Automatic valuation model for Hotel property prices

Milestone Report 2

Preprocessing

As mentioned in the introduction and problem statement, an important goal of this project is to show that practitioners influenced predominantly by applied statistics should adopt common practices from machine learning to validate models. (While these approaches may have originated in statistics, the point is that they have only become common practice in machine learning.)

The first such technique is the idea of withholding data (called a test set) when training the model, and then using these withheld data to assess performance. There are two main approaches of how to divide the data into training and test set: We can either split the data randomly, or we can choose a point in time and use all the data originating from an earlier point in time as the training set, and use the later data as the test set. The former approach works best for independent data, and the latter works best for time-series data.
Our data fall somewhere in between: They are not fully independent, because observations that are closer in time are likely to be more similar, even if we take into account all available predictors. (This is often described as having correlated errors.) On the other hand, though, we are not dealing with a true time-series, because we are not following the same units over time. (An example of a pooled time-series would be repeated sales data.) In this true time-series case, we can commonly observe such an extreme serial correlation of errors that this autoregressive process contains more useful information for prediction than the actual features do.
I decided on a compromise: For the *test* set, I use the latest approximately 20% of the data, which cover the last 1.5 years of the sample. This should make sure we are not overly optimistic, as this makes our calculation of the test accuracy more similar to making predictions for use in practice, when we will probably not have a good estimate of the current year's average price yet. (When I calculate the predictions, I simply treat them as if they were from the last year from which we had data, which should give a better estimate than leaving out the information for year completely, because presumably this year's price will be more similar to last year's prize than to the average price of the last 25 years.)
A disadvantage of this strategy is that the newest portion of the data may not be representative of all years, for example because it is drawn only from a particular point in the business cycle. However, this is likely not a severe problem for our case, because the business cycle presumably mainly influences the *level* of the price (captured, in the case of a linear regression model, by the intercept) while the price determinants should stay pretty stable otherwise. To be on the safe side, I do split the data randomly when creating a validation set from the training set when performing cross-validation. The reason that I pick a different strategy here is that we are

not interested in the *absolute level* of the validation error here, only the *relative* performance of the different hyperparameters. As a result, it's not a problem that random splitting causes an optimistic bias, because this should affect all hyperparameters values equally.

Let's now talk about missing values. I decided to drop all observations with missing values rather than in imputing them. (This was already done in the earlier Python files that cleaned the data.) The reason for this is connected to the fact that, as described earlier, I only had access to the original data for the newer observations. The older observations already came in merged form, and unfortunately all observations with any missing values had already been dropped. Likewise, many *columns* that were seen as less important had already been dropped.
This is because the data were previously used for regression modeling. As a result, the variables were reduced to a small number deemed most relevant by domain experts. To enable easier modeling, any rows with missing observations were dropped. (This is because the regression models were intended for statistical inference, which requires more complicated imputation than predictive modeling, which is the goal here. In particular, in order to get correct standard errors, *multiple* imputation would need to be performed.)

I decided to perform the same processing on the newer data. This leaves us with 7,500 rather than close to 11,000 observations, and 32 rather than around 100 usable features. Note that since gradient boosting does not have any trouble dealing with missing values and is able to use even a high number of features, this decision favors ordinary linear regression. Furthermore, the small size of the data set does not allow gradient boosting to reach anywhere near its full potential.
This is intentional, because it creates what is sometimes called a "least likely case": If it can be shown that gradient boosting is preferable to linear regression even in such a case that is stacked against it, it makes a stronger case in favor of it if it still prevails. If we preprocessed the data specifically for gradient boosting (kept all possible features) and as we increase the number of observations (don't drop observations with missing values, and collected more data over time), the superiority of gradient boosting will be more pronounced.

