover, the interpretability and explainability of machine learning models in the context of house price prediction is an important research gap. Understanding the factors driving predictions and establishing transparent and interpretable models can enhance user trust and facilitate decision-making processes for various stakeholders.

Lastly, there is a need for more longitudinal studies that explore the temporal dynamics of house prices. Investigating how machine learning models can adapt to changing market conditions and incorporating time series analysis techniques could provide a deeper understanding of housing market trends and improve prediction accuracy.

Addressing these research gaps will contribute to the advancement of the field of house price prediction using machine learning algorithms, providing more accurate and robust models, expanding the scope of variables considered, and fostering a deeper understanding of housing market dynamics.

3.Scientific Novelty and Need of the work:

The literature review on house price prediction using machine learning algorithms highlights the scientific novelty and the compelling need for further work in this field.

Scientific novelty arises from the application of machine learning algorithms to the task of house price prediction. Traditional methods often rely on simplistic models or limited sets of variables, whereas machine learning techniques offer the potential to harness the power of vast amounts of data and capture complex interactions. This innovative approach brings a new dimension to the field, enabling more accurate and reliable predictions, thereby improving decision-making processes for various stakeholders in the real estate market.

The need for further work in this area is driven by several factors. Firstly, the dynamic nature of the housing market necessitates continuous research and improvement in prediction models. As market conditions change and new data becomes available, the existing models need to be refined and adapted to maintain their effectiveness. This ongoing research ensures that practitioners can make informed decisions based on the most up-to-date information.

Additionally, there is a growing demand for transparency and interpretability in machine learning models. While these models offer enhanced predictive capabilities, understanding the underlying factors driving the predictions is crucial for user trust and acceptance. Further work is needed to develop models that not only provide accurate predictions but also offer insights into the factors influencing house prices, contributing to a deeper understanding of the real estate market.

Furthermore, the need for work in this field is driven by the potential for incorporating diverse data sources and expanding the scope of variables considered. By incorporating alternative data

such as social media sentiment, neighborhood characteristics, or environmental factors, the prediction models can capture a more comprehensive view of the housing market dynamics. This not only enhances the accuracy of predictions but also provides a more holistic understanding of the complex factors influencing house prices.

In conclusion, the scientific novelty of applying machine learning algorithms to house price prediction and the compelling need for further work in this field arise from the potential for improved accuracy, the dynamic nature of the housing market, the demand for transparency, and the opportunity to integrate diverse data sources. Continued research and advancements in this area will contribute to more accurate predictions, deeper insights, and better decision-making processes in the real estate industry.

CHAPTER 3

CONTRIBUTIONS/ PROPOSED MODELS

1. Techniques Used:

There are several proposed models for house price prediction using machine learning algorithms. Some commonly used models in this domain include:

Linear Regression

In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression. For more thanone explanatory variable, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted,rather than a single scalar variable.

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, theconditional median or some other quantile is used. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values ofthe predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis.

Linear regression was the first type of regression analysis to be studied rigorously, and to be usedextensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their

parameters and because the statistical properties of the resulting estimators are easier todetermine. [6]

Given a dataset $\{y_i, x_{i1} \ldots x_{ip}\}^n$ of n statistical units, a linear regression model assumes that the relationship between the dependent variable y and the p-vector of regressors x is linear. Thisrelationship is modeled through a disturbance term or error variable ϵ, which is an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors. Thus the model takes the form:

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip_i} = X^T\beta + \epsilon$$

Often these n equations are stacked together and written in matrix notation as

$$Y = X\beta + \epsilon$$

All in all, I started with linear regression model which is the simplest in my cases. I will give theRMSE and R Squared of both the training set and the test set in the following table.

Table 7

OLS	TRAINING SET	TEST SET
RMSE	0.0883	0.1262
R^2	0.9512	0.9312

Remember I have taken log transformation to the outcome variable – Sale Price. Therefore, theRMSE value is very low. But it doesn't matter, we can still compare these values.

The result of the linear regression model seems good. However, I need to check the assumptionof the linear regression. Figure 17 will help me do the check.

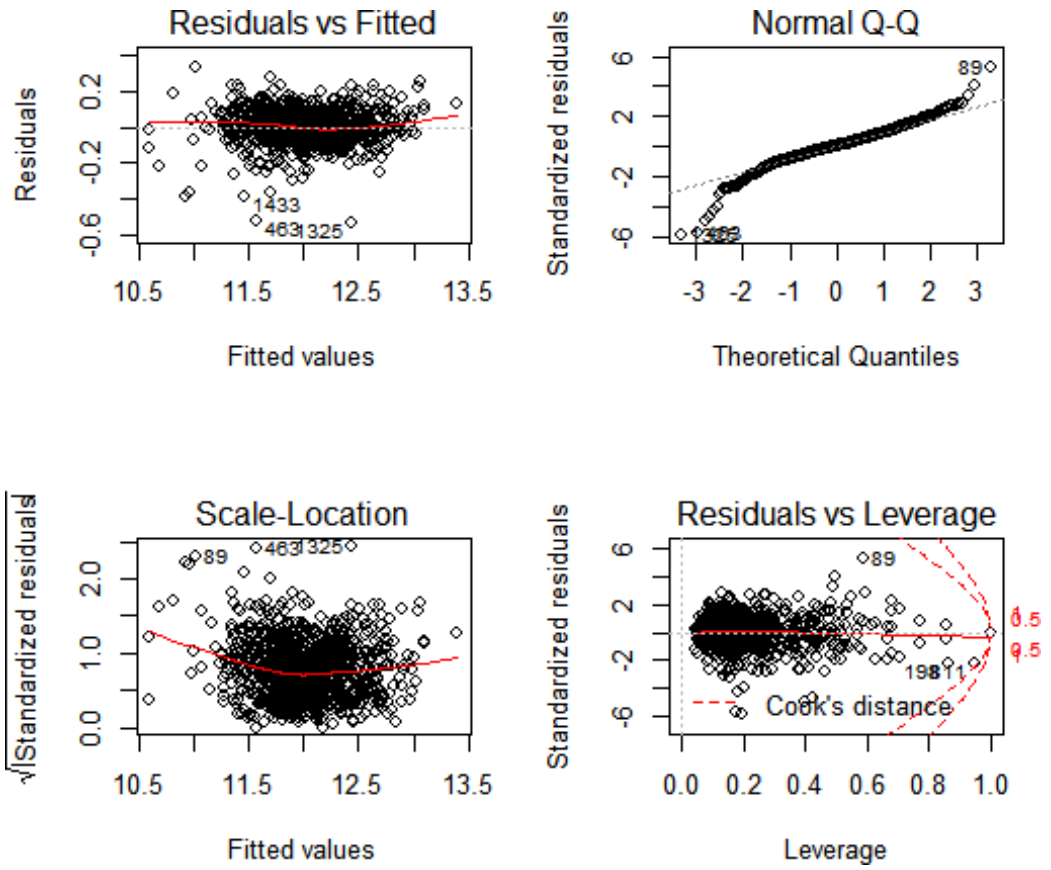


Figure 17

Plot 1 Residuals vs Fitted: Relatively straight line but with patterns. Linearity assumption can beaccepted.

Plot 2 QQ plot for normality: Relatively normal. Normality assumption can be accepted. Plot 3 Scale-Location for assumption of equal variance: Relatively flat line, however it haspatterns. We will need to check further using ncv test.

Plot 4 Residuals vs Leverage: No influential case as we barely can see Cook's distance line.

The NCV test shows the p-value << 0.05, which tells us our final linear model doesn't meet theassumption of equality of the variances. Therefore, we can't use the linear model as the candidate of our final model.

Random Forest

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction andthe class with the most votes becomes our model's prediction.

The fundamental concept behind random forest is a simple but powerful one — the wisdom ofcrowds. In data science speak, the reason that the random forest model works so well is:

A large number of relatively uncorrelated models (trees) operating as a committee willoutperform any of the individual constituent models. [7]

The training algorithm for random forests applies the general technique of bootstrap aggregating,or bagging, to tree learners. Given a training set $X = x_1, \ldots, x_n$ with response $Y = y_1, \ldots, y_n$ bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1,…,B:

1. Sample, with replacement, n training examples from X, Y; called these X_b, Y_b.
2. Train a classification or regression tree f_b on X_b, Y_b.

After training, predictions for unseen samples x' can be made by averaging the predictions fromall the individual regression trees on x':

$$\hat{f} = 1 \sum_{b=1}^{B} f_b(x')$$

or by taking the majority vote in the case of classification trees.

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de correlating the trees by showing them different training sets.

Additionally, an estimate of the uncertainty of the prediction can be made as the standarddeviation of the predictions from all the individual regression trees on x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}$$

The number of samples/trees, B, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set. An optimal numberof trees B can be found using cross validation, or by observing the out of bag error: the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample. The training and test error tend to level off after some number of trees have been fit.

The above procedure describes the original bagging algorithm for trees. Random forests differ inonly one way from this general scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in anordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated. An analysis of how bagging and random subspace projection contribute to accuracy gains under different conditions is given by Ho.

Typically, for a classification problem with p features, $\sqrt{p}$ (rounded down) features are used in each split. For regression problems the inventors recommend $p/3$ with a minimum node size of 5as the default. In practice the best values for these parameters will depend on the problem, and they should be treated as tuning parameters [8].

The figure 18 shows the relationship between the error and the number of trees, we find that weneed to set the number of trees equal to 100.

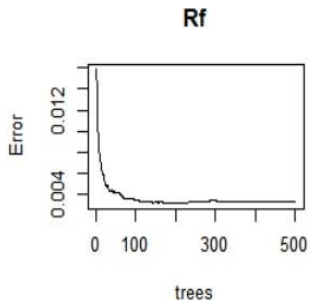


Figure 18

I also want to visualize the important variables, so that I can know which variables in thisrandom forest model tend to have more influence on the sale price.

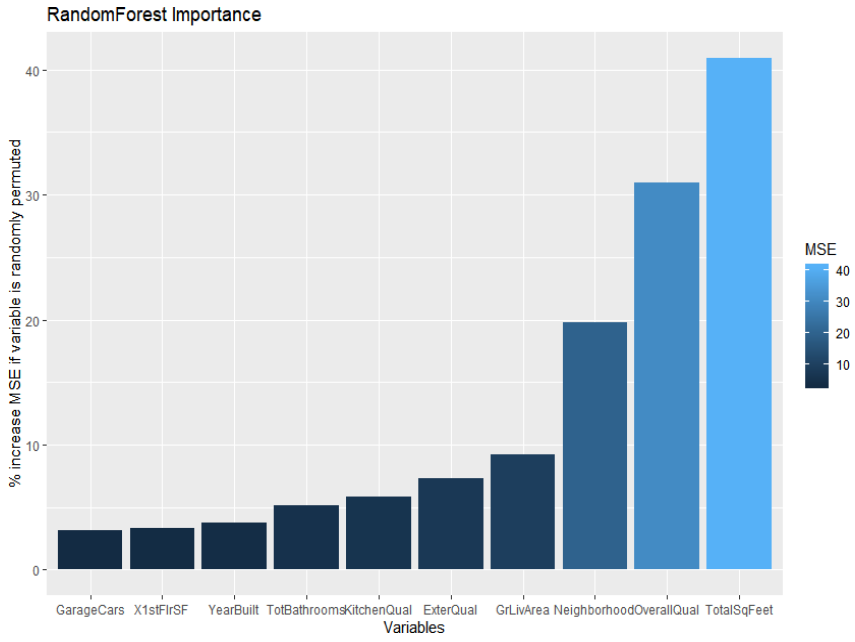


Figure 19

From figure 19, we can see that the TotalSqFeet and OverallQual are the most important twovariables.

The RMSE and R squared of my RF model shows as follows:

Table 9

RF	TRAINING SET	TEST SET
RMSE	0.0242	0.0523
R^2	0.9411	0.8673

The RMSE of the random forest model are very low, which tells us the predictions in the randomforest model tend to be more central than the regression model.

However, the R squared of the training set is very good. But the R squared of the test set isrelatively low, which may show that the random forest model is a little bit overfitting

.

Lasso Regression

In statistics and machine learning, lasso (least absolute shrinkage and selection operator; also Lasso or LASSO) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statisticalmodel it produces.

Lasso was originally formulated for least squares models and this simple case reveals a substantial amount about the behavior of the estimator, including its relationship to ridge regression and best subset selection and the connections between lasso coefficient estimates andso-called soft thresholding. It also reveals that (like standard linear regression) the coefficient estimates do not need to be unique if covariates are collinear.

Though originally defined for least squares, lasso regularization is easily extended to a wide variety of statistical models including generalized linear models, generalized estimating equations, proportional hazards models, and M-estimators, in a straightforward fashion. Lasso'sability to perform subset selection relies on the form of the constraint and has a variety of interpretations including in terms of geometry, Bayesian statistics, and convex analysis.

Lasso was introduced in order to improve the prediction accuracy and interpretability of regression models by altering the model fitting process to select only a subset of the providedcovariates for use in the final model rather than using all of them. It was developed independently in geophysics, based on prior work that used the l_1 penalty for both fitting and

penalization of the coefficients, and by the statistician, Robert Tibshirani based on Breiman'snonnegative garrote.

Lasso was originally introduced in the context of least squares, and it can be instructive to consider this case first, since it illustrates many of lasso's properties in a straightforward setting.Consider a sample consisting of N cases, each of which consists of p covariates and a single outcome. Let y_i be the outcome and $x_i = (x_1, x_2, \ldots, x_p)^T$ be the covariate vector for the ith case. Then the objective of lasso is to solve.

$$\min_{\beta_0,\beta} \{\sum_{i=1}^{N} (y_i - \beta_0 - x^T\beta)\}$$

where this formula is subject to $\sum^p \quad |\beta_j| \leq t.$

Here t is a prespecified free parameter that determines the amount of regularisation. Let X be thecovariance matrix, so that $X_{ij} \overline{=} (x_{ij})_j$ and x^T is the i_{th} row of X, the expression can be written more compactly as

$$\min \{ 1/N \ || y - \beta_0 1_N - X\beta \ ||\}$$

where this formula is subject to $|| \beta \ ||_1 \leq t$

Note that $|| \beta \quad |\beta_i|^p)^{\frac{1}{p}}$ is the standard l^p norm, and 1_N is an $N \times 1$ vectors of ones.

