Automatic valuation model for Hotel property prices

Capstone 2 for Springboard Data Science Career Track

Thomas Loeber

Problem

- Accurate prediction of hotel property prices
- Commonly used approaches:
  - Manual inspection by a human expert
  - Ordinary linear regression
- Both are suboptimal:
  - Manual inspection is expensive but not very accurate
  - Ordinary linear regression is not flexible enough to pick up nonlinearities (e.g., effect of age) and interactions.
- Better alternative: a more flexible machine learning model, such as gradient boosting

Use case

- 1. A buyer deciding how much to pay for a given hotel
- 2. Correctly pricing hotel properties on a company's balance sheet

- In particular, the latter case is a good application area for a machine learning model, because it does not require the last bit of accuracy.
- For the former case, we would probably want to supplement the machine learning model with information from an in-person evaluation of the property.

Data

- Merged two data sets:
  - Hotel sales
  - Survey containing additional attributes of the sold properties
- These data come in 2 waves:
  - For the newer wave, I had access to the original two data sets, and merged them myself
  - For the older wave, I only had access to the already merged data. Since these had been prepared for a traditional analysis using OLS, all observations with any missing values had been deleted, and only columns deemed most important had been kept
- Overall: About 7000 observation and 35 variables.

Data wrangling 1

- First challenge: Merging the data set of sale prices with the data set of detailed hotel attributes
- No key column (or combination of columns)
- Hotel names and addresses were available but formatted inconsistently
- Solution: First match as many observations as possible using (preprocessed) names, then address. Then compute string similarity of name to find observations with similar but not identical spellings.

Data wrangling 2

- Data suffered from various problems, such as information being tacked on to numeric columns. (E.g., the fact that the hotel had been renovated was appended in parentheses to the year built.)
- This required using pandas' string methods to make sure that each column contain information about only a single feature.
- Only after this step was completed, variables could be converted to proper type.

Exploratory data analysis 1

- I mainly used exploratory data analysis to find problems with the data.
- Since I had already cleaned the data, no major additional problems were detected.
- It did help reveal some additional information about the data in one case, namely that a categorical variables was really ordinal. I thus later mapped the categories to ascending integers (only for XGBoost, though, since linear regression is unable to use information about ordering of categories).

Exploratory Data Analysis 2

- I did not use EDA to try to guess which variables are important, or which variables may have nonlinear effects. This often gives a misleading picture, because plots can only look at two or three variables at a time. As a result, this can be biased by things like spurious correlations.
- Instead, I rely on the flexibility of XGBoost to deal with these problems. Since gradient boosting does not make any distributional assumptions. For example, it is not bothered by variables which are extremely skewed, and it detects nonlinearities without any need for feature engineering such as adding polynomial terms or logs.

Machine learning 1

- I’m trying to show two things:
  - Practitioners should use a more flexible model such as gradient boosting rather than linear regression, because it gives better performance.
  - In addition, practitioners should pick up practices from machine learning such as evaluating a model based on the fit on withheld data, and use regularization to avoid overfitting.

