Explaining black-box models using interpretable surrogates

Explaining Black-box Models Using Interpretable Surrogates

Deepthi Praveenlal Kuttichira , Sunil Gupta , Cheng Li , Santu Rana and Svetha Venkatesh

Deakin University, Geelong, Centre for Pattern Recognition and Data Analytics

{dkuttichira,cheng.li,sunil.gupta,santu.rana,svetha.venkatesh}@deakin.edu.au

Abstract. Explaining black-box machine learning models is important for their successful applicability to many real world problems. Existing approaches to model explanation either focus on explaining a particular decision instance or are applicable only to specific models. In this paper, we address these limitations by proposing a new model-agnostic mechanism to black-box model explainability. Our approach can be utilised to explain the predictions of any black-box machine learning model. Our work uses interpretable surrogate models (e.g. a decision tree) to extract global rules to describe the preditions of a model. We develop an optimization procedure, which helps a decision tree to mimic a black-box model, by efficiently retraining the decision tree in a sequential manner, using the data labeled by the black-box model. We demonstrate the usefulness of our proposed framework using two applications: a classification model built using iris dataset and a regression model built for bike sharing dataset.

Keywords: Model Explainability · Bayesian Optimisation · Gaussian Process.

1 Introduction

Application of artificial intelligence (AI) can be found in almost any field ranging from medical diagnosis to making million dollar decisions. AI algorithms are finding popularity in different domains due to their excellent generalization capabilities. Complex machine learning models like deep neural networks make accurate predictions and classifications. However, there is a trade-off between performance and explainability [10]. Simple models e.g. logistic regression are explainable, but have lower predictive power compared to more complex models like deep neural network, Support Vector Machine (SVM) etc. Usually more complex a model gets, the less interpretable it becomes. With the gaining popularity of complex models, their lack of interpretability is posing a problem. For any model to be effective, it is important for it to be both accurate and explainable. If the decisions made by the model cannot be understood, the model might not be deployed, especially in areas where providing explanation is crucial

like medical diagnosis, law making and finance[5, 6, 2]. Due to the accountability and the consequences of our decisions, we need to provide the basis of decision making with appropriate reasons. Even in non-critical scenarios e.g. malware detection, providing an explanation for decision increases trust in the application [1]. Recently European Union proposed in its regulations that people affected by algorithmic decisions have the right to explanation [9]. With these developments providing explanations to a black-box algorithm has become an urgent step.

Researchers have been trying to come up with explanations for black-box models. The term explanation, is not a monolith [14]. It could mean different things. Explanations can be given in terms of relationship between input features and output. It could also mean algorithmic transparency, *i.e.*, the mechanism of the decision making of an algorithm. Since the term explainability is not precisely defined, multiple approaches have been made to address this problem. These approaches can be broadly classified into two categories. The first one is to provide *post-hoc explanation*, i.e., explaining the output of a black-box model while the other is *algorithmic transparency,* i.e., explaining the internal working of an algorithm. Post-hoc explanations aim to provide explanation for an output through another similar input. Deep learning techniques perform well in image and video classifications. Providing descriptions for image and videos through the outputs of other similar images or videos is a form of post-hoc explanation [11, 12, 17]. Algorithmic transparency aims to come up with explanations for existing models [15, 7, 13]. Explaining existing complex models is challenging. Most of the existing methods try to extract rules by perturbing the test points and observing how these affect the outputs [7, 13]. Others try to learn rules from neural network by training it with a subset of inputs, and observing which neurons get activated for the dominant feature in the subset. A recent work has been done to approximate a complex model using a surrogate model [15].