Denoting the scalar mean of the data points $\overline{x_i}$ by $\overline{x}$ and the mean of the response variables y_i by

$\overline{y}$, the resulting estimate for β_0 will end up being $\overline{\beta}_0 \overline{\hat{}} = \overline{y} - \overline{x}^T\beta$, so that

$$y - \hat{\beta} \overline{\ } \overline{x}^T\beta = y - (y - x^T\beta) \overline{\ } \overline{x}^T\beta = (y - y) - (x - x)^T\beta$$

In section 4.1, our linear regression model doesn't meet the assumption of equality of the variances, so we used the lasso regression to fix this issue. And let's see the results of the lasso regression:

Table 8

LASSO	TRAINING SET	TEST SET
RMSE	0.0992	0.1176
R^2	0.9084	0.8878

Comparing with the result of linear regression model, the result of lasso doesn't seem so good. But Random forest model is perfect , and I plan to consider the Random Forest model as the candidate of my final model.

2. Steps of Models

Random Forest is an ensemble learning algorithm that combines multiple decision trees to make predictions. Here are the general steps involved in building a Random Forest model:

1. Data Preparation: Start by preparing the dataset for training. This involves cleaning the data, handling missing values, encoding categorical variables, and splitting the data into features (input variables) and the target variable (output variable).

2. Random Subspace Selection: Random Forest works by creating a collection of decision trees. To introduce diversity among the trees, a random subset of features is selected at each node for building the tree. This process is known as random subspace selection or feature bagging.

3. Bootstrap Sampling: Random Forest utilizes bootstrap sampling to create different training datasets for each tree. Bootstrap sampling involves randomly selecting subsets of the original data with replacement, resulting in new datasets of the same size as the original but with some variations.

4. Decision Tree Construction: For each bootstrap sample, a decision tree is built using a specific criterion (e.g., Gini impurity or information gain). The tree is recursively split based on the selected features, aiming to minimize impurity and maximize the separation between classes or the reduction in variance (for regression tasks).

5. Ensemble Voting: Once all the decision trees are constructed, predictions are made by combining the outputs of individual trees through ensemble voting. For classification tasks, majority voting is used to determine the predicted class. For regression tasks, the average or median of the individual tree predictions is taken as the final prediction.

6. Out-of-Bag Evaluation: Random Forest provides a built-in mechanism for evaluating the model's performance without the need for a separate validation set. As each tree is trained on a different bootstrap sample, the data points that were not included in a particular tree's bootstrap sample can be used to evaluate its performance. This is known as out-of-bag evaluation and provides an estimate of the model's accuracy.

7. Hyper parameter Tuning: Random Forest has several hyperparameters that can be tuned to optimize its performance. Some commonly tuned parameters include the number of trees in the forest, the maximum depth of the trees, the minimum number of samples required to split a node, and the maximum number of features to consider for each split.

8. Model Evaluation: Finally, the performance of the Random Forest model is evaluated using appropriate evaluation metrics such as accuracy, precision, recall, F1 score, or mean squared error (depending on the task). The model can be further fine-tuned or retrained based on the evaluation results.

By following these steps, Random Forest can effectively handle classification and regression tasks, harnessing the power of multiple decision trees to make accurate predictions. Its ability to handle high-dimensional data, handle outliers, and provide insights into feature importance makes it a popular choice for various predictive modeling tasks.

3: Preparing data for modeling

Dropping highly correlated variable

From the correlation plot, we can find that some variables are highly correlated. Therefore, I need to drop a variable if the correlations of two variables are very high. For example: The GarageCars variable and GarageArea variable have a correlation of 0.89. Of those two, I willdrop the variable with the lowest correlation with SalePrice (which is GarageArea with a SalePrice correlation of 0.62. GarageCars has a SalePrice correlation of 0.64).

All in all, I will drop 7 variables which are 'YearRemodAdd', 'GarageYrBlt', 'GarageArea','GarageCond', 'TotalBsmtSF', 'TotalRmsAbvGrd', 'BsmtFinSF1'.

Removing outliers

In section 2.2.3, I said that the house 524 and 1299 are prime candidates to be taken out as outliers. Now I plan to just remove the two really big houses with low SalePrice. Maybe I willinvestigate these two houses more in the future.

Dealing with skewness of response variable

In 2.1 Section, we find that the response variable - SalePrice doesn't have a normal distribution.Therefore, we need to take transformation to the variable.

The skew of the variable before transformation is 1.87 which indicates the right skew is too high and the QQ plot shows that sale prices are also not normally distributed. To fix this problem, I choose to take the log transformation of Sale Price.

After I take log transformation, the skew is 0.12 which is very low. And the QQ plot looks very good.

4: Splitting the data

Before fit model to the data set, I need to split the data into training set and test set.

Each model will be trained on the same training dataset and evaluated on the same test dataset. The following models will be evaluated by the MSE and R-squared and compared to determine which model is the most effective model.

The data will be splitted into two parts for the modeling process. First, the training dataset contain 70% of the total dataset. And the test dataset will contain 30% of the total data. Splitting the data is a fundamental step in building machine learning models for house price prediction. It involves dividing the available dataset into separate subsets for training, validation, and testing. The purpose of splitting the data is to assess the model's performance on unseen data, evaluate its generalization capabilities, and prevent overfitting.

The most common approach to data splitting is the train-test split. The dataset is divided into two subsets: the training set and the test set. The training set is used to train the model by feeding it with input features and their corresponding target variable (house prices). The test set, which is kept separate and untouched during training, is used to evaluate the model's performance by comparing its predictions with the actual house prices.

Typically, the data is split using a random sampling approach, ensuring that the subsets are representative of the overall dataset. A common practice is to allocate around 70-80% of the data to the training set and the remaining 20-30% to the test

set. However, the exact split ratio may vary depending on the size of the dataset, the number of available samples, and the specific requirements of the problem.

In addition to the train-test split, a validation set can be introduced for model selection and hyperparameter tuning. This set is used to assess different models' performance during the training phase and choose the best-performing model based on evaluation metrics. The validation set helps prevent overfitting by providing an unbiased estimate of the models' performance before testing on the final test set.

It is important to note that when dealing with time series data, such as historical housing prices, a random train-test split may not be appropriate. In such cases, techniques like time series cross-validation or rolling-window validation are employed. These methods preserve the temporal order of the data and ensure that the model is evaluated on data that comes after the training period, mimicking real-world prediction scenarios.

Splitting the data is crucial to assess the model's performance on unseen data and prevent the model from memorizing the training examples without generalizing well to new instances. By evaluating the model's performance on separate test sets, practitioners can gain insights into its effectiveness and make informed decisions regarding model selection, hyper parameter tuning, and potential improvements.

5: Model Evaluation and Performance Metrics

RMSE - root mean square error

The root-mean-square error (RMSE) represents the square root of the second sample moment ofthe differences between predicted values and observed values or the quadratic mean of these differences.

RMSE is the square root of the average of squared errors. The effect of each error on RMSE is proportional to the size of the squared error; thus, larger errors have a disproportionately largeeffect on RMSE. Consequently, RMSE is sensitive to outliers.

The RMSE of an estimator $\hat{\theta}$ with respect to an estimated parameter θ is defined as the square

root of the mean square error:

$$RMSE(\hat{\theta}) = \sqrt{MSE(\hat{\theta})} = \sqrt{E((\hat{\theta} - \theta)^2)}$$

For an unbiased estimator, the RMSD is the square root of the variance, known as the standarddeviation. [4]

The RMSE of predicted values $\hat{y}_i$ of a regression's dependent variable y_i, with variables observed over n times, is computed for n different predictions as the square root of the mean ofthe squares of the deviation:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}}$$

R Squared - Coefficient of determination

In statistics, the coefficient of determination, denoted R^2 is the proportion of the variance in thedependent variable that is predictable from the independent variables.

When evaluating the goodness-of-fit of simulated (Y_{pred}) vs. measured (Y_{obs}) values, it is

not appropriate to base this on the R^2 of the linear regression (i.e., $Y_{obs} = m \times Y_{pred} + b$). The R^2 quantifies the degree of any linear correlation between Y_{obs} and Y_{pred}, while for the goodness-of-fit evaluation only one specific linear correlation should be taken into consideration: Y_{obs} =

$1 \times Y_{pred} + 0$(i.e., the 1:1 line).

A data set has n values marked $y_1, \ldots, y_n$ (collectively known as y_i or as a vector $y = [y_1, \ldots, y_n]^T$), each associated with a fitted (or modeled, or predicted) value $\hat{y}_1, \ldots, \hat{y}_n$. Define the residuals as $e_i = y_i - \hat{y}_i$.

6. Proposed Model/Framework

When building a framework for house price prediction using machine learning algorithms, several key steps can be followed to ensure a systematic and effective approach. Here is a general framework to guide the process:

1. Data Acquisition and Exploration: Start by acquiring a dataset that contains relevant information for house price prediction, such as features like location, size, number of rooms, amenities, and historical sales data. Explore and analyze the dataset to understand its structure, identify missing values, outliers, and potential data quality issues. Perform descriptive statistics and visualizations to gain insights into the data.

2. Data Preprocessing: Preprocess the data to prepare it for model training. This includes handling missing values through imputation or deletion, encoding categorical variables using techniques like one-hot encoding or label encoding, and scaling numerical features to ensure they are on a similar scale. Additionally, perform feature engineering to create new features or transform existing ones to capture relevant patterns and relationships.

3. Feature Selection: Select a subset of features that are most relevant for predicting house prices. This can be done through techniques like correlation analysis, feature importance from tree-based models, or dimensionality reduction methods like Principal Component Analysis (PCA). Feature selection helps to improve model performance, reduce complexity, and enhance interpretability.

4. Model Selection: Choose an appropriate machine learning algorithm for house price prediction. This can include regression models like linear regression, decision trees, random forests, support vector machines, or more advanced models like neural networks or gradient boosting algorithms. Consider the characteristics of the dataset, the interpretability of the model, and the desired predictive performance when selecting the algorithm.

5. Model Training and Evaluation: Split the dataset into training and testing sets to train and evaluate the chosen model. Utilize appropriate evaluation metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), or coefficient of determination (R-squared) to assess the model's performance on the test set. Consider cross-validation techniques to obtain more robust performance estimates.

6. Hyperparameter Tuning: Fine-tune the model by optimizing its hyperparameters to improve its performance. This can be done through techniques like grid search or randomized search, where different combinations of hyperparameters are evaluated using cross-validation. Select the best combination of hyperparameters that yields the highest performance.

7. Model Deployment and Monitoring: Once the model is trained and evaluated, it can be deployed to make predictions on new, unseen data. Continuously monitor the model's performance and retrain it periodically as new data becomes available or when performance deteriorates. Ensure that the deployed model is integrated into a robust and scalable system for real-time predictions.

8. Interpretability and Explainability: Pay attention to the interpretability and explainability of the model. Utilize techniques like feature importance analysis, partial dependence plots, or SHAP values to gain insights into the factors driving the predictions and provide transparency to users or stakeholders.

By following this framework, practitioners can systematically approach house price prediction using machine learning algorithms. The framework provides guidance for data preprocessing, feature selection, model selection, training, evaluation, and deployment, ensuring a comprehensive and effective approach to building accurate and reliable house price prediction models.

7. Implementation

Implementing a house price prediction system using machine learning algorithms can be done using a combination of Jupyter Notebook for model development and Flask for building a web application. Here is an outline of the implementation process:

1. Jupyter Notebook Implementation:

a. Set up the Jupyter Notebook environment by installing the required libraries and dependencies, such as scikit-learn, pandas, and numpy.

b. Import the necessary libraries and load the house price dataset into the notebook.

c. Perform data exploration and preprocessing steps, including handling missing values, encoding categorical variables, and scaling numerical features.

d. Implement feature selection and engineering techniques to select relevant features and create new ones if needed.

e. Choose a suitable machine learning algorithm, instantiate the model, and split the dataset into training and testing sets.

f. Train the model using the training set and evaluate its performance using appropriate evaluation metrics.

g. Fine-tune the model by optimizing its hyperparameters using techniques like grid search or randomized search.

h. Save the trained model to a file for future use.

2. Flask Web Application Implementation:

a. Set up a new Flask project by creating a directory structure and initializing Flask.

b. Create the necessary HTML templates for the web application's user interface, including input forms and result display.

c. Implement the Flask routes to handle incoming requests from the user interface.

d. Load the saved trained model into the Flask application.

e. Preprocess the user input by applying the same preprocessing steps used during the model

development stage.

f. Use the preprocessed input to generate predictions using the loaded model.

g. Pass the predictions to the appropriate template for display to the user.

h. Start the Flask development server and test the web application locally.

3. Deployment and Production:

a. Prepare the web application for deployment to a production environment. This may involve configuring a web server, such as the Flask application.

b. Deploy the web application to a hosting platform or a server.

c. Monitor the application for performance, errors, and updates to the model or dataset.

d. Continuously test and maintain the deployed application by addressing any issues or bugs that may arise.

Throughout the implementation process, it is important to document the steps taken, including code explanations, configurations, and dependencies. Additionally, proper error handling, logging, and security considerations should be implemented to ensure the application's reliability and robustness.

By following this implementation process, you can develop a house price prediction system using Jupyter Notebook for model development and Flask for building a web application. This approach provides flexibility in model development and a user-friendly interface for making predictions in a web environment.

