ENS Data Challenge 2022: Real Estate Price Prediction

Victor Hoffmann

March 2023

Abstract

This challenge tackled the issue of a real estate price prediction with tabular data and images. My approach was to build a Multilayer Perceptron concatenated with a Convolutional Neural Network. Lots of preprocessing steps had to be made in order to get the best results, including boxcox transformations and feature selection. My global rank is currently 8th out of the 167 participants.

1 Presentation of the problem

During the year 2022, the ENS Ulm and the Collège de France organized data science challenges, which allowed several companies to publish a challenge on their website. The *Louis Bachelier* Institute published a real estate price prediction challenge, which seems at first sight quite classical. However, this challenge happened to be a bit unusual, since the data which was given was not only tabular but also pictural: each house contained between 1 and 6 images, and each image had its own size. Therefore, dealing with this type of input happened to be challenging. Moreover, one of the main problems of this challenge was to be able to create a model that could handle both tabular data and images.

2 My approach

As the input was multimodal, my approach was to build a model that could handle tables as well as images. Therefore, my choice naturally went towards neural networks, as I could concatenate a Multilayer Perceptron (for tabular data) with a Convolutional Neural Network (for images). The Keras library allowed me to make such a model.

3 Preprocessing

3.1 Tabular Data

The tabular data contained the following variables:

- A listing identifier
- The property type (house, apartment, condo, mansion...),
- The location (approximated latitude, approximated longitude, city, postal code, exposition, floor when applicable...),
- The size (living area and land area when applicable),
- The number of rooms, bedrooms, bathrooms ...
- Energy performance indicators (energy and greenhouse gas emissions)

- The number of photos attached to the listing,
- Indicators whether there is a cellar, a balcony, air conditioning...

Lots of differents preprocessing steps have been made in order to get the best results out of the tabular data.

First, I had to handle **missing values**. I used a quite simple approach. For numerical data, I've decided to impute any missing value of a variable with its median value, except for the variables `land_size` and `floor`. Indeed, I presumed that such values were only missing if there were none. For the categorical data, I replaced the missing values with a `'None'` label. Moreover, for each categorical variable that happened to have missing values, I added a new variable which tells for each value if it has been imputed or not. Therefore, my neural network model would be aware of such imputations. I have also tried some more complicated imputing approaches, such as the K-Nearest-Neighbours Imputer, but it didn't lead to better results.

I have also decided to add a new column named `TotalSize` which added the size of the property and the land size.

Afterwards, I had to encode my categorical data. I chose to one hot encode all of my categorical variables except for `energy_performance_category` and `ghg_category` since these two are ordinal.

Furthermore, I decided transform my numerical data so that each numerical variable follows a Gaussian law. It happened that appying the function $x \to ln(1+x)$ to the label (price) made the label Gaussian. For the other numerical variables, I have used a BoxCox transformation.

Then, I trained an XGBoost model with all of the variables (except for the listing identifier) in order to get an idea of which variables were important. I was then able to make a **feature selection** of the variables. It happened that the variable which told if the apartment/condo was in the upper floors or not was useless. Moreover, the city and postal code variables are just an inaccurate repetition of the longitude and latitude variables, so I decided to drop them as well. Lots of other variables had a small influence on the model but I have decided to keep them because they might slightly improve the model.

I have also normalized the tabular data with the Keras normalization tool and divided each pixel in each image by 255.

3.2 Image Data

As said before, each observation contained between 1 and 6 images and each image had its own size. Therefore, I decided to resize each image to a 70×70 pixel image. In order to have a model that could process a different number of images, I decided to create a 210×140 black image, and to insert each 70×70 pixel image somewhere inside this black image.

An other approach would have been to concatenate a Multilayer Perceptron and 6 Convolutional Neural Networks while only training the n-th CNN if the observation had at least n images. However this approach led to poorer results.

3.3 The model

The number of neurons in each dense layer has been optimized with a Bayesian Op-

timization algorithm. The model has the following structure:

MLP: 32-32-4

CNN: (Conv2D-BN-MaxPool)x3 - FL-16-BN-D-4

MLP + CNN: 4-1

Where Conv2D means 2D Convolution, BN: Batch Normalization, MaxPool: Max Pooling, FL: Flatten and D stands for Dropout.

The model has been optimized with the Adam optimizer. The loss fonction is the mean asbolute percentage error and was predetermined by the challenge organizers. The number of epochs was set to 50.

4 Results

On the public test dataset, this models achieves a mean absolute percentage error of 25,0579%, resulting in the 8th place out of the 167 participants.

5 Ways to improve the model

We can naturally think of a few more improvements to this model. All of them have to deal with the processing of images. First a ResNet pretrained model might be more accurate than our model. However this implies to increase the VRAM, since ResNet only supports 244×244 images. Then, it would have been better to classify each image according to its role in the real estate good (kitchen, bedroom, front of the house...). Some room classifier models already exist and it would therefore be possible to use these models to classify our images. Furthermore, a data augmentation of the images would have likely led to better results.