Although the explainability of black-box model has been tried to tackle in many ways, it still remains an open problem. Post-hoc explanations do not provide any insight into the working of the model. So the underlying model still remains a black-box. Also the mapping of descriptions to outputs remain opaque. The accuracy of novel interpretable algorithms depends a lot on good prior knowledge. Also these algorithms have to be tailor made for each problem. One of the key reasons for the popularity of neural networks is that it can work across different datasets with slight tuning of the hyperparameters. It would be ideal if complex machine learning models that have been successfully deployed could also be explained. Even better if the mechanism of explanation could be agnostic to model type, that is, if explanation mechanism remains same irrespective of whether it is used to explain a neural network, random forest or a gradient boosting model. Such an explanation scheme, then could be implemented as an abstraction layer over the black-box models and can be applied in a wide range of problems. In this paper, we take up this problem and aim to develop a solution. Our work focuses on extracting global rules to describe internal working of a model, while remaining model agnostic. Decision trees are interpretable models as they make decisions based on a hierarchy of decisions. If decision trees can

be modified to mimic a given black-box model, we can use them as a surrogate for the complex black-box model. We make this possible by starting with a black-box model and a decision tree both trained using a training set. Next, we find the region in the input space where the difference between the black-box model and the decision tree model is the highest. We sample a small set of data points in this region and train the decision tree with this data labeled by the black-box model. This process is repeated until the difference between the two models becomes small. After this training process, decision tree rules can be used to explain the internal working of the black-box model. In this paper, we have used a neural network to represent the black-box model. We demonstrate our proposed framework using two applications: a classification model built using iris dataset and a regression model built for bike sharing dataset.

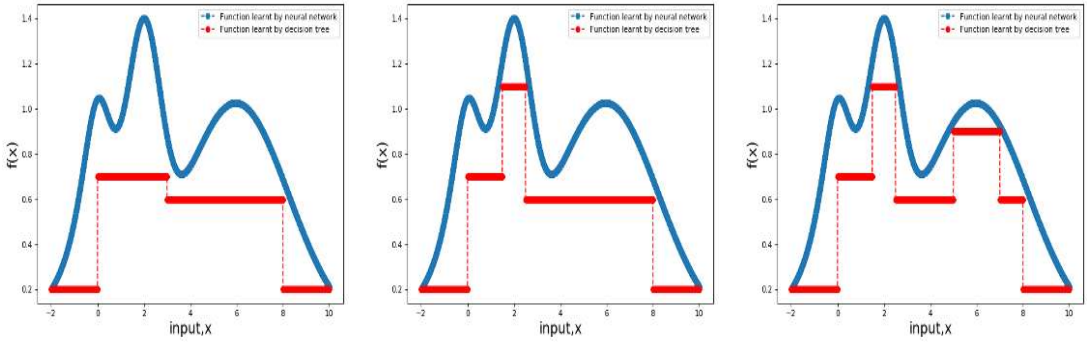
Our contributions can be summarized as below:

- We introduce a novel model-agnostic mechanism for black-box model explainability.
- We develop an optimization procedure to retrain the explainable surrogate model (decision tree) in a sequential manner.
- We demonstrate the usefulness of our proposed framework using two applications: a classification model built using iris dataset and a regression model built for bike sharing dataset.

2 Framework

Complex models are usually black-box. There are many black-box models like neural networks, SVM etc. In our work we have chosen neural network as the black-box model to be explained by a surrogate. Decision tree is capable of modelling non-linear functions. The prediction made by the decision tree can be explained by the hierarchy of decision rules. For this reason, we have chosen decision tree as our surrogate for explanation. The prediction technique of neural network and decision tree are briefly summarized in the subsections 2.1 and 2.2. Both the models are trained with a training set $D^{train} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$,where $x \epsilon \mathbb{R}^d$ and $y = f(x_i)$ is realization from an unknown and smooth function $f : \mathbb{R}^d \rightarrow \mathbb{R}$.

