Notes: Interpretable Machine Learning

Pablo Carrera Flórez de Quiñones

April 2020

Contents

1 Interpretability

1.1 Importance

In the context of Machine Learning, we define interpretability as the ability to explain or to present something in understandable terms to a human [1]. Despite this non-formal and non-technical definition, interpretability is used to confirm some desirable properties of Machine Learning systems [2]:

- **Causality**: only causal relationships are picked up.
- **Fairness**: predictions do not, implicitly or explicitly, discriminate against protected groups.
- **Privacy**: sensitive information in the data is protected.
- **Robustness**: small changes in the input do not lead to large changes in the prediction.
- **Transferability**: capacity to generalize.
- **Trust**: a system can replace a human in its decisions without supervision.
- **Usability**: predictions provide information that assist users to accomplish a task.

However, not all Machine Learning models require interpretability, explanations are not necessary either because there are no significant consequences for unacceptable results or the problem is sufficiently well-studied and validate in real applications that we trust the system's decision, even if the system is not perfect. The need for interpretability stems from an incompleteness in the problem formalization, creating a fundamental barrier to optimization and evaluation. In the presence of an incompleteness, explanations are one of ways to ensure that effects of gaps in problem formalization are visible to us:

- **Ethics**: the human may want to guard against certain kinds of discrimination.
- **Mismatched objectives**: the agent's algorithm may be optimizing an incomplete objective.
- **Multi-objective trade-offs**: two well-defined properties may compete with each other.
- **Safety**: for complex tasks, the end-to-end system is almost never completely testable.
- **Scientific understanding**: the human's goal is to gain knowledge.

1.2 Properties

In order to explain the predictions of a machine learning model, we rely on some explanation method, which can be thought as an algorithm that generates explanations. These methods can be studied by taking into account the following properties:

- **Algorithmic complexity**: describes the computational complexity of the method that generates the explanation.
- **Expressive power**: is the language or structure of the explanations that the method is able to generate.

- **Portability**: describes the range of machine learning models with which the explanation method can be used.
- **Translucency**: describes how much the explanation method relies on looking into the machine learning model, like its parameters. The advantage of high translucency is that the method can rely on more information to generate explanations. The advantage of low translucency is that the explanation method is more portable.

These explanation methods generate explanations that usually relates the feature values of an instance to its model prediction in a humanly understandable way. We can also analyze these individual explanations from different points of view:

- **Accuracy**: How well does an explanation predict unseen data?
- **Certainty**: Does the explanation reflect the certainty of the machine learning model?
- **Comprehensibility**: How well do humans understand the explanations?
- **Consistency**: How much does an explanation differ between models that have been trained on the same task and that produce similar predictions? ("Rashomon Effect")
- **Degree of importance**: How well does the explanation reflect the importance of features or parts of the explanation?
- **Fidelity**: How well does the explanation approximate the prediction of the black box model?
- **Novelty**: Does the explanation reflect whether a data instance to be explained comes from a new region far from the distribution of training data?
- **Representativeness**: How many instances does an explanation cover?
- **Stability**: How similar are the explanations for similar instances?

and make a draft on how a good explanation [3] should to be:

- **Good explanations are contrastive**: humans usually do not ask why a certain prediction was made, but why this prediction was made instead of another prediction (counterfactual explanation).
- **Good explanations are selective**: people do not expect explanations that cover the actual and complete list of causes of an event. We are used to selecting one or two causes from a variety of possible causes as the explanation.
- **Good explanations are social**: they are part of a conversation, or interaction, between the explainer and the receiver of the explanation. The social context determines the content and nature of the explanations.
- **Good explanations focus on the abnormal**: people focus more on abnormal causes to explain events. These are causes that had a small probability but nevertheless happened. The elimination of these abnormal causes would have greatly changed the outcome (counterfactual explanation).
- **Good explanations are truthful**: they prove to be true in reality.
- **Good explanations are consistent with prior beliefs**: humans tend to ignore information that is inconsistent with their prior beliefs. This effect is called confirmation bias. Explanations are not spared by this kind of bias.
- **Good explanations are general and probable**: a cause that can explain many events is very general and could be considered a good explanation.

1.3 Taxonomy

The explanation methods can be classified by different criteria:

- Result of the explanations?
  - Feature summary statistic
  - Feature summary visualization
  - Model internals
  - Data point
  - Intrinsically interpretable model
- Intrinsic or post hoc?
  - Intrinsic: by restricting the complexity of the machine learning model.
  - Post hoc: by applying methods that analyze the model after training.
- Model-specific or model-agnostic?
  - Model-specific interpretation tools are limited to specific model classes.
  - Model-agnostic tools can be used on any machine learning model and are applied after the model has been trained (post hoc). By definition, these methods cannot have access to model internals such as weights or structural information.
- Local or global?
  - Local: the interpretation method explain an individual prediction.
  - Global: the interpretation method explain the entire model behavior. This level of interpretability is about understanding how the model makes decisions, based on a holistic view of its features and each of the learned components such as weights, other parameters, and structures.

2 Interpretable models

The easiest way to achieve interpretability is to use only a subset of algorithms that create interpretable models. Despite this classification is rather subjective, literature commonly agree to use linear regression, logistic regression and the decision tree are as examples of interpretable models. These models are widely used in industry and research and also well studied in the literature [4].

2.1 Linear Models

Linear models can be used to model the dependence of a regression target Y on some features X. The learned relationships are linear and can be written for a single instance of Y as a weighted sum of its features X as:

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n + \varepsilon$$

where β_j represent the learned weights of the features and ε represent the error we still make between the prediction and the final outcome.

The interpretation of a weight depends on the type of the corresponding feature:

- Numerical feature: An increase of feature X_j by one unit increases the prediction for Y by β_j units when all other feature values remain fixed.
- Categorical feature: Changing feature X_j from the reference category to the other category increases the prediction for Y by β_j when all other features remain fixed.
- Intercept: Prediction when all numerical feature values are at zero and the categorical feature values are at the reference categories. The interpretation is only meaningful when the features have been standardised (mean of zero, standard deviation of one), then the intercept reflects the predicted outcome of an instance where all features are at their mean value.

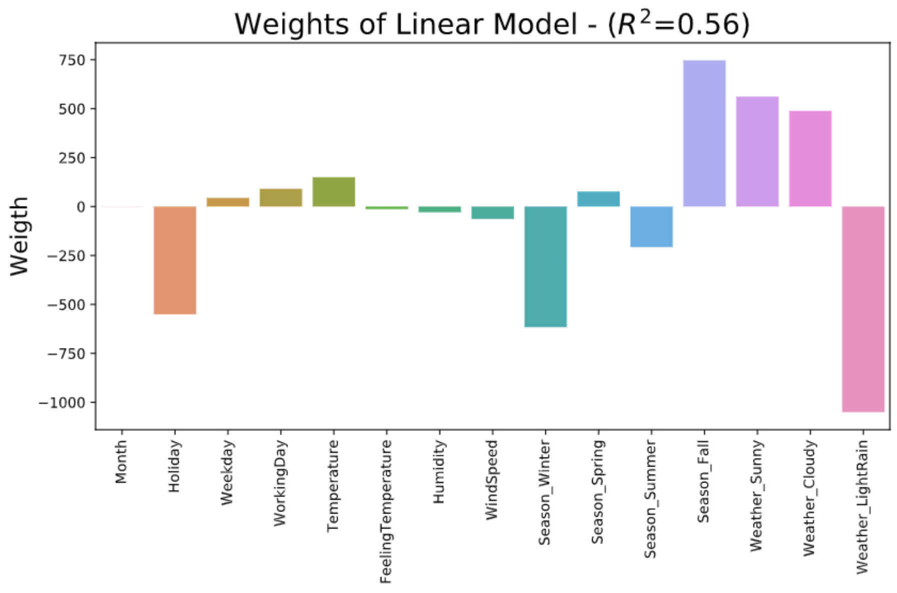

Figure 1: *Example of Linear Regression for the Bike Sharing dataset computed with the library sklearn. In includes both, numerical and categorical features.*

however, if the variables are not on the same scale, the weights cannot completely illustrate the influence of a variable in the predictions. We might not want to lose the interpretability of the real scale of a variable but we still want to perform this comparison, so we can compute the effect of a variable, that is, we can visualize the weights multiplied by the actual feature values in a boxplot.

