Market Analysis with Econometrics and Machine Learning

2b Trees and Forests

Uni Ulm

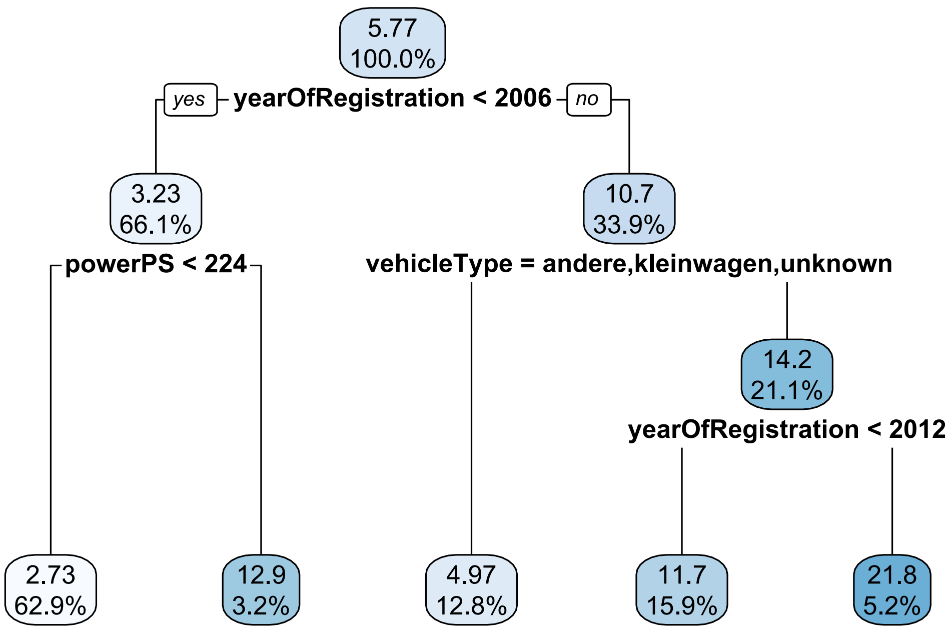
Prof. Dr. Sebastian Kranz

SoSe 2020

Tree based methods

- In this section we will discuss regression *trees* and their very successful extension: *random forests*
  - Random forests typically have good prediction accuracy out-of the box even without extensive parameter tuning. They also often require less data preparation than regression models.
- First look on the graphical representation of a small estimated regression tree on the next slide. It is based on the used cars data set from the RTutor problem set 2a. The model predicts the posted `price` (in 1000 Euro) and has considered in the estimation all explanatory variables except for `model` and `brand`.

Regression tree to predict used car price (in 1000 Euro)



Using the regression tree for predictions

- The resulting estimated tree above uses for the price prediction only 3 variables: `yearOfRegistration`, `vehicleType` and `powerPS`.
- Let's predict the price for a car with `yearOfRegistration=2009`, `vehicleType="Limusine"` and `powerPS=150`.
  - We start with the first decision node on the top with the condition `yearOfRegistration < 2006`. If the condition is true for our considered car, we continue on the left, otherwise on the right. Since we consider a car that is registered in 2009, we continue on the right.
  - The next node asks whether the `vehicleType` is one of the values `andere`, `Kleinwagen`, or `unknown` (`unknown` is how I have encoded missing values). Since we have a `Limusine`, we continue again on the right.
  - The final node asks whether `yearOfRegistration < 2012`. This is true for our car. So we continue on the left and end up in a the 4th node from the left in the bottom row. Such a terminal node that is not split anymore is called a *leaf*.
  - The `11.7` (thousand Euro) shown in the leaf is the predicted price for our car. It is simply the average price of all used cars from our training data set which have ended up into this node when following the tree as described above.
  - The `15.9%` means that 15.9% of the cars in the training data set have ended up in this node.
- We also can get the average price and shares of observation for nodes higher up in the tree. E.g. from the left node in the second line we find that 66.1% of cars have been registered before 2006 and those cars have an average price of 3.23 thousand Euros.
- Note that this tree is of course too simplistic to make good predictions, usually one estimates larger trees.

How is a regression tree estimated?

The most common algorithm to estimate a regression tree is called *recursive binary splitting*.

