

Individual Programming Assignment Report: House Price Prediction with MLFlow

Author: Angus Ferrell

Setup

The dataset and requirements was placed in the GitHub repository:

<https://github.com/angusmf1/HousePricePrediction>

The repository was cloned to my local machine. I then set up and activated a virtual environment to use MLFlow and the rest of the code.

Data Collection/Processing

The dataset chosen for training the models and making predictions was the Kaggle Housing Prices Data, found at <https://www.kaggle.com/datasets/yasserh/housing-prices-dataset>. This dataset is summarized in the accompanying datasheet, found at https://github.com/angusmf1/HousePricePrediction/blob/main/Housing_Prices_Datasheet.docx.

The Kaggle description detailed that the dataset possessed strong multicollinearity among its features, which indicated that the features would need to be analyzed before training the model. The dataset itself was complete – no null/missing values – and contained uniform column types. The data did not require any significant cleaning. The categorical features were converted to integer values using One Hot Encoding(OHE) . Following this, I took two separate approaches with processing the data.

Process 1: The first approach was to use the OHE training data and all initial features as it was.

Process 2: In order to identify the features with strong collinearity, the variance inflation factor (VIF) was calculated for each feature besides 'price'. Any feature with a VIF greater than 5 was removed, leaving the features {'area', 'bedrooms', 'bathrooms', 'stories', 'parking'}. This seems reasonable, as these features normally have the largest influence on housing prices. However, going from 12 to 5 features opened the door to significant information loss.

After both of these processes, the data was split into training and test sets. This completed preprocessing the data was ready to train the models.

Model Training

I choose five models to train and test:

1. Linear Regression
2. Ridge Regression
3. Dabl Simple Regressor
4. Random Forest Classifier
5. LightGBM Model

Linear regression and Ridge regression are typically good initial models to try on a dataset before increasing model complexity. Dabl is a autoML package that is designed to automate machine learning and model selection. The simple regressor tests several different models and hyperparameters, then selects the best one based on mean squared error performance. It is another good starter model to try on the data. Next, I upped the complexity with the random forest regressor, which also added in an ensemble approach. Lastly, I implemented the LightGBM model so that I had also tried a gradient boosting approach as well.

Initially, I took the straightforward approach of training the models without performing any hyperparameter tuning. Next, I used grid search for hyperparameter turning for each model and then compared the results. Surprisingly, the simple approach produced better results. This signifies that I was possibly tuning the wrong hyperparameters, or that the range of hyperparameters I was giving the model for tuning was not optimal. For this reason, I decided to move forward with the simple approach.

Model Performance

Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2) were used to test model performance, which were also used in the model comparisons. I trained and tested all five models using both data processing approaches: Dataset 1 and Dataset 2. Dataset 1 performed better across the board.











Models	Metrics			Tags
	mae	r2	rmse	Training Info
 Ridge_Regr.../11	1127441.5...	0.5464817...	1514047.5...	Dataset 2
 LGB_Model/12	1149333.2...	0.5491045...	1509663.1...	Dataset 2
 Random For.../11	1160912.9...	0.4772450...	1625515.8...	Dataset 2
 Dabl simpl.../15	1123723.2...	0.5512364...	1506089.9...	Dataset 2
 Linear Reg.../11	1127483.3...	0.5464062...	1514173.5...	Dataset 2
 Ridge_Regr.../10	969600.58...	0.6528547...	1324639.5...	Dataset 1
 LGB_Model/11	1022791.0...	0.6188732...	1387959.6...	Dataset 1
 Random For.../10	999124.04...	0.6319306...	1363976.6...	Dataset 1
 Dabl simpl.../14	948929.24...	0.6502095...	1329676.8...	Dataset 1
 Linear Reg.../10	970043.40...	0.6529242...	1324506.9...	Dataset 1

