Ames housing dataset price prediction

William Pedersen & Tommy Tran / Gruppe 4 13.11.2022

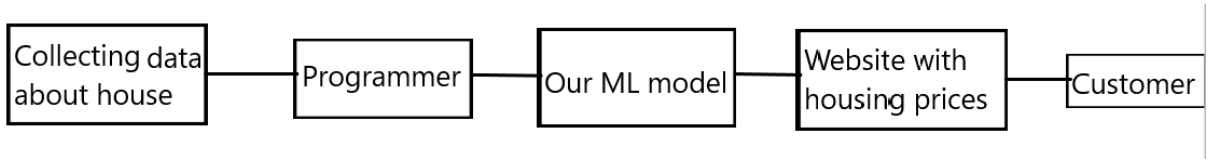
DESCRIBE THE PROBLEM

SCOPE

The goal of this project is to predict the sale price of a house in Ames Iowa. This will for instance greatly impact both time and expenses that real estate agents use to estimate houses manually.

When the solution is produced it will be used by real estate agencies and people that want to quickly predict their house value without the hassle. Real estate agents and appraisers are still used. You pay them and they take their sweet time to give a good estimate. If I were to estimate housing prices manually I would either pay a real estate agent to do the job for me, or look at the prior pricings and estimate the price myself.

if our model were to be used as part of a bigger system it could look something like this:



The flow starts from left to right:

- Someone will start by collecting data about the house or the data already exist and it is then sent forth to the next part of the “pipeline”.
- The programmers job is to make sure that the data is injected into the machine learning model
- Our ML model will predict a price on a house
- it can then be used for various things, and is in this scenario used in a website with housing prices
- Lastly, a customer will check the pricings of the houses predicted by our ML model.

In this “extensive” system a small change in one of the parts can negatively impact each other. For instance:

- Maybe some data sent from the data collector could be too hard for the ML model to handle.
- Or that some data that already exist is sent, but the data is too old and the house is completely different from the housing data actually sent. This can impact the customer who might be fooled that a new house is cheaper than it actually is.

Here is the timeline for how our project went:

1. Understanding the problem and seeing the “big picture”

2. start writing some keywords in the short report
3. Started by getting the data
4. We started exploring the data and getting familiar with it by creating plots and pictures to understand it better
5. We start training the base models and getting some scores
6. We make more models and use more hyperparameters on them to get better scores
7. We need to go forth with a advanced regression technique: stacking ensemble
8. start by learning what it is
9. start by doing some small tests with stacking ensemble
10. fail and repeat until we understand how the stacking works in practice
11. submit our first score to kaggle and check the results
12. we got a bad score, so we decide to start over and check if we can improve.
13. We start using pycaret which will save us time on coding.
14. We finish the model using pycaret and try to deploy our model locally
15. Our deployment works, we try and see if we can improve on our model

To complete the project we would need personnel that could maintain and improve on our model, this would also require them to have some very powerful computers so that the model could quickly and effectively give out predictions for many customers simultaneously. In addition to that we would also need a stream of info for the housing prices and changes that are done to the houses in Ames so that our model always is up to date.

METRICS

The minimal “business metric” for our project to be considered a success would be that it can pretty good estimate of a housing price. It doesn’t have to be a perfect prediction, but that does not mean it should give bad predictions either. It should not predict outrageously higher values that the housing actually is; for example a house that costs 100,000 dollars and our model predicts that it costs 100 million dollars. Then it would not be considered a success. We will be using Root mean square error(Rmse) to calculate if the models are performing well.

DATA

The data we are using is the Ames housing dataset, which is a dataset of residential homes in Ames, Iowa. The dataset has more than 79 labels which cover nearly all aspects of a house. The dataset has pool labels, garage labels, fireplace, etc. However not every house has a pool, which means that there are also a lot of missing values. A lot of the data can conflict with each other and some of them don't really help that much with predicting the housing price. The amount of data we would need would depend on how accurate our model is, and to check how accurate and consistent it is. If the model performs poorly we will increase the amount of data and training until the model only predicts the “ground truth”, or somewhere close enough at least.

Some privacy issues that can occur with having so much data about someone's home is that they may think it's a breach of privacy, and that they would not be willing to give out their data. We might have to pay them a sum of money or persuade them to assist us.

A lot of the data that we have we will give to the models as it is, However, as mentioned above we have to take our time to fix the missing values and take care of obsolete data which will only use up computational resources and time.

MODELING

When we were going to explore which models suited our data the best, we started with checking most models manually and submitting scores to kaggle to see which ones did the best. However, this proved to be very time consuming and dull. So we used pycaret to help us choose the models for us. We sorted the models by Rmse and checked out some of the models. some of the models included: Ridge, Lasso, Elastic net, Orthogonal matching pursuit, etc.

At the start of the project we would check out the baseline performance and behavior by using simple models like linear regression and no stacking or hyperparameters and then submitting the results to kaggle to check how it would perform. At the start we got a measly score of 9.4 (which is very bad). However, towards the end of the project when we added in stacking and the best models and polishing our model, we got a score of 0.16 which is significant improvement compared to our first testing.

DEPLOYMENT

We will deploy our model on a locally hosted website gradio, were our predictions will be used to predict a housing price when given certain inputs from the user. At the start of deployment we were looking for solutions using Flask or Voila, however these were very tedious because of how many variables we had in this task.

If we were going to be deploying further online to services like AWS, we would probably have polished our model and decreased many of the variables to make it less tedious. When we are monitoring and maintaining our machine learning system we will be focusing on detecting the problems that can generate negative business value for us, and also problems that can result in the model giving poor results. If retraining our model doesn’t improve performance, then we would have to consider remodeling or redeveloping our model from scratch. After deployment we will also be focusing on improving the performance and time needed for our model to perform the predictions.

REFERENCES

- *https://towardsdatascience.com/stacked-ensembles-improving-model-performance-on-a-higher-level-99ffc4ea5523*
- *https://neptune.ai/blog/how-to-monitor-your-models-in-production-guide#why*
- *https://colab.research.google.com/github/alu042/DAT158-2022/blob/main/notebooks/DAT158-2.12-PyCaret.ipynb*