The goal of training any model is to learn f as accurately as possible. We denote f^n as the function learned by neural network and f^t as the function learned by decision tree. We expect $f^n$to model the the input data more accurately than f^t. This is usually true as the prediction accuracy of neural network is more than that of decision tree. Our goal is to approximate the function f^t to that of f^n, as they both are likely to be different as shown in Figure (a) 1. For this, we train the decision tree iteratively using the neural network , to make f^t close to f^n, as desirable in Figure (b) 1. In each iteration of retraining, we train the surrogate model with the data points sampled around the point at which the difference between the functions is maximum. Finding the point of maximum difference is an optimization problem. Since the data point at which the functions differ maximum might be virtual, we denote this point as $\boldsymbol{x}^v \in \mathbb{R}^d$. Then our task has



(a) (b) (c)

Fig. 1: The illustration of function differences before and after introducing new data. (a)The functions learnt from neural network and decision tree with only the initial training data; (b)The functions learnt from neural network and decision tree after introducing new data by following our method. (c) Function learnt after another introduction of datapoints.

reduced to iteratively optimizing the following objective function:

$$\boldsymbol{x}^v = \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmax}} |f^n(\boldsymbol{x}) - f^t(\boldsymbol{x})| \tag{1}$$

where $\mathcal{X}$ is the search space for both the training data and the new data. Note that both f^n and f^t have no tractable form and thus the objective function is a black-box function. We can use Bayesian optimization [4] - an efficient and popular unknown function optimizer to solve the Eq1 .

We illustrate our framework in Figure 2. The first step is to training a neural network and decision tree based on the initial training data (Step 1). The neural network learns a function f^n and decision tree learns f^t. The function f^n by neural network is fixed and we use it only to label a new data. We run Bayesian optimization for several iterations to suggest a new data $\boldsymbol{x}^v$, where function differences (Eq 1) are maximum (Step 2). In order to approximate the function of decision tree to that of neural network at the point $\boldsymbol{x}^v$, we uniformly sample points from $[(1-\delta)\boldsymbol{x}^v, (1+\delta)\boldsymbol{x}^v]$ with the radius coefficient δ. These new samples are labeled using the prediction of the trained neural network (Step 3).We next augment the training data with these samples and retrain the decision tree (Step 4). The steps 2, 3, and 4 are repeated until the performance of the decision tree converges. The algorithm can be found in Alg.1. We give a detailed explanation of the key steps below.

2.1 Training Neural Network

Neural network has a long developing history and it became popular after it has been successfully used for image classification. Neural network is a framework

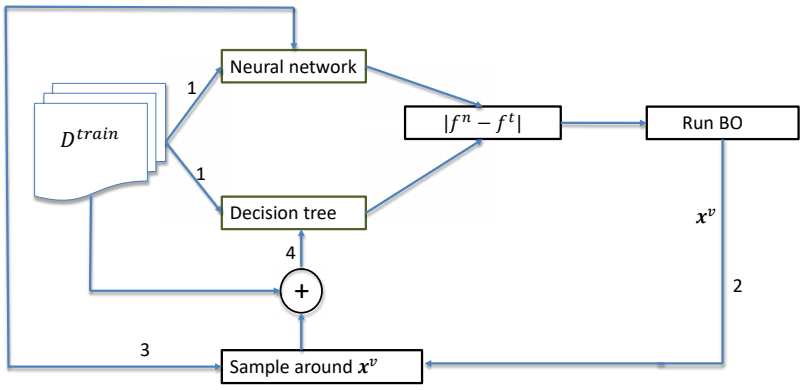


Fig. 2: The diagrams of our algorithm using decision tree to interpret Neural network. We illustrate the detail steps in texts.

describing the relationship between input and output by connected neurons in the hidden layers. A neuron is a decision unit based on the input vector $\boldsymbol{x}$ and the weight vector $\boldsymbol{\omega}$. The output y of a neuron can be formalized as

$$y = f(\boldsymbol{\omega}^\intercal \boldsymbol{x} + \boldsymbol{b})$$

where $\boldsymbol{b}$ is the bias vector and f is the activation function. The activation function has several choices. The popular ones include rectified linear units (ReLU), tanh and sigmoid function.