Figure 1: MLFlow Housing Prices Prediction Metrics

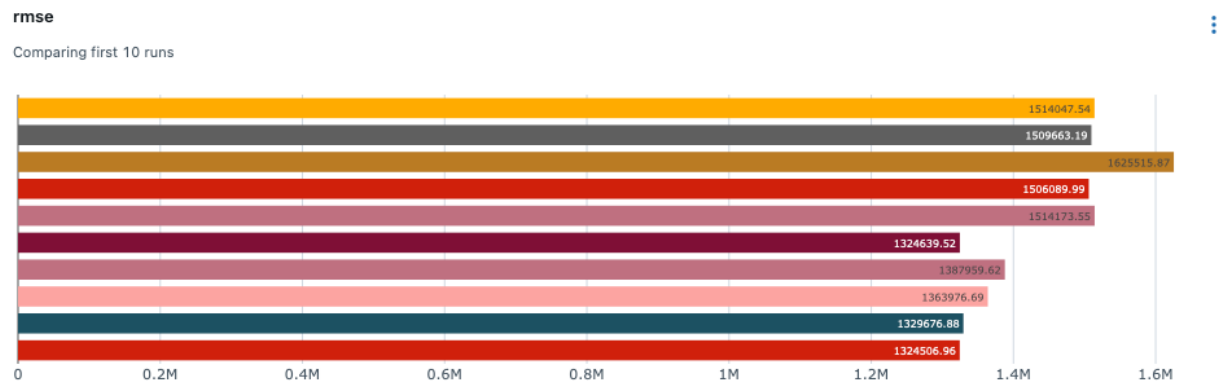


Figure 2: MLFlow Housing Prices Prediction Graphical Results (RMSE)

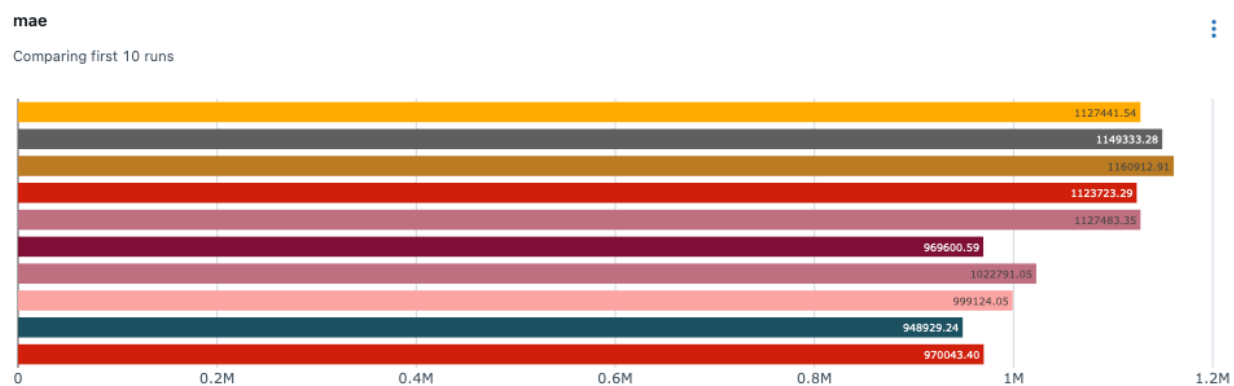


Figure 3: MLFlow Housing Prices Prediction Graphical Results (MAE)

