NAME OF THE PROJECT

HOUSING PRICE PREDICTION

Submitted by:

DEEPAK KUMAR

Table of Contents

INTRODUCTION

Housing prices are an important reflection of the economy, and housing price ranges are of great interest for both buyers and sellers. In this project, house prices will be predicted given explanatory variables that cover many aspects of residential houses. The goal of this project is to create a regression model that are able to accurately estimate the price of the house given the features.

Business Problem Framing

You are required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

Conceptual Background of the Domain Problem

Housing prices are an important reflection of the economy, and housing price ranges are of great interest for both buyers and sellers. In this project, house prices will be predicted given explanatory variables that cover many aspects of residential houses. The goal of this project is to create a regression model that are able to accurately estimate the price of the house given the features.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

Generally speaking, machine learning projects follow the same process. Data ingestion, data cleaning, exploratory data analysis, feature engineering and finally machine learning.

The pipeline is not linear and you might find you have to jump back and forth between different stages. It's important I mention this because tutorials often make you believe the process is much cleaner than in reality. So please keep this in mind, your first machine learning project might be a mess.

Data Sources and their formats

The objective of the project is to perform data visualization techniques to understand the insight of the data. Machine learning often required to getting the understanding of the data and its insights. This project aims apply various Python tools to get a visual understanding of the data and clean it to make it ready to apply machine learning and deep learning models on it.

Data Pre-processing Done

Data Cleaning: However, we mustn't become lazy, there are always surprises in the data. We shouldn't skip the data cleaning phase, it's boring but it will save you hours of headache later on.

Duplicates & NaNs: I started by removing duplicates from the data, checked for missing or NaN (not a number) values. It's important to check for NaNs (and not just because it's socially moral) because these cause errors in the machine learning models.

Categorical Features: There are a lot of categorical variables that are marked as N/A when a feature of the house is nonexistent. For example, when no alley is present. I identified all the cases where this was happening across the training and test data and replaced the N/As with something more descriptive. N/As can cause errors with machine learning later down the line hence it's better to get rid of them.

Exploratory Data Analysis (EDA): This is where our data visualization journey often begins. The purpose of EDA in machine learning is to explore the quality of our data. A question to keep in mind is; are there any strange patterns that leave us scratching our heads?

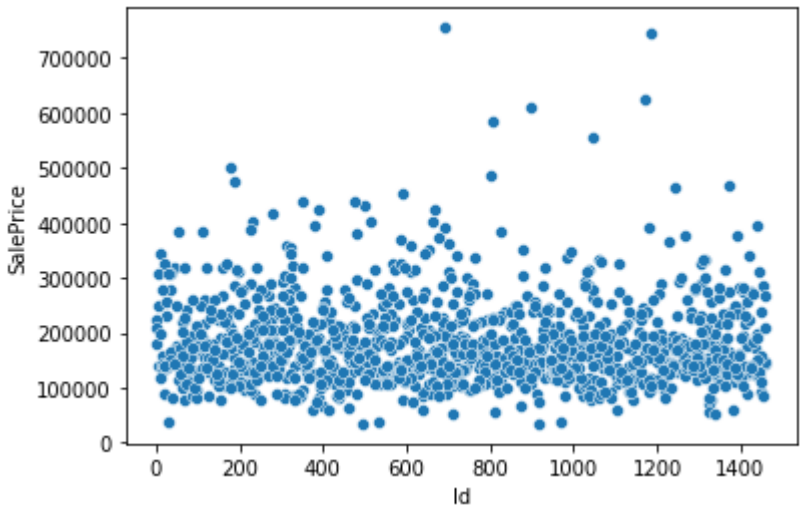


Figure gives us the scatter plot of the sale price. Most of the points are assembled on the bottom. And there seems to be no large outliers in the sale price variable

Outliers can have devastating effects on models that use loss functions minimizing squared error. Instead of removing outliers try applying a transformation.