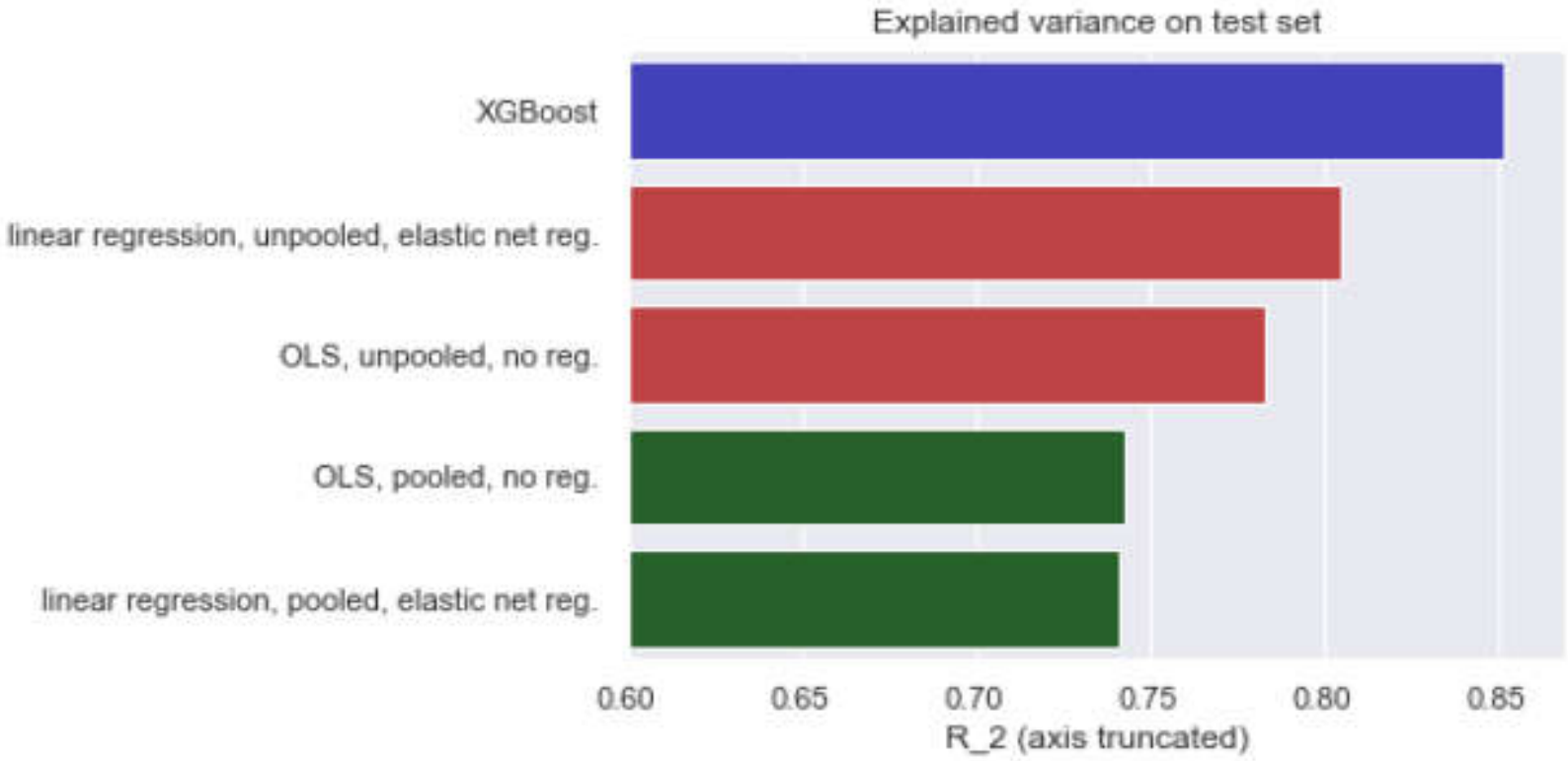
Machine learning 2

- Primary model used: gradient boosting (XGBoost)
- In general, this is the best-performing model on anything but huge data sets (where deep learning yields better results)
- Hyperparameters are tuned using Bayesian optimization (Hyperopt), which requires less iterations to attain a given performance. While the data set was small, the problem still ended up being quite computationally expensive, since I used 10 folds for cross-validation in order to not lose too many data. In addition, I wanted to use a large number of iterations in order to squeeze out the last bit of accuracy, in order to partially compensate for the small data size.

Machine learning 3

- The baseline model: ordinary linear regression
- Used 2 different versions:
  - Unpooled (dummy variables included for categorical variables with a large number of categories, e.g. MSA and the hotel's affiliation before the sale)
  - Pooled (only dummies for categorical variables with less then 10 categories).
- In addition, i.e. re-estimated both models with elastic net regularization to show the benefits from this technique.

