Tree Based Estimators

Decision trees and other tree based models (random forest, boosting) are popular for both classification and regression. However, as with everything, buzz and online chatter do not necessarily make them a good choice in every or even most circumstances. As I'll discuss further below, continuous estimators (logistic regression, support vectors, neural networks) are frequently a better fit for data relationships. In fact, in most circumstances, the value of tree based estimators is more likely to be found in how the results are being used than the true nature of what you are predicting. Below, I will provide a discussion of the pros and cons of tree based estimators and then provide an example.

Advanced Tree Estimators

I'll start with discussing trendy estimators like random forests and boosting. Random forests generally work by generating many decision trees with both a random subset of your data and and random subset of available features. These decision trees are then ensembled with some method such as majority voting (for classification) or averaging (for regression). Boosting, on the other hand, runs one decision tree model and then runs subsequent decision trees on the residuals of the prior models. Alternatively, you can run subsequent decision trees on the same data but instead of weighting observations equally you can give more poorly predicted observations in the previous models more weight.

By calling these models a trend or fad, I'm not at all saying they might not be useful and shouldn't be considered. For instance, one benefit of the random forest might occur when you are modling a dataset too large for your existing memory. Since random forests only use a subset of the data at a time, they essentially 'chunk' your estimation process without bringing the entire dataset into memory. (Note: you might need to select the correct options and syntax for this to work that way.) It's also true that you could simply run these models relatively easily and see how they predict.

But there is still a value in understanding the underlying assumptions of these models and considering whether they match your current application. Even if you plan to compare multiple estimators, you still need to decide what these estimators will be. Further, I don't completely buy into the theory that a model is best chosen based only on a small advantage in predictive performance on a test dataset. As any student of basic statistics will tell you, you can easily make false conclusions about population parameters based on a dataset. Predictive performance is a random variable in and of its own.

Before one considers any tree based estimator, he should consider whether the relationships in the data match the functional form of a decision tree. I will discuss this more below. But just because you are combining multiple decision trees in an advanced methodology, doesn't mean you escape the limitations of each of your component trees.

Further, one should also consider whether the pertinent method of combining multiple decision trees makes sense in your application. For instance, imagine you have a dataset comprised of a thousand potential predictors. You don't know which of these predictors are important and what you don't know is going to hurt you, specifically that only 3 of the 1,000 predictors have any relevance whatsoever. You decide to employ a random forest because everyone is talking about it online. Unfortunately, the random forest builds each tree on a random subset of predictors. In other words, most of your trees won't even include your three predictors unless your settings for number of features selected is so large that it moots the advantage of using a random forest in the first place.

One could also consider the benefit of using boosting. For instance, using a purely linear estimator in both predictors and residuals, boosting would have no impact whatsoever. Decision trees are a nonlinear estimator. But, as with any case you try to glean information from the residuals of a nonlinear estimator, you are left to wonder whether you are modeling information from the actual data or that imposed by your nonlinear functional form. This will depend, again, on whether your data actually exhibits a tree like relationship.

Basic Decision Trees

I will now delve into some of the practical aspects of basic decision trees, what type of data structure fits them well, and finally discuss a specific simulated example. This discussion is relevant not only to the application of the basic decision tree but to more advanced estimators (discussed above) that are built on basic decision trees.

I will start by saying that one might choose a decision tree methodology even if it doesn't make sense based on the actual relationships in the data (i.e. most of the time). For instance, I love using decision trees for exploratory analysis of a large number of potential predictors. That's because decision trees are relatively robust to missing and nonsensical values. In exploratory analysis, I don't necessarily want to perform data cleaning on variables that are mostly unlikely predictors but I might want to run a decision tree to determine if something surprising occurs.

Decision trees might also be used if the decision based on the model is categorical in nature. For instance, the underlying data relationships might be strictly continuous (see more below). However, at the end of the day, your decision will be completely binary. For instance, you will either label an account as fraud or you will not. You will contact a prospectice lead or you will not. You will send a customer to collections or you will not. Therefore, given no other information, forcing your data points into various leaves or categories with expected probabilities and values might be convenient even if the data points don't naturally fall into this sort of structure.

But one could easily make the argument that decision trees are suboptimal even if your action is binary. The thresholds used by decision trees to divide data points into categories are based on creating categories with the purest predictions (see documentation for gini, entropy, or mean squared errors). However, the optimal thresholds might be based on underlying models of loss or profitability. For instance, given a continuos model you might determine which decision threshold leads to the expected benefit of contacting a marketing lead or sending a customer to collections to be double the cost of these actions. You might also consider whether benefits of labeling an account as fraud might outweight the costs of follow up or inconveniencing a customer who makes a legitimate purchase.

Decision Trees Performing Well

Decision trees, especially decision trees with many layers, create complex nonlinear relationships which may or may not be a good fit to your data. Now I will consider a case where a decision tree might be a good fit for a specific problem. Assume you have a population of people of all ages, health conditions, and personal characteristics and you are trying to predict which people have a driver's license. Your three predictors are age, a granular list of serious health conditions/disabilities, a granular list of religious affiliation, and eye color.

In order to give this data to common decision tree algorithms, you will enter age as a continous variable probably after ensuring it contains sensible values. Each health condition, religious affiliation, and eye color will be one hot encoded with its own dummy variable (you might leave one out, but the dummy variable trap does not occur for decision trees as in linear models).

The decision tree will peform many splits on each variable calculating a purity statistic (such as gini) for each split. For instance, it will perform one split for 0 years old and 1+ years old. Then another for 1- years old and 2+ years old and so on. It will then calculate the purity for each individual dummy variable. After considering all of these options, it will choose the best one, probably age.