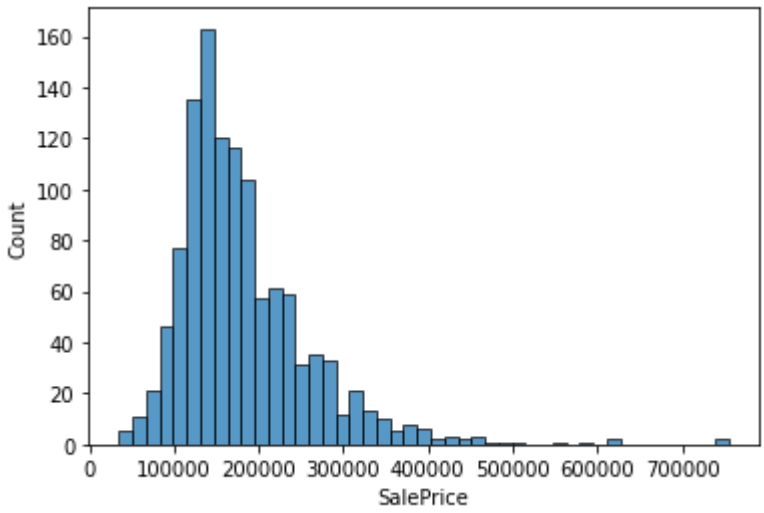


Figure shows that the distribution of sale prices are right skewed, which shows the distribution of the sale prices isn't normal. It is reasonable because few people can afford very expensive houses. I need to take transformation to the sale prices variable before model fitting.

Labels: I plotted sales price on a histogram. The distribution of sale prices is right skewed, something that is expected. In your neighborhood it might not be unusual to see a few houses that are relatively expensive.

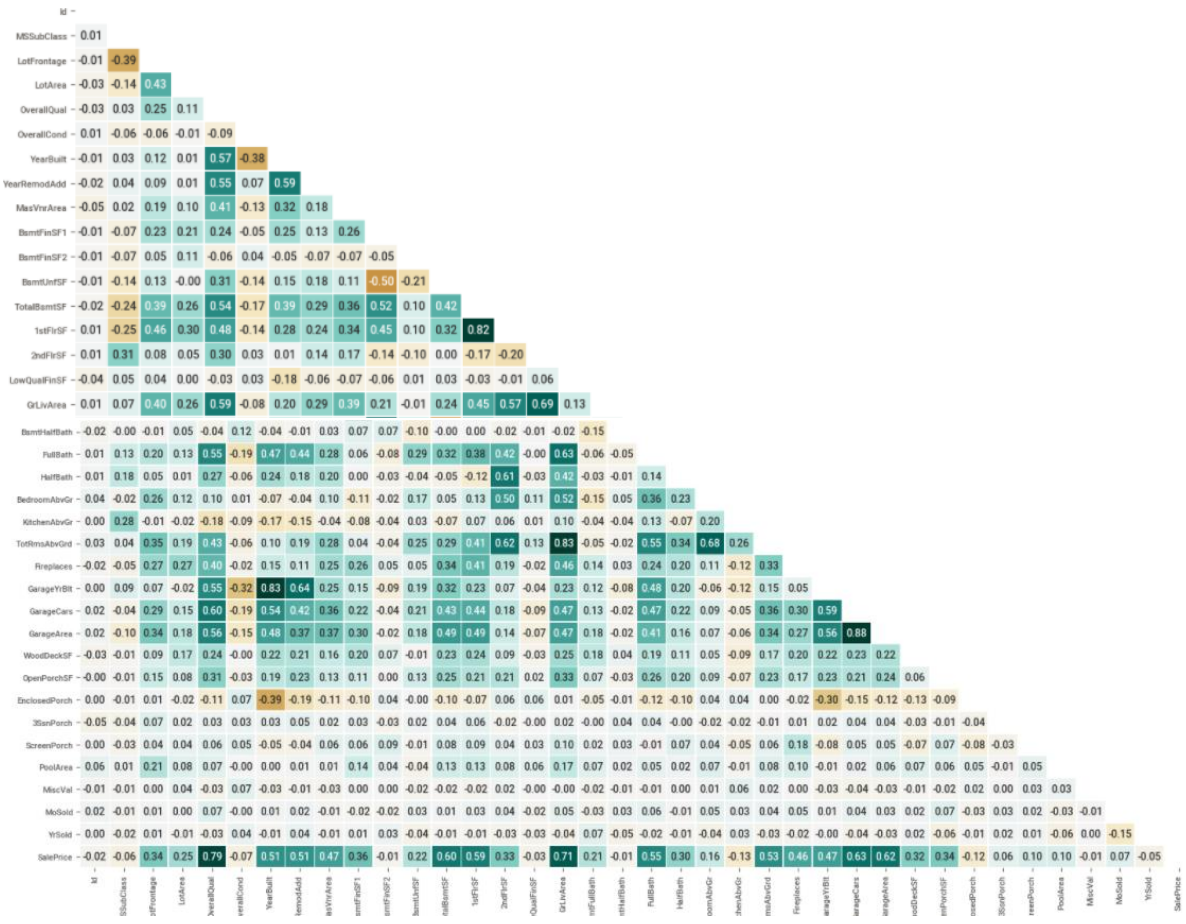
Here I perform my first bit of feature engineering (told you the process was messy). I'll apply a log transform to sales price to compress outliers making the distribution normal.