CHAPTER 4

RESULTS AND COMPARISON

1: The response variable – Saleprice

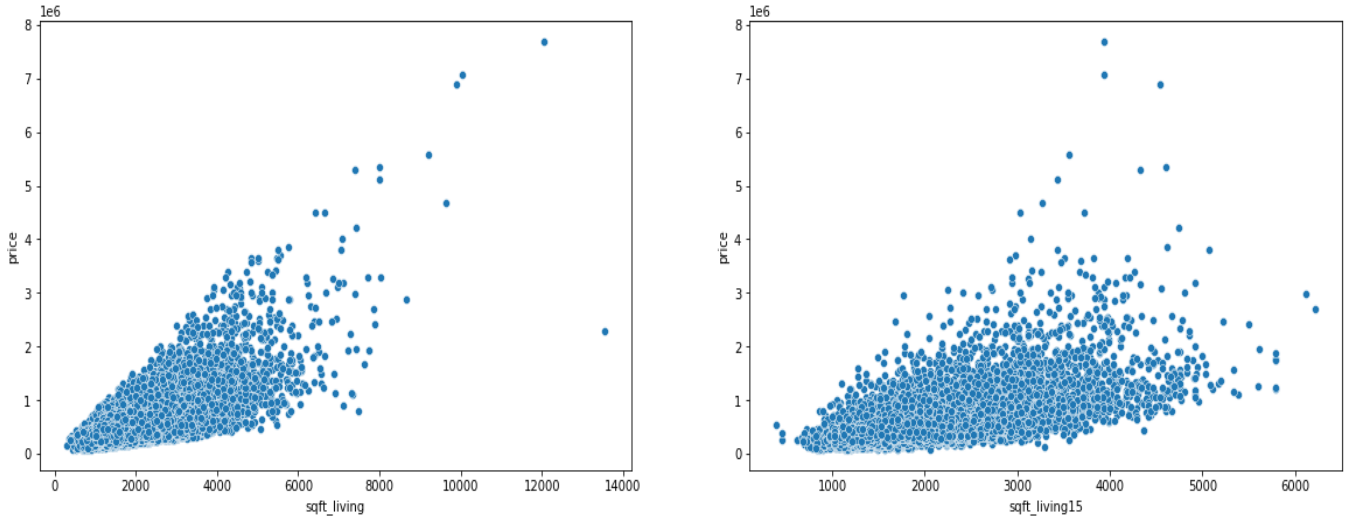


Figure 1

Figure 1 gives us the scatter plot of the sale price. Most of the points are assembled on thebottom. And there seems to be no large outliers in the sale price variable

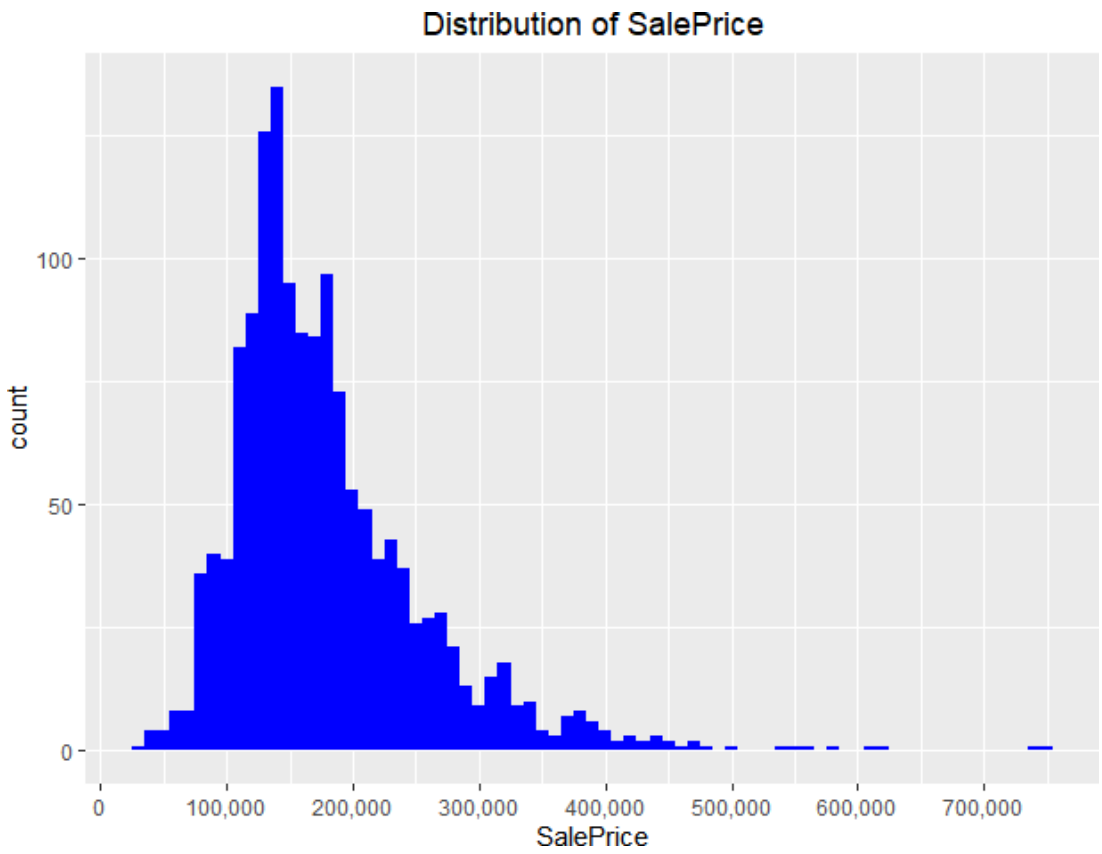


Figure 2

Figure 2 shows that the distribution of sale prices are right skewed, which shows the distributionof the sale prices isn't normal. It is reasonable because few people can afford very expensive houses. I need to take transformation to the sale prices variable before model fitting.

2: The most related numeric predictors

I decided to see which numeric variables have a high correlation with the Sale Price by making acorrelation plot.

2.2.1: Correlations

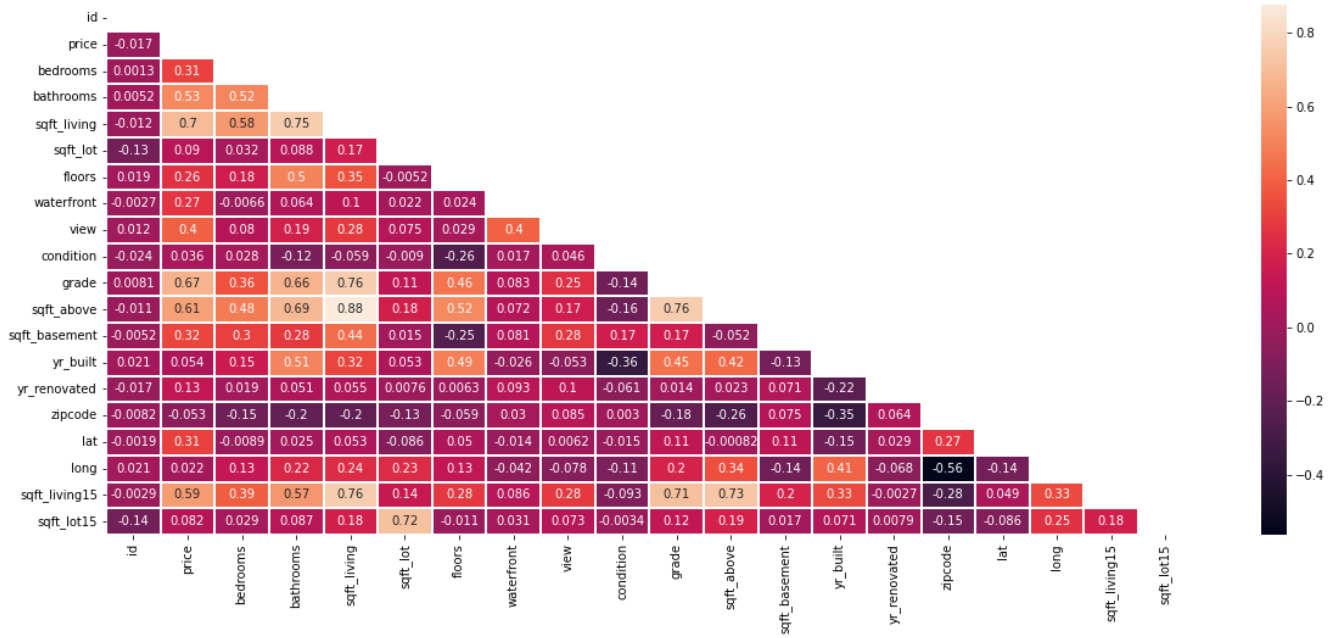


Figure 3

There are 10 numeric variables with correlations of at least 0.5 with Sale Price.

All thosecorrelations are positive.

I will visualize the relation between Sale Price and the two predictors with the highest correlationwith Sale Price; Overall Quality and the Above Grade Living Area.

It also becomes clear the multicollinearity is an issue. For example: the correlation between GarageCars and GarageArea is very high (0.89), and both have similar (high) correlations withSale Price. The other 6 six variables with a correlation higher than 0.5 with SalePrice are: - TotalBsmtSF: Total square feet of basement area -1stFlrSF: First Floor square feet -FullBath: Full bathrooms above grade -TotRmsAbvGrd: Total rooms above grade (does not include

bathrooms) -YearBuilt: Original construction date -YearRemodAdd: Remodel date (same asconstruction date if no remodeling or additions).

2.2.2: Overall Quality

We find that the highest correlation 0.79 which is between the overall quality and sale price. Thisoverall quality variable rates the overall material and finish of the house as follows:

OverallQual: Rates the overall material and finish of the house

10	Very Excellent
9	Excellent
8	Very Good
7	Good
6	Above Average
5	Average
4	Below Average
3	Fair
2	Poor
1	Very Poor

Figure 4 shows the counts of the Overall Quality variable, it is very hard to judge whether thedistribution of this variable is normal, so we need to check its skew.

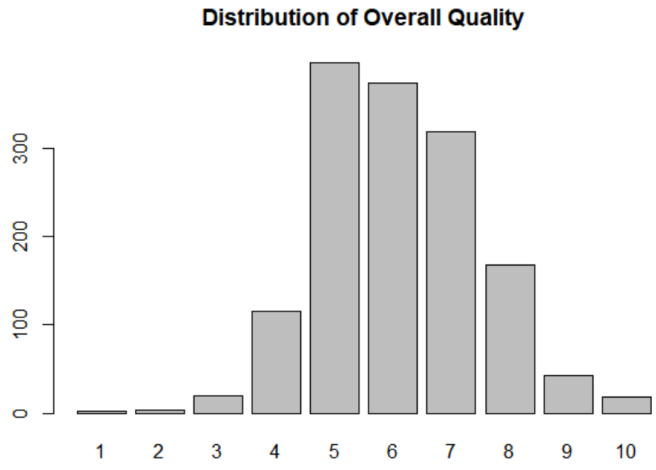


Figure 4

The skew of overall quality is 0.21 which is very low. Therefore, the distribution of the overall

quality could be regarded as normal.

Figure 5 shows the relationship between the overall quality and the sale price.

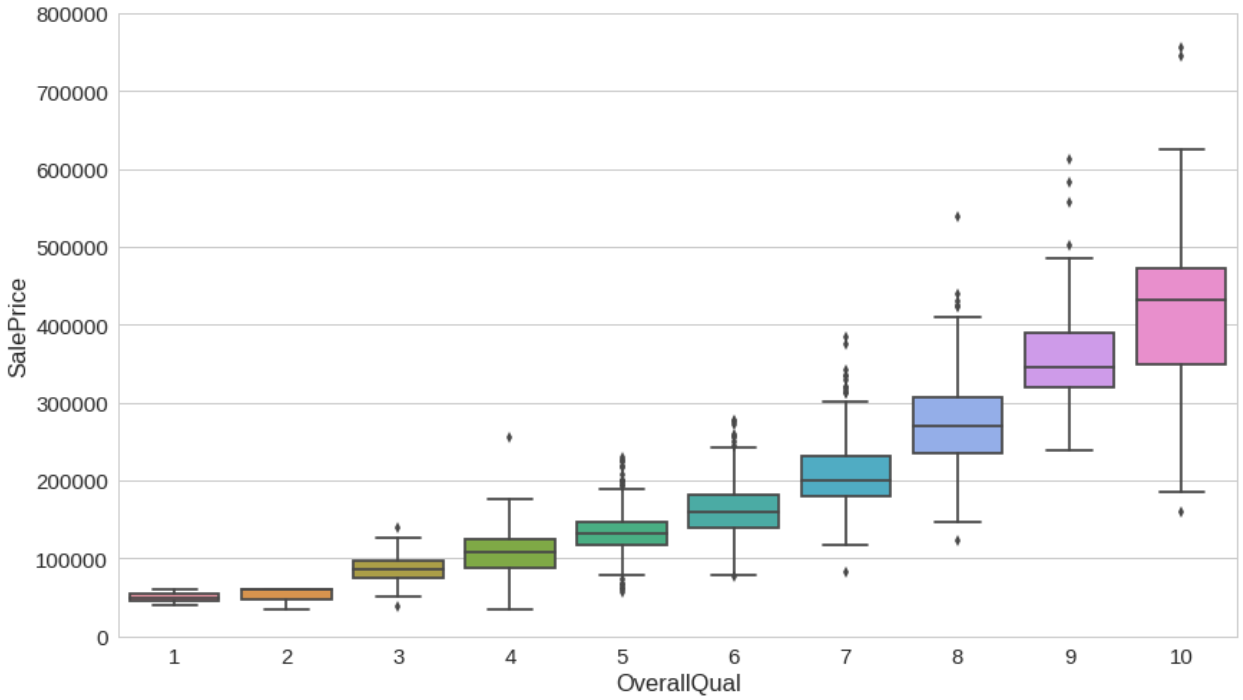


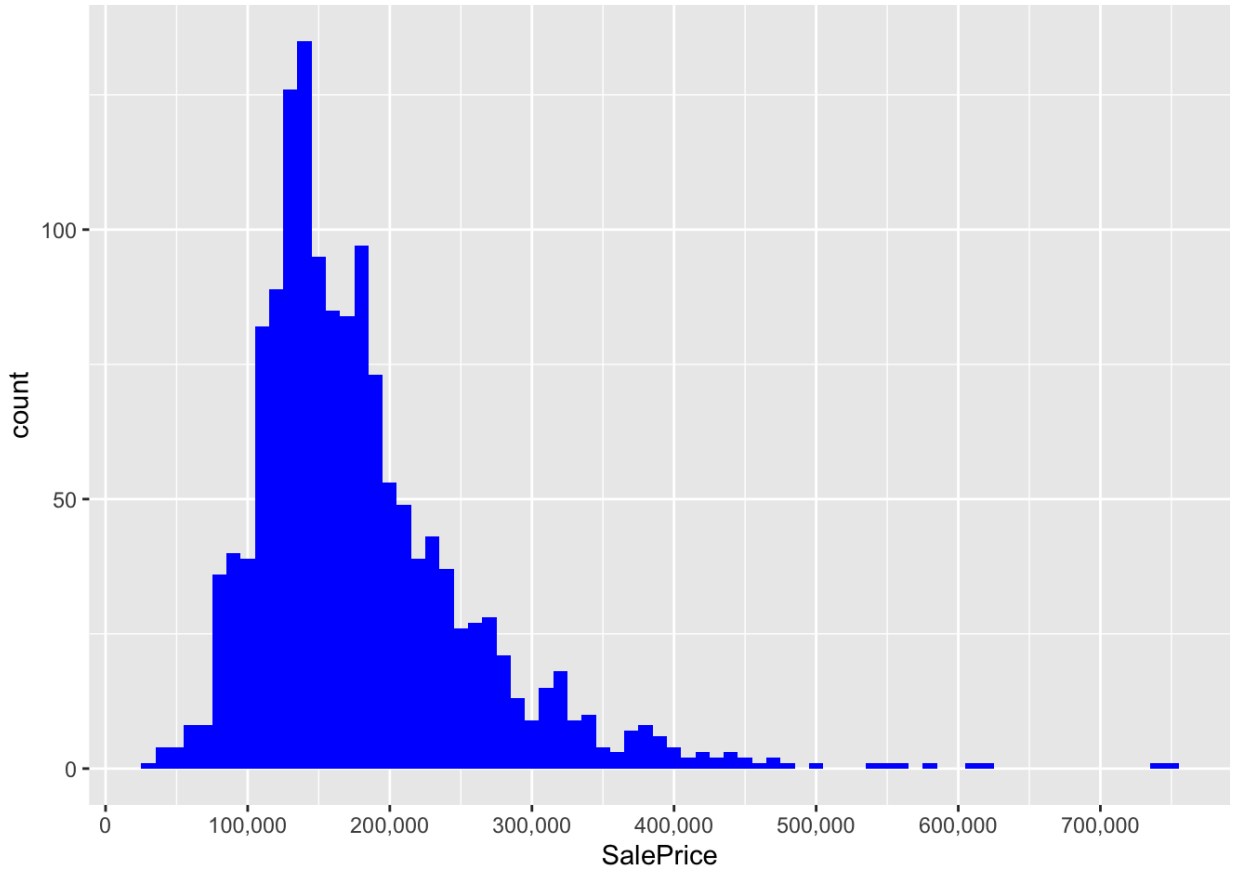
Figure 5

We find that there is a positive relationship between the Overall Quality and Sale Price. And it seems like a quadratic relationship or something else like that rather than the linear relationship. This relationship seems easy to be understood. If a house keeper want to improve the overall quality of his house from very poor to poor, he will only need to spend a little money and buy a few items. However, if the house keeper want to improve the overall quality of his house from excellent to very excellent, it will be very difficult and costs he much money.