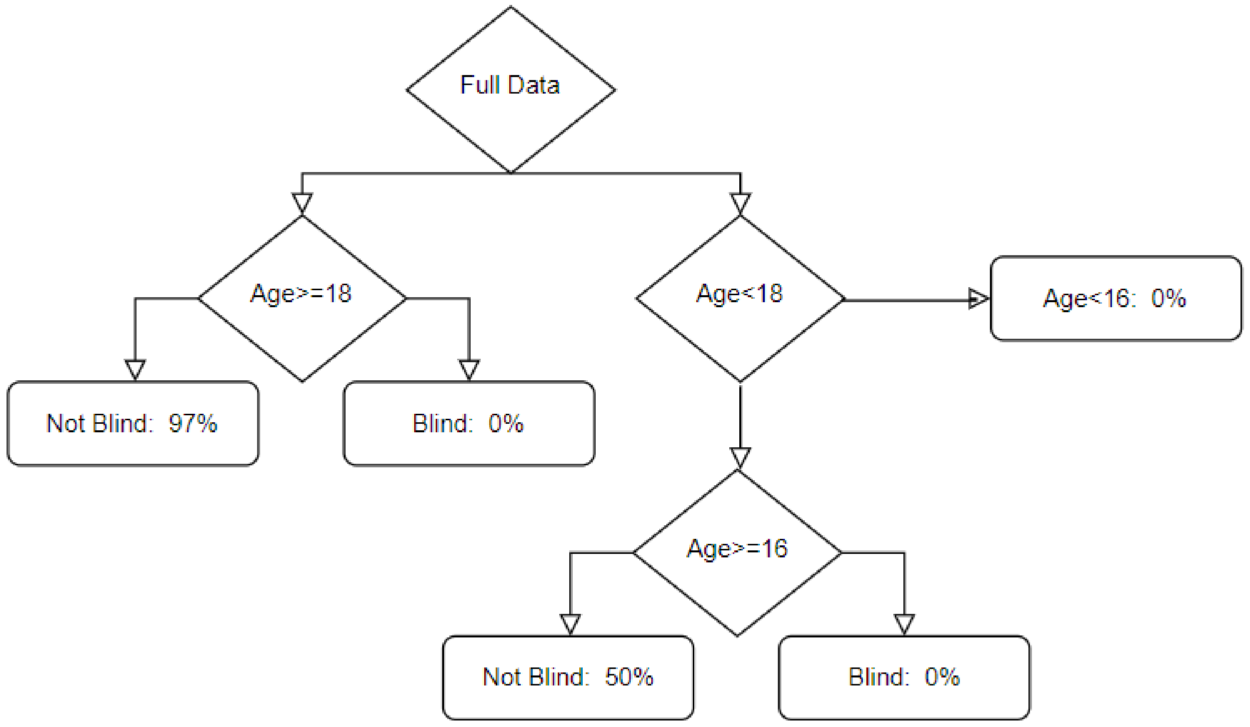
The reason that this problem is such a great prospect for a decision tree is that the key variable (age) experiences catgegorical jumps in the probability of having a driver's license. The first split chosen by the decision tree might be between 16 and 18, let's say 18 as those over 18 only need a drivers test. After performing the first split, the algorithm will perform the same process for those above and below 18 (separately) to determing the subsequent best split. This might be 16, when a person becomes old enough to attend driver's training. Subsequent splits might occur at 17 when more individuals complete training or 75 when many individuals suffer from health issues. But more on health issues next.

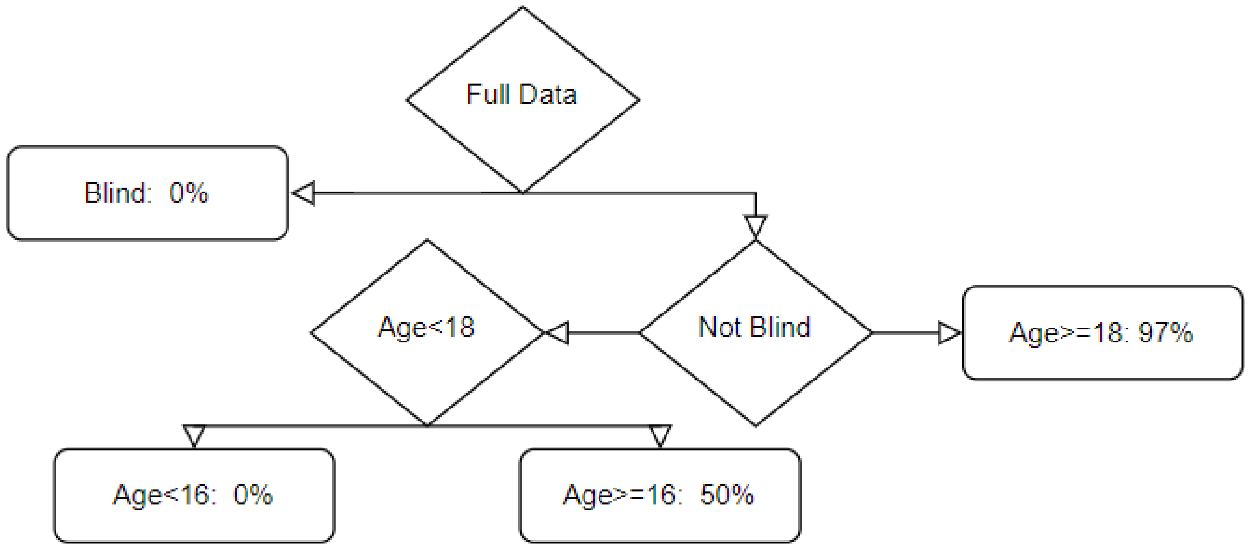
The splits after age might be based on health issues such as blindness, being paraplegic, severe dementia, or an intellectual disability. Note that each of these health conditions will have its own binary variable and will be chosen on its own. Also note that, in our case, age was unfortunately chosen first. So individual splits for blindness might be made for each individual age category. If we were lucky enough that blindess was chosen first, then this would not be necessary. Instead, there would be a blind category and then the non-blind category could be futher split by age.