The training for neural network aims to compute the weights between connected neurons. A reasonable choice is to compute the gradients of the output with respect to the weights and then we can use any gradient based optimizer. Back-propagation makes it possible since we can transmit the gradients by using chain rules. Further details can be found in [3]. The difficulty of the interpretability of neural network is attributed to its highly interconnected hidden layers.

2.2 Training Decision Tree

A decision tree is a tree-like predictive model, which normally starts with a single node and then splits into different branches according to the information gain and finally forms a tree structure. Decision tree can be a classification tree or a regression tree, and has been widely used in data mining and machine learning. A natural question to be asked is which feature can be used to split the node. The standard method is to select the feature which can maximize the purity of classes. The metrics for the purity can be Gini index and information gain.

From the tree structure, we can easily interpret the decision process from an input to its output in a decision tree. Using decision tree as a surrogate model, we expect the function learned by decision tree to approximate the function learned

by neural network. As we stated before, the existing training data might be not sufficient to have the decision tree function, closely approximate the neural network function, since the neural network generally performs well. We next need to find additional data to refine the function of decision tree.

2.3 Bayesian optimization to search new data

Having a trained decision tree and neural network from the original training set, our goal is to minimize the Eq. 1 to find the region in the input space, where the two functions are the most different. We choose the technique of Bayesian optimization to accomplish it, since Bayesian optimization (BO) has demonstrated superior power on a black-box function optimization. To simplify, we describe a generic Bayesian optimization procedure given observations $\{x_j, y_j\}_{j=1}^{J}$, where J is the number of observations, $y_j = f(\boldsymbol{x}_i) + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ and σ^2 is the noise variance. For our problem, the initial training set for BO is $D_0^{BO} = \{x_i, \triangle y_i\}_{i=1}^{n}$ where, x is the training input and $\triangle y$ is the corresponding difference $|f^n(\boldsymbol{x}) - f^t(\boldsymbol{x})|$. In case of regression problem $\triangle y$ is the difference of the predicted real values. In classification the predicted values are class labels. We take the prediction by neural network as the ground truth and calculate the difference as the difference between the probabilities assciated with the class labels.

In Bayesian optimization, we often first use Gaussian process to model the latent function based on observations. Then we can construct an acquisition function to query the next point. Specifically, Gaussian process is a stochastic process where the joint distribution of any point in the domain space is still a Gaussian distribution. Therefore, for a predicted point $\boldsymbol{x}^*$, its predictive posterior distribution is a Gaussian distribution $\mathcal{N}(\mu(\boldsymbol{x}^*), \sigma^2(x^*))$. With a typical zero-mean assumption for GP mean function, we can write the mean and variance:

$$\mu(\boldsymbol{x}^*) = \boldsymbol{k}_{*J}^T (K_{JJ} + \sigma^2 I)^{-1} \boldsymbol{y}_{1:J}$$

$$\sigma^2(\boldsymbol{x}^*) = k_{**} - \boldsymbol{k}_{*J}^T (K_{JJ} + \sigma^2 I)^{-1} \boldsymbol{k}_{*J}$$

where k_{**} is the kernel function, $\boldsymbol{k}_{*J} = [k(\boldsymbol{x}^*, \boldsymbol{x}_1), \cdots, k(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)]$ and K_{JJ} is the Gram matrix between $\boldsymbol{x}_{1:J}$. Note that k is the kernel function representation of the smoothness of the latent function. The popular choice includes the SE-kernel and Matern kernel [16]. The assumptions we make about the data such as the function that models it is smooth etc is incorporated by choosing the kernel function appropriately. In out framework, we have used squared exponential kernel $exp(-\frac{1}{2l^2}||x_i - x_j||^2)$. The hyperparameter l decided the width of then kernel function.