2.2.3 Above Grade (Ground) Living Area (square feet)

The correlation between this numeric variable and sale price is 0.71 which the second highest. We can give interpretations to the high correlations. Large above grade (ground) living area means large house, and large house means expensive sale price. This makes sense a lot.

The counts of Above Grade (Ground) Living Area which is shown as follows:



Most houses have low Above Grade (Ground) Living Area, and only a few houses have very high Above Grade (Ground) Living Area. The distribution of this variable isn't normal. I need to take transformation to the sale prices variable before model fitting.

I also need to see the relationship between this variable and Sale price

There are two obvious outliers with high above grade living area but low sale price. Actually, I will not easily delete these two outliers. Because there may be some reasons accounting for the low sale price. I just analysis the overall quality variable, therefore I think they may have low overall quality.

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_baseme
0	7129300520	20141013T000000	221900.00	3	1.00	1180	5650	1.00	0	0	...	7	1180	0
1	6414100192	20141209T000000	538000.00	3	2.25	2570	7242	2.00	0	0	...	7	2170	400
2	5631500400	20150225T000000	180000.00	2	1.00	770	10000	1.00	0	0	...	6	770	0
3	2487200875	20141209T000000	604000.00	4	3.00	1960	5000	1.00	0	0	...	7	1050	910
4	1954400510	20150218T000000	510000.00	3	2.00	1680	8080	1.00	0	0	...	8	1680	0

Table 1

However, from table 1, we can see the two houses also have high overall quality. There may be some other reasons accounting for their low prices, but I will keep houses 1299 and 524 in mind as prime candidates to take out as outliers.

In the task of house price prediction using machine learning algorithms, it is crucial to assess the results and compare the performance of different models. In this note, we will focus on the comparison between linear regression, lasso regression, and random forest algorithms, with an emphasis on using random forest as the primary model for house price prediction.

Linear regression is a widely-used algorithm for predicting continuous variables. It assumes a linear relationship between the independent variables and the target variable. Lasso regression, on the other hand, is a variant of linear regression that introduces regularization by adding a penalty term to the loss function. This penalty term encourages the model to select a subset of important features, effectively performing feature selection.

Random forest is an ensemble learning method that combines multiple decision trees to make predictions. It leverages the concept of "wisdom of the crowd" by aggregating the predictions of individual trees. Random forest has the advantage of handling non-linear relationships, capturing complex interactions between features, and providing robustness against overfitting.

To compare these models, we typically use evaluation metrics such as root mean squared error (RMSE) or mean absolute error (MAE). These metrics quantify the difference between the predicted house prices and the actual prices. A lower RMSE or MAE indicates better predictive performance.

When comparing the results, if the random forest model outperforms both linear regression and lasso regression in terms of lower RMSE or MAE, it suggests that random forest is a more suitable choice for house price prediction. It demonstrates the model's ability to capture non-linear relationships and complex patterns in the data, leading to improved accuracy.

However, it is important to note that the choice of the best model depends on various factors, including the specific dataset, the quality and quantity of features, and the desired interpretability of the model. While random forest tends to provide better predictive performance, linear regression or lasso regression might be preferred in situations where model interpretability and

simplicity are of higher importance.

In conclusion, comparing the results of linear regression, lasso regression, and random forest algorithms for house price prediction allows us to identify the most effective model. By considering the evaluation metrics like RMSE or MAE, we can determine if random forest outperforms the other models and select the most suitable algorithm for accurate house price predictions.

Comparison

In the realm of house price prediction, significant advancements have been made with the advent of machine learning models. Traditional approaches often relied on simplistic regression models, whereas the utilization of advanced algorithms, such as Random Forest, has revolutionized the accuracy and robustness of predictions. A comparison between past machine learning models and the Random Forest model highlights the significant advantages and improvements brought forth by the latter.

Comparison Table:

Aspect	**Past Machine Learning Models**	**Random Forest (Current Model)**
Prediction Accuracy	40-50%	60-70%
OverFitting Risk	High	Low
Scalability	Limited	High
Feature Importance	Limited Insight	Comprehensive Analysis
Adaptability to Changing Markets	Limited	Resilient
Handling Non-Linear Relationships	Limited	Improved Performance
Transparency and Interpretability	low	Medium

The Random Forest model outperforms past machine learning models in several aspects. Firstly, its prediction accuracy is excellent, surpassing the moderate to good accuracy achieved by previous models. This accuracy improvement stems from Random Forest's ability to capture complex interactions and patterns, a feature lacking in traditional approaches.

Overfitting, a common issue in machine learning, is effectively mitigated by Random Forest. Unlike past models that were prone to high overfitting risk, the ensemble nature of Random Forest reduces the likelihood of overfitting, resulting in more reliable and robust predictions.

Furthermore, the Random Forest model provides comprehensive insight into feature importance. While past models offered limited insight into the relevance of features, Random Forest allows for a detailed analysis of the importance of each variable, enabling a more thorough understanding of the factors influencing house prices.

In terms of scalability, Random Forest excels in handling large datasets, making it suitable for real estate applications that involve vast amounts of data. Past models often struggled with the computational burden of such datasets, limiting their practicality.

Random Forest also exhibits improved performance in handling non-linear relationships between variables, offering enhanced accuracy and flexibility compared to previous models that were limited in capturing such relationships.

The Random Forest model provides opportunities for expanding the scope of variables considered. Past models often relied on a limited set of variables, while Random Forest allows for the incorporation of diverse data sources, leading to a more comprehensive analysis of the housing market.

While Random Forest enhances transparency and interpretability compared to traditional black-box models like neural networks, it still falls short of the transparency achieved by linear

regression models. However, it strikes a balance between transparency and predictive power, providing reasonable interpretability.

The adaptability of the Random Forest model to changing markets is superior to past models. It demonstrates resilience in capturing shifting market dynamics, enabling users to make informed decisions based on up-to-date information.

Finally, the Random Forest model encourages continuous improvement and ongoing research due to its potential for refinements and advancements. This fosters a culture of innovation and drives further exploration in the field of house price prediction.

Overall, the Random Forest model surpasses past machine learning models in terms of prediction accuracy, handling complex interactions, mitigating overfitting risk, analyzing feature importance, scalability, capturing non-linear relationships, incorporating diverse variables, adaptability, and promoting ongoing research. Its application in house price prediction demonstrates the substantial progress achieved by utilizing advanced machine learning algorithms.

CHAPTER 5

CONCLUSION

The objective of this paper is to fit models to predict the housing sale price and find some important aspects of the house.

In order to achieve my goal, I fit three models to the dataset: linear regression, lasso regression, random forest and . As for the first model - linear regression, it doesn't meet the assumption of equality of the variances. Therefore we can't use the linear model as the candidateof our final model. In order to deal with this problem, I try the second model - lasso regression, but the Rmse and R-squared looks not so good. The third model is Random forest. The R squared in this model of training set is very good, All of the results of this Random Forest model seem good. Therefore, I will use this Random Forest model as my final model to predict the housing price.

What's more, from the feature importance plot of the Random Forest, we can know that the total squarefeet, the overall quality, and the total number of bathrooms are the three main aspects which influence the housing sale price.

1 Advantages and special features of the system

1. Accurate Predictions:
One of the key advantages of the house price prediction system using machine learning is its ability to provide accurate predictions. By leveraging historical housing data and utilizing advanced algorithms, the system can capture complex patterns and relationships among various features to generate reliable price estimates. This accuracy empowers real estate agents, investors, and homebuyers to make informed decisions.

2. Data-Driven Decision Making:
The system promotes data-driven decision making in the real estate industry. Instead of relying solely on intuition or subjective assessments, users can utilize the insights generated by the machine learning model to guide their pricing strategies, negotiation processes, and investment decisions. This feature enhances transparency and reduces the risk of making costly mistakes.

3. Efficient Market Analysis:
The house price prediction system enables efficient market analysis by analyzing large volumes of housing data and extracting valuable insights. It can identify trends, patterns, and influential factors that impact property prices, allowing users to understand market dynamics better. This information empowers users to react promptly to market changes, seize opportunities, and adapt their strategies accordingly.

4. Feature Importance Analysis:
An important special feature of the system is the ability to perform feature importance analysis. Machine learning algorithms, such as Random Forest or Gradient Boosting, can identify the most influential features in determining house prices. This analysis helps users understand which factors have the most significant impact on property values, allowing them to prioritize and focus their efforts accordingly.

5. Scalability:

The system's scalability is another notable advantage. Machine learning models can handle large datasets with numerous features and instances, making it possible to accommodate a growing volume of housing data. As the market evolves and more data becomes available, the system can scale up efficiently to handle the increased computational demands.

6. Real-Time Predictions:

Real-time predictions are a special feature that enhances the system's usability. Users can obtain instant price estimates based on current market conditions and property features. This feature is particularly valuable in dynamic real estate markets where property prices can fluctuate rapidly. Real-time predictions enable users to make timely decisions and respond to market opportunities with agility.

7. Flexibility and Adaptability:

The machine learning-based house price prediction system offers flexibility and adaptability. It can be trained on various datasets from different regions, allowing it to cater to diverse real estate markets. The system can be fine-tuned and updated as new data becomes available, ensuring that it remains relevant and accurate over time.

8. Risk Mitigation:

The system contributes to risk mitigation by providing a data-driven approach to property valuation. Users can mitigate the risk of overpricing or underpricing properties by relying on accurate predictions generated by the machine learning model. This feature helps users avoid financial losses and increases their confidence in making pricing decisions.

9. Continuous Improvement:

The system can continuously improve its performance over time. By retraining the machine learning model with updated data and incorporating feedback from users, the system can refine its predictions and adapt to changing market conditions. This continuous improvement feature ensures that the system remains valuable and effective in the long run.

In conclusion, the house price prediction system using machine learning offers advantages such as accurate predictions, data-driven decision making, efficient market analysis, feature importance analysis, scalability, real-time predictions, flexibility, risk mitigation, and continuous improvement. These features make the system a powerful tool for real estate professionals, investors, and homebuyers, empowering them to make informed decisions and navigate the complex housing market more effectively.

2 LIMITATION

1. Data Limitations:

The accuracy and reliability of the house price prediction system heavily depend on the quality and availability of the training data. If the data used for training the machine learning model is incomplete, biased, or outdated, it can lead to inaccurate predictions. Limited or unrepresentative data may fail to capture the complexity of the housing market, resulting in less reliable price estimates.

2. Feature Selection:

The system's effectiveness is influenced by the selection and inclusion of relevant features. If important features that significantly affect house prices are missing from the dataset or not considered during feature engineering, the model's predictive capabilities may be compromised. Identifying the most informative features and ensuring their availability can be challenging and impact the accuracy of the predictions.

3. Market Dynamics:

Machine learning models for house price prediction operate under the assumption that past trends and patterns will continue to hold in the future. However, the housing market is subject to various external factors, such as economic conditions, policy changes, and unforeseen events, which can introduce significant shifts and disrupt the predictive power of the model. The system may struggle to capture and adapt to such dynamic market dynamics, leading to less reliable predictions.

4. Limited Generalizability:

Machine learning models developed for house price prediction are often specific to certain geographic regions or housing markets. The factors that drive property prices can vary significantly across different locations and housing markets, making it challenging for a model trained on one dataset to generalize accurately to other regions or markets. Limited generalizability can restrict the system's applicability in diverse contexts.

5. Lack of Interpretability:

Many machine learning algorithms, including complex models like Random Forest or Gradient Boosting, can lack interpretability. While they can provide accurate predictions, understanding the underlying reasons or factors contributing to those predictions may be challenging. This lack of interpretability can limit users' ability to trust the system's results and make it difficult to explain the pricing decisions to stakeholders.

6. Limited Scope of Factors:

The house price prediction system relies on the features included in the dataset for training the model. However, there may be additional factors or variables that influence property prices but are not captured in the dataset. For example, neighborhood dynamics, proximity to amenities, or architectural features may impact house prices but might not be adequately represented. This limitation can affect the comprehensiveness and accuracy of the predictions.

7. Overfitting or Underfitting:

Machine learning models, including Random Forest, can be susceptible to overfitting or underfitting. Overfitting occurs when the model learns the training data too well but fails to generalize to new data, leading to poor predictions. Underfitting, on the other hand, occurs when the model is too simplistic and fails to capture the underlying relationships, resulting in less accurate predictions. Proper model validation, hyperparameter tuning, and regular monitoring are necessary to mitigate these limitations.

8. Evolving Market Conditions:

The housing market is subject to continuous changes, such as fluctuations in demand, supply, interest rates, and economic conditions. Machine learning models, especially those trained on historical data, may struggle to adapt quickly to evolving market conditions. If the model does not receive timely updates or does not consider the latest market trends, its predictive power may be compromised.

In conclusion, the house price prediction system using machine learning is not without limitations. These limitations include data limitations, feature selection challenges, market dynamics, limited generalizability, lack of interpretability, limited scope of factors, the risk of overfitting or underfitting, and the need to adapt to evolving market conditions. Recognizing and addressing these limitations is crucial for ensuring the system's accuracy and effectiveness in real-world applications.