Further splits might be made by religion and eye color. For instance, Amish people are unlikely to have driver's licenses. Eye color is an irrelevant predictor but I included it for illustrative purposes. Unless you have an effective method to stop the decision tree from splitting, your decision tree will likely make several splits on eye color that aren't actually predictive. In fact, you might get splits on eye color even if you are using an effective stop criteria. That is not only due to random chance, but because eye color is correlated with other factors such as race and ethnicity. This is why having a theoretical understanding of what should be a predictor for your problem and what shouldn't is often very important. Some may consider any predictor valuable, even if it predicts only by correlation with a true predictor. But it's plain to see that some of these outcomes may be undesirable.

One final point to note is that decision trees do an ok job at addressing nonlinearity. As I already said, the impact of age on drivers license is best characterized by uneven jumps not a constant line and a decision tree can easily capture this specific form of nonlinearity. Decision trees can also capture some interactions. For instance, some health issues may only have an impact for people in higher age brackets. Or, assuming we had a variable for sex/gender, this variable may only have an impact for some religions.

The following two figures demonstrate the hypothetical impact of split order on predicting whether someone has a driver's license. The first example which splits on age first has four splits but the second requires only three. In this case, the actual results of the trees are identical, but this might frequently not be the case in real life.





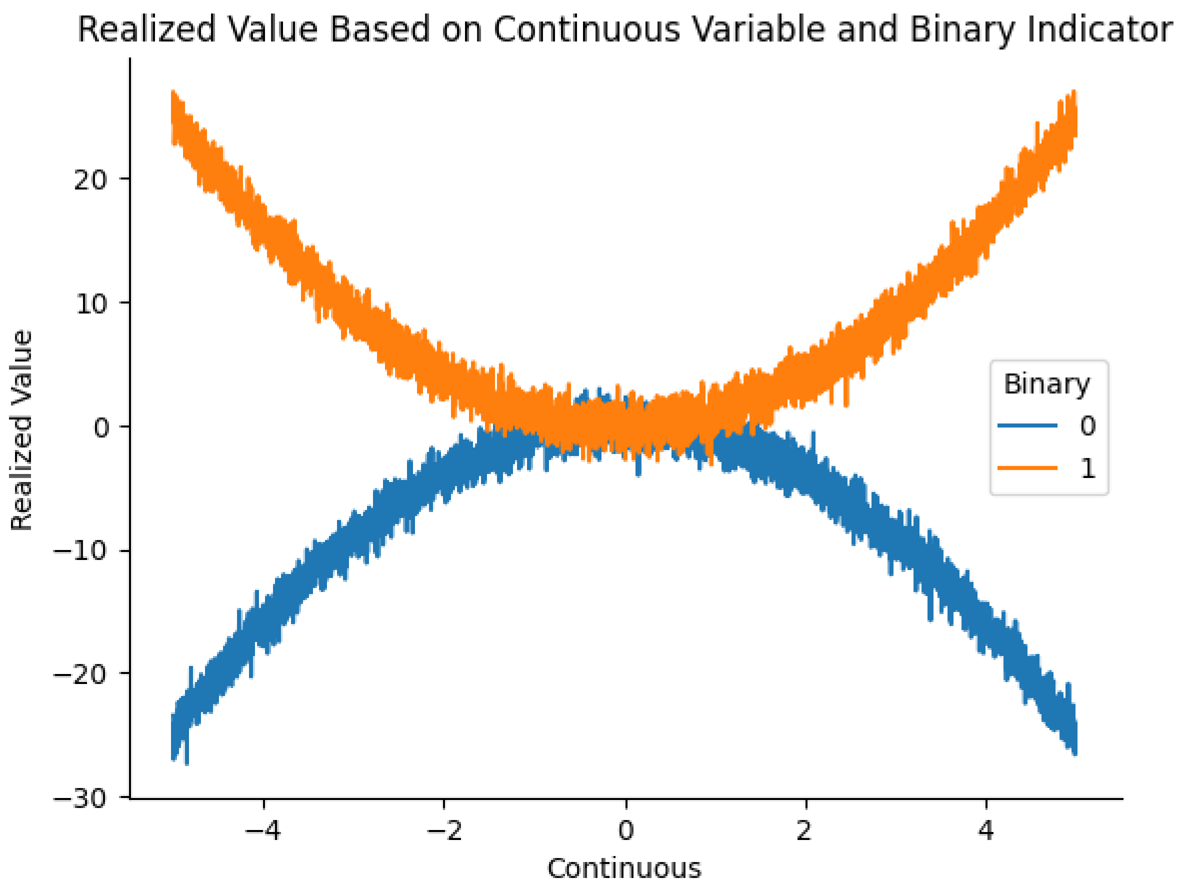
Decision Trees Behaving Badly

Unfortunately, however, decision trees might lead to poor results even in relatively optimal circumstances. As I said before relating blindness and age to driver's licenses, the order that a decision tree makes splits can lead to a very convoluted functional form or even differing results. Also, it's incorrect to say that a decision tree performs best even with purely categorical data. Assume that you were trying to predict the probability that someone was going to vote for a Democratic party candidate based on state of residence. In this case, the optimal approach would be to feature engineer an indicator for red state, blue state, and swing state. Otherwise, based on the most common settings for decision tree algorithms with pairwise splits, each individual state will enter as its own binary indicator which the algorithm will only consider one at a time. So the first split may be made on California, which is relatively populous and Democrat majority. The second may be on Texas which is also populous and Republican majority. However, states such as Wyoming may never meet the split criteria despite the fact that these voters should clearly be categorized as less likely to vote for a Democratic candidate (knowing nothing else about the voter).

Further, most continuous variables (especially those that don't involve regulatory or policy thresholds) have either a linear or smooth nonlinear relationship with the dependent variable. These effects are poorly captured by decision trees. For instance, imagine if we were to consider the impact of temperature on pedestrian traffic. A decision tree (or regression tree) would categorize the temperatures in groups such as (-,37], (37,54], (54,71], (71,79], (79,86], (86,89] (89,+). In reality, pededstrian traffic at 38 degrees is probably different than traffic at 54 degrees. Yet the decision tree will give both these temperatures the average for this entire range.

