feature selection

determining which predictors should be included in a model is becoming one of the most critical questions as data are becoming increasingly high-dimensional

from a practical point of view, a model with less predictors may be more interpretable and less costly especially if there is a cost to measuring the predictors

statistically, it is often more attractive to estimate fewer parameters — also some models may be negatively affected by non-informative predictors

feature selection

some models are naturally resistant to non-informative predictors.

tree- and rule-based models, MARS and the lasso, for example, intrinsically conduct feature selection.

for example, if a predictor is not used in any split during the construction of a tree, the prediction equation is functionally independent of the predictor

feature selection

an important distinction to be made in feature selection is that of supervised and unsupervised methods

when the outcome is ignored during the elimination of predictors, the technique is unsupervised.

removing predictors that have high correlations with other predictors or near-zero variance predictors — outcome is independent of the filtering calculations.

for supervised methods, predictors are specifically selected for the purpose of increasing accuracy or to find a subset of predictors to reduce the complexity of the model — outcome is typically used to quantify the importance of the predictors

consequences of using non-informative predictors

feature selection is primarily focused on removing non-informative or redundant predictors from the model.

the importance of feature selection depends on which model is being used. many models, especially those based on regression slopes and intercepts, will estimate parameters for every term in the model

the presence of non-informative variables can add uncertainty to the predictions and reduce the overall effectiveness of the model

consequences of using non-informative predictors

random forests show a moderate degradation in performance with the issue being that the random selection of predictors for splitting can coerce the model into including some unimportant predictors

parametrically structured models (such as linear regression), partial least squares, and neural networks are most affected

neural networks have issues, often due to the excess number of parameters added to the model

consequences of using non-informative predictors

given the potential negative impact, there is the need to find a smaller subset of predictors.

the basic goal is to reduce their number in a way that maximizes performance — this is similar to the previous discussions:

how can we reduce complexity without negatively affecting model effectiveness?

approaches for reducing the number of predictors

apart from models with built-in feature selection, most approaches for reducing the number of predictors can be placed into two main categories

wrapper methods evaluate multiple models using procedures that add and/or remove predictors to find the optimal combination that maximizes model performance

filter methods evaluate the relevance of the predictors outside of the predictive models and subsequently model only the predictors that pass some criterion

both approaches have advantages and drawbacks

filter methods are usually more computationally efficient than wrapper methods, but the selection criterion is not directly related to the effectiveness of the model

most filter methods evaluate each predictor separately, and, redundant predictors may be selected while variable interactions will not be quantified

the downside of the wrapper method is that many models are evaluated (which may also require parameter tuning) and thus an increase in computation time

there is also an increased risk of over-fitting with wrappers

wrapper methods — forward selection linear regression classical example

the predictors are evaluated (one at a time) in the current linear regression model

a statistical hypothesis test can be conducted to see if each of the newly added predictors is statistically significant (at some predefined threshold)

if at least one predictor has a p-value below the threshold, the predictor associated with the smallest value is added to the model and the process starts again

the algorithm stops when none of the p-values for the remaining predictors are statistically significant

classical forward selection for linear regression

```
Create an initial model containing only an intercept term.
repeat
    for each predictor not in the current model do
        Create a candidate model by adding the predictor to the current model
        Use a hypothesis test to estimate the statistical significance of the new model term
    end
    if the smallest p-value is less than the inclusion threshold then
        Update the current model to include a term corresponding to the most statistically significant predictor
    else
        Stop
    end
until no statistically significant predictors remain outside the model
```

in this scheme, linear regression is the base learner and forward selection is the search procedure.

the objective function is the quantity being optimized which, in this case, is statistical significance as represented by the p-value.

1. forward search is greedy — it does not reevaluate past solutions
2. the use of repeated hypothesis tests in this manner invalidates many statistical properties since the same data are being evaluated numerous times
3. maximizing statistical significance may not be the same as maximizing more relevant accuracy-based quantities

wrapper methods

suppose that the RMSE was the objective function instead of statistical significance

the algorithm would be the same but would add predictors to the model that results in the smallest model RMSE continuing until some predefined number of predictors has been reached or the full model is used

the RMSE can be monitored to determine a point where the error began to increase and the subset size associated with the smallest RMSE is chosen

forward, backward, and stepwise selection

stepwise selection is a popular modification where, after each candidate variable is added to the model, each term is reevaluated for removal from the model

in backward selection, the initial model contains all P predictors which are then iteratively removed to determine which are not significantly contributing to the model

these procedures can be improved using non-inferential criteria, such as the AIC statistic, to add or remove predictors from the model.

recursive feature elimination is a backward selection algorithm that avoids refitting many models at each step of the search

a measure of variable importance is computed that ranks the predictors from most important to least through a model based approach (e.g., the random forest importance criterion) or using a more general approach that is independent of the full model.

the least important predictors are iteratively eliminated prior to rebuilding the model

the process continues for some predefined sequence, and the subset size corresponding to the best value of the objective function is used as the final model

while it is easy to treat the RFE algorithm as a black box, there are some considerations that should be made.

when the outcome has more than two classes, some classes may have a large degree of separation from the rest of the training set.