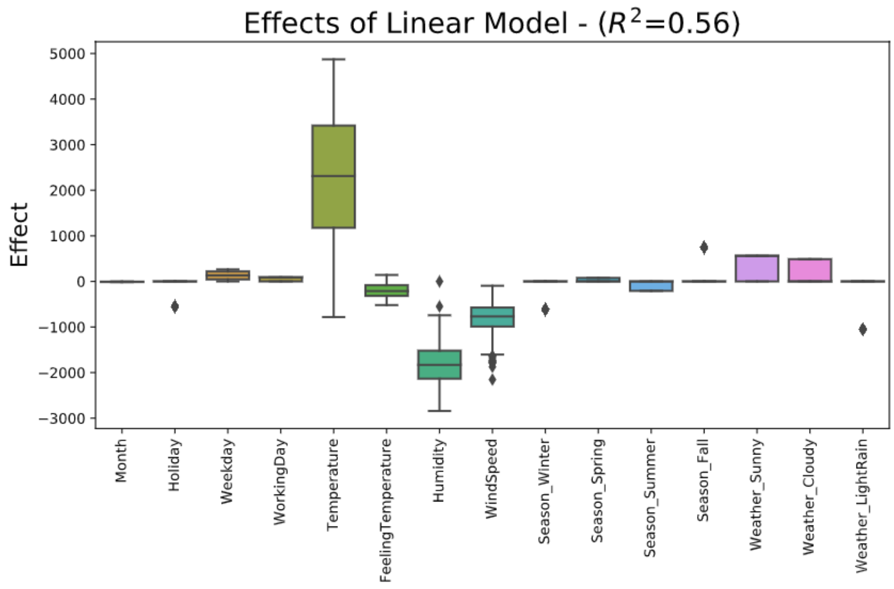

Figure 2: *Effect plot of Linear Regression for the Bike Sharing dataset. We can see that despite its lower weights, temperature and humidity have the highest influences over the predictions, as we could expect.*

Linear models can also be used to solve classification problems, the most straightforward case, logistic regression, models the probabilities for classification problems with two possible outcomes. It's an extension of the linear regression model for classification problems, in the linear regression model, we have modelled the relationship between outcome and features with a linear equation, but, for classification we prefer probabilities between 0 and 1, so we wrap the the equation into the logistic function:

$$P(y=1) = \frac{1}{1+\exp[-(\beta_0+\beta_1 x_1+...+\beta_n x_n)]}$$

The interpretation of the weights in logistic regression differs from the interpretation of the weights in linear regression, since the outcome in logistic regression is a probability between 0 and 1. The weights do not influence the probability linearly any longer, but computing the odds

$$\frac{P(y=1)}{1-P(y=1)} = \exp[\beta_0+\beta_1 x_1+...+\beta_n x_n]$$

we can get interpretations similar to the ones of linear regression:

- Numerical feature: An increase of feature X_j by one unit increases the odds by a factor e^{β_j} when all other feature values remain fixed.
- Categorical feature: Changing feature X_j from the reference category to the other category increases the odds by a factor e^{β_j} when all other feature values remain fixed.
- Intercept: Odds when all numerical feature values are at zero and the categorical feature values are at the reference categories. The interpretation is only meaningful when the features have been standardised.

Do linear models create a good explanation?

- The explanations of a linear model are contrastive, the show differences respect to a reference instance. However, this reference instance is a data point where all numerical features are zero and the categorical features are at their reference categories. This is usually an artificial, meaningless instance that is unlikely to occur in your data or reality.
- The explanations of a linear model are selective, in some sense we can explain the contribution of each variable of the model separated to rest. In some other sense, we cannot explain intuitively a prediction as a whole if we have so many features. However, this selectivity can be achieved in linear models by selecting features or by training sparse linear models such as Lasso.
- The explanations of a linear model are truthful, as long as the linear equation is an appropriate model for the relationship between features and outcome. The more non-linearities and interactions there are, the less accurate the linear model will be and the less truthful the explanations become. However, we can solve this easily, before we train a linear model we can add a feature that literally represents the interaction between the features and then train the model as usual. The solution is elegant in a way, since it does not require any change of the linear model, only additional features to the data.

2.2 Tree Models

Tree based models split the data multiple times according to certain cutoff values in the features. Through splitting, different subsets of the dataset are created, with each instance belonging to one subset. The final subsets are called terminal or leaf nodes and the intermediate subsets are called internal nodes or split nodes. Trees can be used for classification and regression as long as to predict the outcome in each leaf node, the average outcome of the training data in this node is used.

There are various algorithms that can grow a tree, which differ in the possible structure of the tree, the criteria how to find the splits, when to stop splitting and how to estimate the simple models within the leaf nodes. The major implementations are:

- ID3: the algorithm creates a multiway tree, finding for each node the categorical feature that will yield the largest information gain for categorical targets. The algorithm continues recursively on each subset, considering only attributes never selected before. Trees are grown to their maximum size and then a pruning step is usually applied to improve the ability of the tree to generalise to unseen data.
- C4.5: is the successor to ID3 and removed the restriction that features must be categorical by dynamically defining a discrete attribute (based on numerical variables) that partitions the continuous attribute value into a discrete set of intervals. It converts the trained trees into sets of if-then rules. The accuracy of each rule is then evaluated to determine the order in which they should be applied. Pruning is done by removing a rule's precondition if the accuracy of the rule improves without it.
- CART: the algorithm constructs binary trees using the feature and threshold that yield the largest information gain at each node in a classification task, or the largest variance reduction in a regression task. The algorithm continues this search-and-split recursively until a stop criterion is reached. For categorical features, the algorithm tries to create subsets by trying different groupings of categories.

The interpretation of tree models is simple, starting from the root node, we can go to the next nodes though the edges, which tell us which subsets we are looking at, and once we reach the leaf node we get the predicted outcome. That is, we can track a decision through the tree and explain a prediction by the contributions added at each decision node, interpreting each individual prediction as a rule composed by the nodes.