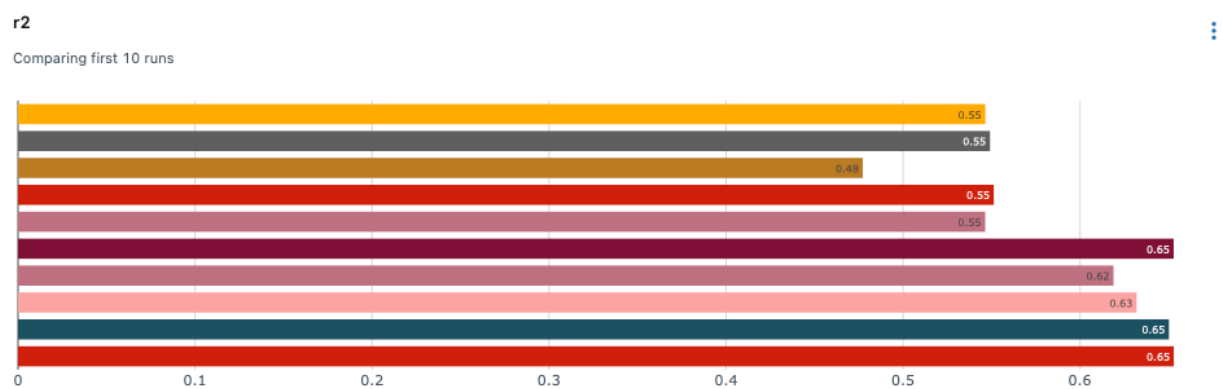


Figure 4: MLFlow Housing Prices Prediction Graphical Results (R^2)

Dataset 1			
Model	RMSE	MAE	R ²
Linear Regression	1.324507e+06	9.700434e+05	0.652924
Dabl Simple Regressor	1.329677e+06	9.489292e+05	0.650210
Random Forest	1.363977e+06	9.991240e+05	0.631931
LightGBM	1.387960e+06	1.022791e+06	0.618873
Ridge Regression	1.324640e+06	9.696006e+05	0.652855

Dataset 2			
Model	RMSE	MAE	R ²
Linear Regression	1.514174e+06	1.127483e+06	0.546406
Dabl Simple Regressor	1.506090e+06	1.123723e+06	0.551236
Random Forest	1.625516e+06	1.160913e+06	0.477245
LightGBM	1.509663e+06	1.149333e+06	0.549105
Ridge Regression	1.514048e+06	1.127442e+06	0.546482

Figure 5: Dataset Metrics Comparison

Figure 5 depicts that removing 7 features did lead to information loss, which led to larger error and poorer performance from the models trained on Dataset 2. Overall, the best model was Linear Regression with $RMSE = 1324506$ and $R^2 = 0.652924$, with Ridge Regression coming in a close second place with $RMSE = 1324639$ and $R^2 = 0.652855$. The MAE scores give the edge to the Dabl Simple Regressor, with Linear Regression and Ridge Regression coming in close second and third. In this experiment, the simple models performed better than the more complex models, with the LightGBM model achieving the lowest performance. I believe that if the more complex models were tuned optimally (correct selection of hyperparameter ranges), they would outperform the simple models. However, for now the edge goes to both Linear Regression and Ridge Regression.

Model Comparison

The first model that was chosen for comparison was the top performers of Zillow's Home Value Prediction (Zestimate) competition, Abdelwahed Assklou:

<https://www.kaggle.com/code/abdelwahedassklou/only-cat-boost-lb-0-0641-939>.

While only the python script was provided and the Kaggle Score does not directly match with my model metrics for performance, it is still useful to compare models. The model used here was an ensemble of CatBoost models, a decision-tree gradient boosting model, which is similar to the LightGBM model that I used. The dataset that this model used had some significant differences from the Kaggle Housing Prices dataset, specifically in regards to the number of and type of features that include temporal data, taxes, location, etc. Therefore, a direct comparison between the our models cannot be made, but can be inferred. This model underwent extensive data cleaning/preprocessing, however, the hyperparameters appear to have been tuned in an unseen prior step. The richer dataset, extensive preprocessing, ensemble of 5 CatBoost models and the assumed-to-be-tuned hyperparameters stand out as to why this model was so successful, and safe to say better than my models.