Let us first explain how we make a single binary split:

- Consider a set of n observations that we split into two subsets S_1 and S_2.
- We denote by $\bar{y}_{S_1}$ and $\bar{y}_{S_2}$ the average values of the dependent variable y in each subset.
- The predicted value $\hat{y}_i$ of an observation i shall be either $\bar{y}_{S_1}$ or $\bar{y}_{S_2}$, depending to which subset observation i is assigned to.
- The residual sum of squares of the split observations therefore is

$$RSS(S_1, S_2) = \sum_{i \in S_1} (y_i - \bar{y}_{S_1})^2 + \sum_{i \in S_2} (y_i - \bar{y}_{S_2})^2$$

Computing a single split (continued)

- A split into S_1 and S_2 will be based on an explanatory variable, call it x:
  - If x is numeric (like `yearOfRegistration`) we split by choosing a threshold x^* and putting all observations i with $x_i < x^*$ into S_1 and all other observations into S_2. We will choose that threshold x^* that minimizes the resulting residual sum of squares $RSS(S_1, S_2)$.
  - The case that x is a nominal variable (like `vehicleType`) will be explained on a later slide.
- We go through every explanatory variable and compute the resulting RSS if we would base the split on it. We then base the split on the explanatory variable that yields the lowest RSS.

Computing the whole tree

- To estimate a whole tree we first compute the initial split for all observations in the training data set and thereby get two new child nodes.
- Then in every round, we look at all nodes who have not yet been split and pick the node whose subset has the highest RSS. We split this node again into two child nodes using the procedure above.
- The step above is repeated until some stopping criterion is reached (e.g. as long as the number of nodes is below some specified maximum number of nodes). Stopping criteria involve parameters, which can be determined by usual parameter tuning methods (like cross-validation).

Optimal splitting based on a nominal variable

- A split based on a factor variable x with M different categories is achieved by finding an optimal subset $\mathcal{M}_1$ of categories such that all observations i with $x_i \in \mathcal{M}_1$ belong to subset S_1 and all others to S_2. The optimal subset of categories $\mathcal{M}_1$ is the one that minimizes the residual sum of squares $RSS(S_1, S_2)$.
- There are $2^{(M-1)}$ ways to split M categories into two non-empty subsets. For example if we have 40 categories there are more than 549 billion ways to split them.
- Luckily, there is a quick way to compute the split that minimizes the RSS. One first computes for each category m the mean of the dependent variable $\bar{y}_m$ of all observations with $x_i = m$. Then one splits by finding a threshold y^* such that all observations where x_i is a category m with $\bar{y}_m < y^*$ belong to S_1 and all others to S_2. (For more details see Section 5 here)

Replicating a regression tree with a linear regression

- We can in principle replicate the predictions of a regression tree by estimating an equivalent linear regression.
- Consider a regression tree that has K different leaves (terminal nodes) and the corresponding linear regression:

$$y = \beta_1\delta_1 + ... + \beta_K\delta_K + \varepsilon$$

  where δ_k is a dummy variable that corresponds to the `k'th` leaf in our regression tree.
- More concretely $\delta_{k,i}$ is 1 if observation i satisfies all conditions to end up in the k'th leaf of our regression tree. Otherwise $\delta_{k,i}$ is zero.
  - For an example, look again at the previously estimated regression tree of our car data set.
  - The first dummy variable $\delta_{1,i}$ is 1 if and only if `yearOfRegistration_i< 2006` and `powerPS_i < 224` .
  - The second dummy variable $\delta_{2,i}$ is 1 if and only if `yearOfRegistration_i< 2006` and `powerPS_i >= 224` ,
  - ...

Replicating a regression tree with a linear regression

- If we estimate that linear regression

$$y = \beta_1\delta_1 + ... + \beta_K\delta_K + \varepsilon,$$

  the OLS estimate $\hat{\beta}_k$ will be equal to the average value $\bar{y}_k$ of the dependent variable for all observations of the training data that fall into leaf k.
- In practice there is no need to estimate such a regression, after we have estimated the corresponding regression tree.
- Conceptually, however, we see that a regression tree can be interpreted as a method to pick explanatory variables with good predictive power by discretizing the original variables and by finding important interaction effects between those discretized variables.