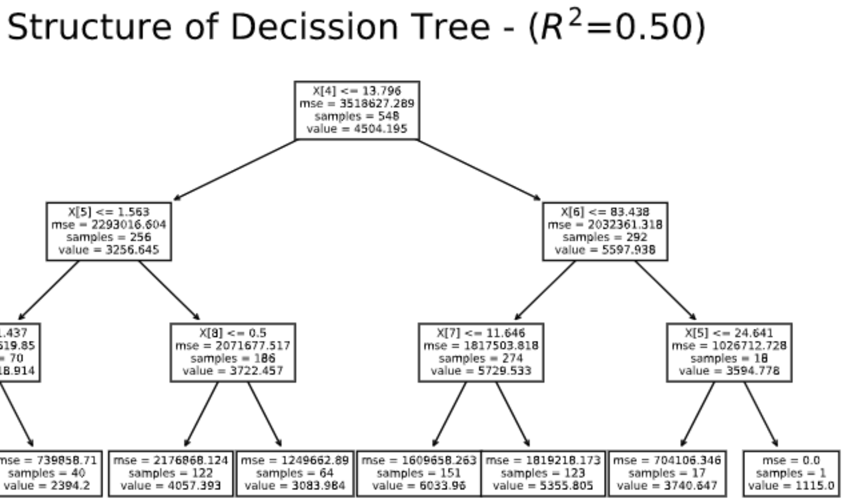
Structure of Decission Tree - (R^2=0.50)
X[4] <= 13.796 mse = 3518627.289 samples = 548 value = 4504.195
X[5] <= 1.563 mse = 2293016.604 samples = 256 value = 3256.645
X[6] <= 83.438 mse = 2032361.318 samples = 292 value = 5597.938
X[5] <= -1.437 mse = 772619.85 samples = 70 value = 2018.914
X[8] <= 0.5 mse = 2071677.517 samples = 186 value = 3722.457
X[7] <= 11.646 mse = 1817503.818 samples = 274 value = 5729.533
X[5] <= 24.641 mse = 1026712.728 samples = 18 value = 3594.778
mse = 378134.449 samples = 30 value = 1518.533
mse = 739858.71 samples = 40 value = 2394.2
mse = 2176868.124 samples = 122 value = 4057.393
mse = 1249662.89 samples = 64 value = 3083.984
mse = 1609658.263 samples = 151 value = 6033.96
mse = 1819218.173 samples = 123 value = 5355.805
mse = 704106.346 samples = 17 value = 3740.647
mse = 0.0 samples = 1 value = 1115.0

Figure 3: *Example of Regression Tree for the Bike Sharing dataset computed with the library sklearn. In includes both, numerical and categorical features.*

We can also interpret the model globally trough feature importance, this can be computed by going through all the splits for which the feature was used and measuring how much it has reduced the variance, or Gini index, compared to the parent node. Then we can scale this with the sum of all importances, to have it in the range $[0, 1]$, so that each importance can be interpreted as share of the overall model.

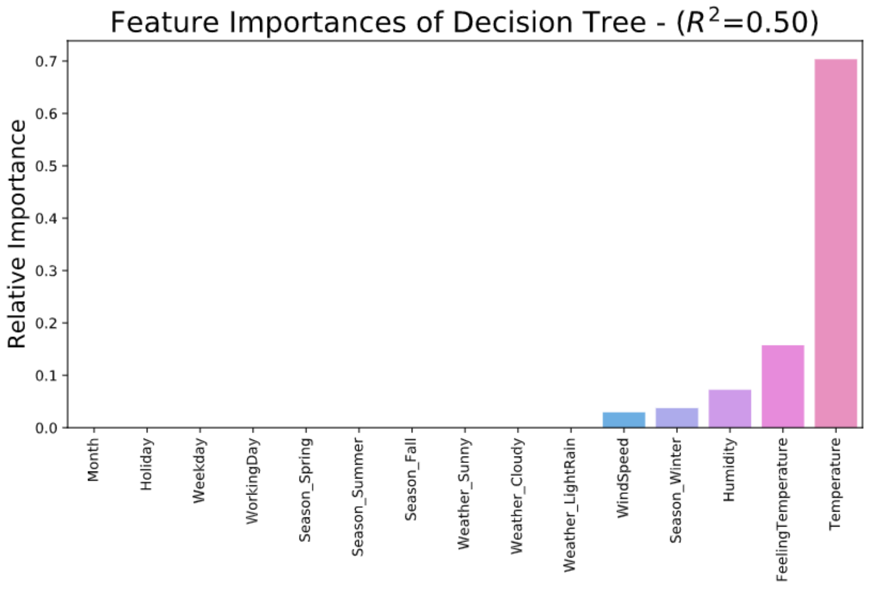

Figure 4: *Feature importances of Regression Tree for the Bike Sharing dataset.*

Do tree models create a good explanation?

- The explanations of a tree model are contrastive, the tree structure automatically invites to think about predicted values for individual instances as counterfactuals. We can always compare the prediction of an instance with relevant alternative scenarios, as defined by the tree, that are simply the other leaf nodes of the tree.
- The explanations of a tree model are selective, if the tree is short, like one to three splits deep, the resulting explanations are not too overwhelming.
- The explanations of a tree model are social, the data ends up in distinct groups that are often easier to understand than points on a multi-dimensional hyperplane as in linear regression.
- The explanations of a tree model are truthful depending on the predictive performance of the tree. Trees fail to deal with linear relationships, any linear relationship between an input feature and the outcome has to be approximated by splits, creating a step function, so slight changes in the input feature can have a big impact on the predicted outcome, which is usually not desirable. Trees are also quite unstable, a few changes in the training dataset can create a completely different tree. This is because each split depends on the parent split. And if a different feature is selected as the first split feature, the entire tree structure changes. It does not create confidence in the model if the structure changes so easily.
- The explanations of a tree model are general, the tree structure has a natural visualization, with its nodes and edges. For each split the instance falls into either one or the other leaf, and binary decisions are easy to understand.

2.3 Rule Models

The problem of association rule mining is defined as follows: let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n binary attributes called items and let $T = \{t_1, t_2, \ldots, t_m\}$ be a set of transactions called the database. A rule is defined as an implication of the form $X \Rightarrow Y$ where $X, Y \subseteq I$, where X is called antecedent and Y consequent. Rules typically take the form of an $\{$`IF 'condition' THEN 'result'`$\}$ expression. An individual rule is not in itself a model, since the rule is only applicable when its condition is satisfied. Therefore rule-based machine learning methods typically identify a set of rules that collectively comprise the prediction model, or the knowledge base.In order to select interesting rules from the set of all possible rules, constraints on various measures of significance and interest are used. Let X, Y be itemsets, $X \Rightarrow Y$ an association rule and T a set of transactions of a given database. Then we define:

- Support: is an indication of how frequently the itemset appears in the dataset. The support of X with respect to T is defined as the proportion of transactions in the dataset which contains the itemset X.
$$supp(X) = \frac{|\{t \in T | X \in t\}|}{|T|}$$
- Confidence: Confidence is an indication of how often the rule has been found to be true. The confidence value of a rule, $X \Rightarrow Y$, with respect to a set of transactions T, is the proportion of the transactions that contains X which also contains Y.
$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$$
- Lift: is defined as the ratio of the observed support to that expected if X and Y were independent.
$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) supp(y)}$$

The best-known constraints are minimum thresholds on support and confidence. Association rules are usually required to satisfy a user-specified minimum support and a user-specified minimum confidence at the same time. Association rule generation is usually split up into two separate steps:

- A minimum support threshold is applied to find all frequent item sets in a database.
- A minimum confidence constraint is applied to these frequent itemsets in order to form rules.

wile the second step is straightforward, the first step needs more attention.

-To create a good classifier you might need to learn not only one rule, but maybe hundreds. Then things can get more complicated and you can run into one of the following problems:

- Rules can overlap: there are two main strategies for combining multiple rules: Decision lists, that introduce an order to the decision rules (if the condition of the first rule is true for an instance, we use the prediction of the first rule), and decision sets, that resemble a democracy of the rules, except that some rules might have a higher voting power.
- No rule applies: this can be resolved by introducing a default rule. The default rule is the rule that applies when no other rule applies. The prediction of the default rule is often the most frequent class of the data points which are not covered by other rules.

There are many ways to learn rules from data:

- OneR learns rules from a single feature.
- Sequential covering is a general procedure that iteratively learns rules and removes the data points that are covered by the new rule.
- Bayesian Rule Lists combine pre-mined frequent patterns into a decision list using Bayesian statistics.