Data Inputs- Logic- Output Relationships

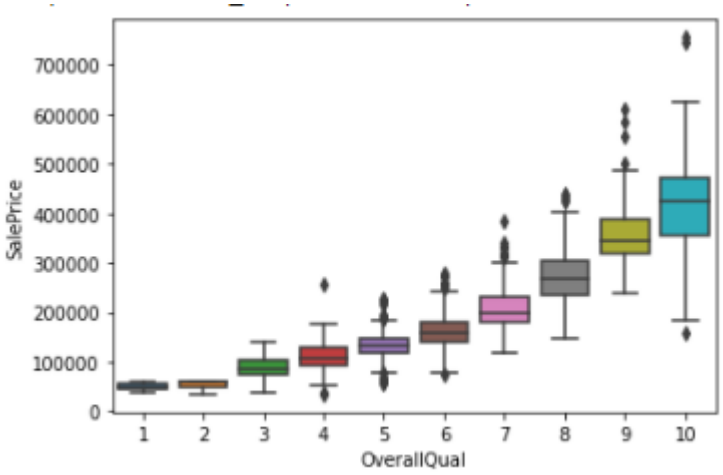
Correlations: It's often good to plot a correlation matrix to give you an idea of relationships that exist in your data. It can also guide your model building. For example, if you see a lot of your features are correlated with each other you might want to avoid linear regression. State the set of assumptions (if any) related to the problem under consideration.

The correlation measure used here is Pearson's correlation. In our case the lighter the square the stronger the correlation between two variables.

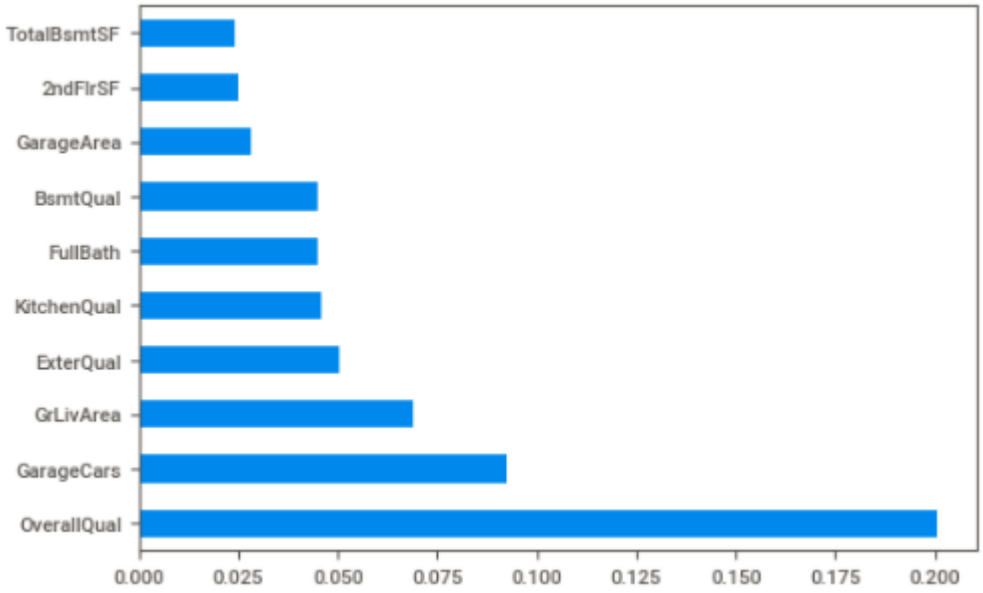
Features related to space such as lot frontage, garage area, ground living area were all positively correlated with sale price as one might expect. The logic being that larger properties should be more expensive. No correlations look suspicious here.