Discussion and Extensions

- Regression trees are very intuitive and can be nicely illustrated.
- However, a single tree is prone to over-fitting and tends not to make very good predictions.
- Yet, two extensions: *random forests* and *gradient boosted trees* are very successful machine learning methods. Both estimate a large number of trees (called an ensemble) and combine their predictions.
- We will only cover random forests in this course, but there are many good tutorials for *gradient boosting* in the internet. The most popular R package to estimate gradient boosted trees probably is xgboost.

Random Forests

- A (regression) random forest estimates a large number of trees (e.g. 500) and takes the mean of the predicted values of every tree as total prediction.
- Every tree is estimated a bit differently be introducing two random factors (explained further below):

1. Bagging
2. Select a random subset of considered features for each node split.

- The goal of the random factors is to make the overall prediction robust against over-fitting.

Bagging in Random Forests

- Bagging is a technique inspired by bootstrap re-sampling.
- Instead of using the complete training data set to estimate a tree, a random sample with replacement of the same size than the training data set will be drawn to train the tree.
- This means some rows of the training data will not be used to train a particular tree, others occur twice or even more often.
- The non-used-observations are called *out-of-bag* sample, and can be used as a sort of validation data set to assess the prediction accuracy of a single tree.

Random feature subset for splitting nodes

- When splitting a specific node in a tree of a random forest not all explanatory variables (aka features) are considered.
- Instead, a random subset of explanatory variables is drawn (by default one third of all features in a regression tree) and one chooses the explanatory from that subset which creates the lowest residual sum of squares to be the splitting variable.

Discussion of random forests

- Random forests typically yield good out-of-the-box prediction quality even without parameter tuning.
- A drawback compared to linear regression, lasso and trees is that random forests are more like a black-box. It is hard to intuitively reconstruct how 500 estimated trees come up with particular predictions.
  - There are several approaches to make black box models easier interpretable, see e.g. here for an overview, or the R packages modelStudio, Dalex or iml.
- A good implementation of random forests in R is the package ranger. It can be called as comfortably as `lm` via a formula interface. You see an example in the exercise sheet.

Using machine learning methods to estimate causal effects

- An active modern research field is how machine learning methods like tree-based methods can help for estimating causal effects.
- One new method are *causal forests* introduced by Wager and Athey (2018). They can help to estimate heterogeneous treatment effects in data from a randomized experiment.
- We want to explain that idea in the following slides.

Potential Outcomes in an Experiment

- Consider a randomized experiment with n subjects indexed by i. As example, we consider an experiment for testing a new drug, but you can also think of other experiments like testing some marketing tool.
- Subjects are randomly split into a treatment group (who get the new drug) and a control group (who only get a placebo). Let w_i be a dummy variable that is equal to 1 if subject i is in the treatment group and 0 otherwise.
- Let y_i be a measurable outcome of interest, e.g. a measure of subject i's health after some weeks taking the drug.
- Let $y_i^{(0)}$ and $y_i^{(1)}$ be the *potential outcomes* for a subject if it would be in the control group or treatment group respectively. Since a subject can be assigned to only one group, we only observe one of the two potential outcomes:

$$y_i = \begin{cases} y_i^{(0)} & \text{if } w_i = 0 \\ y_i^{(1)} & \text{if } w_i = 1 \end{cases}$$

Treatment Effect

- The treatment effect for individual i is the difference between the potential outcomes:

  $$\tau_i = y_i^{(1)} - y_i^{(0)}$$

  In principle this treatment effect can be heterogeneous, i.e. differ between individuals. Yet, since we cannot assign a subject i at the same time to the treatment and control group, we can never directly measure i's treatment effect.
- What we *can* nicely estimate with a randomized experiment is the average treatment effect (ATE) over all subjects

  $$E\tau_i = E(y_i^{(1)} - y_i^{(0)}).$$

  We simply take the difference of the average y_i in the treatment and control groups.
- We can alternatively estimate the linear regression

  $$y_i = \beta_0 + \beta_1 w_i + \varepsilon_i$$

  Given that we have a randomized experiment, w_i is exogenous (uncorrelated with ε_i), and the average treatment effect (ATE) is consistently estimated by the OLS estimator $\hat{\beta}_1$.