Do rule models create a good explanation?

- The explanation of rule models are contrastive,: Decision rules are probably the most interpretable prediction models. Their IF-THEN structure semantically resembles natural language and the way we think, provided that the condition is built from intelligible features, the length of the condition is short (small number of feature=value pairs combined with an AND) and there are not too many rules
- The explanation of rule models are selective, if the conditions of the rules are short (maximum 3 I would say) and if the rules are organized in a decision list or a non-overlapping decision set. IF-THEN rules usually generate sparse models, which means that not many features are included. They select only the relevant features for the model.
- **Good explanations are social**: Decision rules can be as expressive as decision trees, while being more compact. The prediction with IF-THEN rules is fast, since only a few binary statements need to be checked to determine which rules apply.

- **Good explanations are truthful**: Decision rules are robust against monotonic transformations of the input features, because only the threshold in the conditions changes. They are also robust against outliers, since it only matters if a condition applies or not. However, as trees, rules are bad in describing linear relationships between features and output, they can only produce step-like prediction functions, where changes in the prediction are always discrete steps and never smooth curves. This is related to the issue that the inputs have to be categorical.
- **Good explanations are general and probable**: The research and literature for IF-THEN rules focuses on classification and almost completely neglects regression. While you can always divide a continuous target into intervals and turn it into a classification problem, you always lose information. In general, approaches are more attractive if they can be used for both regression and classification.

ADD THE COMMENT ABOUT GOOD EXPLANATIONS

2.4 K-Nearest Neighbors

The k-nearest neighbor method can be used for regression and classification and uses the nearest neighbors of a data point for prediction. For classification, the k-nearest neighbor method assigns the most common class of the nearest neighbors of an instance. For regression, it takes the average of the outcome of the neighbors. The tricky parts are finding the right k and deciding how to measure the distance between instances, which ultimately defines the neighborhood.

The k-nearest neighbor model differs from the other interpretable models presented in this book because it is an instance-based learning algorithm. How can k-nearest neighbors be interpreted? First of all, there are no parameters to learn, so there is no interpretability on a modular level. Furthermore, there is a lack of global model interpretability because the model is inherently local and there are no global weights or structures explicitly learned. Maybe it is interpretable at the local level? To explain a prediction, you can always retrieve the k neighbors that were used for the prediction. Whether the model is interpretable depends solely on the question whether you can 'interpret' a single instance in the dataset. If an instance consists of hundreds or thousands of features, then it is not interpretable, I would argue. But if you have few features or a way to reduce your instance to the most important features, presenting the k-nearest neighbors can give you good explanations.

ADD EXAMPLE ADD THE COMMENT ABOUT GOOD EXPLANATIONS

2.5 Other models

- **GLM**:The linear regression model assumes that the outcome given the input features follows a Gaussian distribution. This assumption excludes many cases: The outcome can also be a category, a count, the time to the occurrence of an event or a very skewed outcome with a few very high values. The linear regression model can be extended to model all these types of outcomes. This extension is called Generalized Linear Models (GLMs). The GLM mathematically links the weighted sum of the features with the mean value of the assumed distribution (from the exponential family) using the link function g, which can be chosen flexibly depending on the type of outcome:

$$g(E_Y(y|x)) = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$$

where for the classic linear model (Gaussian distribution), as a special case of a GLM, the link function is simply the identity function and for the logistic regression (Bernoulli distribution), also a GLM, the link function is the logit function.

- **GAM**: Generalized Additive Models (GAMs) relax the restriction that the relationship must be a simple weighted sum, and instead assume that the outcome can be modeled by a sum of arbitrary functions of each feature

  $$g(E_Y(y|x)) = f_0 + f_1(x_1) + ... + f_n(x_n)$$

  so in the core of a GAM is still a sum of feature effects, but you have the option to allow nonlinear relationships between some features and the output. Linear effects are also covered by the framework, because for features to be handled linearly, you can make fj(xj)=jxj. These nonlinear functions are usually obtained by a approximation by splines.

- **Naïve Bayes**:

ADD NAIVE BAYES COMMENT THE OTHER MODELS EXAMPLES FOR CLASSIFICATION

3 Model-agnostic Methods

Separating the explanations from the machine learning model has some advantages [5]:

- **Model flexibility**: for most real-world applications, it is necessary to train models that are accurate for the task, irrespective of how complex or uninterpretable the underlying mechanism may be. In model-agnostic interpretability, the model is treated as a black box, the separation of explanations from the model free us it to be as flexible as necessary for the task, enabling the use of any machine learning approach.
- **Explanation flexibility**: different kinds of explanations meet different information needs. By keeping the model separate from explanations, one is able to tailor the model explanation for the information need, while keeping the model fixed, that is, the same model can be explained with different types of explanations, and different degrees of interpretability for each type of explanation.
- **Representation flexibility**: in domains such as images, audio and text, many of the features used to represent instances in state-of-the-art solutions are themselves not interpretable. While an interpretable model trained on such features is still uninterpretable, model-agnostic approaches can generate explanations using different features than the one used by the underlying model.
- **Lower cost to switch**: switching models is not an uncommon operation in machine learning pipelines. If one uses model-agnostic explanations, switching the underlying model for a new one is trivial, while the way in which the explanations are presented is maintained.
- **Comparing two models**: when deploying machine learning in the real world, a system designed often has to decide between one or more contenders. With model-agnostic explanations, the models being compared can be explained using the same techniques and representations.

3.1 Visualization of features

3.1.1 Partial Dependence Plot (PDP)

Partial Dependence Plot (PDP) were developed in [6], and extended in [4], with the idea of studying the effect of a single variable in the predictions of a model. Consider the subset X_S of $l < p$ of the input variables $X = (X_1, ..., X_d)$, indexed by $S \in 1, ..., d$. Let C be the complement set, with $S \cup C = 1, ..., d$. A general function $f(X)$ will in principle depend on all the input variables, so one way to define the average, or partial, dependence of $f(X)$ on X_S is

$$f_S^{PDP}(x_S) = \mathbb{E}_{X_C}[f(x_S, X_C)] = \int_{X_C} f(x_S, x_C)\mathbb{P}(x_C)dx_C$$

in such a way that each subset of predictors S has its own partial dependence f_S^{PDP}, which gives the average value of f when x_S is fixed and x_C varies over its marginal distribution $\mathbb{P}(x_C)$. These partial dependence functions can be estimated by

$$\hat{f}_S^{PDP}(x_S) = \frac{1}{N}\sum_{i=1}^{N} f(x_S, x_{C,i})$$

where $x_{C,i}, ..., x_{C,N}$ are the values of X_C occurring in the training data. Viewing plots (PDP) of the partial dependence approximations on selected variables can help to provide a qualitative description of its properties.

This is a visualization tool in the sense that, if $\hat{f}_S^{PDP}$ is evaluated at the x_S observed in the data, we will end with a set of N ordered pairs $\{x_{S,l}, \hat{f}_{S,l}^{PDP}\}_{l=1}^{N}$, where $\hat{f}_{S,l}^{PDP}$ refers to the estimated partial dependence function evaluated at the l-th coordinate of x_S, denoted as $x_{S,l}$. Then for a one or two dimensional x_S we can plot these N $x_{S,l}$ versus their associated $\hat{f}_{S,l}$, usually with a previous discretization of x_s, conventionally joined by lines, as can be seen in Figure 5. The resulting graphic, which is called a Partial Dependence Plot (PDP), displays the average value of $\hat{f}^{PDP}$ as a function of x_S.