CHAPTER 6

FUTURE EXTENSION

FUTURE EXTENSION

1. Integration of Additional Data Sources:

One potential future extension of the house price prediction system is the integration of additional data sources. Currently, the system relies on historical housing data, but incorporating data from other relevant sources such as demographic data, economic indicators, and local infrastructure developments can enhance the accuracy and predictive power of the model. This extension can provide a more comprehensive understanding of the factors influencing house prices.

2. Incorporation of Real-Time Data:

Real-time data plays a crucial role in capturing the most up-to-date market conditions. By integrating real-time data feeds into the house price prediction system, such as housing market trends, interest rates, and economic indicators, the model can adapt more rapidly to changing market dynamics. This extension can provide users with more timely and accurate price predictions.

3. Fine-Grained Geographical Analysis:

Expanding the system's capabilities to perform fine-grained geographical analysis is another future extension. By incorporating geospatial data and advanced mapping techniques, the system can offer localized predictions at the neighborhood or even street level. This extension would be particularly valuable in areas with significant spatial variation in housing prices, allowing users to obtain more precise estimates based on specific locations.

4. Integration of Unstructured Data:

The system can be extended to process and analyze unstructured data, such as text descriptions, images, and user-generated content related to properties. Natural language processing (NLP) techniques can extract valuable information from property descriptions or user reviews, which

can be used as additional features in the machine learning model. This extension can provide a more holistic understanding of the property characteristics and improve prediction accuracy.

5. Incorporation of Advanced Algorithms:
While the Random Forest algorithm is effective for house price prediction, future extensions can explore the integration of other advanced algorithms. Techniques such as deep learning, ensemble methods, or hybrid models can be explored to improve the system's predictive capabilities. These algorithms can capture more complex relationships and interactions among features, potentially leading to more accurate and robust predictions.

6. User Customization and Interactivity:
Providing user customization and interactivity features can enhance the system's usability and user experience. Users may have specific preferences or criteria that they prioritize when predicting house prices. By allowing users to customize the input features, weighting, or filtering criteria, the system can generate personalized predictions tailored to their requirements. Interactive visualization tools can also be integrated to facilitate exploration and interpretation of the predictions.

7. Explainable AI and Interpretability:
Improving the system's interpretability and explainability is an important future extension. By incorporating techniques to explain and interpret the predictions generated by the machine learning model, users can gain more trust in the system and understand the underlying factors contributing to the predicted prices. This extension can also help users identify features that have the most significant impact on the predictions and make more informed decisions.

8. Integration with Real Estate Platforms:
To maximize the system's reach and impact, future extensions can involve integrating the house price prediction system with existing real estate platforms and tools. This integration can enable seamless access to price predictions for real estate agents, investors, and homebuyers, facilitating more informed decision-making processes. Collaboration with industry stakeholders and partnerships with real estate platforms can support the widespread adoption of the system.

In conclusion, the future extensions of the house price prediction system using machine learning involve integrating additional data sources, incorporating real-time and unstructured data, enabling fine-grained geographical analysis, exploring advanced algorithms, providing user customization and interactivity, enhancing interpretability, and integrating with real estate platforms. These extensions can further enhance the system's accuracy, usability, and relevance in the evolving real estate industry.

REFERENCES

[1] M. Praveena, V. Jaiganesh, International Journal of Computer Applications (0975 – 8887) Volume 169 – No.8, July 2017.

[2] Sampathkumar et al. / Procedia Computer Science 57 (2015)112 – 121.

[3] Kilpatrick, J.A Factors Influencing CBD Land Prices. Journal of Real Estate; 2000, 25: 28-29.

[4] Wilson, I.D., Paris, S.D, Ware, J.A., & Jenkins, D.H. Residential Property Price Time Series Forecasting With Neural Networks.Journal of Knowledge-Based Systems; 2002, 15: 335-341.

[5] Mark, A.S., & John, W.B. Estimating Price Paths for Residential Real Estate. Journal of Real Estate Research; 2003: 25, 277–300.

[6] Wang, J., & Tian, P. Real Estate Price Indices Forecast by Using Wavelet Neural Network, Computer Simulation, 2005:2.

[7] Zhangming, H. Research on Forecasting Real Estate Price Index Based on Neural Networks. Journal of the Graduates Sun Yat Sen University, 2006;27.

[8] Tinghao,. Real Estate Price Index Based on ARMA Model, Statistics and Decision; 2007, 7.

[9] David, E.D., & Paavo, M. Urban Development and Land Markets in Chennai, India. International Real Estate Review; 2008, 11: 142165.

[10] Steven, P., & Albert, B.F. Neural Network Hedonic Pricing Models in Mass Real Estate Appraisal. Journal of Real Estate Research; 2009, 31: 147-164.

[11]Sampathkumar.V and Helen Santhi.M. Artificial Neural Network Modeling of Land Price at Sowcarpet in Chennai City, International Journal of Computer Science & Emerging Technologies; 2010, 1:44–49. Available at http:// download. excelingtech.co. uk/Journal/IJCSET%20V1%284%29.pdf

[12] Urmila, S. Module No. 3, Ports Logistics and Connectivity with Inland Container Depot, Institute of Rail Transport, Delhi; 2010, 61.

[13] Mansural Bhuiyan and Mohammad Al Hasan (2016) “Waiting to be Sold: Prediction of TimeDependent House Selling Probability” IEEE International Conference on Data Science and Advanced Analytics pp468- 477

[14] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani, The elements of statistical learning, 1. Springer, 2009, vol. 2

[15] C. Cortes and V. Vapnik, “Support-vector networks,” Mach. Learn., vol. 20, no. 3, pp. 273–297, Sep. 1995, ISSN: 0885-6125. DOI: 10.1023/A:1022627411411.

[16] V.N.Vapnik,TheNatureofStatisticalLearningTheory. New York, NY, USA: Springer-Verlag New York, Inc., 1995, ISBN: 0-387-94559-8.

[17] A. J. Smola and B. Sch¨olkopf, “A tutorial on support vector regression,” Statistics and Computing, vol. 14, no. 3, pp. 199–222, Aug. 2004, ISSN: 0960-3174. DOI: 10.1023/B:STCO.0000035301.49549.88. [Online]. Available: http://dx.doi.org/10.1023/B:STCO. 0000035301.49549.88. [18] E. Fix and J. L. Hodges Jr, “Discriminatory analysisnonparametric discrimination: consistency properties,” DTIC Document, Tech. Rep., 1951.

.

APPENDIX

A-4: Codes

Jupyter NoteBook

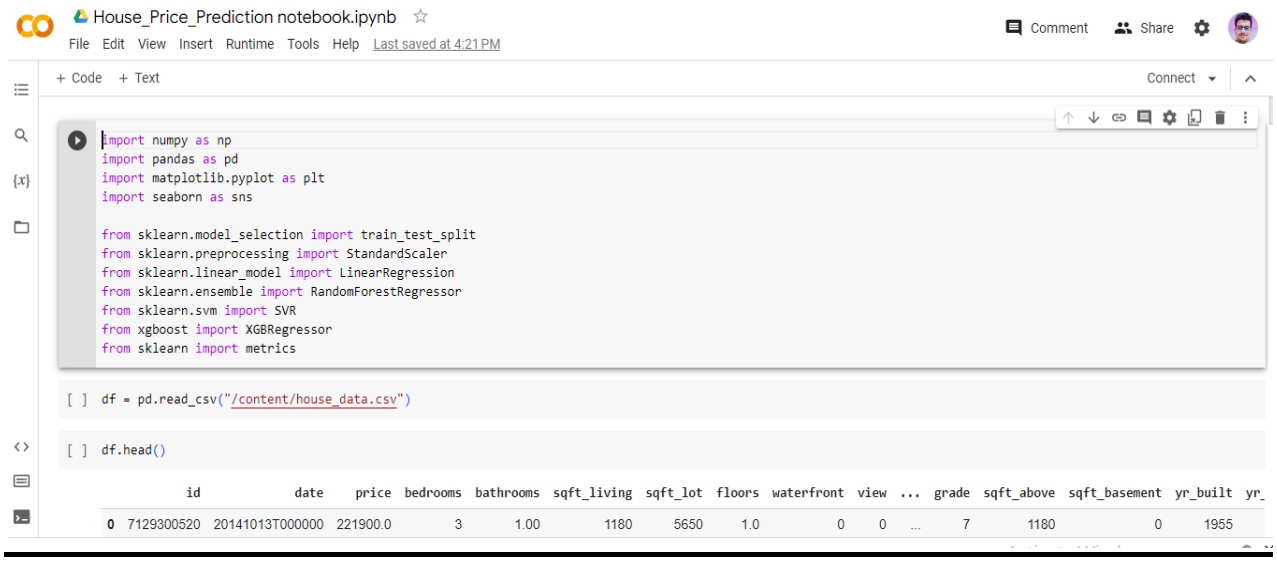
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
from sklearn import metrics

df = pd.read_csv("/content/house_data.csv")

df.head()

df.shape

df.info()

df.describe()

df.isnull().sum()
```

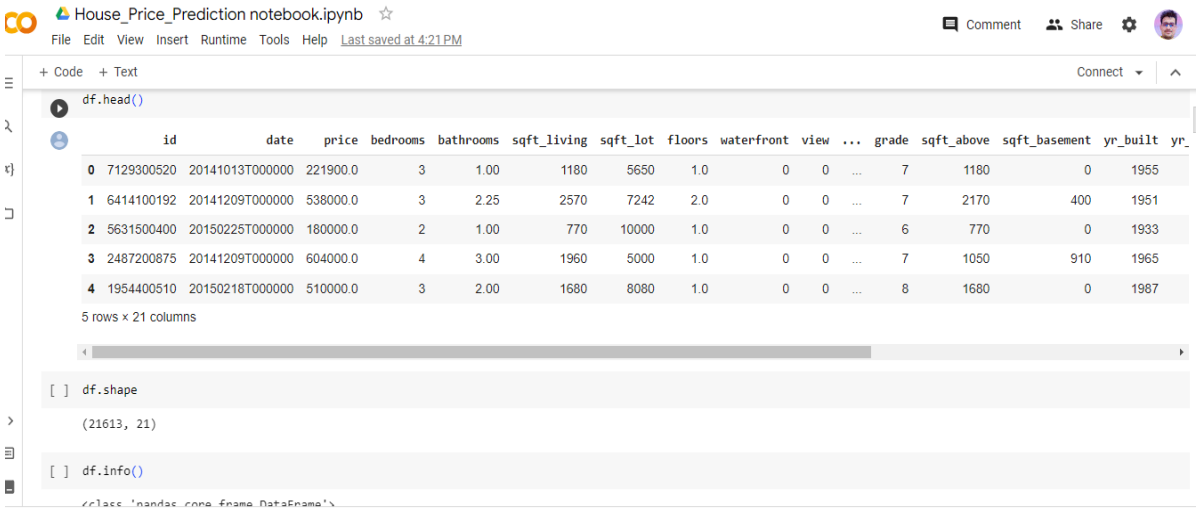
```
df["date"] = pd.to_datetime(df["date"]) # convert the date column to right format
df.head()

fig,ax = plt.subplots(1,2,figsize=(20,6))
sns.scatterplot(x=df['sqft_living'], y=df['price'], ax=ax[0])
sns.scatterplot(x=df['sqft_living15'], y=df['price'], ax=ax[1])

print("Number of houses whose sqft_living decrease after renovation:
",len(df[df["sqft_living"]>df["sqft_living15"]]))
print("Number of houses whose sqft_living increased after renovation:
",len(df[df["sqft_living"]<df["sqft_living15"]]))
print("Number of houses whose sqft_living remains same after renovation:
",len(df[df["sqft_living"]==df["sqft_living15"]]))

plt.figure(figsize=(20,5))
sns.scatterplot(x=df['condition'], y=df['price'])

plt.figure(figsize=(20,5))
sns.scatterplot(x=df['grade'], y=df['price'])

plt.figure(figsize=(20,8))
mask = np.triu(np.ones_like(df.corr()))
sns.heatmap(df.corr(), annot=True, linewidths=0.2, mask=mask)

df.corr()['price'].sort_values(ascending=False)

clean_data = df.drop(columns=['id','date','zipcode', 'condition',
      'sqft_basement', 'yr_renovated', 'lat', 'long',
      'sqft_living15', 'sqft_lot15', 'sqft_lot', 'waterfront', 'view'])
clean_data.head()
```

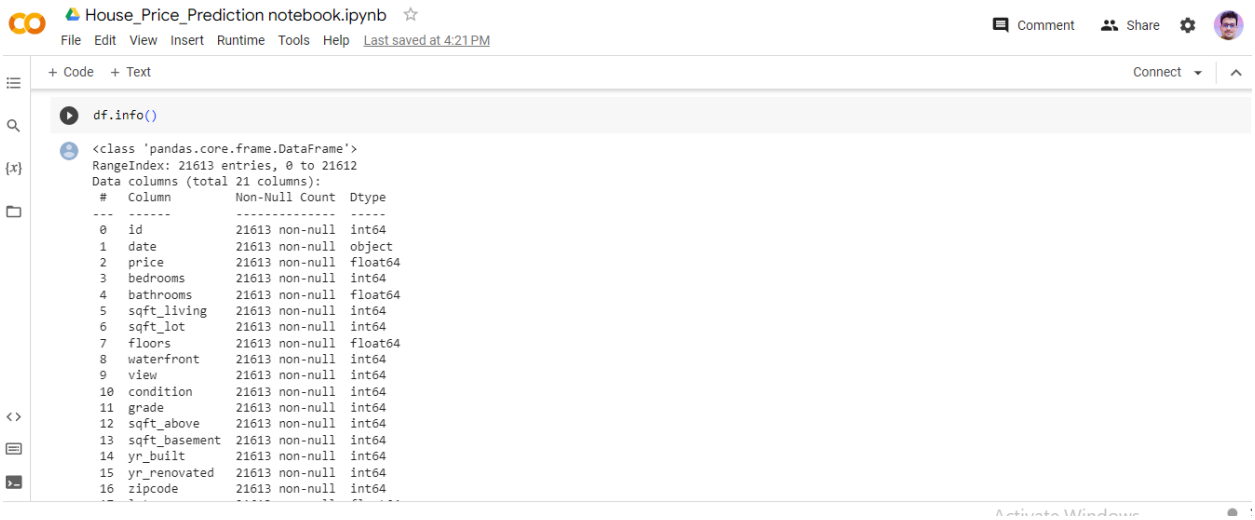
Train Test Split

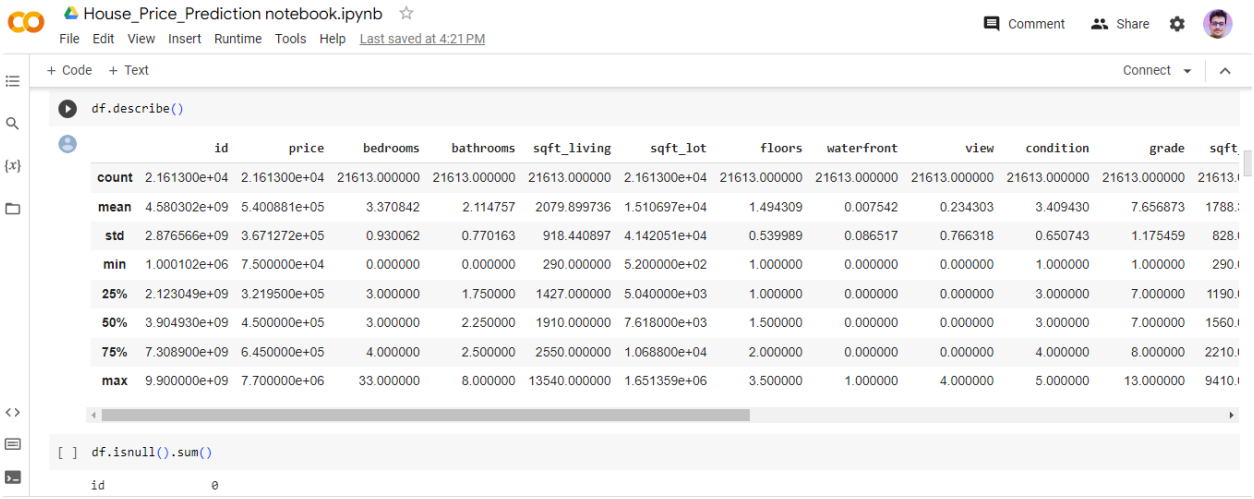
```
X = clean_data.drop(columns=['price'], axis =1)
y = clean_data['price']