An Example

In my simulated example, I will give the decision tree a fair chance, the linear regression a run for its money, and first place will be given handily to the neural network. I simulate a dataset of 10,000 observations split evently into a train and test dataset. The dataset includes two interesting aspects: non-monotonic nonlinearity and an interaction. The impact the the uniformly distributed ([-5,5]) continuous variable depends on the bernoulli (p=0.5) indicator. If the indicator is 1, then the impact of the continuous variable on the realized value is x^2. If the indicator is 0, the impact is -x^2. I further add a standard normal error term to achieve the realized value. The full dataset is seen in the figure below.



I said that I would give the linear regression a run for its money. And this problem certainly does just that. Linear regressions often perform reasonably well with monotonic nonlinearity as they still provide a linear average impact of the continuous variable. However, due the the non-monotonic quadratic function, the average linear impact is zero. Note, however, the binary indicator does capture an informative average impact and the overall R-Squared is 0.55.

While many analysts would respond to this problem by inserting an interaction between the binary indicator and the continuous variable, this will not help at all in this case, again due to the non-monotonic quadratic function.

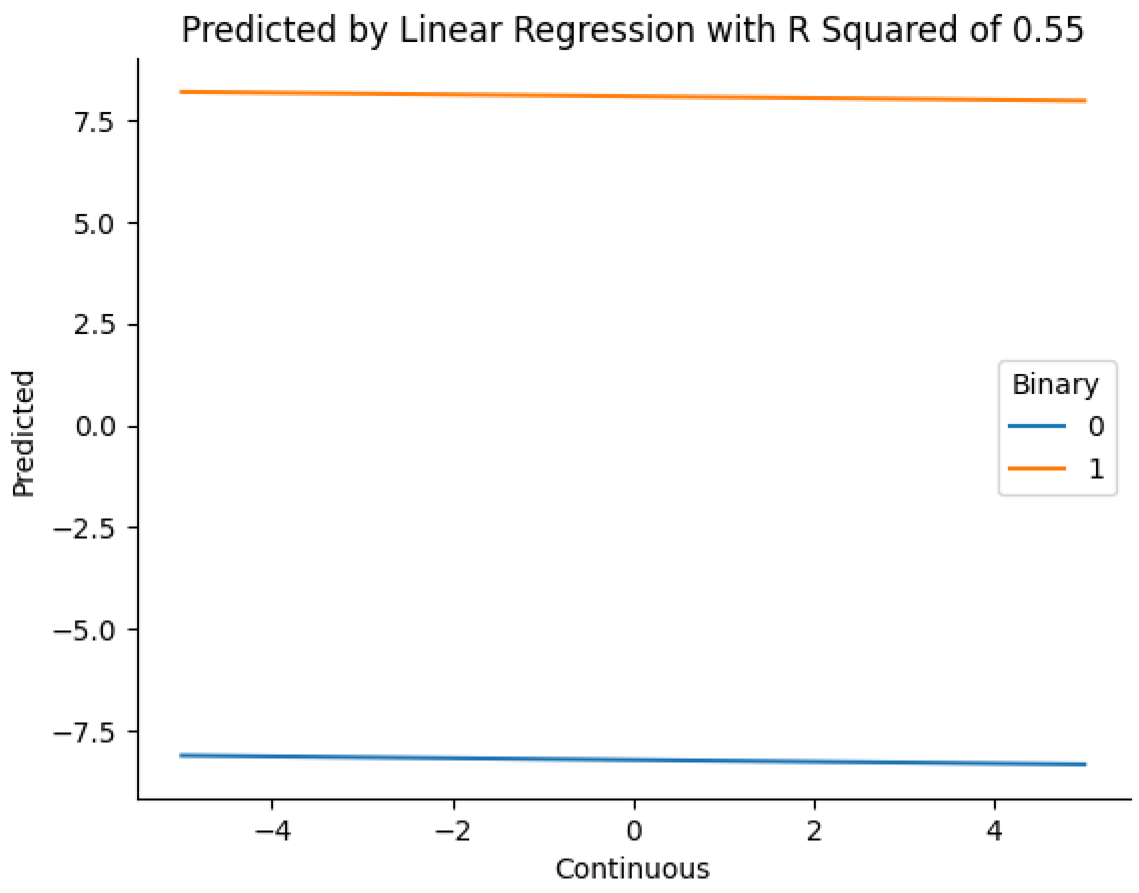
OLS Regression Results

Dep. Variable:	Realized Value	**R-squared:**	0.548
Model:	OLS	**Adj. R-squared:**	0.547
Method:	Least Squares	**F-statistic:**	2962.
Date:	Mon, 19 Feb 2024	**Prob (F-statistic):**	0.00
Time:	02:30:15	**Log-Likelihood:**	-16745.
No. Observations:	4895	**AIC:**	3.350e+04
Df Residuals:	4892	**BIC:**	3.351e+04
Df Model:	2		
Covariance Type:	nonrobust		