it may be easier to achieve smaller error rates for these classes than the others. when the predictors are ranked for selection, the predictors associated with the “easy” classes may saturate the positions for the highest ranks

as a result, the difficult classes are neglected and maintain high error rates. In this case, class-specific importance scores can aid in selecting a more balanced set of predictors in an effort to balance the error rates across all the classes

backward selection via the RFE algorithm

1 Tune/train the model on the training set using all P predictors
2 Calculate model performance
3 Calculate variable importance or rankings
4 **for** *each subset size* S_i, $i = 1 \ldots S$ **do**
5 | Keep the S_i most important variables
6 | [Optional] Pre-process the data
7 | Tune/train the model on the training set using S_i predictors
8 | Calculate model performance
9 | [Optional] Recalculate the rankings for each predictor
10 **end**
11 Calculate the performance profile over the S_i
12 Determine the appropriate number of predictors (i.e. the S_i associated with the best performance)
13 Fit the final model based on the optimal S_i

simulated annealing (again) for feature selection

an initial subset of predictors is selected and is used to estimate performance of the model (denoted here as E_1, for the initial error rate)

the current predictor subset is slightly changed, and another model is created with an estimated error rate of E_2

if the new model is an improvement over the previous one (i.e., $E_2 < E_1$), the new feature set is accepted

if it is worse, it may still be accepted based on some probability p_{a_i}, where i is the iteration of the process

simulated annealing (again) for feature selection

this probability is configured to decrease over time so that, as i becomes large, it becomes very unlikely that a suboptimal configuration will be accepted

the process continues for some pre-specified number of iterations and the best variable subset across all the iterations is used

the idea is to avoid a local optimum (a solution that is currently best but is not best overall).

by accepting “bad” solutions, the algorithm is able to continue the search in other spaces and therefore is less greedy

simulated annealing (again)

Generate an initial random subset of predictors
for *iterations* $i = 1 \ldots t$ **do**
  Randomly perturb the current best predictor set
  [Optional] Pre-process the data
  Tune/train the model using this predictor set
  Calculate model performance (E_i)
  if $E_i < E_{best}$ **then**
    Accept current predictor set as best
    Set $E_{best} = E_i$
  else
    Calculate the probability of accepting the current predictor set

$$p_i^a = \exp\left[(E_{best} - E_i)/T\right]$$

    Generate a random number U between $[0, 1]$
    if $p_i^a \leq U$ **then**
      Accept current predictor set as best
      Set $E_{best} = E_i$
    else
      Keep current best predictor set
    end
  end
end
Determine the predictor set associated with the smallest E_i across all iterations
Finalize the model with this predictor set

genetic algorithm (again) for feature selection

the problem of feature selection is inherently a complex optimization problem, where we seek the combination of features that provides an optimal prediction of the response

to employ GAs towards this end, we must frame the feature selection problem in terms of the GA machinery.

genetic algorithm (again)

in the context of feature selection, the chromosome is a binary vector that has the same length as the number of predictors in the data set

each binary entry of the chromosome, or gene, represents the presence or absence of each predictor in the data

the fitness of the chromosome is determined by the model using the predictors indicated by the binary vector

GAs are therefore tasked with finding optimal solutions from the 2^n possible combinations of predictor sets.

genetic algorithm (again)

to begin the search process, GAs are often initiated with a random selection of chromosomes from the population of all possible chromosomes.

each chromosome’s fitness is computed, which determines the likelihood of the chromosome’s selection for the process of reproduction

two chromosomes from the current population are then selected based on the fitness criterion and are allowed to reproduce.

genetic algorithm (again)

in the reproduction phase, the two parent chromosomes are split at a random position (also called loci), and the head of one chromosome is combined with the tail of the other chromosome and vice versa

after crossover, the individual entries of the new chromosomes can be randomly selected for mutation in which the current binary value is changed to the other value

genetic algorithm (again)

the crossover phase drives subsequent generations towards optimums in subspaces of similar genetic material

the search subspace is narrowed to the space defined by the most fit chromosomes

the algorithm could become trapped in a local optimum— in the context of feature selection, this means that the selected features may produce an optimal model, but other more optimal feature subsets may exist

genetic algorithm (again)

the mutation phase enables the algorithm to escape local optimums by randomly perturbing the genetic material

usually the probability of mutation is kept low (say, $p_m < 0.05$).

If their are concerns about local optimums, then the mutation probability can be raised

the effect of raising the mutation probability is a slowing of the convergence to an optimal solution

genetic algorithm (again)

```
1  Define the stopping criteria, number of children for each generation
   (GenSize), and probability of mutation (p_m)
2  Generate an initial random set of m binary chromosomes, each of
   length p
3  repeat
4      for each chromosome do
5          Tune and train a model and compute each chromosome's
           fitness
6      end
7      for reproduction k = 1 ... GenSize/2 do
8          Select two chromosomes based on the fitness criterion
9          Crossover: Randomly select a loci and exchange each
           chromosome's genes beyond the loci
10         Mutation: Randomly change binary values of each gene in
           each new child chromosome with probability, p_m
11     end
12 until stopping criteria is met
```

filter methods

filter methods evaluate the predictors prior to training the model, and, based on this evaluation, a subset of predictors are entered into the model

most of these techniques are univariate, meaning that they evaluate each predictor in isolation. In this case, the existence of correlated predictors makes it possible to select important, but redundant, predictors

the obvious consequences of this issue are that too many predictors are chosen and, as a result, collinearity problems arise

filter methods

if hypothesis tests are used to determine which predictors have statistically significant relationships with the outcome (such as the t-test), the problem of multiplicity can occur

for example, if a confidence level of $\alpha = 0.05$ is used as a p-value threshold for significance, each individual test has a theoretical false-positive rate of 5%.

however, when a large number of simultaneous statistical tests are conducted, the overall false-positive probability increases exponentially.

filter methods

to account for this, p-value adjustment procedures can control the false positive rate.

the Bonferroni correction is one such procedure.

if a p-value cutoff of α is used to define statistical significance for each of M tests, using an alternative cutoff of α/M increases the stringency and will help control the probability of a false-positive results — this procedure can be very conservative and limit the number of true-positive results.

filter methods

while filter methods tend to be simple and fast, there is a subjective nature to the procedure.

most scoring methods have no obvious cut point to declare which predictors are important enough to go into the model.

even in the case of statistical hypothesis tests, the user must still select the confidence level to apply to the results.

in practice, finding an appropriate value for the confidence value α may require several evaluations until acceptable performance is achieved.