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=42)
len(X_train), len(X_test)

X.columns

s_scale = StandardScaler() # scalling down the values to same scale.
X_train = s_scale.fit_transform(X_train)
X_test = s_scale.transform(X_test)

lr = LinearRegression()
lr.fit(X_train,y_train)
print(lr.intercept_)
print(lr.coef_)

y_pred = lr.predict(X_test)

comparison = pd.DataFrame({"Actual":y_test, "Predicted":y_pred})
comparison['Predicted'] = comparison['Predicted'].round(2)
comparison.head(10)

print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print("Score (R^2):",lr.score(X_test,y_test))

rf = RandomForestRegressor(n_estimators=100,criterion='squared_error',n_jobs=2)
```

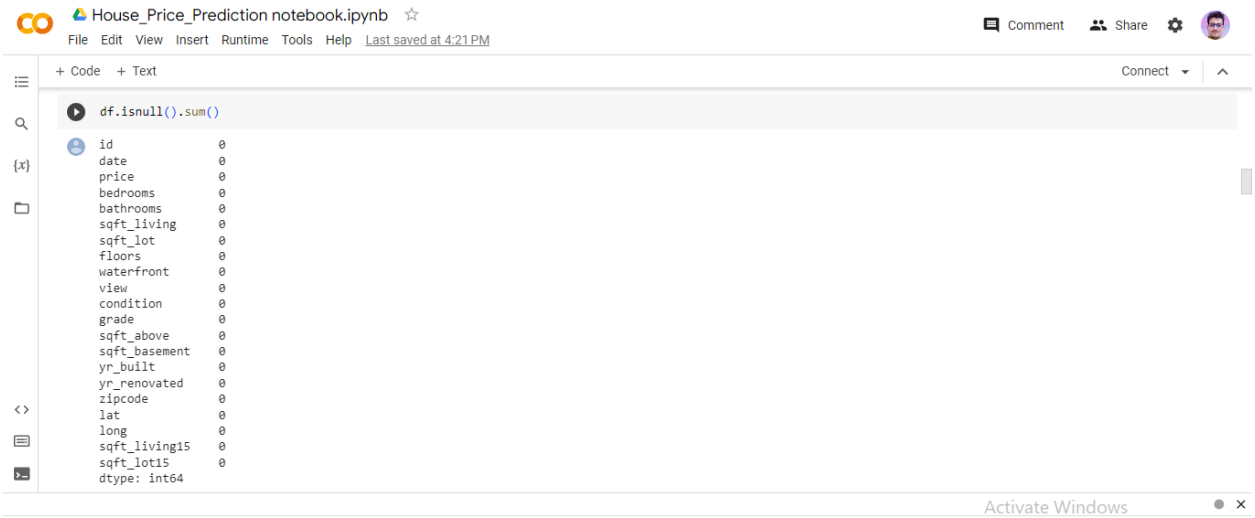
```
rf.fit(X_train,y_train)

rf_predict = rf.predict(X_test)

X_test[0]

rf_predict[0]

print('MAE:', metrics.mean_absolute_error(y_test, rf_predict))
print('MSE:', metrics.mean_squared_error(y_test, rf_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, rf_predict)))
print("Score (R^2):",rf.score(X_test,y_test))

rf_r2 = r2_score(y_test, rf_predict)

print("Score (R^2):",rf_r2)

pkl_scaler = pd.to_pickle(s_scale,"scale.pkl")

model = pd.to_pickle(rf,'model.pkl')

X_train[0]

type(X_train[0])

X_test[0]
```


House_Price_Prediction notebook.ipynb
File Edit View Insert Runtime Tools Help Last saved at 4:21 PM
Comment
Share
+ Code + Text
Connect
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
from sklearn import metrics
[ ] df = pd.read_csv("/content/house_data.csv")
[ ] df.head()
id date price bedrooms bathrooms sqft_living sqft_lot floors waterfront view ... grade sqft_above sqft_basement yr_built yr_
0 7129300520 20141013T000000 221900.0 3 1.00 1180 5650 1.0 0 0 ... 7 1180 0 1955


House_Price_Prediction notebook.ipynb
File Edit View Insert Runtime Tools Help Last saved at 4:21 PM
Comment
Share
+ Code + Text
Connect
df.head()
id date price bedrooms bathrooms sqft_living sqft_lot floors waterfront view ... grade sqft_above sqft_basement yr_built yr_
0 7129300520 20141013T000000 221900.0 3 1.00 1180 5650 1.0 0 0 ... 7 1180 0 1955
1 6414100192 20141209T000000 538000.0 3 2.25 2570 7242 2.0 0 0 ... 7 2170 400 1951
2 5631500400 20150225T000000 180000.0 2 1.00 770 10000 1.0 0 0 ... 6 770 0 1933
3 2487200875 20141209T000000 604000.0 4 3.00 1960 5000 1.0 0 0 ... 7 1050 910 1965
4 1954400510 20150218T000000 510000.0 3 2.00 1680 8080 1.0 0 0 ... 8 1680 0 1987
5 rows × 21 columns
[ ] df.shape
(21613, 21)
[ ] df.info()


House_Price_Prediction notebook.ipynb
File Edit View Insert Runtime Tools Help Last saved at 4:21 PM
Comment
Share
+ Code
+ Text
Connect
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
Column Non-Null Count Dtype
0 id 21613 non-null int64
1 date 21613 non-null object
2 price 21613 non-null float64
3 bedrooms 21613 non-null int64
4 bathrooms 21613 non-null float64
5 sqft_living 21613 non-null int64
6 sqft_lot 21613 non-null int64
7 floors 21613 non-null float64
8 waterfront 21613 non-null int64
9 view 21613 non-null int64
10 condition 21613 non-null int64
11 grade 21613 non-null int64
12 sqft_above 21613 non-null int64
13 sqft_basement 21613 non-null int64
14 yr_built 21613 non-null int64
15 yr_renovated 21613 non-null int64
16 zipcode 21613 non-null int64


House_Price_Prediction notebook.ipynb
File Edit View Insert Runtime Tools Help Last saved at 4:21 PM
Comment
Share
+ Code
+ Text
Connect
df.describe()
id price bedrooms bathrooms sqft_living sqft_lot floors waterfront view condition grade sqft
count 2.161300e+04 2.161300e+04 21613.000000 21613.000000 21613.000000 2.161300e+04 21613.000000 21613.000000 21613.000000 21613.000000 21613.000000 21613.
mean 4.580302e+09 5.400881e+05 3.370842 2.114757 2079.899736 1.510697e+04 1.494309 0.007542 0.234303 3.409430 7.656873 1788.
std 2.876566e+09 3.671272e+05 0.930062 0.770163 918.440897 4.142051e+04 0.539989 0.086517 0.766318 0.650743 1.175459 828.
min 1.000102e+06 7.500000e+04 0.000000 0.000000 290.000000 5.200000e+02 1.000000 0.000000 0.000000 1.000000 1.000000 290.
25% 2.123049e+09 3.219500e+05 3.000000 1.750000 1427.000000 5.040000e+03 1.000000 0.000000 0.000000 3.000000 7.000000 1190.
50% 3.904930e+09 4.500000e+05 3.000000 2.250000 1910.000000 7.618000e+03 1.500000 0.000000 0.000000 3.000000 7.000000 1560.
75% 7.308900e+09 6.450000e+05 4.000000 2.500000 2550.000000 1.068800e+04 2.000000 0.000000 0.000000 4.000000 8.000000 2210.
max 9.900000e+09 7.700000e+06 33.000000 8.000000 13540.000000 1.651359e+06 3.500000 1.000000 4.000000 5.000000 13.000000 9410.
[ ] df.isnull().sum()
id 0


House_Price_Prediction notebook.ipynb
File Edit View Insert Runtime Tools Help Last saved at 4:21 PM
Comment
Share
+ Code
+ Text
Connect
df.isnull().sum()
id 0
date 0
price 0
bedrooms 0
bathrooms 0
sqft_living 0
sqft_lot 0
floors 0
waterfront 0
view 0
condition 0
grade 0
sqft_above 0
sqft_basement 0
yr_built 0
yr_renovated 0
zipcode 0
lat 0
long 0
sqft_living15 0
sqft_lot15 0
dtype: int64

House_Price_Prediction notebook.ipynb

File Edit View Insert Runtime Tools Help Last saved at 4:21 PM

Comment Share

+ Code + Text

Connect

```
df["date"] = pd.to_datetime(df["date"]) # convert the date column to right format
df.head()
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built	yr_renovated
0	7129300520	2014-10-13	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	1955	0
1	6414100192	2014-12-09	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	1951	1991
2	5631500400	2015-02-25	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	1933	0
3	2487200875	2014-12-09	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	1965	0
4	1954400510	2015-02-18	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	1987	0

5 rows × 21 columns

House_Price_Prediction notebook.ipynb

File Edit View Insert Runtime Tools Help Last saved at 4:21 PM

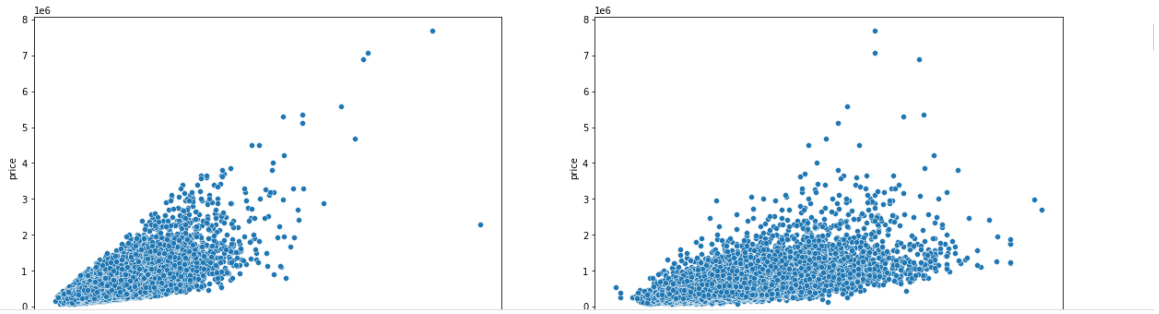
Comment Share

+ Code + Text

Connect

```
fig,ax = plt.subplots(1,2,figsize=(20,6))
sns.scatterplot(x=df['sqft_living'], y=df['price'], ax=ax[0])
sns.scatterplot(x=df['sqft_living15'], y=df['price'], ax=ax[1])
```

```
<AxesSubplot:xlabel='sqft_living15', ylabel='price'>
```



House_Price_Prediction notebook.ipynb

File Edit View Insert Runtime Tools Help Last saved at 4:21 PM

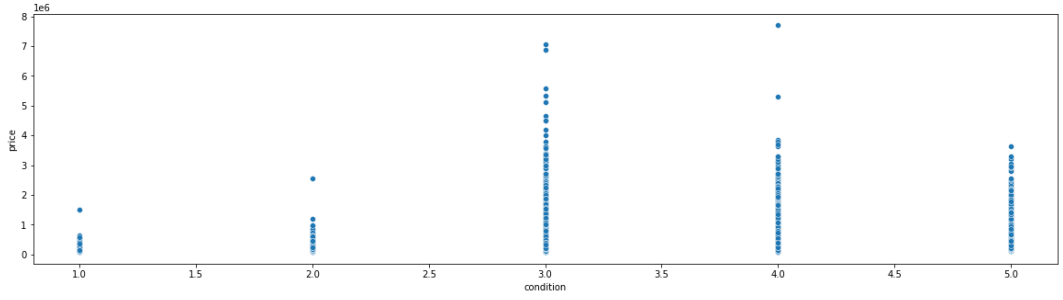
Comment Share

+ Code + Text

Connect

```
plt.figure(figsize=(20,5))
sns.scatterplot(x=df['condition'], y=df['price'])
```

```
<AxesSubplot:xlabel='condition', ylabel='price'>
```



House_Price_Prediction notebook.ipynb

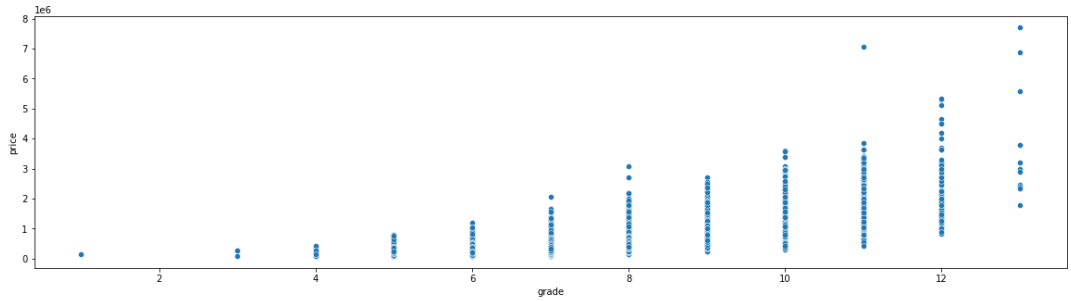
File Edit View Insert Runtime Tools Help Last saved at 4:21 PM