The second model that was chosen for comparison is the LilHomie Price Prediction Model by Vivek Pandey from the GitHub repository on house price prediction models, located at <https://github.com/Viveckh/LilHomie>. This prediction model utilized both linear regression and random forest models for training and predictions. This model used Root Mean Squared Error (RMSE) between the actual housing prices and predicted housing prices as its performance metric. The dataset that this model used again had significant differences from the Kaggle Housing Prices dataset, specifically in regards to the number of and type of features. This model's dataset also included temporal data, taxes, location, etc. Therefore, this model also cannot be directly compared to my models, but the comparison is still useful. The random forest model achieved a $RMSE = 73832$ and the linear regression model achieved a $RMSE = 90803$. Both values are two orders of magnitude lower than my model scores, significantly outperforming them. The models used here are similar to the models used in my predictions, however the training datasets were very different and extensive hyperparameter tuning was done. The richer dataset, extensive preprocessing, and tuned hyperparameters stand out as to why this model was so successful and outperformed my models.

In summary, both of these models did extensive data preprocessing on richer datasets and in-depth hyperparameter tuning. These are the largest reasons why both models outperformed my models. However, my best models in Dataset 1 had acceptable performance, with R^2 values above 0.65.

Docker

My code and dependencies were containerized and pushed to DockerHub.

The name of the repository: aferrell/predict

The screenshot shows the Docker Hub interface for the repository `aferrell/predict`. The top navigation bar includes the Docker Hub logo, links to Explore, Repositories, and Organizations, a search bar, and user account options. The breadcrumb trail indicates the path: `aferrell` / `Repositories` / `predict` / `General`. The repository page has tabs for General, Tags, Builds, Collaborators, Webhooks, and Settings. The `General` tab is active, showing the repository name, a description, and a Docker command. Below this, there are sections for Tags and Automated Builds.

aferrell / predict

Description
Machine Learning code for predicting housing prices

⌚ Last pushed: about 1 hour ago

Docker commands [Public View](#)
To push a new tag to this repository:
`docker push aferrell/predict:tagname`

Tags
This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	44 minutes ago	an hour ago

[See all](#)

Automated Builds
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.
Available with Pro, Team and Business subscriptions. [Read more about automated builds](#) .


[Upgrade](#)

MLFlow Project Registry

mlflow 2.10.0 Experiments **Models**

Registered Models

Filter registered models by name or tags ⓘ 🔍

Name 	Latest version
Dabl simple regressor	Version 15
LGB_Model	Version 12
Linear Regression Model	Version 11
Random Forest Model	Version 11
Ridge_Regression_Model	Version 11

The link to the project registry:

Run the code (docker container). Replace <your_machine_ip> with your machine's IP.

http://<your_machine_ip>/#/experiments/859833248773557952?searchFilter=&orderByKey=attributes.start_time&orderByAsc=false&startTime=ALL&lifecycleFilter=Active&datasetsFilter=W10%3D&modelVersionFilter=All%20Runs&selectedColumns=attributes.%60Source%60,attributes.%60Models%60,metrics.%60rmse%60&compareRunCharts=W3sidXVpZCI6IjE3MDc1MzlyNzkyNzdwZTd6cGFxZCIsInR5cGUiOiJCQVliLCJydW5zQ291bnRUb0NvbXBhcmUiOiEwLCJkZWxldGVkljpmYWxzZSwiaXNHZW5lcmF0ZWQiOnRydWUsIm1ldHJpY0tleSI6Im1hZSJ9LHsidXVpZCI6IjE3MDc1MzlyNzkyNzdwMmg1Zmo1YiIsInR5cGUiOiJCQVliLCJydW5zQ291bnRUb0NvbXBhcmUiOiEwLCJkZWxldGVkljpmYWxzZSwiaXNHZW5lcmF0ZWQiOnRydWUsIm1ldHJpY0tleSI6InlyIn0seyJ1dWlkljoiMTcwNzUzMjl3OTI3NzdkNXJqcHZ3liwidHlwZSI6IkJBUiIsInJ1bnNDb3VudFRvQ29tcGFyZSI6MTAsImRlbGV0ZWQiOmZhbnHNILCJpc0dlbmVyYXRIZCI6dHJ1ZSwibWV0cmIjS2V5Ijoicm1zZSJ9XQ%3D%3D