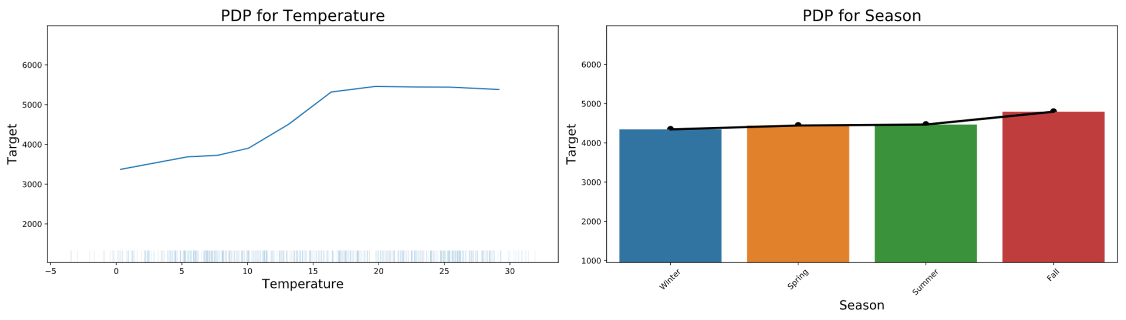

Figure 5: *Example of PDPs for the Bike Sharing dataset modelled by a Random Forest algorithm. We show both, numerical and categorical variables. It can be seen that the temperature has intuitive effect over the number of bikes shared but the season doesn't have an intuitive interpretation.*

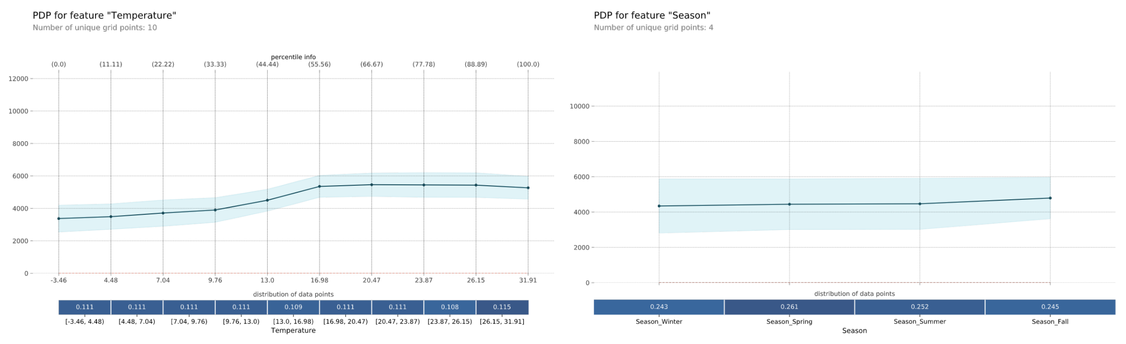

Figure 6: *Sophisticated version of Figure 5 using the library PDPbox.*

On the one hand, the computation of PDPs is intuitive, easy to implement and the interpretation is clear, the plot shows how the average prediction in our dataset changes when the j-th feature is changed. Note here that partial dependence functions represent the effect of X_S on $f(X)$ after accounting for the effects of the other variables X_C, they are not the effect of X_S on $f(X)$ ignoring the effects of X_C. Note also that the realistic maximum number of features in a partial dependence function is two.

But on the other hand, we can observe that it is assumed that the features for which the partial dependence is computed are not correlated with other features, if this assumption is violated the averages calculated for the PDP will include data points that are very unlikely or even impossible. And also heterogeneous effects might be hidden because PDPs only show the average marginal effects.

3.1.2 Individual Conditional Expectation (ICE)

As we said before, one problem of PDPs is that the computation of the average might be hidden heterogeneous effects. One way to address this issue is proposed in [7]. Now consider the observations $(x_{S,i}, x_{C,i})_{i=1}^{N}$, and the estimated response function f. For each $x_{C,i}$ of the N observed and fixed values of x_C, a curve

$$f_{S,i}^{ICE}(x_S) = f(x_S, x_{C,i})$$

is plotted against observed values $x_{S,i}$ of x_S. Therefore, at each coordinate x_S is fixed and the x_C varies across N observations. Each curve defines the conditional relationship between x_S and f at fixed values of x_C. Thus, the ICE algorithm gives the user insight into the several variants of conditional relationships estimated by the black box model. Note that the PDP curve is the average of the N ICE curves.

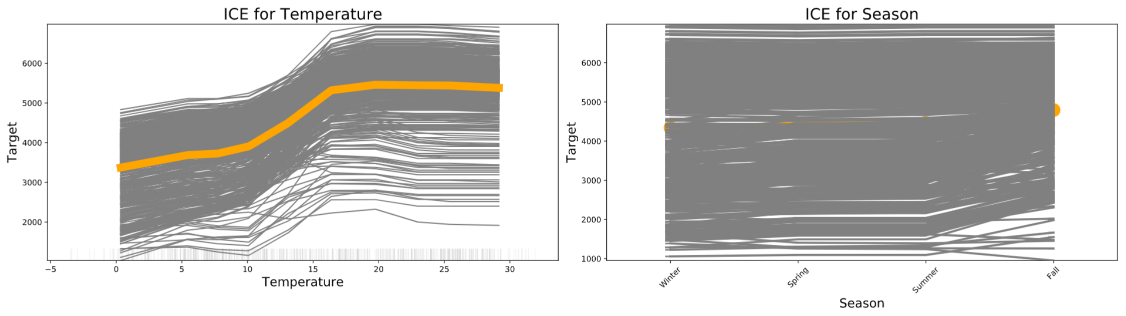

Figure 7: *Example of ICEs for the Bike Sharing dataset modelled by a Random Forest algorithm. We show both, numerical and categorical variables. We can see that, despite the trend in predictions is practically the same in all samples, some of the seem to bee more affected by a change of the dependent variable than anothers.*

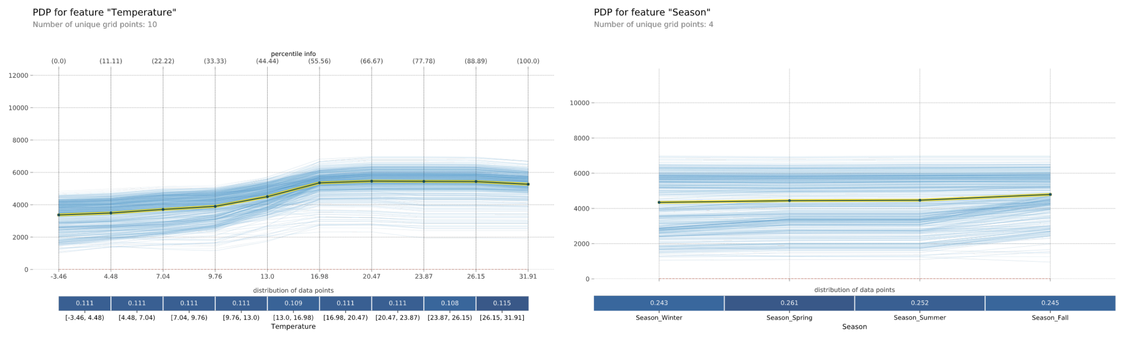

Figure 8: *Sophisticated version of Figure 7 using the library PDPbox.*

When the curves have a wide range of intercepts and are consequently stacked on each other, heterogeneity in the model can be difficult to discern. In such cases the centered ICE plot (c-ICE), which removes level effects, is useful. For constructing a c-ICE choose a location $x*$ in the range of x_S and join all prediction lines at that point. Choosing that $x*$ as the minimum or the maximum observed value results in the most interpretable plots. For each curve $f_{S,i}$ in the ICE plot, the corresponding c-ICE curve is given by

$$f_{S,i}^{ICE,c}(x_S) = f_{S,i}^{ICE}(x_S) - f(x^*, x_{C,i})$$

where the point $(x*, f(x*, x_{C,i}))$ acts as a base case for each curve. If x^* is the minimum value of x_S, for example, this ensure that all curves originate at 0, thus removing the differences in level due to different $x_{C,i}$. At maximum x_S value, each centered curve's level reflects the cumulative effect of x_S on f relative to the base case.