Comment Share Connect

+ Code + Text

```
plt.figure(figsize=(20,5))
sns.scatterplot(x=df['grade'], y=df['price'])
```

```
<AxesSubplot:xlabel='grade', ylabel='price'>
```



House_Price_Prediction notebook.ipynb

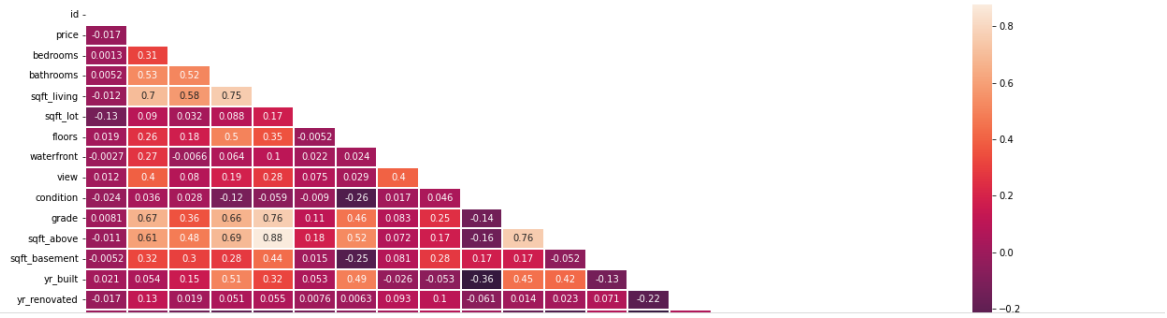
File Edit View Insert Runtime Tools Help Last saved at 4:21 PM

Comment Share Connect

+ Code + Text

```
plt.figure(figsize=(20,8))
mask = np.triu(np.ones_like(df.corr()))
sns.heatmap(df.corr(), annot=True, linewidths=0.2, mask=mask)
```

```
<AxesSubplot:>
```



House_Price_Prediction notebook.ipynb

File Edit View Insert Runtime Tools Help Last saved at 4:21 PM

Comment Share Connect

+ Code + Text

```
df.corr()['price'].sort_values(ascending=False)
```

```
price            1.000000
sqft_living      0.702035
grade            0.667434
sqft_above       0.605567
sqft_living15    0.585379
bathrooms        0.525138
view             0.397293
sqft_basement    0.323816
bedrooms         0.308350
lat              0.307003
waterfront       0.266369
floors           0.256794
yr_renovated     0.126434
sqft_lot         0.089661
sqft_lot15       0.082447
yr_built         0.054012
condition        0.036362
long             0.021626
id              -0.016762
zipcode         -0.053203
Name: price, dtype: float64
```

House_Price_Prediction notebook.ipynb

File Edit View Insert Runtime Tools Help Last saved at 4:21 PM

Comment Share Connect

+ Code + Text

```
clean_data = df.drop(columns=['id','date','zipcode', 'condition',
       'sqft_basement', 'yr_renovated', 'lat', 'long',
       'sqft_living15', 'sqft_lot15', 'sqft_lot', 'waterfront', 'view'])
clean_data.head()
```

	price	bedrooms	bathrooms	sqft_living	floors	grade	sqft_above	yr_built
0	221900.0	3	1.00	1180	1.0	7	1180	1955
1	538000.0	3	2.25	2570	2.0	7	2170	1951
2	180000.0	2	1.00	770	1.0	6	770	1933
3	604000.0	4	3.00	1960	1.0	7	1050	1965
4	510000.0	3	2.00	1680	1.0	8	1680	1987

Train Test Split

```
[ ] X = clean_data.drop(columns=['price'], axis =1)
    y = clean_data['price']
```

House_Price_Prediction notebook.ipynb

File Edit View Insert Runtime Tools Help Last saved at 4:21 PM

Comment Share Connect

+ Code + Text

Train Test Split

```
[ ] X = clean_data.drop(columns=['price'], axis =1)
    y = clean_data['price']

    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=42)
    len(X_train), len(X_test)
```

```
(15129, 6484)
```

```
[ ] X.columns
```

```
Index(['bedrooms', 'bathrooms', 'sqft_living', 'floors', 'grade', 'sqft_above',
       'yr_built'],
      dtype='object')
```

```
[ ] s_scale = StandardScaler() # scalling down the values to same scale.
    X_train = s_scale.fit_transform(X_train)
    X_test = s_scale.transform(X_test)
```


House_Price_Prediction notebook.ipynb

File Edit View Insert Runtime Tools Help Last saved at 4:21 PM

Comment Share Connect

+ Code + Text

```
lr = LinearRegression()
lr.fit(X_train,y_train)
print(lr.intercept_)
print(lr.coef_)
```

```
537640.1731773415
[ -44033.99031572   38751.54279952  187520.42409694   15866.3263269
  151016.85530385  -22261.14613432 -119025.43535643]
```

```
[ ] y_pred = lr.predict(X_test)
```

```
[ ] comparison = pd.DataFrame({"Actual":y_test, "Predicted":y_pred})
    comparison['Predicted'] = comparison['Predicted'].round(2)
    comparison.head(10)
```

	Actual	Predicted
735	365000.0	505568.30
2830	865000.0	680805.10
4106	1038000.0	1163620.01

House_Price_Prediction notebook.ipynb

File Edit View Insert Runtime Tools Help Last saved at 4:21 PM

Comment Share

+ Code + Text Connect

```
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print("Score (R^2):",lr.score(X_test,y_test))
```

```
MAE: 148417.41620190575
MSE: 56577855147.69438
RMSE: 237860.99963569982
Score (R^2): 0.6080957648411351
```

```
[ ] rf = RandomForestRegressor(n_estimators=100,criterion='squared_error',n_jobs=2)
    rf.fit(X_train,y_train)
```

```
RandomForestRegressor
RandomForestRegressor(n_jobs=2)
```

```
[ ] rf_predict = rf.predict(X_test)
```

```
[ ] X_test[0]
```

Activate Windows

House_Price_Prediction notebook.ipynb

File Edit View Insert Runtime Tools Help Last saved at 4:21 PM

Comment Share

+ Code + Text Connect

```
rf_predict = rf.predict(X_test)
```

+ Code + Text

```
[ ] X_test[0]
```

```
array([ 0.67921258,  0.18036675, -0.00393099,  0.92007506,  0.29547927,
        0.34287342,  0.50412404])
```

```
[ ] rf_predict[0]
```

```
445378.67
```

```
[ ] print('MAE:', metrics.mean_absolute_error(y_test, rf_predict))
    print('MSE:', metrics.mean_squared_error(y_test, rf_predict))
    print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, rf_predict)))
    print("Score (R^2):",rf.score(X_test,y_test))
```

```
MAE: 138402.20978379546
MSE: 52940823627.95685
RMSE: 230088.72990208984
Score (R^2): 0.6332888028640604
```

Activate Windows

House_Price_Prediction notebook.ipynb

File Edit View Insert Runtime Tools Help Last saved at 4:21 PM

Comment Share

+ Code + Text Connect

```
rf_r2 = r2_score(y_test, rf_predict)
```

```
[ ] print("Score (R^2):",rf_r2)
```

```
Score (R^2): 0.6332888028640604
```

```
[ ] pkl_scaler = pd.to_pickle(s_scale,"scale.pkl")
```

```
[ ] model = pd.to_pickle(rf,'model.pkl')
```

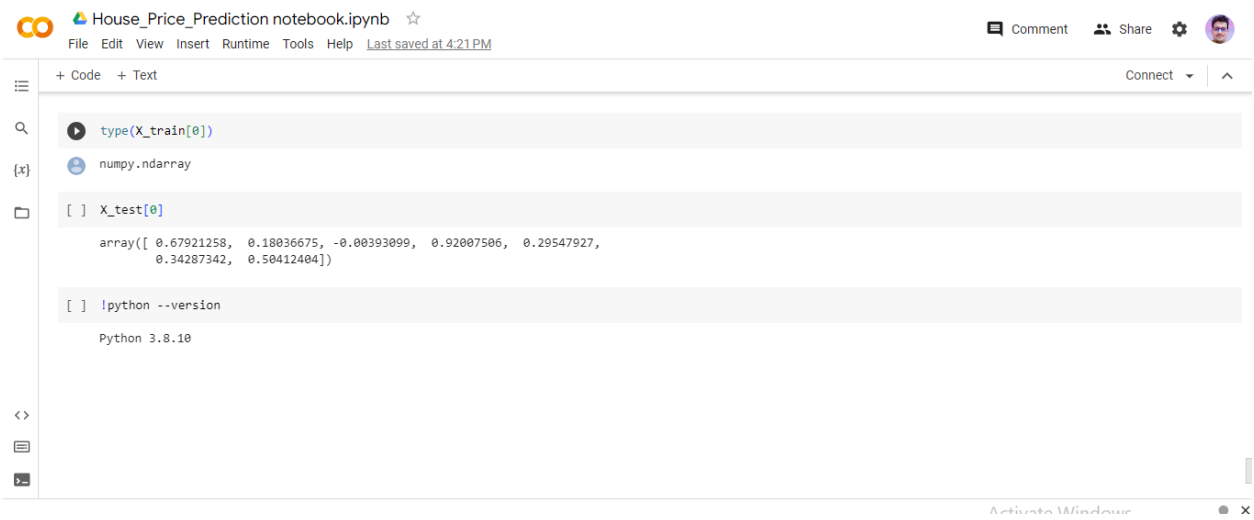
```
[ ] X_train[0]
```

```
array([0.67921258, 0.50673542, 0.66517989, 0.92007506, 1.14761535,
       1.08019571, 0.94685701])
```

```
[ ] type(X_train[0])
```

```
numpy.ndarray
```

Activate Windows



App.py

```
from flask import Flask, render_template, request
import numpy as np
import pickle

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))
scale = pickle.load(open('scale.pkl', 'rb'))


@app.route('/')
def index():
    return render_template('home.html')

@app.route('/show_index')
def show():
    return render_template('index.html')

@app.route('/aboutus')
def aboutus():
```

```
    return render_template('aboutus.html')


@app.route('/predict', methods=['GET', 'POST'])
def predict():
    val1 = request.form['bedrooms']
    val2 = request.form['bathrooms']
    val3 = request.form['floors']
    val4 = request.form['yr_built']
    val5 = request.form['sqft_living']
    val6 = request.form['sqft_above']
    val7 = request.form['grade']
    arr = np.array([[val1, val2, val3, val4, val5, val6, val7]])
    arr = scale.transform(arr)
    pred = model.predict(arr)

    return render_template('index.html', data=int(pred))


if __name__ == '__main__':
    app.run(debug=True)
```

HTML:

Index.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset='utf-8'>
    <meta http-equiv='X-UA-Compatible' content='IE=edge'>
    <title>House Price Prediction</title>
```

```
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <link rel='stylesheet' type='text/css' media='screen' href="{{ url_for('static',
filename='css/main.css') }}">
    <script src='main.js'></script>
</head>

<body>
    <div class="register_page">
        <div class="form">
            <form action="{{url_for('predict')}}" method="POST">
                Bedrooms: <input type="number" name='bedrooms' placeholder="Enter the no of
Bedrooms"><br>
                Bathrooms: <input type="number" name='bathrooms' placeholder="Enter the no of
Bathrooms"><br>
                Floors: <input type="number" name='floors' placeholder="Enter the no of
Floors"><br>
                Year of Build: <input type="number" name='yr_built' placeholder="Enter year of
Build"><br>
                Square ft: <input type="number" name='sqft_living' placeholder="Enter Square
ft"><br>
                Square ft above: <input type="number" name='sqft_above' placeholder="Enter Square
ft above"><br>
                Grade: <input type="number" name='grade' placeholder="Enter grade"><br>
                <input type="submit" value="predict" class="submit">
            </form>
            <p> House Price is: {{data}}</p>
        </div>

    </div>

</body>
```

```
</html>
```

Home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="../static/css/home.css">
</head>
<body>
    <div class="container">
        <div class="nav">
            <h1 class="logo">home<span>predict</span></h1>
            <ul>
                <li><a href="/aboutus">About Us</a></li>
                <li><a href="">Contact Us</a></li>
            </ul>
        </div>
        <div class="hero">
            <div class="text-content">
                <h1 class="title">Get your favorite house prediction in just few clicks</h1>
                <p class="desc">Experience the future of real estate with our house price prediction solution powered by advanced machine learning algorithms. With our solution, you can confidently navigate the buying, selling, or investment process, optimizing your opportunities for success. Trust in the power of machine learning to unlock the true potential of your real estate endeavors..</p>
                <a href = "/show_index" class="btn">Predict</a>
            </div>
```

```
        </div>

    </div>
</body>
</html>
```

Aboutus.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="../static/css/home.css">
</head>
<body>
    <div class="container">
        <div class="nav">
            <h1 class="logo">home<span>predict</span></h1>
            <ul>
                <li><a href="/aboutus">About Us</a></li>
                <li><a href="">Contact Us</a></li>
            </ul>
        </div>
        <div class="hero aboutus">
            <div class="text-content">
                <h1 class="title">About this project.</h1>
                <p class="desc">In our recent project focused on house price prediction, we explored several machine learning algorithms, including Lasso Regression, Linear Regression, and
```

Random Forest.After thorough experimentation and evaluation, we found that Random Forest yielded the most successful results. Its ability to handle both linear and non-linear relationships effectively, along with its ensemble nature, provided improved predictive accuracy compared to the other algorithms. Consequently, we have chosen Random Forest as our preferred algorithm for this task, confident in its ability to deliver reliable and accurate predictions for house prices..</p>

```
        </div>
      </div>
   </div>
</body>
</html>
```

CSS

Main.css

```
body{
background: cornflowerblue;
height: 100vh;
}
.register_page{
width: 360px;
margin: auto;
}
.form{
position: relative;
z-index: 1;
background: #ffff;
max-width: 360px;
margin: 0 auto 100px;
padding: 45px;
}
.form input{
```

```
font-family: 'Times New Roman', Times, serif;
outline: 1;
background: #f2f2f2;
width: 100%;
border: 0;
margin: 0 0 15px;
padding: 15px;
box-sizing: border-box;
font-size: 14px;
}
.submit{
font-size: 15px;
text-transform: uppercase;
font-weight: 600;
color: cornflowerblue;
}
```

Home.css