Categorical Relations: Sales price appears to be approximately normally distributed within each level of each category. No observations appear, untoward. Some categories contain little to no data, whilst other show little to no distinguishing ability between sales class. See full project on GitHub for data visualization.



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 wants 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 wants to improve the overall quality of his house from excellent to very excellent, it will be very difficult and costs he much money.



Get the importance of the variables from the correlations, we can get an overview of some important numeric variables such as the Overall quality and Above Grade (Ground) Living Area. Also, we can find some variables such as the Garage Car and the Garage Area have multicollinearity. However, I want to get more details about the importance of variables which including the factor variables. Therefore, I conduct a simple and quick random forest with only 100 trees and see the most 20 important variables.

Hardware and Software Requirements and Tools Used

Tools: I have used Python and Jupyter notebooks for the competition. Jupyter notebooks are popular among data scientist because they are easy to follow and show your working steps. Please be aware this code is not for production purposes, it doesn't follow software engineering best practices.

Libraries: These are frameworks in python to handle commonly required tasks. I Implore any budding data scientists to familiarize themselves with these libraries:

Pandas — For handling structured data

Scikit Learn — For machine learning

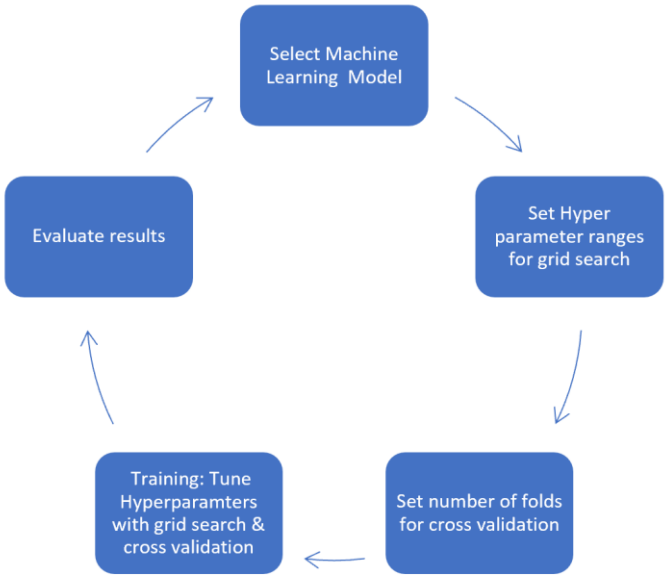
NumPy — For linear algebra and mathematics

Seaborn — For data visualization

Model/s Development and Evaluation

Testing of Identified Approaches (Algorithms)

I follow a standard development cycle for machine learning. As a beginner or even a pro, you'll likely have to go through many iterations of the cycle before you are able to get your models working to a high standard. As you gain more experience the number of iterations will reduce (I promise!).



Run and Evaluate selected models

Model Selection: As mentioned at the start of the article the task is supervised machine learning. We know it's a regression task because we are being asked to predict a numerical outcome (sale price).

Therefore, I approached this problem with three machine learning models. Decision tree, random forest and gradient boosting machines. I used the decision tree as my baseline model then built on this experience to tune my candidate models. This approach saves a lot of time as decision trees are quick to train and can give you an idea of how to tune the hyperparameters for my candidate models.

Model mechanics: I will not go into too much detail about how each model works here. Instead I'll drop a one-liner and link you to articles that describe what they do "under the hood".

> **Decision Tree** — A tree algorithm used in machine learning to find patterns in data by learning decision rules.
>
> **Random Forest** — A type of bagging method that plays on 'the wisdom of crowds' effect. It uses multiple independent decision trees in parallel to learn from data and aggregates their predictions for an outcome.
>
> **Gradient Boosting Machines** — A type of boosting method that uses a combination of decision tree in series. Each tree is used to predict and correct the errors by the preceding tree additively.

Random forests and gradient boosting can turn individually weak decision trees into strong predictive models. They're great algorithms to use if you have small training data sets like the one we have.

Training: In machine learning training refers to the process of teaching your model using examples from your training data set. In the training stage, you'll tune your model hyperparameters.

Before we get into further detail, I wish to briefly introduce the bias-variance trade-off.

> **Model Bias** — Models that underfit the training data leading to poor predictive capacity on unseen data. Generally, the simpler the model the higher the bias.