selection bias

while some filtering methods or search procedures are more effective than others, the more important question is related to how model performance is calculated (especially when the sample size is small)

over-fitting the predictors to the training data can occur and, without a proper validation, may go unnoticed

selection bias

when using other search procedures or filters for reducing the number of predictors, there is still a risk.

the following situations increase the likelihood of selection bias:

- the data set is small
- the number of predictors is large
- the predictive model is powerful (e.g., black-box models), which is more likely to over-fit the data
- no independent test set is available

When the data set is large, we recommend separate data sets for selecting

selection bias

when the data set is large, we recommend separate data sets for selecting features, tuning models, and validating the final model (and feature set).

for small training sets, proper resampling is critical.

if the amount of data is not too small, we also recommend setting aside a small test set to double check that no gross errors have been committed.

measuring predictor importance

often, we desire to quantify the strength of the relationship between the predictors and the outcome.

ss the number of attributes becomes large, exploratory analysis of the all the predictors may be infeasible and concentrating on those with strong relationships with the outcome may be an effective triaging strategy

ranking predictors in this manner can be very useful when sifting through large amounts of data

measuring predictor importance

the notion of variable importance is taken to mean an overall quantification of the relationship between the predictor and outcome

most of the methodologies discussed cannot inform the modeler as to the nature of the relationship, such as “increasing the predictor results in a decrease in the outcome.”

such detailed characterizations can only result from models having precise parametric forms such as linear or logistic regression, multivariate adaptive regression splines, and a few others

measuring predictor importance

variable importance scores (e.g random forests) are a measurement of predictor relevance derived by permuting each predictor individually and assessing the loss in performance when the effect of the predictor is negated

a substantial drop in performance is indicative of an important predictor. While this can be an effective approach, it does not enlighten the modeler as to the exact form of the relationship

despite this these measurements can be useful for guiding the user to focus more closely on specific predictors via visualizations and other means

numeric outcomes

for numeric predictors, the classic approach to quantifying each relationship with the outcome uses the sample correlation statistic

this quantity measures linear associations; if the relationship is nearly linear or curvilinear, then Spearman’s correlation coefficient may be more effective

caveat: these metrics should be considered rough estimates of the relationship and may not be effective for more complex relationships

numeric outcomes

an alternative is to use more flexible methods that may be capable of modeling general nonlinear relationships, for instance, the locally weighted regression model (LOESS)

this technique is based on a series polynomial regressions that model the data in small neighborhoods (similar to computing a moving average). The approach can be effective at creating smooth regression trends that are extremely adaptive

from a model fit, a pseudo-R^2 statistic can be calculated derived from the residuals

numeric outcomes

each of these techniques evaluates each predictor without considering the others —this can be potentially misleading in two ways:

if two predictors are highly correlated with the response and with each other, then the univariate approach will identify both as important. some models will be negatively impacted by including this redundant information.

the univariate importance approach will fail to identify groups of predictors that together have a strong relationship with the response. For example, two predictors may not be highly correlated with the response; however, their interaction may be

numeric outcomes

a sensible strategy would be to further explore these aspects of the predictors instead of using the rankings as the sole method for understanding the underlying trends

knowing which relationship to explore often requires expert knowledge about the data

numeric outcomes

the most natural method for comparing the mean of two groups is the standard t-statistic, which is essentially a signal-to-noise ratio

a p-value can be produced by this procedure where the null hypothesis is that there is no difference between the groups

the assumption for the statistic is that the data are normally distributed. If this assumption is unlikely to be true use a non-parametric version of a t-test

numeric outcomes

when the predictor has more than two values, an analysis of variance (ANOVA) model can be used to characterize the statistical significance of the predictors.

however, if the means of the categories are found to be different, the natural next step is to discover which are different.

it may be helpful to decompose the categories into multiple binary dummy variables and apply the procedure outlined above to determine the relevance of each category.

categorical outcomes

one approach when there are two classes is to use the area under the ROC curve to quantify predictor relevance.

here, we use the predictor data as inputs into the ROC curve.

if the predictor could perfectly separate the classes, there would be a cutoff for the predictor that would achieve a sensitivity and specificity of 1 and the area under the curve would be one.

its remains that a completely irrelevant predictor would have an area under the curve of approximately 0.5

categorical outcomes

when there are multiple classes, one solution is to use a set of “one versus all” ROC curves can be created by lumping all but one of the classes together.

in this way, there would be a separate AUC for each class and the overall relevance can be quantified using the average or maximum AUC across the classes

categorical outcomes

another, more simplistic approach is to test if the mean values of predictors within each class are different.

the t-statistics can be used to compare the predictors on different scales.

categorical outcomes

when the predictor is categorical, there are several metrics that may be appropriate.

for binary predictors and two classes, one effective method for measuring the relevance is the odds ratio

recall that the odds of a probability are p/(1 – p). The probability of an event can be calculated for both levels of the predictor. If these are denoted as p_1 and p_2, the odds ratio is

$$\mathrm{OR} = \frac{p_1(1-p_2)}{p_2(1-p_1)}$$

and represents the increase in the odds of the event when going from the first level of the predictor to the other.

categorical outcomes

although there are formulas for calculating the confidence interval for the odds ratio, a more common approach is to conduct a statistical hypothesis test that the odds ratio is equal to one (i.e., equal odds between the predictor levels)

when there are two categories and two values for the predictor, Fisher's exact test can be used to evaluate this hypothesis and can be summarized by the resulting p-value

categorical outcomes

when there are more than two classes or the predictors have more than two levels, other methods can be applied.