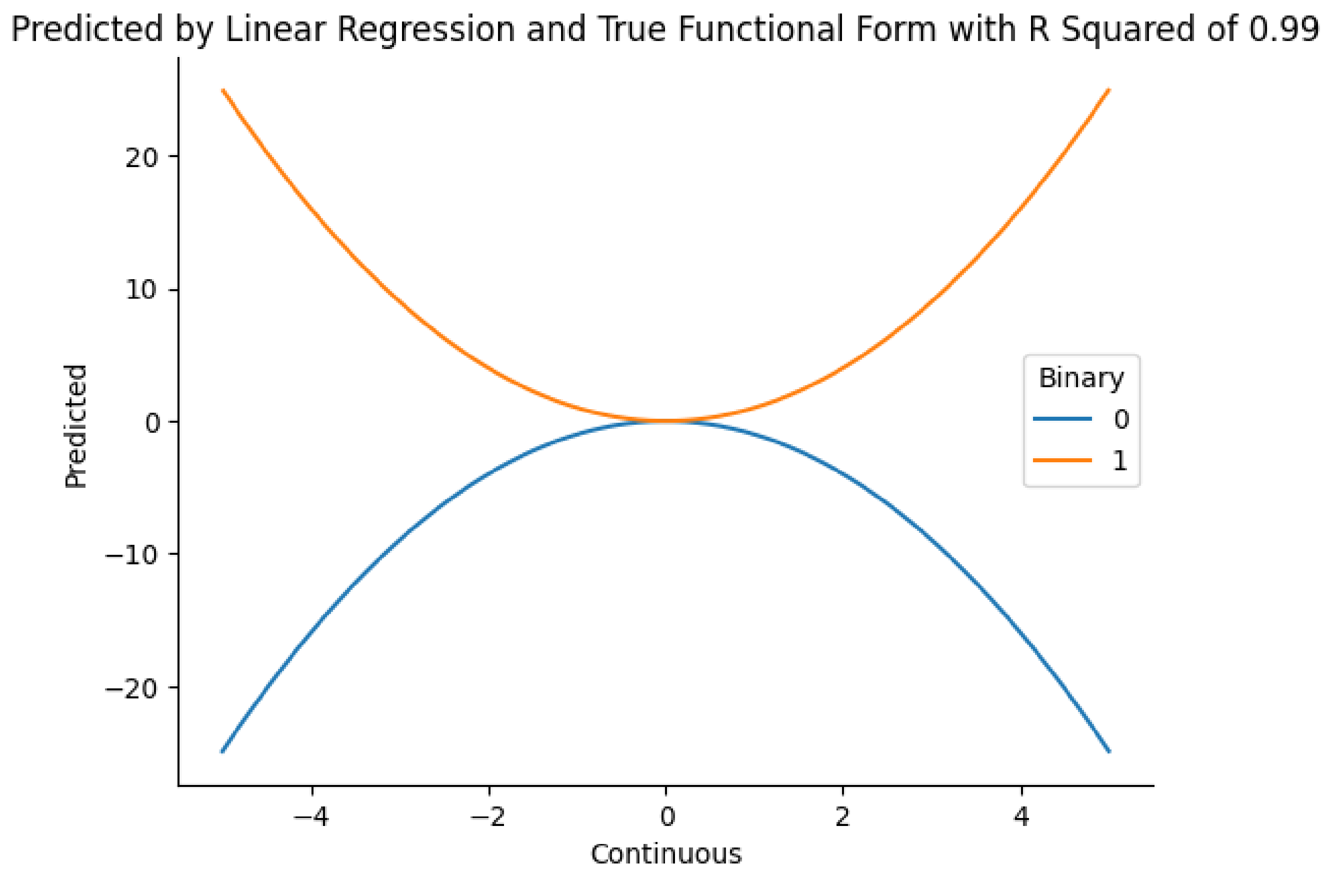
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
ones	-8.2091	0.150	-54.906	0.000	-8.502	-7.916
Continuous	-0.0217	0.037	-0.585	0.558	-0.094	0.051
Binary	16.2883	0.212	76.941	0.000	15.873	16.703

Omnibus:	331.565	**Durbin-Watson:**	1.974
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	115.962
Skew:	0.006	**Prob(JB):**	6.59e-26
Kurtosis:	2.246	**Cond. No.**	6.53

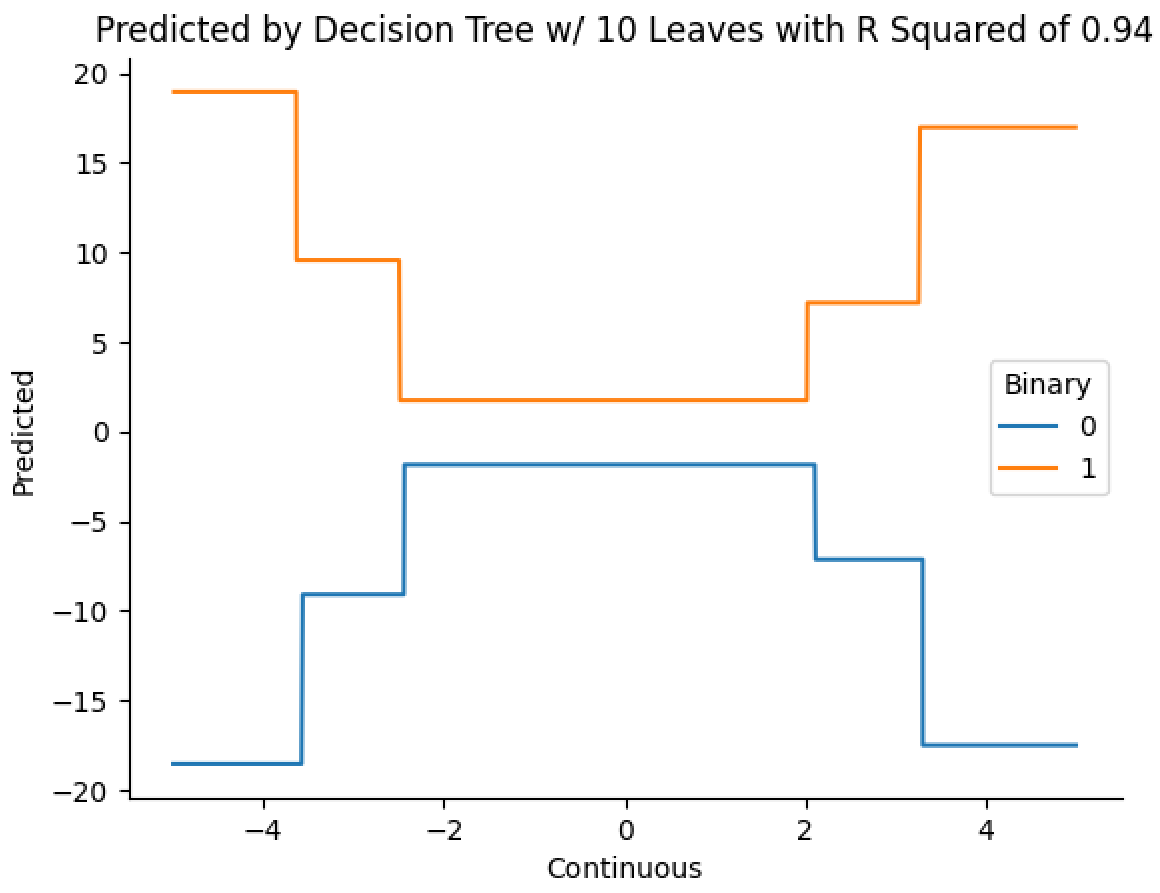
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.