Next based on the built GP before, we want an acquisition function to suggest the next evaluation point. A natural choice is to use a function to measure the possible improvement over the best observation so far (minimal or maximal). The popular ones such as probability of improvement (PI), expected improvement (EI) [8] have been derived. In our work we opted for EI, although other acquisition functions can also be used. We can maximize the acquisition function

to obtain the next point and then update Gaussian process and these steps will be repeated. The BO will return the maxima of the difference function. It is a new data point in the function space that maximizes our objective function. We sample around this new data point in a window of $[(1-\delta)\boldsymbol{x}^v, (1+\delta)\boldsymbol{x}^v]$. The size of δ vary with the complexity of data. It is ideally small, so as to limit sampling close to the optima. This new set of points form X_{new}. The target values Y_{new} for these data points are found using neural networks as $Y_{new} = f^n(X_{new})$. The training data is augumented with these new data points. The steps are repeated until the performance of the decision tree converges. The performance of a predictive model can be measured in multiple ways. A common measure is to use percentage error in classification data and Root Mean Square Error(RMSE) in regression data.

Algorithm 1 The proposed decision tree retraining algorithm.

1. **Input**: Training set $D^{train} = \{X, Y\}$ with $X = \{\boldsymbol{x}_i\}_{i=1}^n$ and $Y = \{y_i\}_{i=1}^n$, the number of rounds K, the number of iterations M at each round, a small radius δ.
2. Train Neural network and Decision tree functions $f^n(\boldsymbol{x})$ and $f^t(\boldsymbol{x})$ using D^{train}.
3. Initialize a set for BO $D_0^{BO} = \{X, \Delta Y\}$ with $\Delta Y = \{\Delta y_i\}_{i=1}^n$, where $\Delta y_i = f^n(\boldsymbol{x}_i) - f^t(\boldsymbol{x}_i)$.
4. **For** $k = 1 : K$ **Do**
5. Build up a GP using D^{BO}
6. **For** $m = 1 : M$ **Do**
7. Recommend $\boldsymbol{x}_m$ using EI acquisition function and compute $\Delta y_m = f^n(\boldsymbol{x}_m) - f^t(\boldsymbol{x}_m)$
8. Update the GP based on the data $D^{BO} \bigcup \{\boldsymbol{x}_i, \Delta y_i\}_{i=1}^m$
9. **end For**
10. Obtain the point $\boldsymbol{x}^v$ corresponding to the maximum function difference $\boldsymbol{x}^v = \text{argmax}_{\boldsymbol{x}_m \in \boldsymbol{x}_{1:M}} |f^n(\boldsymbol{x}_m) - f^t(\boldsymbol{x}_m)|$
11. Obtain a new data set $X_{new} = \{\boldsymbol{x}^v \cup \boldsymbol{x}_{1:J}\}$, where $\boldsymbol{x}_{1:J} \sim \mathcal{U}((1-\delta)\boldsymbol{x}^v, (1+\delta)\boldsymbol{x}^v)$ is uniformly sampled.
12. Use Neural network to label $Y_{new}^n = f^n(X_{new})$
13. Augment the training data with the new data $D^{train} = D^{train} \bigcup \{X_{new}, Y_{new}^n\}$
14. Update f^t via retraining decision tree on D^{train} and recompute $\triangle Y$ for all points in $X = X \bigcup X_{new}$
15. Reset the data for BO $D_0^{BO} = \{X, \Delta Y\}$
16. **end For**

3 Experiments

In this section we show the results obtained by our framework on different datasets. The black-box model we have chosen for our experiments is a simple neural network and the surrogate used to explain it, is a decision tree. We first show the results on synthetic dataset. We demonstrate how the decision

boundary of the decision tree changes after it has been trained by neural network. Further we show the graphs that show the improvement of the accuracy of the predictions of the surrogate model. We have also used our framework on regression and classification data. In both these data, our framework helps in the improvement of accuracy of the prediction.

Experiments with synthetic dataset

The synthetic data was generated using the function $x_1^2 - x_2^2$. If the function value was greater than zero, then the data point was assigned class 1, else class 2. The decision space between -15 and 15 was considered for the experiments. The neural network used for the synthetic data was a simple neural network with 10 input nodes, 10 nodes in the hidden layer and two nodes in the output layer. The decision tree used has a maximum depth of 6. Initially a set of 250 data points was generated. The initial points were randomly split into 75% training data and 25% testing data.