Fisher’s exact test can still be used to measure the association between the predictor and the classes.

alternatively, the gain ratio can be applied to quantify relationship between two variables, where larger is better

other approaches

the Relief algorithm is a generic method for quantifying predictor importance

it was originally developed for classification problems with two classes but has been extended to work across a wider range of problems.

it can accommodate continuous predictors as well as dummy variables and can recognize nonlinear relationships between the predictors and the outcome

it uses random selected points and their nearest neighbors to evaluate each predictor in isolation

other approaches

for a particular predictor, the score attempts to characterize the separation between the classes in isolated sections of the data.

for a randomly selected training set sample, the algorithm finds the nearest samples from both classes (called the “hits” and “misses”)

for each predictor, a measure of difference in the predictor’s values is calculated between the random data point and the hits and misses.

other approaches

for continuous predictors, the distance between the two points be divided by the overall range of the predictor:

$$\text{diff}(x, y) = (x - y)/C$$

where C is a constant for the predictor that scales the difference to be between 0 and 1. For binary (i.e., 0/1) data, a simple indicator for equivalence can be used

$$\text{diff}(x, y) = |x - y|$$

so that these values are also between 0 and 1

other approaches

the overall score (S_i) is an accumulation of the differences such that the score is decreased if the hit is far away from the randomly selected value but is increased if the miss is far away

the idea is that a predictor that shows a separation between the classes should have hits nearby and missed far away

given this, larger scores are indicative of important predictors

other approaches

1 Initialize the predictor scores S_j to zero
2 **for** $i = 1 \ldots m$ *randomly selected training set samples* (R_i) **do**
3 | Find the nearest miss and hit in the training set
4 | **for** $j = 1 \ldots p$ *predictor variables* **do**
5 | | Adjust the score for each predictor based on the proximity of R_j to the nearest miss and hit:
6 | | $S_j = S_j - diff_j(R_j, Hit)^2/m + diff_j(R_j, Miss)^2/m$
7 | **end**
8 **end**

other approaches

this procedure has subsequently been improved

the modified algorithm, called ReliefF, uses more than a single nearest neighbor, uses a modified difference metric, and allows for more than two classes as well as missing predictor values.

additionally, it has been adapted the algorithm for numeric outcomes

other approaches

XgboostExplainer and LIME are newcomers with the goal making black-box models more easily interpretable

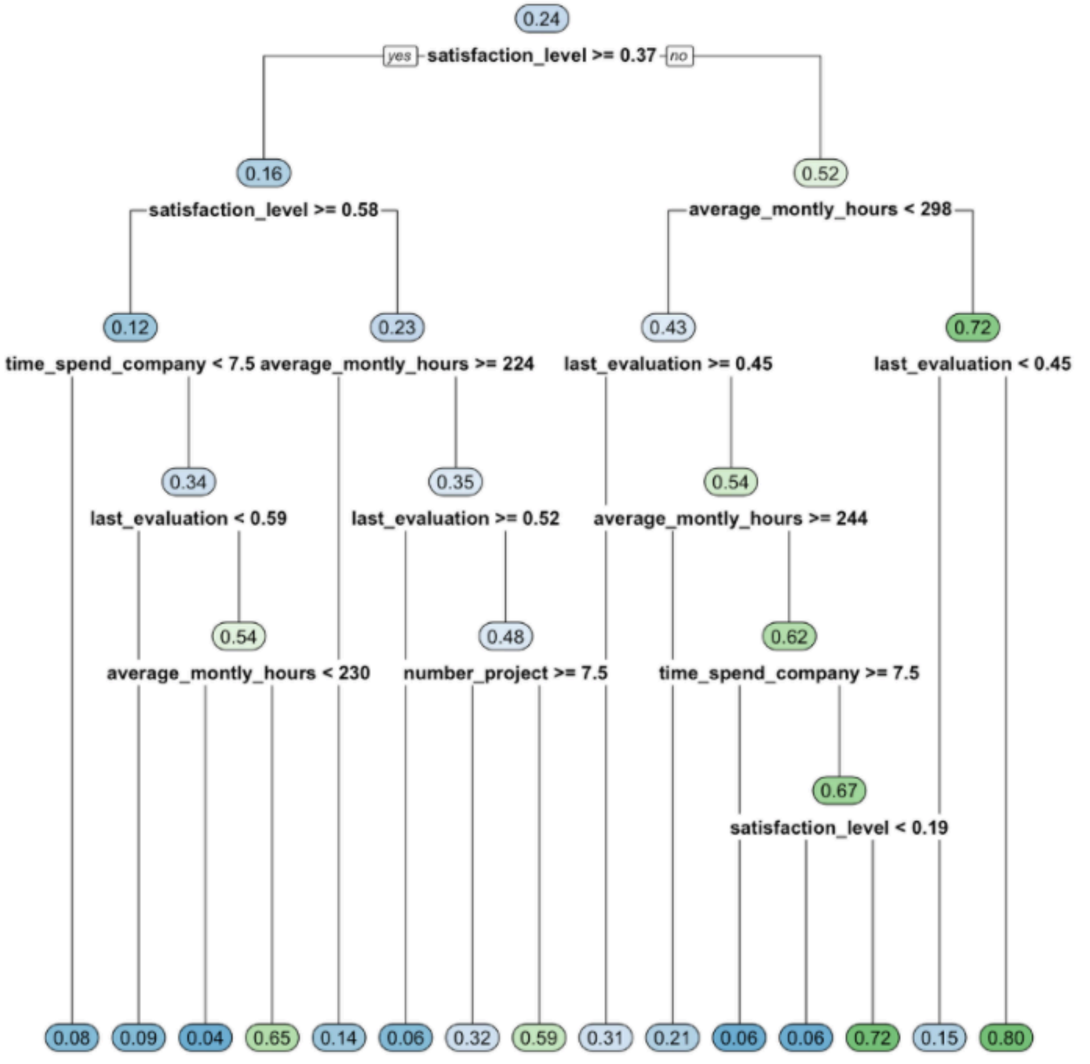
> XgboostExplainer is specifically built for explaining XgBoost (obviously). It has also been extended to lightGBM but won't be discussed here
>
> LIME is a more flexible framework with is named being derived as an acronym for the desired properties of Local-fidelity, Interpretability, Model-agnosticism, Explainability

let's take a look at an example of trying to predict employee attrition from a fictional company using XgboostExplainer