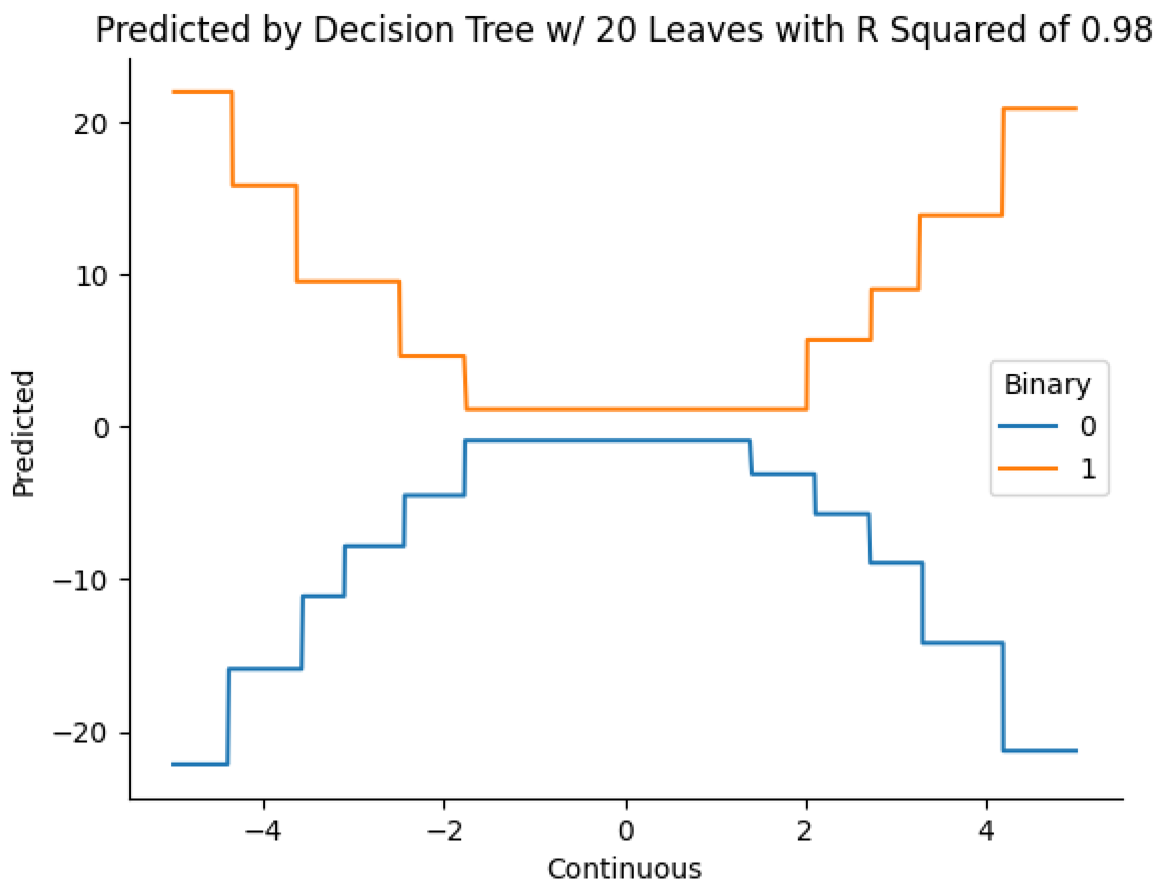
However, we shouldn't give up on the linear regression just yet. With proper nonlinear feature transformation, the linear regression will match the data. In fact, I use the proper transformation and achieve an R-Squared of 0.99 and a prediction that exactly matches the actuals except for the normal error. While many may scoff at this example as a true real-life data relationship will be unknown, in many cases it may be possible to closely determine the true relationship in the data through exploratory analysis, business knowledge, or theory.