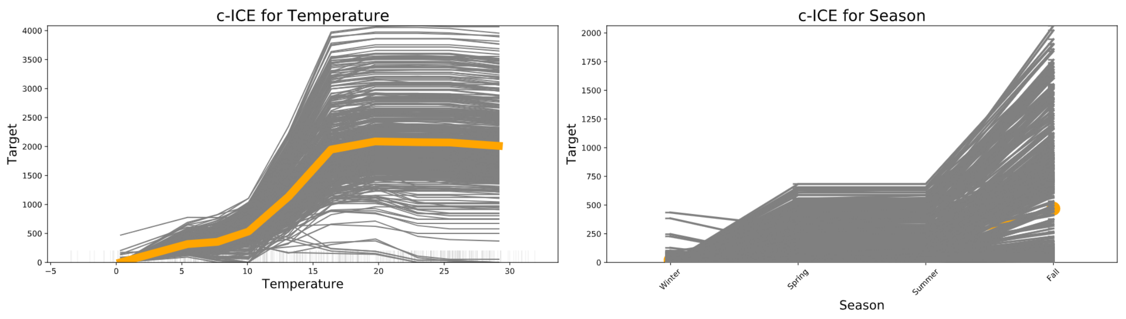

Figure 9: *Example of c-ICEs for the Bike Sharing dataset modelled by a Random Forest algorithm. We show both, numerical and categorical variables. We can see that, with this setup, the cumulative effects of the change in predictions due to changes in a variable can be interpreted more intuitively.*

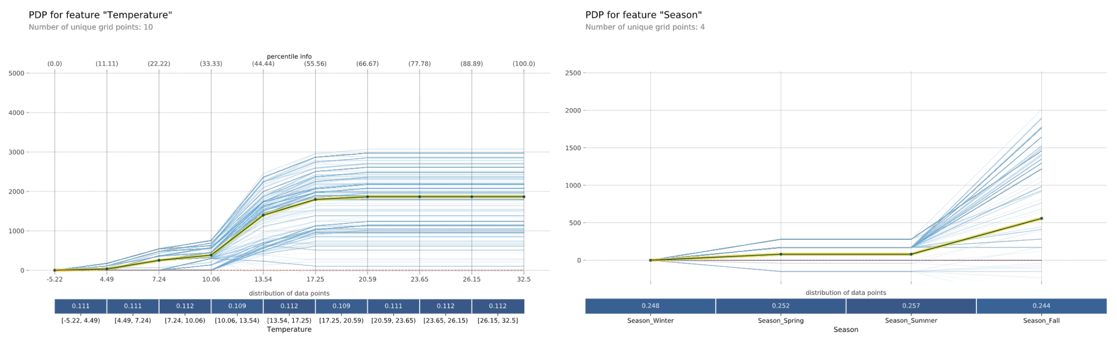

Figure 10: *Sophisticated version of Figure 9 using the library PDPbox.*

On the one hand ICE curves are more intuitive to understand than PDP curves, one line represents the predictions for one instance if we vary the feature of interest and they can uncover heterogeneous relationships.

But on the other hand, ICE curves can only display one feature meaningfully and if many ICE curves are drawn, the plot can become overcrowded and it might not be easy to see the average. If the feature of interest is correlated with the other features, as in the case of PDPs, then some points in the lines might be invalid data points according to the joint feature distribution.

3.1.3 Accumulated Local Effects (ALE)

The other problem of PDPs and ICEs is that both approaches does not take into account the possible correlation between the variables in X_S and the variables in X_C, generating points in regions that might not be available. One solution to this is replacing the marginal probability $\mathbb{P}(x_C)$ by the conditional probability $\mathbb{P}(x_C|x_S)$, this creates the so called Marginal Plots(MP), described by

$$f_S^M(x_S) = \mathbb{E}_{X_C}[f(x_S, x_C)|X_S = x_S] = \int_{X_C} f(x_S, x_C)\mathbb{P}(x_C|x_S)dx_C$$

that can be estimated by

$$\hat{f}_S^M(x_S) = \frac{1}{n(x_S)} \sum_{i \in N(x_S)} f(x_S, x_{C,i})$$

where $N(x_s)$ is the subset of data points for which $x_{S,i}$ falls in some small, appropriately selected neighborhood of x_S, and $n(x_s)$ is the number of observations in the neighborhood. Namely, using these approach is like regressing $Y = f(X_S, X_C)$ onto X_S while ignoring X_C, which is equivalent to consider the joint distribution of (Y, X_S) after marginalizing across X_C. However, despite solving the problem of correlation, omitting the effects of X_C usually leads to bad results.

The solution to the problems of PDPs and MPs is proposed in [8] with the concept of Accumulated Local Effects (ALE). Now, we assume that each variable X_j has a compact support $\mathcal{S}_j = [x_{j,min}, x_{j,max}]$, then let $\mathcal{P}_j^K = \{z_{j,k}^K\}_{k=0}^K$ be a partition of S_j into K intervals, with $z_{j,0}^K = x_{j,min}$ and $z_{j,K}^K = x_{j,max}$. We define $\delta_j^K = \max_k |z_{j,k}^K - z_{j,k-1}^K|$ as a measure of fineness of the partition. For any $x_j \in S_j$ define $k_j^K(x_j)$ to be the index of the interval $\mathcal{P}_j^K$ into which x_j falls, that is, $x_j \in (z_{j,k-1}^K, z_{j,k}^K]$ for $k = k_j^K(x_j)$. Finally, let $X_{\setminus j}$ denote the subset of $d-1$ predictors excluding X_j. We define the uncentered ALE main effect function as

$$g_j^{ALE}(x_j) = \lim_{K \to \infty} \sum_{k=1}^{k_j^K(x_j)} \mathbb{E}_{X_{\setminus j}} \left[f(z_{j,k}^K, X_{\setminus j}) - f(z_{j,k-1}^K, X_{\setminus j}) | X_j \in (z_{j,k-1}^K, z_{j,k}^K] \right]$$

which converges to

$$g_j^{ALE}(x_j) = \int_{x_{j,min}}^{x_j} \mathbb{E}_{X_{\setminus j}} \left[f^j(X_j, X_{\setminus j}) | X_j = z_j \right] dz_j$$

where $f^j(x_j, x_{\setminus j})$ denotes the partial derivative of $f(x)$ with respect to x_j when the derivative exists. Then, the centered ALE main effect of X_j is defined the same as $g_j^{ALE}(x_j)$, but centered so that is has mean of zero with respect to the marginal distribution of X_j, that is,

$$f_j^{ALE}(x_j) = g_j^{ALE}(x_j) - \mathbb{E}_{X_j}[g_j^{ALE}(x_j)]$$

that can be interpreted as the accumulated local effect of X_j in the following sense: we calculate the local effect $f^j(X_j, X_{\setminus j})$ at $x_j = z_j$, then average this local effect across all values of $x_{\setminus j}$, and finally accumulate this averaged local effect over all values of z_j up to x_j.