XgBoostExplainer

remember, a single classification and regression trees are fully explainable but are weak learners

predictions made using a CART tree are entirely transparent - ie you can say *exactly* how each feature has influenced the prediction



A decision tree to predict employee attrition. The prediction is the label on each leaf node (eg 0.59 means 59% chance of leaving)

lets say we have an employee with the following attributes

satisfaction level = 0.23

average monthly hours = 200

last evaluation = 0.5

the model would estimate the likelihood of this employee leaving at 0.31 (ie 31%).

following this employee's path through the tree, we can easily see the contribution of each feature

```
0.24 ::: Baseline
+0.28 ::: Satisfaction Level (prediction is now 0.52)
-0.09 ::: Average Monthly Hours (prediction is now 0.43)
-0.12 ::: Last Evaluation (prediction is now 0.31)

= 0.31 ::: Prediction
```

the model would say that the poor satisfaction score is making it more likely that the employee is going to leave (+0.28), whereas the average monthly and last evaluation scores actually decrease the likelihood of leaving (by -0.09 and -0.12 respectively).

the decision tree scores 0.823 AUC but could be better - any further growth of the tree would have led to over-fitting

training an Xgboost ensemble model with 42 trees drastically outperform the single decision tree with an AUC of 0.919

unfortunately to-date we can’t follow a prediction the same way we have with a single tree

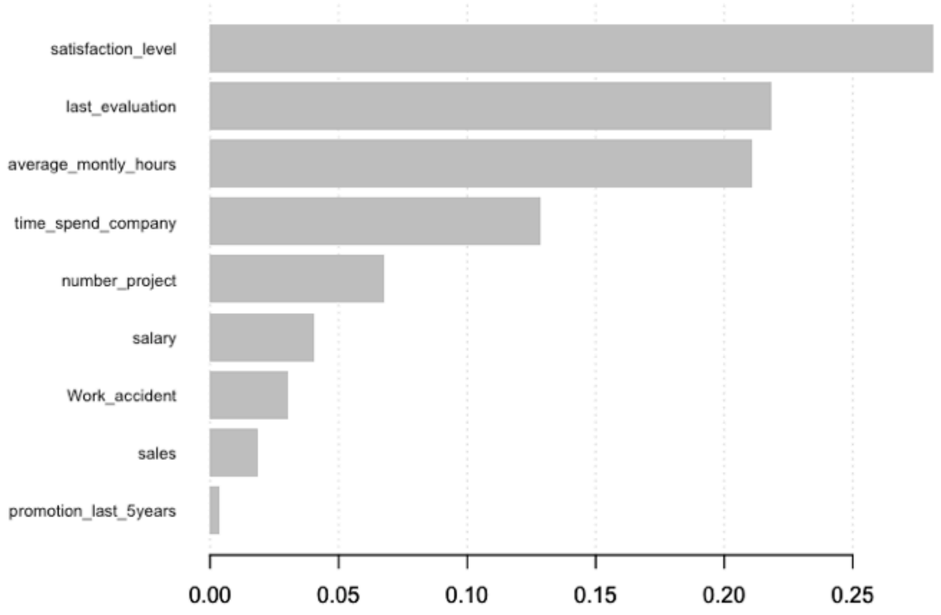
and is ultimately a main disadvantage of these models

satisfaction_level	0.6389651
last_evaluation	0.8507813
number_project	7
average_montly_hours	268
time_spend_company	6
Work_accident	0
promotion_last_5years	0
sales	RandD
salary	medium

now for this employee the model predicts a 21.4% likelihood of leaving

since there are now 42 trees, each contributing to the prediction, it becomes difficult to judge the influence of each individual feature

If someone wants to know why this employee had a 21.4% chance of leaving the best you can do is show the importance graph of each variable