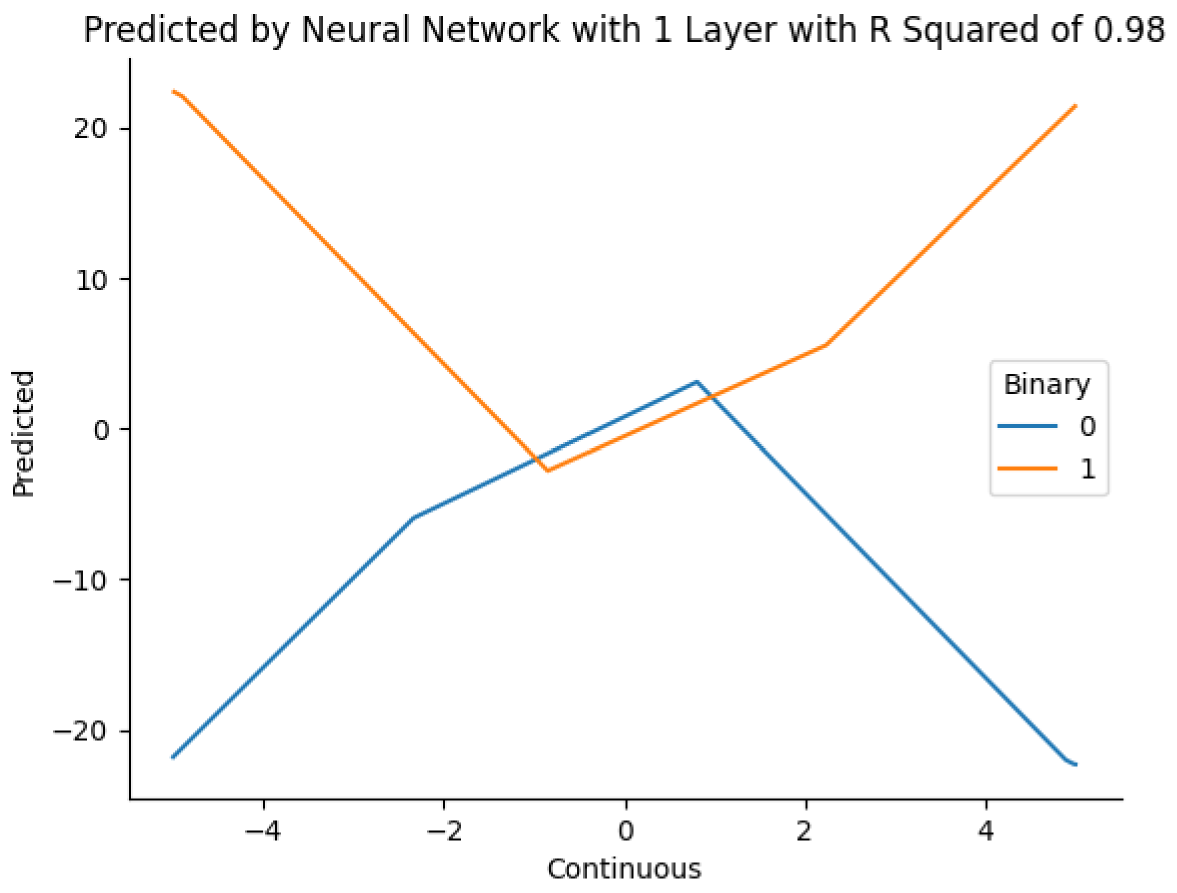
Now we might wonder how close we can get to the true equation using a decision tree. A decion tree can capture nonlinearity, to include non-monotonic non-linearity. It can also capture interactions. However, it will imperfectly capture the continuous nature of the continuous variable. In the figure below, I run a decision tree with 10 leaves, which is probably the largest tree that is easy to diagram and explain. As we can see, the model predictions are jumpy and unattractive. However, we achieve an R-Squared of 0.94 without knowing anything about the underlying relationships in the data other than which variables to include.



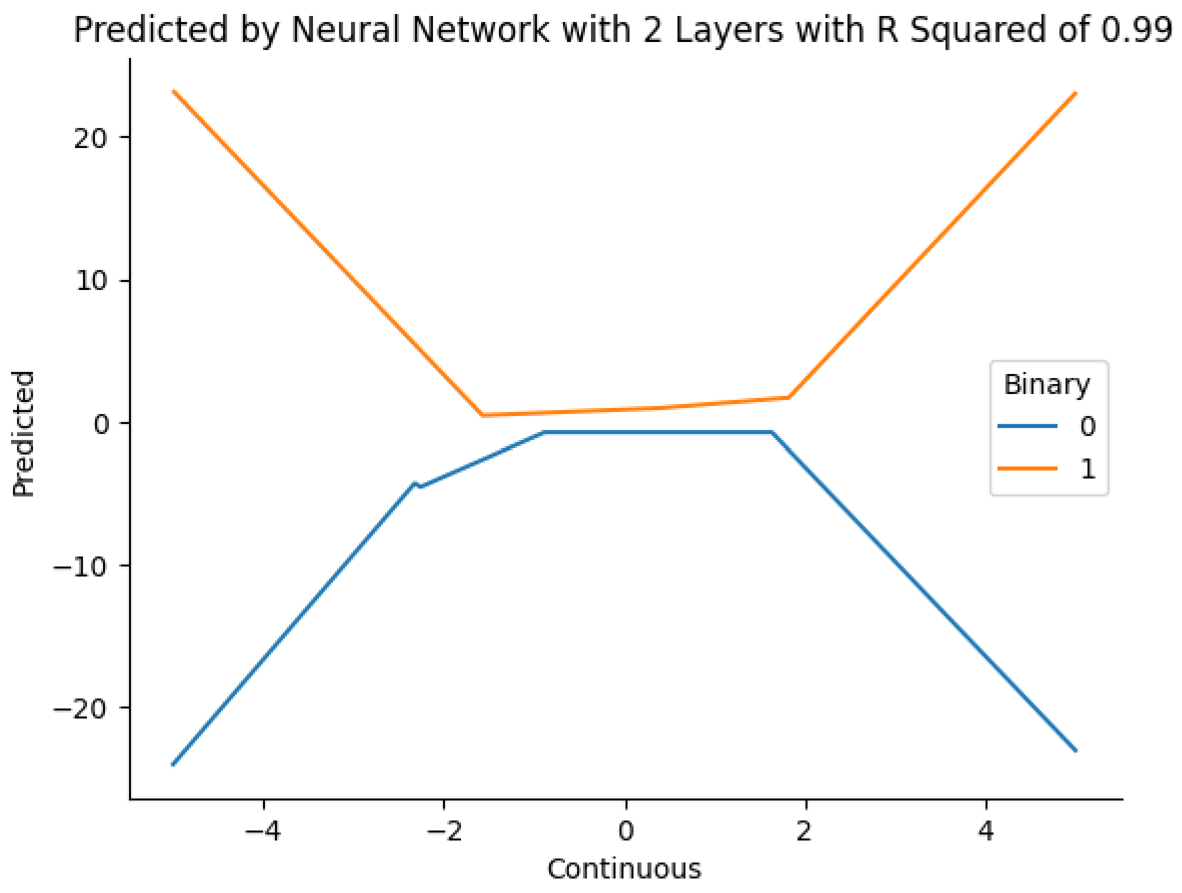
Naturally, in this case we can achieve a more predictive decision tree by adding more leaf nodes. In the following example, I use 20 leaf nodes and achieve an R-Squared of 0.98.