To estimate the main effect function $f_{ALE,j}$ of a predictor X_j, let $\{N_j(k) = (z_{j,k-1}, z_{j,k}]\}$ be a sufficiently fine partition of the sample range of $\{x_{i,j}\}_{i=1}^N$ into K intervals, and since this number is fixed we are going to omit it from the notation. Usually $z_{j,k}$ are chosen to be the k/K quantile of the empirical distribution of $\{x_{i,j}\}_{i=1}^N$, with $z_{j,0}$ just below the smallest observation and $z_{j,K}$ as the largest observation. Also let $n_j(k)$ denote the number of training observations that fall into the k-th interval $N_j(k)$, in such a way that $\sum_{k=1}^K n_j(k) = N$. We first compute an estimate of the uncentered effect $g_{ALE,j}$ as

$$\hat{g}_j^{ALE}(x_j) = \sum_{k=1}^{k_j^K(x_j)} \frac{1}{n_j(k)} \sum_{\{i: x_{i,j} \in N_j(k)\}} \left[f(z_{j,k}^K, x_{i,\setminus j}) - f(z_{j,k-1}^K, x_{i,\setminus j}) \right]$$

for each $x \in (z_{0,j}, z_{K,j}]$. The ALE main effect estimator $f_{ALE,j}$ is estimated by subtracting an estimate of the expected value of $\hat{g}_j^{ALE}(x_j)$, that is,

$$\hat{f}_j^{ALE}(x_j) = \hat{g}_j^{ALE}(x_j) - \frac{1}{N} \sum_{i=1}^N \hat{g}_j^{ALE}(x_{i,j}) = \hat{g}_j^{ALE}(x_j) - \frac{1}{N} \sum_{k=1}^K n_j(k) \hat{g}_j^{ALE}(z_{j,k})$$

ADD EXAMPLE

3.2 Global Surrogate Models

The purpose of (interpretable) surrogate models is to approximate the predictions of the underlying model as accurately as possible and to be interpretable at the same time. The idea of surrogate models can be found under different names: approximation model, metamodel, response surface model, emulator, ...

We want to approximate our black box prediction function f as closely as possible with the surrogate model prediction function g, under the constraint that g is interpretable. For the function g any interpretable model – for example from the interpretable models chapter – can be used. For example a linear model Or a decision tree:

Training a surrogate model is a model-agnostic method, since it does not require any information about the inner workings of the black box model, only access to data and the prediction function is necessary. If the underlying machine learning model was replaced with another, we could still use the surrogate method. The choice of the black box model type and of the surrogate model type is decoupled.

One way to measure how well the surrogate replicates the black box model is the R-squared measure, SSE stands for sum of squares error and SST for sum of squares total. The R-squared measure can be interpreted as the percentage of variance that is captured by the surrogate model. If R-squared is close to 1 (= low SSE), then the interpretable model approximates the behavior of the black box model very well. If the interpretable model is very close, you might want to replace the complex model with the interpretable model. If the R-squared is close to 0 (= high SSE), then the interpretable model fails to explain the black box model.

Note that we have not talked about the model performance of the underlying black box model, i.e. how good or bad it performs in predicting the actual outcome. The performance of the black box model does not play a role in training the surrogate model. The interpretation of the surrogate model is still valid because it makes statements about the model and not about the real world. But of course the interpretation of the surrogate model becomes irrelevant if the black box model is bad, because then the black box model itself is irrelevant.

Advantages

- The surrogate model method is flexible: Any model from the interpretable models chapter can be used. This also means that you can exchange not only the interpretable model, but also the underlying black box model. Suppose you create some complex model and explain it to different teams in your company. One team is familiar with linear models, the other team can understand decision trees. You can train two surrogate models (linear model and decision tree) for the original black box model and offer two kinds of explanations. If you find a better performing black box model, you do not have to change your method of interpretation, because you can use the same class of surrogate models.
- I would argue that the approach is very intuitive and straightforward. This means it is easy to implement, but also easy to explain to people not familiar with data science or machine learning.
- With the R-squared measure, we can easily measure how good our surrogate models are in approximating the black box predictions.

Disadvantages

- You have to be aware that you draw conclusions about the model and not about the data, since the surrogate model never sees the real outcome.
- It is not clear what the best cut-off for R-squared is in order to be confident that the surrogate model is close enough to the black box model?
- We can measure how close the surrogate model is to the black box model. Let us assume we are not very close, but close enough. It could happen that the interpretable model is very close for one subset of the dataset, but widely divergent for another subset. In this case the interpretation for the simple model would not be equally good for all data points.
- The interpretable model you choose as a surrogate comes with all its advantages and disadvantages.
- Some people argue that there are, in general, no intrinsically interpretable models (including even linear models and decision trees) and that it would even be dangerous to have an illusion of interpretability.

ADD THE COMMENT ABOUT GOOD EXPLANATIONS IMPLEMENTATIONS? JUST THE PROPOSAL?

3.3 Local Surrogate Models

3.3.1 Local Interpretable Model-agnostic Explanations (LIME)

Developed in [9] and [5], LIME's goal is to identify an interpretable model that is locally faithful to the classifier. Even though an interpretable model may not be able to approximate the original model globally, approximating in the vicinity of an individual instance may be feasible.

Formally, we define an explanation as a model $g \in G$, where G is a class of potentially interpretable models, such as the ones described in section 2, so g can presented to a human with visual or textual artifacts. Let $\Omega(g)$ be a measure of complexity, as opposed to interpretability, of the explanation g. Let the model being explained be denoted as $f : \mathbb{R}^d \to \mathbb{R}$. We further use $\pi_x(z)$ as a proximity measure between an instance z to x, so as to define locality around x. Finally, let $\mathcal{L}(f, g, \pi_x)$ be a measure of how unfaithful is g approximating f in the locality of x in order to ensure interpretability and local fidelity, we must minimize $\mathcal{L}(f, g, \pi_x)$ while having $\Omega(g)$ be low enough to be interpretable by humans, that is,

$$\xi(x) = \text{argmin}_{g \in G}[\mathcal{L}(f, g, \pi_x) + \Omega(g)]$$

so this formulation can be used with different families G, fidelity functions $\mathcal{L}$ and complexity measures Ω.

We estimate $\mathcal{L}$ by generating perturbed samples around x, making predictions with the origin black-box model f and weighting them according to π_x.

ADD LASSO IMPREMENTATION FROM THE PAPER

https://github.com/marcotcr/lime

ADD THE COMMENT ABOUT GOOD EXPLANATIONS

Advantages

- Even if you replace the underlying machine learning model, you can still use the same local, interpretable model for explanation. Suppose the people looking at the explanations understand decision trees best. Because you use local surrogate models, you use decision trees as explanations without actually having to use a decision tree to make the predictions. For example, you can use a SVM. And if it turns out that an xgboost model works better, you can replace the SVM and still use as decision tree to explain the predictions.
- Local surrogate models benefit from the literature and experience of training and interpreting interpretable models.
- When using Lasso or short trees, the resulting explanations are short (= selective) and possibly contrastive. Therefore, they make human-friendly explanations. This is why I see LIME more in applications where the recipient of the explanation is a lay person or someone with very little time. It is not sufficient for complete attributions, so I do not see LIME in compliance scenarios where you might be legally required to fully explain a prediction. Also for debugging machine learning models, it is useful to have all the reasons instead of a few.
- LIME is one of the few methods that works for tabular data, text and images.
- The fidelity measure (how well the interpretable model approximates the black box predictions) gives us a good idea of how reliable the interpretable model is in explaining the black box predictions in the neighborhood of the data instance of interest.
- LIME is implemented in Python (lime library) and R (lime package and iml package) and is very easy to use.
- The explanations created with local surrogate models can use other (interpretable) features than the original model was trained on.. Of course, these interpretable features must be derived from the data instances. A text classifier can rely on abstract word embeddings as features, but the explanation can be based on the presence or absence of words in a sentence. A regression model can rely on a non-interpretable transformation of some attributes, but the explanations can be created with the original attributes. For example, the regression model could be trained on components of a principal component analysis (PCA) of answers to a survey, but LIME might be trained on the original survey questions. Using interpretable features for LIME can be a big advantage over other methods, especially when the model was trained with non-interpretable features.