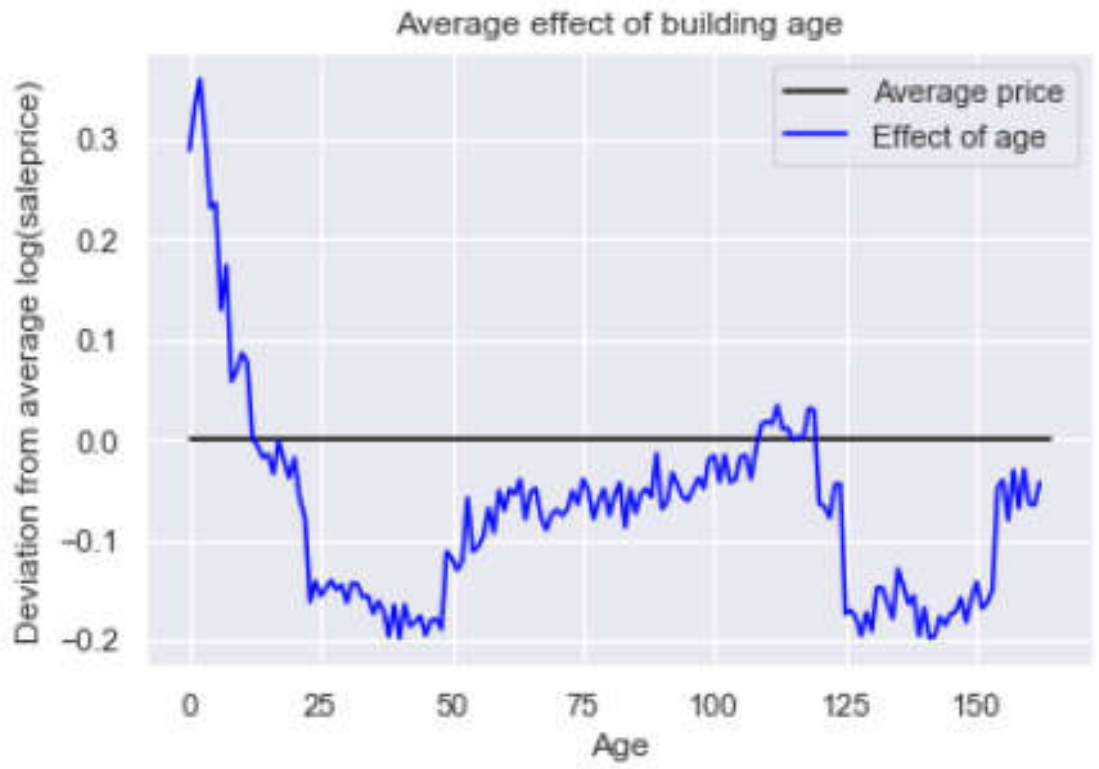
Results


Explained variance on test set
XGBoost
linear regression, unpooled, elastic net reg.
OLS, unpooled, no reg.
OLS, pooled, no reg.
linear regression, pooled, elastic net reg.
0.60
0.65
0.70
0.75
0.80
0.85
R_2 (axis truncated)

Results 2

- Gradient boosting performs best on test data (R_2 = 0.85), with a substantial margin. Remember that the odds were stacked against it, because the data had been optimized for linear regression (observations with missing variables dropped; features had been hand-selected by substantive experts and all other features dropped; data set very small relative to gradient boosting's capacity)
- Unpooled model performs 2nd-best. Thus, it seems worth keeping even dummy variables with hundreds of categories. (This would've been hard to decide without using a test set.
- Regularization improves performance, but only for the unpooled model (because it has a lot more variables)

The nonlinear effect of age, as estimated by gradient boosting



Conclusion 1

- Even on a small data set, gradient boosting perform substantially better than a linear model.
- While gradient boosting is in some ways more complex to understand, it is easier to understand in other ways: By contrast to models originating in statistics, it relies on less (often highly mathematical) assumptions about the distribution of the data. As a result, there are less things we have to worry about.

Conclusion 2

- Likewise, the techniques popular in machine learning to evaluate the fit of a model – namely examining the performance on withheld data – is not only more trustworthy, but also easier to understand than the statistical alternatives (fit statistics such as AIC or BIC, residual plots, etc.)
- In addition, practitioners influenced by applied statistics should also borrow another useful technique from machine learning, namely regularization, which can substantially improve performance at basically no cost. (While it does require a little bit of extra time to implement, it tends to lead to even bigger time-savings elsewhere because it automates feature selection.)

Future scope

- Most importantly, get more data:
  - Start with original data used to create the first wave, and don't drop any observations with missing values
  - Likewise, don't drop any columns, but let regularization take care of shrinking the coefficients of features that contain mostly noise
- Model location on a more fine-ingrained scale (e.g. using geolocation)
- Using an ensemble of different learning algorithms to further increase performance, if necessary.
- Estimate learning curve, to see if there is a minimum threshold of observations (and columns), below which gradient boosting tends to perform worse than a linear model.