Next, let's talk about the encoding of categorical and ordinal variables. I took different steps here for different models: For gradient boosting, I performed one-hot encoding for actual categorical variables, but mapped categorical variables that were in fact ordinal to integers corresponding to the correct order. This works because tree-based models do not distinguish between ordinal and numeric variables. By contrast, for linear regression, I will perform one-hot encoding for both categorical and ordinal variables. Leaving ordinal variables as numeric would otherwise amount to assuming that the differences between categories are equal, which is usually not the case.
Note that I did not perform any standardization, because it is not necessary for the models I use: As already mentioned, gradient boosting does not care about the distribution of the data, because it treats them as ordinal. For linear regression, standardization can be useful to speed up convergence of gradient descent, but our data set is so small that this is not an issue. Thus, I

left variables at the original scale, which has the additional benefit of making the coefficients of linear regression more easily interpretable.

Goal

The main goal of this analysis is to show that models of hotel property prices should use machine learning techniques that are able to model more complex relationships between variables, rather than using the linear regression models that have been traditionally the main workhorse for statistical inference.
A secondary goal of mine is to show that if linear regression models are used – e.g. as a first step or a point of comparison – it still pays to borrow two related techniques from machine learning, regularization and the practice of assessing the performance of the model by measuring performance on withheld data.

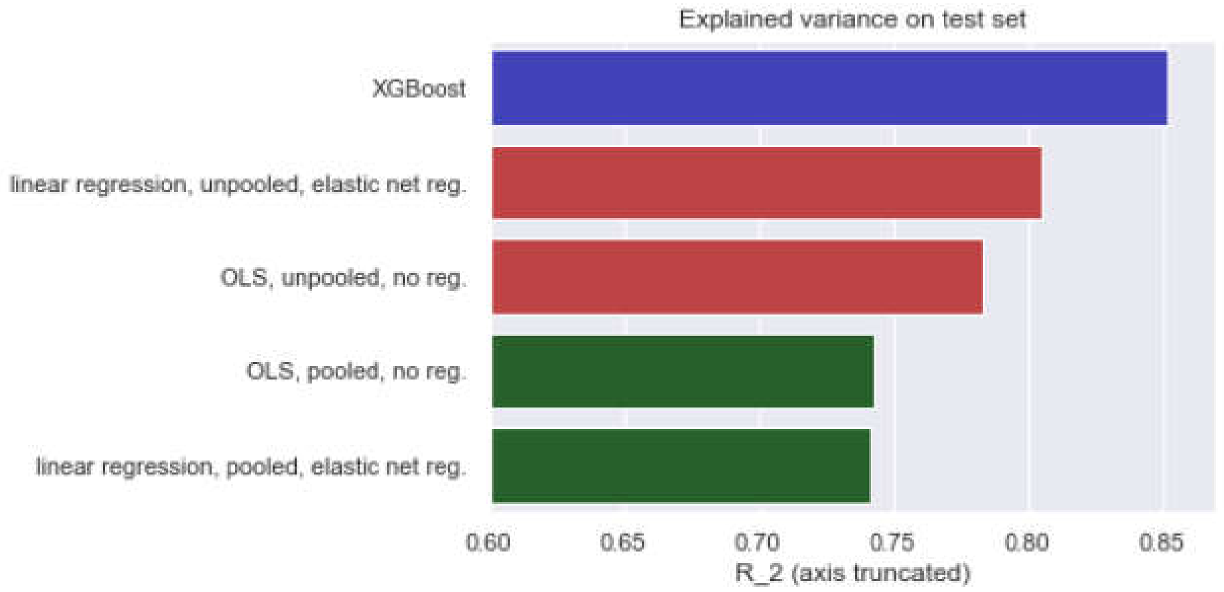
While cross-validation may have originated in statistics, the most common way of assessing the trustworthiness of a model in applied statistics is based on verifying theoretical assumptions combined through model checking, e.g. analysis of the residuals. By contrast, it is common practice in machine learning to instead verify the fit of a model through more direct means, namely by withholding part of the data and then assessing the model's performance on the withheld data. This is usually preferable because it avoids relying on dubious assumptions about the distribution of the data. Furthermore, in practice, model checking is often neglected, or the findings are not presented with the results. Furthermore, the model checking that is done tends to be biased towards not rejecting wrong models. This is probably a major cause of the replication crisis in many scientific disciplines . For a more in-depth discussion of these issues, see Leo Breiman's famous paper "Statistical modeling: The two cultures," which is available at http://www2.math.uu.se/~thulin/mm/breiman.pdf.