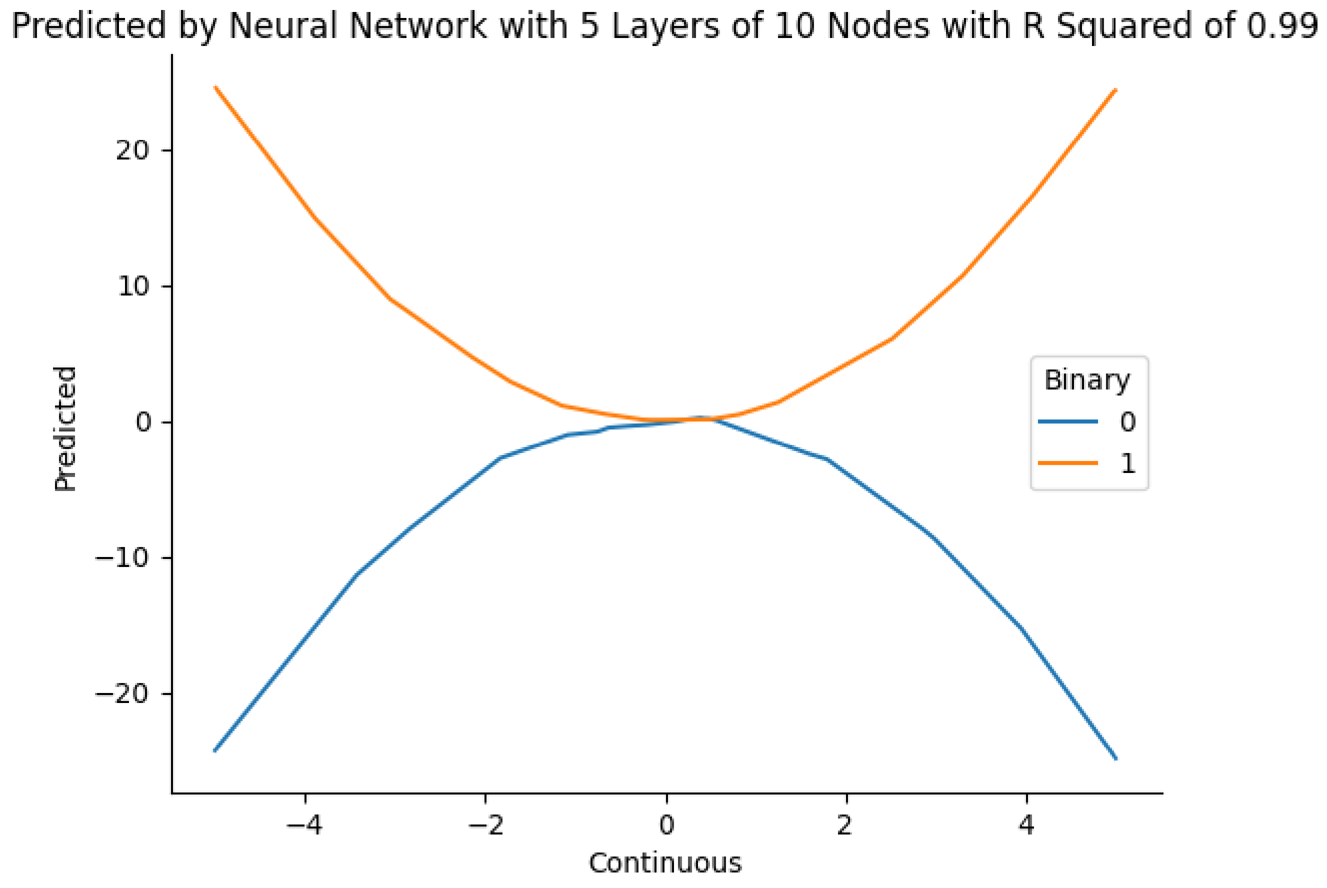


Next, I will discuss what would happen if I were to employ a model more appropriate for the continuous relationship in our data, such as a neural network. In the following example, I use a neural network with RELU activation function and only one layer of five nodes. Doing this, I achieve the same R-Squared as a decision tree with 20 leaves. Again, this would require no exploratory analysis, feature engineering, nor knowledge of the underlying data beyond data validation and cleaning.



Now, I use a neural network with two layers of five nodes. I achieve a R-Squared of 0.99 which ties the linear regression with perfect transformation, though the prediction chart is less attractive. A larger number of layers and nodes (5x10) almost perfectly matches the underlying function.


Predicted by Neural Network with 2 Layers with R Squared of 0.99
Predicted
Continuous
Binary
0
1
20
10
0
−10
−20
−4
−2
0
2
4


Predicted by Neural Network with 5 Layers of 10 Nodes with R Squared of 0.99
Predicted
Continuous
Binary
0
1
20
10
0
−10
−20
−4
−2
0
2
4

In conclusion, decision trees can approximately capture complex nonlinearities in data relationships without indepth exploratory analysis or theoretical knowledge often required for feature engineering. It's also important to mention that the decision tree could have achieved similar results even if we had a dataset full of dozens, hundreds, or even thousands of irrelevant predictors. Aided by adequate sample size and cross validation, the decision tree would have disregarded most of these irrelevant predictors.

However, methods designed for continuous variable relationships more closely approximate many real world data relationships, particularly those not determined by binary indicators or policy/regulatory thresholds.