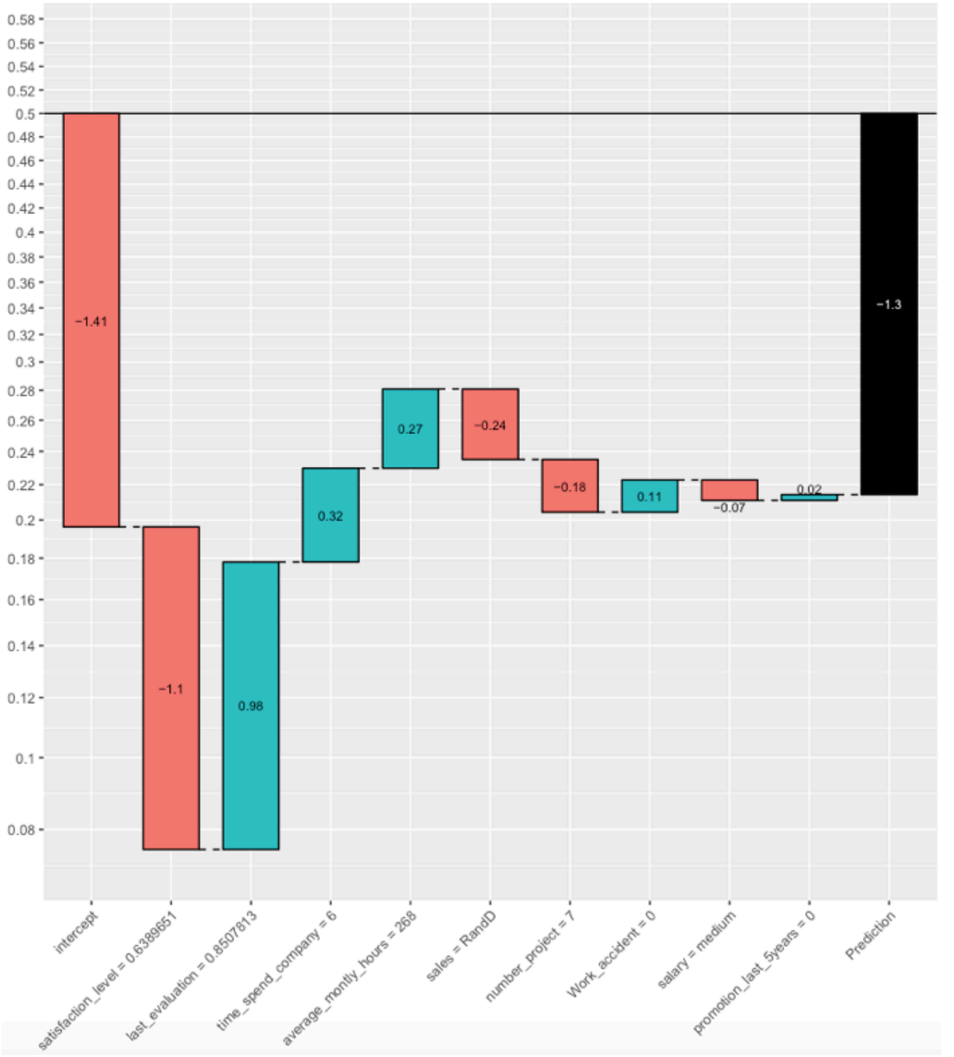


this shows that satisfaction level is the most important variable across all predictions, but there is not guarantee its most important for this particular employee

also, the x-axis is not particular intuitive in a business setting

through the lens of the the XgBoostExplainer

```
Prediction:  0.2142523
Weight:  -1.299481
Breakdown
                 intercept     satisfaction_level
                -1.40792649            -1.10388912
           last_evaluation     time_spend_company
                 0.97700268             0.32087247
  average_montly_hours                      sales
                 0.27201501            -0.24048891
            number_project          Work_accident
                -0.17556633             0.11294757
                    salary promotion_last_5years
                -0.07439384             0.01994550
```



The explanation of the log-odds prediction of -1.299 (y-axis shows the probability, the bar labels show the log-odds impact of each variable)

the prediction of 21.4% is broken down into the impact of each individual feature.more specifically, it breaks down the log-odds of the prediction, which in this case is -1.299.

walking through step by step, just like we did for the decision tree, except this time working with log-odds

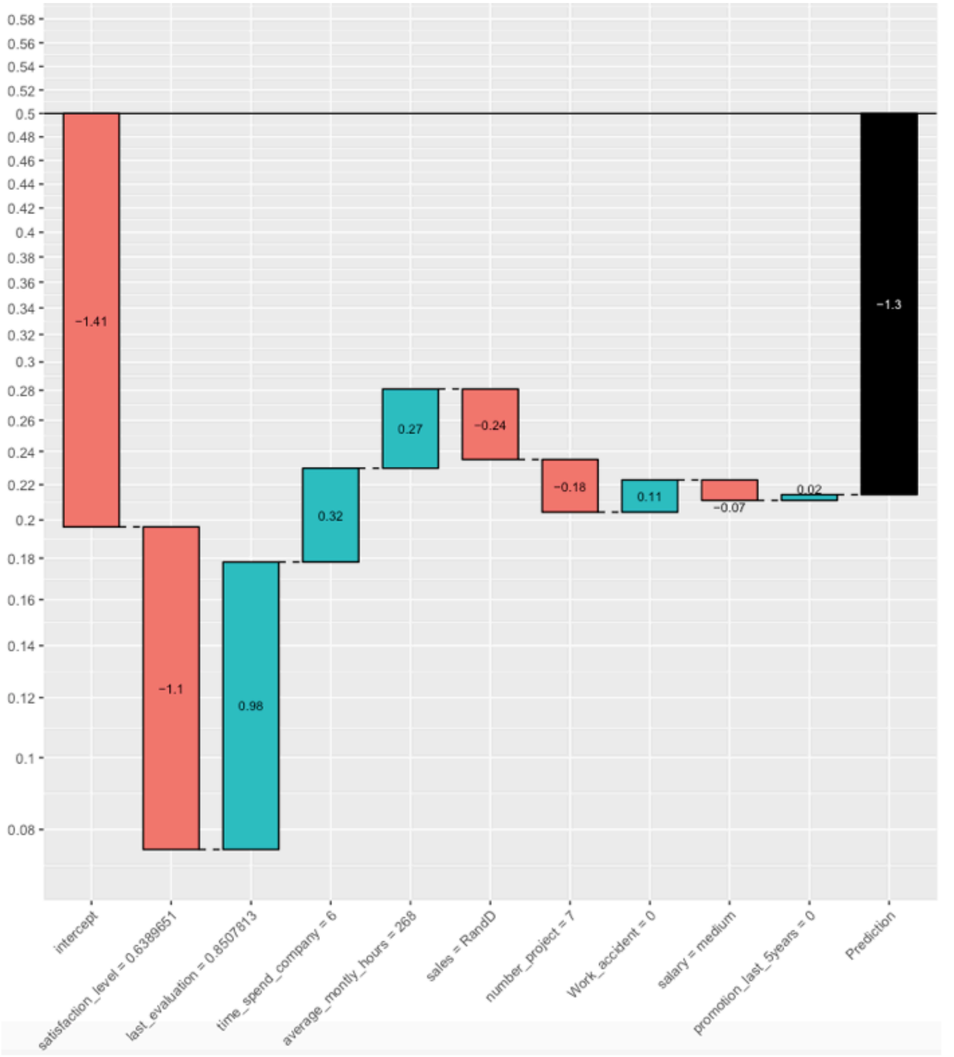
```
-1.41 ::: Baseline (Intercept)
-1.10 ::: Satisfaction Level (prediction is now -2.51)
+0.98 ::: Last Evaluation (prediction is now -1.53)
+0.32 ::: Time Spent At Company (prediction is now - 1.21)
+0.27 ::: Hours Average Monthly (prediction is now -0.94)
-0.24 ::: Sales (prediction is now -1.18)
-0.18 ::: Number of Projects (prediction is now -1.36)
+0.11 ::: Work Accident (prediction is now -1.25)
-0.07 ::: Salary (prediction is now -1.32)
+0.02 ::: Promotion Last 5 Years (prediction is now -1.30)

= -1.299 ::: Prediction
```

the waterfall chart shows how probability of leaving changes with the addition of each variable, in reverse order of the absolute impact on the log-odds.