The second technique borrowed from machine learning that should be standard practice in applied statistics is regularization. The purpose is to avoid overfitting our model to the data we train our model on. This problem is usually dealt with in applied statistics by practices such as using the adjusted R_2 for model selection. However, the adjustment of the R_2 for the lost degrees of freedom is not rooted in any principled way. By contrast, fit statistics derived from information theory, such as AIC or BIC, are based on a theoretical framework that tries to adjust for the overfitting to the training set. However, these are not only much less widely used than the adjusted R_2, but are also based on a lot of assumptions about the distribution of our data that are rarely completely satisfied in practice. As a result, it is often hard to say how confident we can be in its prescriptions.
One might think that for our sample size of over 7000, regularization might not be that important for a simple linear model. However, remember that we had three categorical variables with several hundred different categories. If we perform one-hot encoding on these, we end up with about 1000 predictors. Since two out of these three categorical variables refer to the location of the property (e.g., the Metropolitan Statistical Area), we can expect that it would indeed be useful if we could include these variables. The only good way to find out if it is worth losing so

many degrees of freedom is to estimate the model on part of the data and then seeing which model performs better on the withheld data.

Results

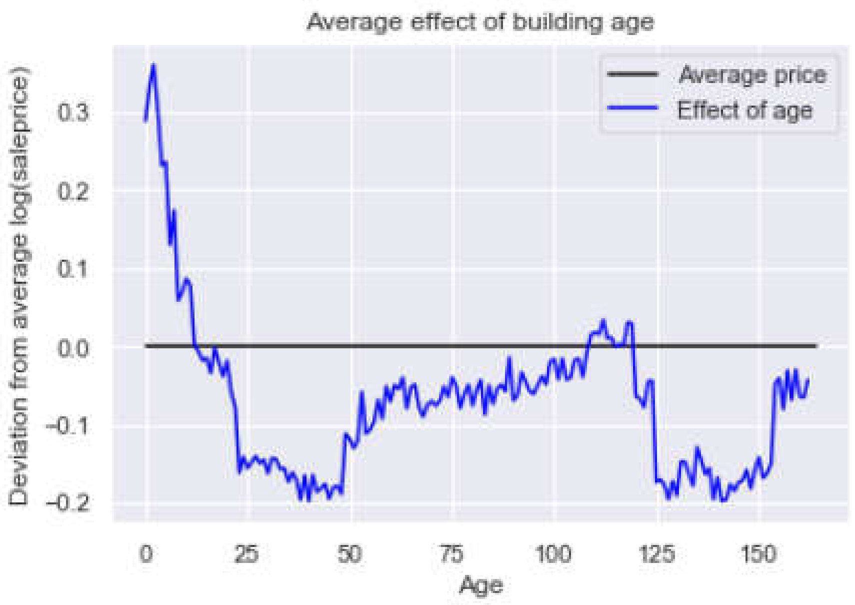
The below table sums up the explained variance on the test set:



We see that gradient boosting gives us the best performance on new data, with an R_2 of 0.85. By contrast, OLS only achieves 0.78 and 0.74 for the unpooled and pooled case, respectively, if we do not use regularization. Adding elastic net regularization improves the performance in the unpooled case from 0.78 to 0.81 to make it the second best performing model. By contrast, regularization does not make any noticeable difference for the pooled case. This is not too surprising: In the unpooled case, adding dummies for hundreds of locations and affiliations of the sold hotel led to a proliferation of variables, so regularization is necessary to prevent overfitting.
Conversely, in the unpooled case, since we have about 7000 observations and only 37 variables, a linear model seems to have virtually reached its capacity: The number of observations is high enough that there is basically no overfitting, and as a result regularization does not yield any noticeable improvement. A second reason why regularization turned out not to be necessary in this case is that I'm using a data set in which the variables have already been pre-selected to optimize it for use with OLS. If I had access to the full data set, it is likely that regularization would have an edge, because it is able to intelligently select which variables contain useful information in which mainly contain noise.