Model Variance — Models that overfit the training data leading to poor predictive capacity on unseen data. Generally, the more complexity in the model the higher the variance.

Complexity can be thought of as the number of features in the model. Model variance and model bias have an inverse relationship leading to a trade-off. There is an optimal point for model complexity that minimizes the error. We seek to establish that by tuning our hyper parameters.

Here's a good article to help you explore this stuff in more detail.

Hyperparameters: Hyperparameters help us adjust the complexity of our model. There are some best practices on what hyperparameters one should tune for each of the models. I'll first detail the hyperparameters, then I'll tell you which I've chosen to tune for each model.

max_depth — The maximum number of nodes for a given decision tree.

max_features — The size of the subset of features to consider for splitting at a node.

n_estimators — The number of trees used for boosting or aggregation. This hyperparameter only applies to the random forest and gradient boosting machines.

learning_rate — The learning rate acts to reduce the contribution of each tree. This only applies for gradient boosting machines.

Decision Tree — Hyperparameters tuned are the max_depth and the max_features

Random Forest — The most important hyperparameters to tune are n_estimators and max_features [1].

Gradient boosting machines — The most important hyperparameters to tune are n_estimators, max_depth and learning_rate [1].

Grid search: Choosing the range of your hyperparameters is an iterative process. With more experience you'll begin to get a feel for what ranges to set. The good news is once you've chosen your possible hyperparameter ranges, grid search allows you to test the model at every combination of those ranges. I'll talk more about this in the next section.

Cross validation: Models are trained with a 5-fold cross validation. A technique that takes the entirety of your training data, randomly splits it into train and validation data sets over 5 iterations.

You end up with 5 different training and validation data sets to build and test your models. It's a good way to counter overfitting.

More generally, cross validation of this kind is known as k-fold cross validation. More on k-fold cross validation here.

Implementation: Scikit Learn helps us bring together hyperparameter tuning and cross validation with ease in using GridSearchCV. It gives you options to view the results of each of your training runs.

Evaluation: This is the last step in the process. Here's where we either jump with joy or pull our hair with frustration (just kidding, we don't do that…ever). We can use data visualisation to see the results of each of our candidate models. If we are not happy with our results, we might have to revisit our process at any of the stages from data cleaning to machine learning.

CONCLUSION

In this study, our models are trained with 18-year of housing property data utilising stochastic gradient descent (SGD) based support vector regression (SVM), random forest (RF) and gradient boosting machine (GBM). We have demonstrated that advanced machine learning algorithms can achieve very accurate prediction of property prices, as evaluated by the performance metrics. Given our dataset used in this paper, our main conclusion is that RF and GBM are able to generate comparably accurate price estimations with lower prediction errors, compared with the SVM results.

First, our study has shown that advanced machine learning algorithms like SVM, RF and GBM, are promising tools for property researchers to use in housing price predictions. However, we must be cautious that these machine learning tools also have their own limitations. There are often many potential features for researchers to choose and include in the models so that a very careful feature selection is essential.

Second, many conventional estimation methods produce reasonably good estimates of the coefficients that unveil the relationship between output variable and predictor variables. These methods are intended to explain the real-world phenomena and to make predictions, respectively. They are used for developing and testing theories to perform causal explanation, prediction, and description (Shmuell, 2010). Based on these estimates, investigators can interpret the results and make policy recommendations. However, machine learning algorithms are often not developed to achieve these purposes. Although machine learning can produce model predictions with tremendously low errors, the estimated coefficients (or weights, in machine learning terminology) derived by the models may sometimes make it hard for interpretation.

Third, the computation of machine learning algorithms often takes much longer time than conventional methods such as hedonic pricing model. The choice of algorithm depends on consideration of a number of factors such as the size of the data set, computing power of the equipment, and the availability of waiting time for the results. We recommend property valuers and researchers to use SVM for making forecasts if speed is a primary concern. When predictive accuracy is a key objective, RF and GBM should be considered instead.

To conclude, the application of machine learning in property research is still at an early stage. We hope this study has moved a small step ahead in providing some methodological and empirical contributions to property appraisal, and presenting an alternative approach to the valuation of housing prices. Future direction of research may consider incorporating additional property transaction data from a larger geographical location with more features, or analysing other property types beyond housing development.

Using different methods that match the time-series data will be used in the future research to obtain smaller error prediction values and using more data to get the better result.