the intercept reflects the fact that this is an imbalanced dataset - only around 20% of employees in the training set were leavers.

the high satisfaction score of the employee does indeed pull the predicted log-odds **down** (by-1.1)**,** but this is more than cancelled out by the impacts of the last evaluation, time at company and average hours variables, all of which pull the log-odds **up**.



The explanation of the log-odds prediction of -1.299 (y-axis shows the probability, the bar labels show the log-odds impact of each variable)

how does it work

the key point to understand is how the log-odds contribution of each feature is calculated— nothing more than adding up the contributions of each feature for every tree in the ensemble, in the same way as we did for the decision tree.

The R xgboost package contains a function 'xgb.model.dt.tree' that exposes the calculations that the algorithm is using to generate predictions

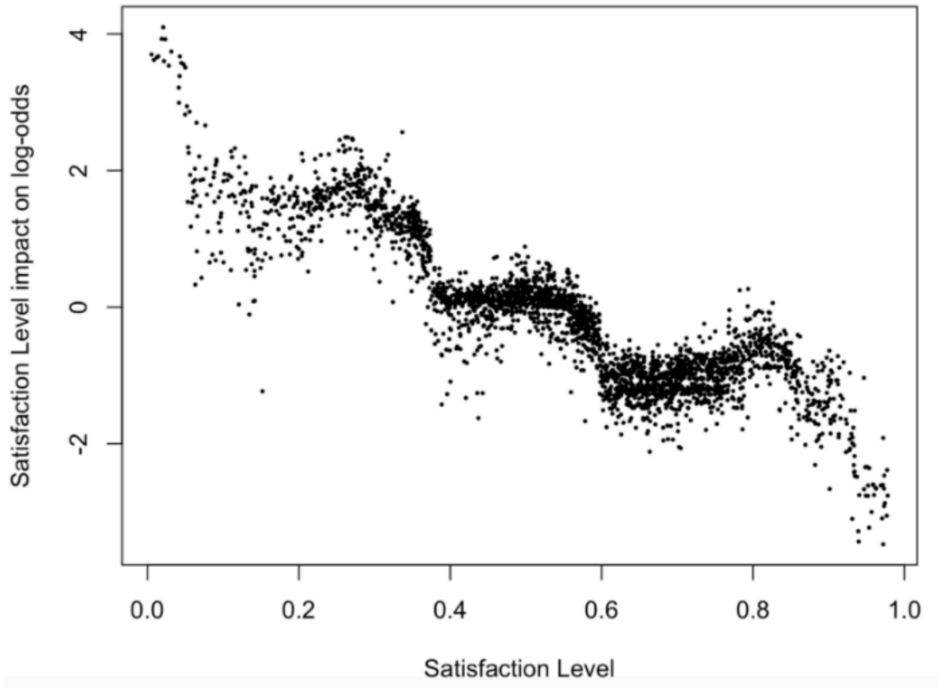
xgboostExplainer extends this, performing the additional calculations necessary to express each prediction as the sum of feature impacts

note: the 'impacts' here are not static coefficients as in a logistic regression. the impact of a feature is dependent on the specific path that the observation took through the ensemble of trees

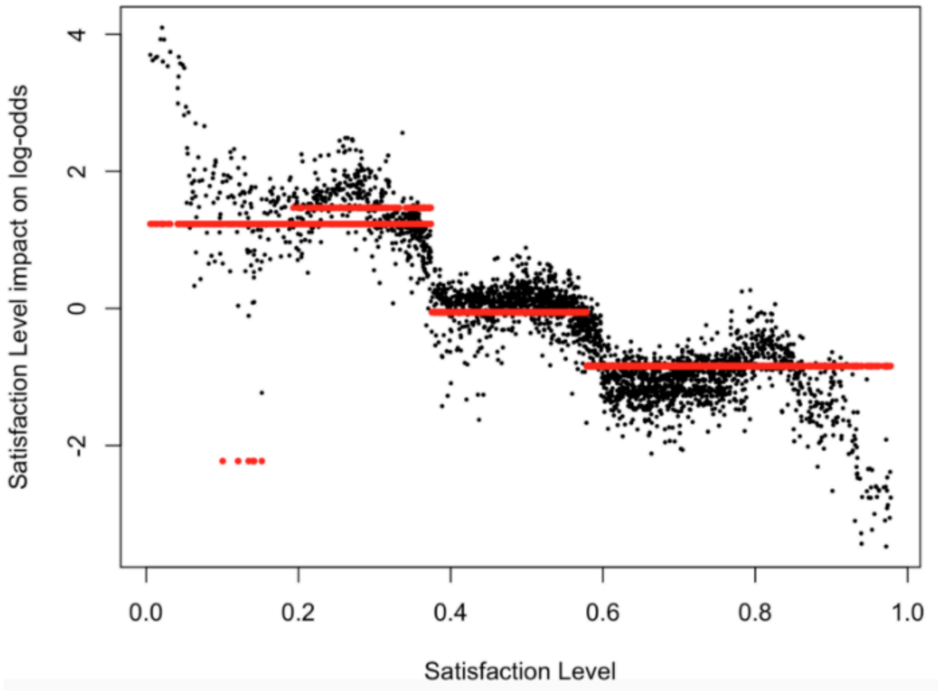
we can plot the impact against the value of the variable, for say, satisfaction level and get insightful graphs like this

if this were a logistic regression, the points would be on a straight line, with the slope dependent on the value of the coefficient

the xgboostExplainer captures the non-linearity, but is not restricted by the requirement of a single straight line or steps



