Zach Simmons

Corey Smith

Final Project Report

Introduction

House price prediction could help the average home buyer or seller understand what each home is worth. The Kaggle competition: House Prices – Advanced Regression Techniques aims to develop models that can predict the sale price of a home from a set of 79 explanatory variables. The data was provided by Kaggle and includes the data of 2920 homes in Ames, Iowa split into a training set and a test set. In order to increase the complexity of the project, we decided to utilize two different models, LightGBM and RandomForestRegressor. After the data exploration and cleaning processes were completed, the models achieved a root mean squared error (RMSE) value of approximately $37000 and $30000 respectively. Additionally, the average runtimes for training the models were around 22 seconds for LightGBM and around 2 seconds for RandomForestRegressor.

Problem Statement and Data Description

With housing prices increasing at alarming rates around the country, a way to predict how prices will continue in the future would be lifesaving for new renters and home buyers. Our original idea was to develop a mechanism for determining the areas most impacted by price increases and potentially discover the attributes that cause it. However, the problem was too complex, and it would be difficult to find or create a usable dataset. So, our group decided to base our project on the Kaggle competition: House Prices – Advanced Regression Techniques [1].

This competition aimed to analyze a dataset for houses in Ames, Iowa and included 79 explanatory variables including lot area, number of bedrooms, and garage size. A regression model trained on the data would be tasked with predicting the eventual sale price of the house based on the numerous parameters. Since this was a Kaggle competition, the data was in a highly usable format and already separated into a training set with all of the explanatory variables and the sale price as well as a test set which lacked the sale prices.

Approaches, Algorithms, and tools

We decided to utilize tools we had the most familiarity with from our lab work in the class, namely pandas in python to process and work with the data and Jupyter notebooks to house our work. Kaggle also has a self-contained kernel to run and maintain your notebooks. To begin our work on the project, we began the data exploration process using a few tools within pandas. We used the built-in data frame functionality to remove any rows with a NaN value for the sale price in the train dataset and devised a method for handling NaN values for the wide range of variables used in both the train and testing data.

For numeric variables where NaN indicated that a house did not have the requisite component being described, such as GarageCars, or GarageArea, we simply replaced the null with 0. Additionally, for numerical data that possessed NaN values, but appeared to indicate an error or missing value, we replaced the null value with the mode of the column so that a 0 would not overly skew the regression based on that column. For categorical data, most of the NaN values could be attributed to the requisite feature not being in the house, such as a garage or basement, so replacing these null values with the mode would not be correct. In order to preserve the affect of the features on the regression, we replaced the null values with 'Missing'.

Another tool we utilized for data exploration was Seaborn. Seaborn allowed us to create a heatmap for the correlation matrix of the data and visually inspect for highly correlated variables. Specifically, we determined that GarageArea and OverallQual as well as OverallQual and YearRemodAdd were two pairs of highly correlated variables.

On the topic of data cleaning, this dataset had 42 of its 79 explanatory variables as categorical variables. In order for the models to operate correctly, we needed to devise a way to deal with these values. We came up with two options: LabelEncoder and One Hot Encoding. LabelEncoder, which we used with our LightGBM model assigned all objects of type 'object' a numerical value within their respective columns. This basically converts categorical values into a factor of numeric values and allowed LightGBM to fit the data. One Hot Encoding, which we used on our RandomForestRegressor model executes a similar task but makes each column a vector of all possible categorical values, where if the cell had that value, it would have a value of 1, or a 0 otherwise. This drastically increases the number of columns for the dataset but did not have an effect on runtime.

We decided to analyze two different models for the project and compare their performances. LightGBM is a machine learning algorithm, specifically a gradient boosting decision tree, and was designed to be more efficient and scalable than previous implementations [2]. Random Forest Regressor also utilizes decision trees, but specifically bags the data and trains each tree on a different sample of the data to increase accuracy and reduce over-fitting [3]. Comparing these models was done using cross-validation as well as the RMSE which was used as the evaluation metric for the Kaggle competition.