Heterogeneous treatment effects for subgroups

- Often one is also interested in the average treatment effect for some subgroup, e.g. for females.
- We have already seen in earlier chapters two ways to estimate heterogeneous effects: running separate regressions for each subgroup or adding interaction terms.
  - Example for a regression with interaction terms:

    $$y_i = \beta_0 + \beta_1 w_i + \beta_2 female_i + \beta_3 female_i \cdot w_i + \varepsilon$$

    Then β_1 measures the average treatment effect for males and $\beta_1 + \beta_3$ the average treatment effect for females.

Difficult when looking at fine-grained subgroups

- Yet, what if we want to estimate a personalized treatment effect that could be based on many potentially relevant explanatory variables and interaction effects?
- More precisely, we would like to have an unbiased estimate of the *conditional average treatment effect*

  $$\tau(x) = E(y_i^{(1)}(x) - y_i^{(0)}(x) | x_i = x)$$

  i.e. the average treatment effect of all subjects that have the same *feature vector* x, where x can include many variables like age, gender, previous diseases, body mass, ...
  - Note the difference in notation compared to earlier sections: both x and x_i are a $K \times 1$ vector that contains one value for each explanatory variable. I follow here the notation in the causal forest papers.
- Potentially, the treatment effect $\tau(x)$ could differ between relatively fine-grained subgroups, e.g. we could have a strong effect for the subgroup of older males that have a certain previous chronic disease.
- But running separate regressions for many subgroups (or equivalently adding a lot of interaction terms) is problematic, because we will have very little observations for each subgroup: this leads to over-fitting and imprecise estimations.
- Great would be an estimation method that finds relevant subgroups that differ in their treatment effects.

Causal Forests

- Causal forests (Athey and Wager, 2018) are a new method that allow to make unbiased predictions $\hat{\tau}(x)$ of the conditional average treatment effect for subjects with feature vector x.
- A causal forest aggregates the predictions of many *causal trees* very much in the same way that a regression forest aggregates many regression trees.
- A trained causal tree makes a prediction for a new observation with feature vector x in the following way:
  - It first finds the corresponding leaf (terminal node) for x.
  - Then its estimate $\hat{\tau}(x)$ of the treatment effect is the difference $\bar{y}^{(1)} - \bar{y}^{(0)}$ where $\bar{y}^{(1)}$ and $\bar{y}^{(0)}$ are the average outcomes y_i for the training data observations in that leaf which were treated and untreated, respectively.

Training a causal tree

- Training a causal tree differs in two main ways from estimating a regression tree:

1. There is a different criterion how to split a node. The idea is to split a node such that the subgroups in the two child nodes differ strongly in their estimated treatment effect $\hat{\tau}(x)$. At the same time we also want to have sufficiently many treated and untreated observations in both child nodes, so that we can estimate $\tau(x)$ sufficiently precise.
    - I don't show the exact criterion. It is a bit complex since it also is formulated such that splits can be computed in a fast way.

Training a causal tree

2. **Honesty** If we use the training data set to split nodes in order to find those subgroups that differ substantially in their treatment effects, we cannot use the same training data to get an unbiased estimate of the treatment effect for those subgroups.

- That is an always repeating story: if you use a data set for estimating, tuning parameters, or selecting something, you typically cannot use the same data set to get unbiased predictions.
- Often there is a solution: data splitting. Use a different data set to make an unbiased prediction.
- A causal tree splits the training data set used to estimate a tree in half. Half of the data is used to determine the node splits. The other half of the data is used to compute the estimated treatment effect

$$\hat{\tau}(x) = \bar{y}^{(1)} - \bar{y}^{(0)}$$

  for each leaf.
- The authors call this data splitting procedure *honesty*, because it yields unbiased = honest estimates of the treatment effect in each leaf.

Honesty is not costly in a random forest

- Data splitting is a very powerful tool. But it can also be costly in the sense that we use only half of the data for some estimation.
    - E.g. in a causal tree we only use half of the training observations to estimate the causal effect for a particular leaf.
    - Using fewer observations makes estimators less precise.
- However, this actually seems to be no big problem for a causal forest. Recall, that in a random forest we anyway choose a different random data set to train each tree in order to reduce over-fitting.
    - In a causal forest we also randomly determine separately for each tree which half of the training data is used for splitting and which half to estimate $\tau(x)$.
    - This means every data point is used in some, but not all, trees for estimating $\tau(x)$.
    - The authors note that this additional random split of data seems to make the causal forest predictions even a bit more robust against over-fitting.
    - Since every tree effectively uses fewer observations, it is recommended, though, to train more trees in a causal forest than in a regression forest. The default settings are 2000 trees in a causal forest (`grf` package), compared to 500 trees in a regression forest (`ranger` package).