Finally, let's examine to what extent we can get interpretable results from gradient boosting. As mentioned above, I argue that it is not the "fault" of more complex machine learning models that

they lack easy interpretability; rather, reality is too complex to fit well into simple linear models. Thus, better fitting models inevitably tend to be less interpretable, but this is simply due to the limitations of the human mind. While interpretability is good to have, it is not worth sacrificing accuracy and fit.
I illustrate this problem by looking at the effect of age. Gradient boosting is able to model both the nonlinear impact of a feature, as well as the interaction between different features. In order to make the effect of age more interpretable, I abstract from the interaction, and simply calculate the effect of age, averaged over all situations. To do so, I use a resampling technique: I draw a large number of samples, but randomly assign age values to them. Then, I generate predictions, and plot how the average prediction varies by age.



The resulting graph shows the nonlinear shape that would be impossible to generate by a lower-degree polynomial. This high capacity to adapt to the actual shape of the data is one of the reasons why models such as gradient boosting outperform linear models. As the number of observations grows, this advantage of gradient boosting becomes even more pronounced.

In this way, we could also model feature interactions. For example, we could randomize not only age, but also hotel category. Then we could plot the effect of age for each category. Note, however, that it is only possible to model a small amount of interactions this way, because otherwise the plots become too complicated to generate any insights.

Conclusion

Overall, this analysis successfully demonstrates that a state-of-the-art machine learning model is preferable to OLS, even in situations that seem to be stacked against it, such as a small data set (7000 observations) and not a lot of variables (35). While gradient boosting is more complex to implement, there have been a lot of recent developments that make advanced machine learning techniques more accessible, such as the rise Automated machine learning (AutoML). Furthermore, precisely because machine learning models are more complex, there has been a greater emphasis on developing tools to make model checking easier. In particular, the practice of measuring predictive performance on withheld data offers a simpler and more reliable alternative to applied statistics' heavy use of test statistics that are based on a lot of mathematics as well as often dubious assumptions.

This brings us to the second goal of this analysis, namely to show that – independently of whether more sophisticated models from machine learning such as gradient boosting are adopted – there are also gains to be made from adapting practices from machine learning such as regularization and the use of a separate test set. While using regularization does not always yield noticeable improvements (see the pooled case), it is still good practice to always do so, because it is not always easy to intuitively decide beforehand whether it will. Furthermore, the additional effort involved is small, and it actually saves time in other ways, because we do not have to think as hard about which variables to include and which to exclude.

Likewise, this analysis demonstrated how important it is to use a second central technique from machine learning, namely the practice of withholding data when training the model and then evaluating the model only on these data. Without this it would have been impossible to say, for example, whether the pooled or unpooled model is preferable, or exactly how much better gradient boosting is likely to perform on new data.

Future scope

While we could try different learning algorithms, and combine these into an ensemble to get a presumably small improvement in accuracy, the biggest gains are likely to be had from training the gradient boosting model on better data. As already mentioned, I used data optimized for OLS to show that gradient boosting is preferable even in cases where the odds are stacked against it.
Firstly, it would be very interesting to see how much the edge of gradient boosting increases if we added back the extra features that were discarded. Ideally, we would go back and get the original data for the older wave. However, it might also be sufficient to only use the data we have, and add the extra features from the newer wave while leaving them as missing for the older wave.
Likewise, we should try to increase the number of observations. This could simply be done by not discarding any rows with missing values. (Note that this was done so as to not disadvantage OLS, since there are no good tools available yet that perform a *multivariate* imputation on the training set and then apply the same transformation on the test set. The next version of scikit-learn is scheduled to fix this, though.)
Since location is one of the most important attributes affecting price, we should also focus on engineering better features that measure location at a more fine-ingrained level. For example, since hotel sales are much more sparse than data on residential housing prices, we could use

the latter to estimate the desirability of a specific micro-location, and then include this as an additional feature.
Finally, it would be interesting to estimate learning curves, to see if there are any circumstances in which linear regression should still be the preferred model. In this calculation, it might also be interesting to vary not only the number of observations but also the number of features.