Disadvantages

- The correct definition of the neighborhood is a very big, unsolved problem when using LIME with tabular data. In my opinion it is the biggest problem with LIME and the reason why I would recommend to use LIME only with great care. For each application you have to try different kernel settings and see for yourself if the explanations make sense. Unfortunately, this is the best advice I can give to find good kernel widths.
- Sampling could be improved in the current implementation of LIME. Data points are sampled from a Gaussian distribution, ignoring the correlation between features. This can lead to unlikely data points which can then be used to learn local explanation models.
- The complexity of the explanation model has to be defined in advance. This is just a small complaint, because in the end the user always has to define the compromise between fidelity and sparsity.
- Another really big problem is the instability of the explanations. In an article 38 the authors showed that the explanations of two very close points varied greatly in a simulated setting. Also, in my experience, if you repeat the sampling process, then the explantions that come out can be different. Instability means that it is difficult to trust the explanations, and you should be very critical.

3.3.2 Scoped Rules (Anchors)

. [10]

3.4 Shapley Values

.

ORIGINAL DEFINION 1956 VS. APPLICATIONS TO ML?

3.5 Shapley Additive Explanations (SHAP)

.[11]

3.6 Para ampliar

- Feature Interaction: If a machine learning model makes a prediction based on two features, we can decompose the prediction into four terms: a constant term, a term for the first feature, a term for the second feature and a term for the interaction between the two features. The interaction between two features is the change in the prediction that occurs by varying the features after considering the individual feature effects. Computed as relative difference between PDPs [12]
- Permutation Feature Importance: We measure the importance of a feature by calculating the increase in the model's prediction error after permuting the feature. A feature is "important" if shuffling its values increases the model error, because in this case the model relied on the feature for the prediction. A feature is "unimportant" if shuffling its values leaves the model error unchanged, because in this case the model ignored the feature for the prediction. [13]

4 Example-Based Explanations

4.1 Counterfactual Explanations

[14]

In the existing literature, "explanation" typically refers to an attempt to communicate the internal state of logic of an algorithm that leads to a decision. In contrast, counterfactuals describe a dependency on the external facts, the inputs of the model, that led to that decision. (Example: You were denied a loan because your annual income was 30k, if your income had been 45k you would have been offered a loan)

Here the statement of a decision is followed by a counterfactual, that is, a statement of how the world would have to be different for a desirable outcome to occur. The concept of "closest possible world" denotes the smallest change that can be made to obtain a desirable outcome. Multiple counterfactuals are possible, as multiple desirable outcomes can exist, and there may be several ways to achieve any of there outcomes, that is, there may exist many "closest possible worlds" and their relevance will depend on the mutability of the variables and its real world probability of change. (Example: A person can increase his annual income but not reduce his age)

Given a model $f_w(x)$ with optimal weights w, we want to find a counterfactual x' as close to the original point x_i as possible such that $f_w(x')$ is equal to a new target y' different from y_i. We can find x' by holding w fixed and minimizing the cost function

$$\text{argmin}_{x'}\text{max}_{\lambda}\mathcal{L}(x_i, x', y', \lambda), \qquad \mathcal{L}(x_i, x', y', \lambda) = \lambda(f_w(x') - y') + d(x_i, x')$$

where $d(.,.)$ is a distance function that measures how far the counterfactual x' is from the original point x_i, this distance is usually the Manhattan distance weighted feature-wise with the inverse median absolute deviation (MAD). The parameter λ balances the distance in prediction, first term, against the distance in feature values, second term. A higher value of λ means that we prefer counterfactuals x' that come close to the desired outcome y', a lower value means that we prefer counterfactuals x' that are very similar to x_i in the feature values. To minimize this loss function, any suitable optimization algorithm can be used, the loss function is minimized for x', and the locally optimal counterfactual x' returned while increasing until a sufficiently close solution is found, that is, $|f_w(x') - y'| < \varepsilon$.

Pros:

- The interpretation of counterfactual explanations is very clear. If the feature values of an instance are changed according to the counterfactual, the prediction changes to the predefined prediction.
- The counterfactual method does not require access to the data or the model. It only requires access to the model's prediction function. This is attractive for companies which are audited by third parties or which are offering explanations for users without disclosing the model or data.
- The method works also with systems that do not use machine learning. We can create counterfactuals for any system that receives inputs and returns outputs.
- The counterfactual explanation method is relatively easy to implement, since it is essentially a loss function that can be optimized with standard optimizer libraries.

Cons:

- For each instance you will usually find multiple counterfactual explanations.
- There is no guarantee that for a given tolerance a counterfactual instance is found. That is not necessarily the fault of the method, but rather depends on the data.
- The method does not handle categorical features with many different levels well.

4.2 Adversarial Examples

4.3 Prototipes and Criticism

5 Model-specific Explanations

5.1 Tree based models

5.2 Neural Networks

Para la motivación: [15] [16]

References

[1] Finale Doshi-Velez and Been Kim. *Towards A Rigorous Science of Interpretable Machine Learning*. 2017. arXiv: 1702.08608 [stat.ML].

[2] Zachary C. Lipton. *The Mythos of Model Interpretability*. 2016. arXiv: 1606.03490 [cs.LG].

[3] Tim Miller. *Explanation in Artificial Intelligence: Insights from the Social Sciences*. 2017. arXiv: 1706.07269 [cs.AI].

[4] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009. ISBN: 9780387848846.

[5] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *Model-Agnostic Interpretability of Machine Learning*. 2016. arXiv: 1606.05386 [stat.ML].

[6] Jerome H. Friedman. "Greedy function approximation: A gradient boosting machine". In: *Ann. Statist.* 29 (2001), pp. 1189–1232. DOI: 10.1214/aos/1013203451.

[7] Alex Goldstein et al. *Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation*. 2013. arXiv: 1309.6392 [stat.AP].

[8] Daniel W. Apley and Jingyu Zhu. *Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models*. 2016. arXiv: 1612.08468 [stat.ME].

[9] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. 2016. arXiv: 1602.04938 [cs.LG].

[10] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *Anchors: High-Precision Model-Agnostic Explanations*. 2018. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16982.

[11] Scott Lundberg and Su-In Lee. *A Unified Approach to Interpreting Model Predictions*. 2017. arXiv: 1705.07874 [cs.AI].

[12] Jerome H. Friedman and Bogdan E. Popescu. "Predictive learning via rule ensembles". In: *The Annals of Applied Statistics* 2.3 (2008), pp. 916–954. DOI: 10.1214/07-aoas148. arXiv: 0811.1679 [stat.AP].

[13] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. *All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously*. 2018. arXiv: 1801.01489 [stat.ME].

[14] Sandra Wachter, Brent Mittelstadt, and Chris Russell. *Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR*. 2017. arXiv: 1711.00399 [cs.AI].

[15] Leo Breiman. "Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author)". In: *Statist. Sci.* 16 (2001), pp. 199–231. DOI: 10.1214/ss/1009213726.

[16] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. https://christophm.github.io/interpretable-ml-book/. 2019.