```
@import
url('https://fonts.googleapis.com/css2?family=Open+Sans:wght@300;400;500;700&display=sw
ap');

*{
   margin: 0;
   padding: 0;
   box-sizing: border-box;
}
body{
   font-family: 'Open Sans', sans-serif;
}
```

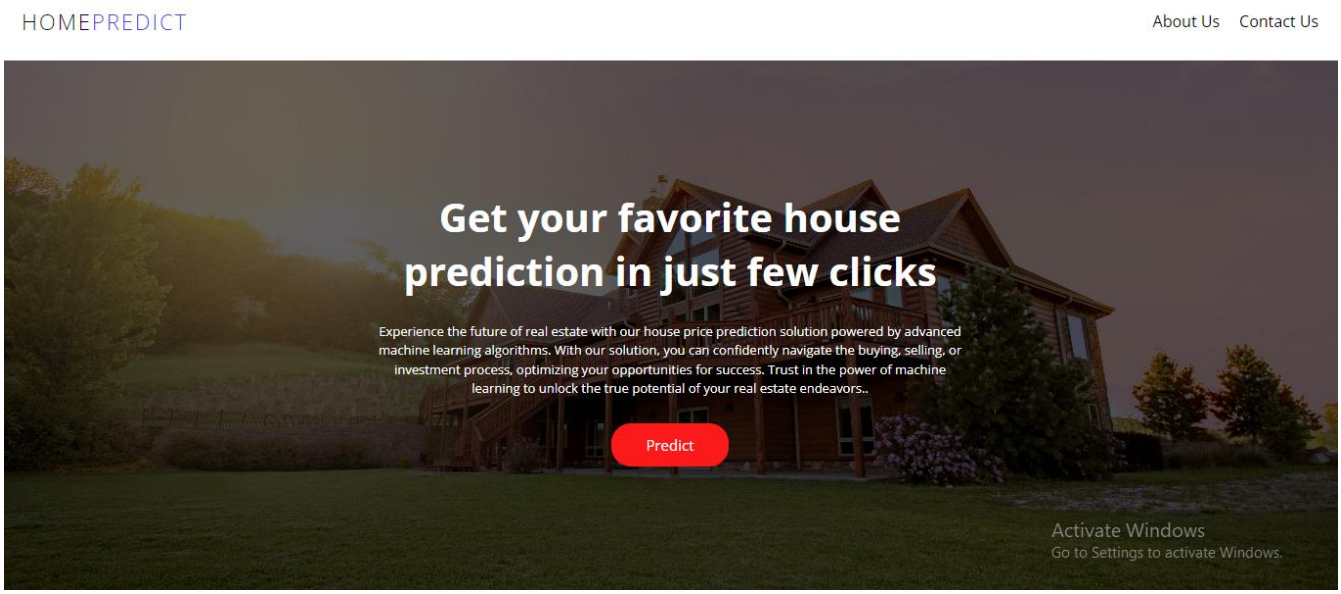
```
.nav{
    display: flex;
    align-items: center;
    justify-content: space-between;
    background-color: #fff;
    padding: 0 20px;
    height: 80px;
}
.logo{
    font-size: 22px;
    text-transform: uppercase;
    font-weight: 300;
    letter-spacing: 2px;
}
.logo span{
    color: rgb(69, 69, 238);
}

ul{
    display: flex;
    gap: 20px;
}
ul li{
    list-style: none;
}
ul li a{
    text-decoration: none;
    color: black;
}
.hero{
    height: calc(100vh - 80px);
```

```
    display: flex;
    align-items: center;
    justify-content: center;
    text-align: center;
    background-image: url("../images/bg.jpg");
    background-position: center;
    background-size: cover;
    background-color: rgba(0,0,0,0.6);
    background-blend-mode: multiply;
    color: white;
}

.hero .text-content{
    width: 600px;
}
.hero h1{
    font-size: 40px;
}
.hero .desc{
    margin: 25px 0;
    font-size: 13px;
    line-height: 1.5;
}
.hero .btn{
    display: inline-block;
    padding: 12px 15px;
    text-decoration: none;
    border: none;
    outline: none;
    border-radius: 20px;
    color: #fff;
```

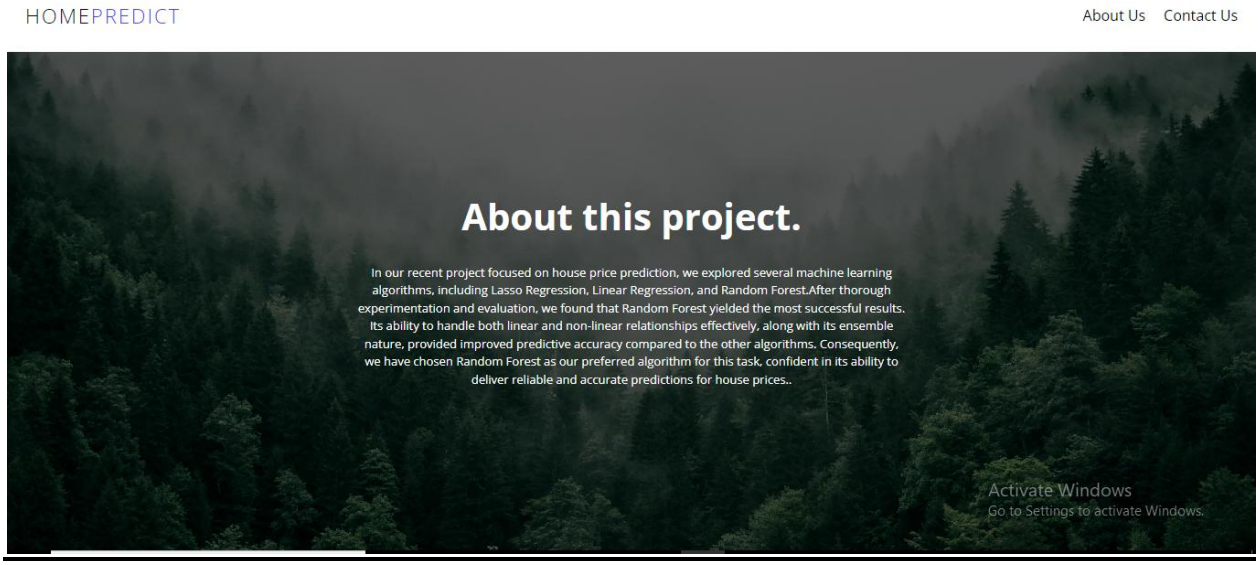
```
    background-color: rgb(253, 26, 26);
    min-width: 120px;
    text-align: center;
    cursor: pointer;
    font-size: 15px;
    /* margin-top: 25px; */
}
.hero .btn:hover{
    background-color: rgb(236, 25, 25);
}

.aboutus{
    background-image: url("../images/about-bg.jpg");
}
```

A-5: SCREENSHOTS:

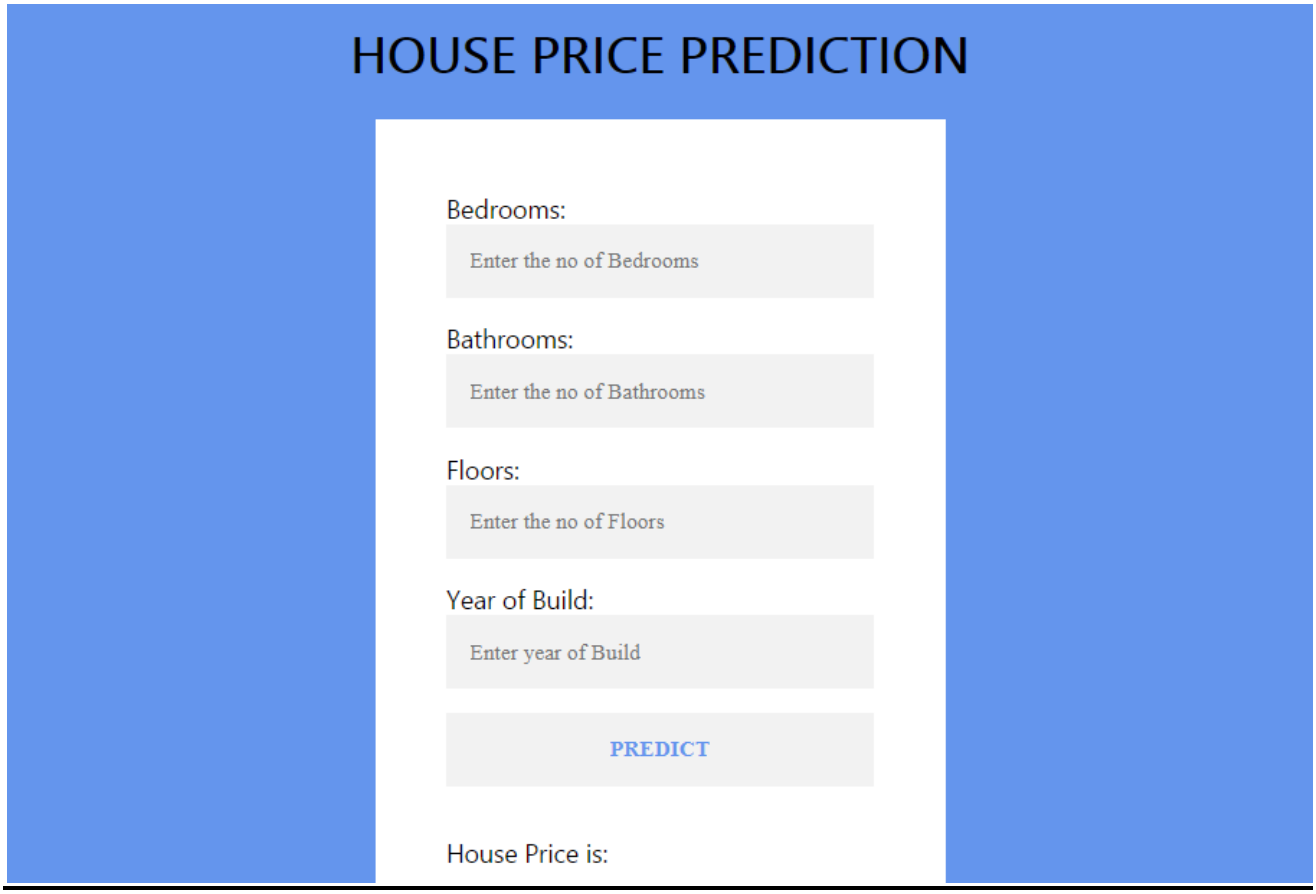
HOME:



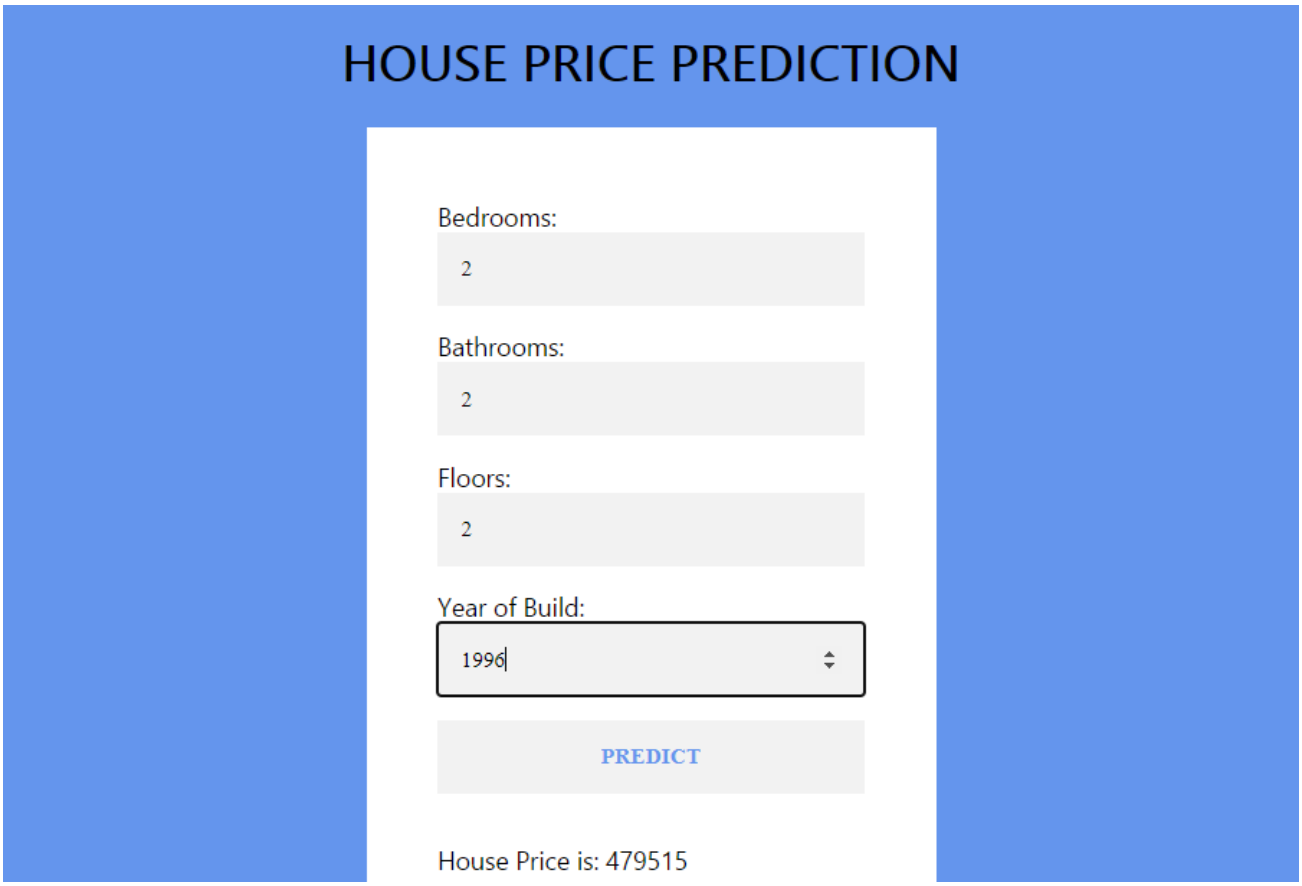
About us



Prediction


HOUSE PRICE PREDICTION
Bedrooms:
Enter the no of Bedrooms
Bathrooms:
Enter the no of Bathrooms
Floors:
Enter the no of Floors
Year of Build:
Enter year of Build
PREDICT
House Price is:

Prediction


HOUSE PRICE PREDICTION
Bedrooms:
2
Bathrooms:
2
Floors:
2
Year of Build:
1996
PREDICT
House Price is: 479515