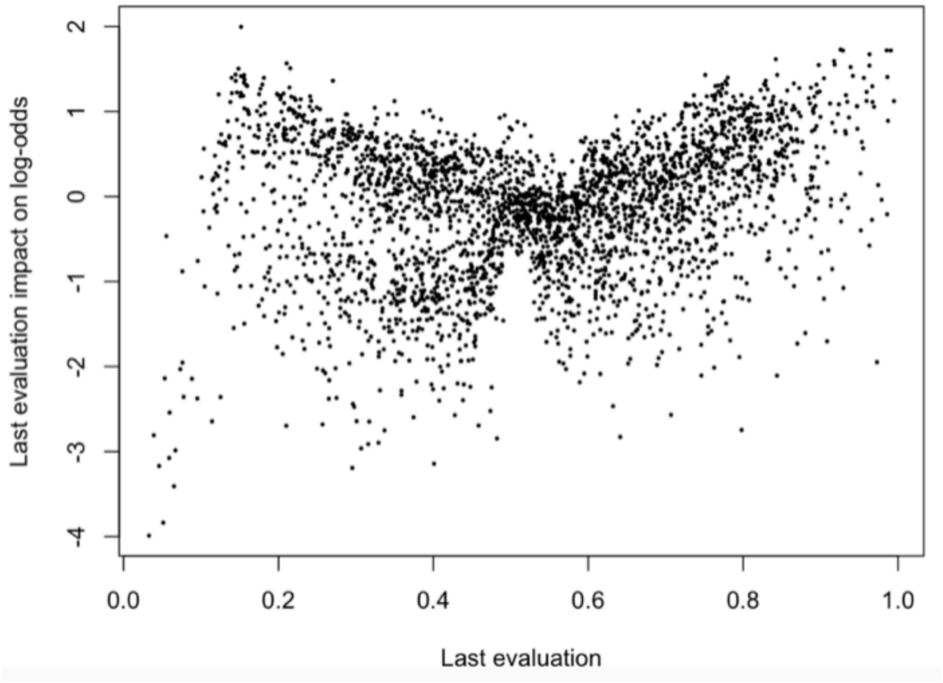
XgBoost model in comparison to single tree



Black: XGBoost Explainer, Red: single decision tree

on first inspection, this doesn’t look too interesting — basically saying that there is large variance in the impact of the last evaluation variable, except perhaps for values around 0.5, where the impact is more likely to be close to zero

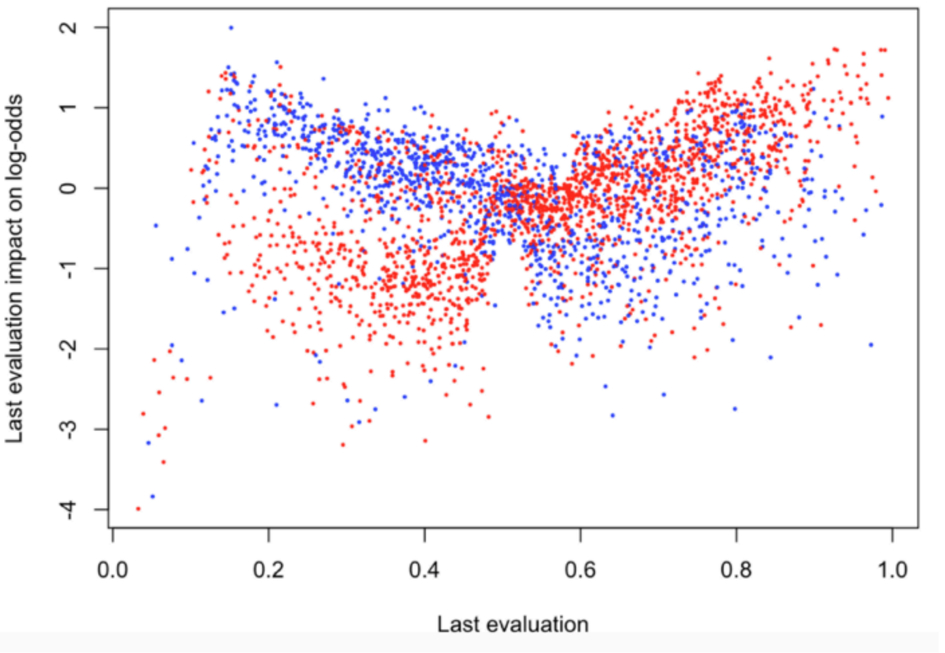
if we color the plot by satisfaction level <0.5 (blue) or ≥0.5 (red), the story becomes clear



employees that are happy (red) are more likely to leave after a **higher** last evaluation score (i.e. talented happy employees are more likely to be poached than less talented happy employees)

employees that are unhappy (blue), are more likely to leave after a **lower** last evaluation score (i.e. the poor evaluation score may be due to a lack of motivation, resulting in the employee switching jobs)

perhaps worryingly, for this fictional company, the model is judging that employees who are happy but scored poorly on the last evaluation are more likely to stick around and not leave.



the abstract of the paper ‘Why Should I Trust?’ Explaining the Predictions of Any Classifier discussing LIME:

> LIME, a novel explanation technique that explains the predictions of any classifier in an interpretable and faithful manner, by learning an interpretable model locally around the prediction. We also propose a method to explain models by presenting representative individual predictions and their explanations in a non-redundant way, framing the task as a submodular optimization problem. We demonstrate the flexibility of these methods by explaining different models for text and image classification (e.g. neural networks)

an R example exists on the course website