Source Code

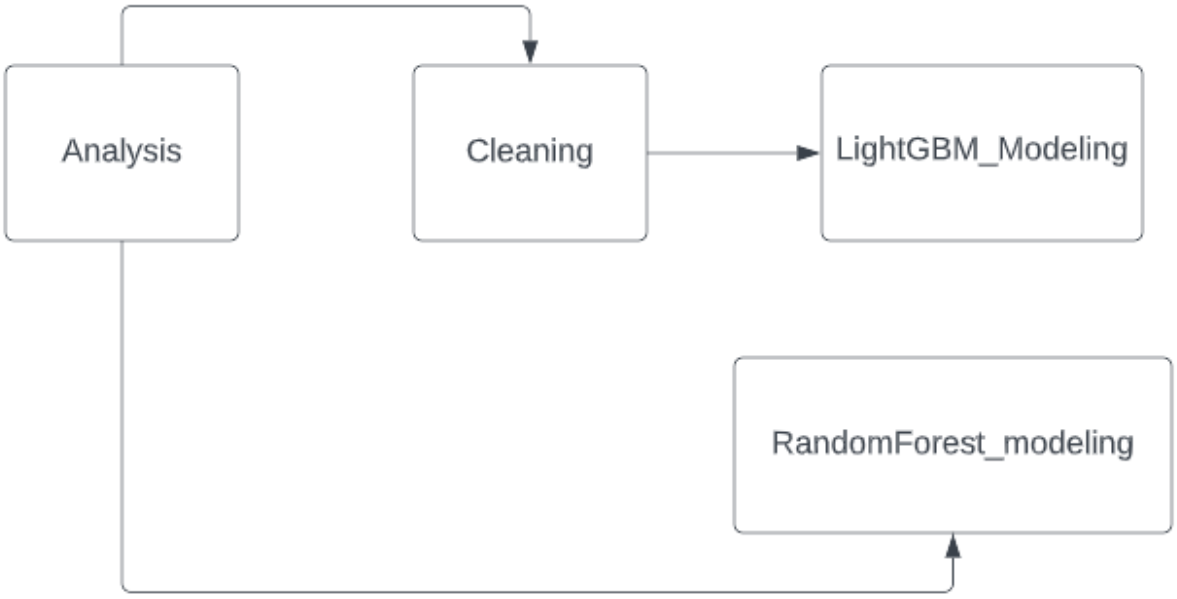
The source code is structured into three main sub-sections: Analysis, Cleaning, and Modeling. Each sub-section is its own Jupyter Notebook file written in Python (.ipynb). Modeling is split into two notebooks, with one handling LightGBM and the other handling RandomForest.

The analysis.ipynb covers all code involved in the preliminary analysis of the features. To understand how to modify the data in the cleaning phase, it was important to understand the data and how it is being presented.

Cleaning includes all the steps that are necessary to clean the data and prepare it for the machine learning algorithms. The cleaning phase uses information from the analysis, such as null values, and data type of the data, but does not directly use any of the analysis phase code and is independent as a notebook. The output of this notebook will create a train and test dataset in .csv format.

Modeling has two notebooks, and LightGBM is dependent on the cleaning phase. To use LightGBM_modeling.ipynb, the cleaning notebook must run at least once to generate clean data csv

files. Both LightGBM_modeling and RandomForest_modeling include code for fitting the model, cross-validation, and training. RandomForest_modeling includes cleaning steps.



Source Code flow chart

Metrics and Evaluation

Metrics for the goodness of the model are the comparison of our model to others in the Kaggle challenge, cross-validation scoring, and RSME (Root Square Mean Error) of the predictive prices against the training data's provided sale price data. To better understand the RMSE, an output of the predicted prices is provided as a data frame in the LightGBM_modeling code. The approach for evaluation comes in two steps: analysis, and score.

The main goal of analysis in the evaluation was to understand the data and answer questions such as: What was a strong variable (housing feature), and what wasn't? What are the data types? Does the data have null values? Are housing features correlated with each other?

The analysis utilized multivariate and univariate comparisons of the data to conclude that there are correlations between features. Correlation matrices and heat maps were also generated to format a generalization of feature importance. A categorical feature analysis was performed by separating all data of the type 'object' into a data set using the .select_dtypes() function and pulling out those that represented categorical data. A null feature analysis was also done by searching using the Python function .isnull().

For scoring our models, cross-validation was performed. This evaluated the fit of each model and identified how well it should perform on the clean data.For LightGBM, our evaluation metric was an R-squared. This was calculated using lgb.booster() to create a framework of the model for validation.

This framework was then tested against predictions on the base data sets train, and test, and the result was given an R-squared valuation. In RandomForest cross_val_score() is used to generate a mean squared error against the sale prices.

Results and Reflection

The results of both models were close in terms of RSME. LightGBM and RandomForest both have margins of error and do not predict prices to exact real-life values. RandomForest, for our data set, tended to perform more accurately with its best training being slightly above $30,000 off of the training data's sale prices. LightGBM was slightly higher at ~$37,000. The RSME cross-validation for RandomForest confirmed accuracy of ~$29,000, so the model had a good fit and the training of the data agreed with each other. LightGBM had an R-squared of 0.76, since the best fit is at an R-squared of 1.0, the expected results were to be off by some margin, but not a significant amount. The predictions are done by the model after training agreed with this behavior.

This project was our first-time using machine learning. A large challenge came in learning and understanding the way algorithms are set up. How to clean, process, and validate models before using them became extensive time-consuming lessons. Cleaning data is the most important step and the bulk of machine learning. Once data was clean, models had to be tested for their usefulness and when training could finally happen, understanding the output and visualization of it became another challenge. A lesson learned for future machine learning projects comes in the importance of finding out as much about your data as possible. Machines interpret data in a concrete way, and the dataset must be prepared for interpretation by being conclusively researched. For this project, more research could be done into the variables in the future. There may be more optimal ways of associating the data with the two model's parameters that give a lower mean squared error and a higher R-squared. This is why this project is suitable for Kaggle as a community will come up with different approaches and ideas and can collaborate on what is best.

References:

[1] Kaggle.com, "House Prices - Advanced Regression Techniques" Web: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques, last accessed: 24 April 2022.

[2] LightGBM, "LightGBM" Web: https://lightgbm.readthedocs.io/en/latest/, last accessed: 26 April 2022.

[3] scikit-learn.org, "sklearn.ensemble.RandomForestRegressor," Web: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html, last accessed: 27 April 2022.