R implementation, prediction error and confidence interval

- Causal forests are implemented in the R package grf
    - The package documentation also gives a good explanation of the algorithm under https://grf-labs.github.io/grf/REFERENCE.html which is easier accessible than the research paper.
- A great feature of causal forests is that the `predict` function for causal forests also can provide an estimate of the variance of the estimator $\hat{\tau}(x)$ for given features x.
    - A theoretical result also establishes that $\hat{\tau}(x)$ is approximately normally distributed.
    - This means we can use our rule of thumb to compute 95% confidence intervals for the predicted treatment effect given some x.
- Also note that in a causal forest `predict` yields consistent estimates $\hat{\tau}(x)$ already for the training data set, i.e. splitting away a test data set is not needed unless we want to search for subgroups with high treatment effects (see following slides).

Small sample bias towards mean treatment effect

- We will look at a simple causal forest example in the lecture.
- The simulation suggests that the estimates of the treatment effects $\hat{\tau}(x)$ for subgroups with large effects seem to be biased towards the average treatment effect.
    - That is because the leaves of the different trees typically bunch together observations with smaller and larger treatment effects and thereby flatten out heterogeneity in the predictions.
    - This *flattening effect* makes a causal tree in some sense conservative since it underestimates heterogeneity.
- It is theoretically shown, however, that the estimate $\hat{\tau}(x)$ is consistent, i.e. the flattening out of heterogeneity goes away as the number of observations grow (and the forests parameters are accordingly adjusted).

Be careful when systematically searching for subgroups with the largest effect

- It is tempting to use a causal forest to especially search for subgroups who have a large estimated treatment effect.
- Yet, you should be aware that pre-selecting such subgroups can introduce a bias if you present their treatment effect.

- To better understand the problem. Let us write the resulting prediction of a causal forest as

$$\hat{\tau}(x) = \tau(x) + \varepsilon(x)$$

  where $\varepsilon(x)$ shall be the prediction error.
- Assume $\hat{\tau}(x)$ would be an unbiased estimator (ignore the flattening out effect)
  - Then if we choose some random feature vector x, the expected prediction error $\varepsilon(x)$ would be 0.
  - Yet, if we selectively search for subgroups with large predicted treatment effects $\hat{\tau}(x)$, we are likely to systematically select subgroups were we have a positive prediction error $\varepsilon(x)$, i.e. $\hat{\tau}(x)$ for those subgroups would have an upwards bias.
- This is a similar problem to the multiple testing problem we have discussed in chapter 1d. We would have the same problem if we ran a lot of regressions for subgroups and then selected the subgroup that had the largest effect.
- This bias actually goes in a different direction than the flattening effect described on the previous slide. But it is hard to know which effect is larger.

But what if we want to search for subgroups with large treatment effects?

- This does not mean we cannot look which subgroups seem to have particular strong or weak treatment effects, but you should be aware and transparent about the potential biases.
- Assume we have identified *one* interesting subgroup that seems to have a particular strong treatment effect. How could we get an unbiased estimate of its treatment effect that corrects for both the flattening and selection effect?
  - You could run a new randomized experiment just for this subgroup.

Endogeneity Problems

- Do causal forests solve endogeneity problems?
- No. We have assumed that we have performed a well randomized experiment. Applying causal forests on data sets where the treatment assignment is endogenous yields to inconsistent estimates of treatment effects.
- However, causal forest don't require a perfectly randomized experiment. It also works for settings that a subject's probability to be treated depends on the considered explanatory variables x (That is similar to the fact that in a linear regression the treatment dummy w could be correlated with control variables x. Only a correlation with ε creates an endogeneity problem.)
- The package `grf` also has a function `instrumental_forest` that can help to estimate heterogeneous treatment effects in an instrumental variable setting. But we won't explore this method in this course.

More Literature

- A nice overview of recent developments in the interaction of machine learning and econometrics can be found in the survey paper Machine Learning Methods Economists Should Know About (Athey and Imbens, 2019).