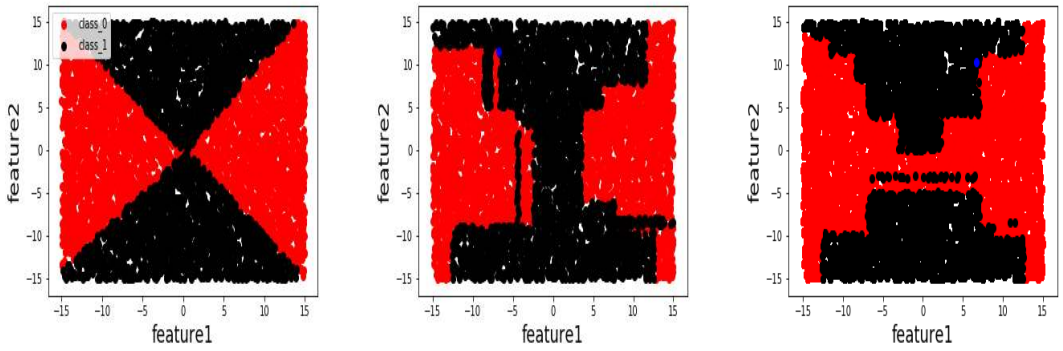
The decision tree was iteratively trained for 30 rounds. At each round a set of 60 data points was appended to the original training set. These 60 datapoints were sampled around the point , were the difference between the decision made by neural network and decision tree was maximum. The class of a point is decided by the class probability associated to it. The difference is calculated as the difference in the class probability predicted by neural network and decision tree. This difference in the function is denoted as Δy. Bayesian optimization was used to find the point of maximum difference. New data points are sampled around this point within a window of $\delta = 0.01$. In Figure 1 the decision boundary of the neural network and the decision boundary of the surrogate before and after the training are shown. After the training the decision boundary of the decision tree resembles more of that of the neural network.

As the boundaries are more similar, the predictions of the surrogate is more similar to that of the black-box model. This is further indicated by the improvement in the accuracy of the decision tree and also by the decrease in the difference of the decision probabilities of the black-box model and the surrogate. This is illustrated in Figure 2.

Experiments with Bike sharing dataset

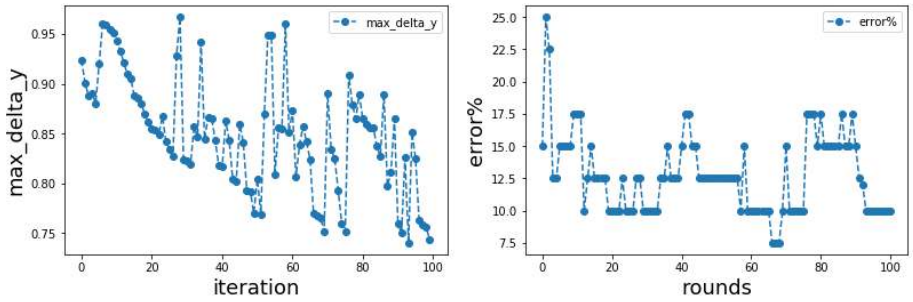
It is a regression dataset from UCI repository. It contains the hourly and daily count of rental bikes with the corresponding weather and seasonal information. We use only the hourly count of rental bikes as the dependent variable. We have selected the most related 5 dependent variables out of 14 based on the correlation between input and output variables. We later use these 5 selected variables ('season', 'hour', 'holiday', 'actual temperature' and 'apparent temperature') to predict the daily count of rental bikes. We randomly split the total 17389 data points into 75% training data and 25% test data.

We train a fully-connected neural network with 1 hidden layer consisting of 30 neurons. We also train a decision tree, for which the appropriate depth



(a) (b) (c)

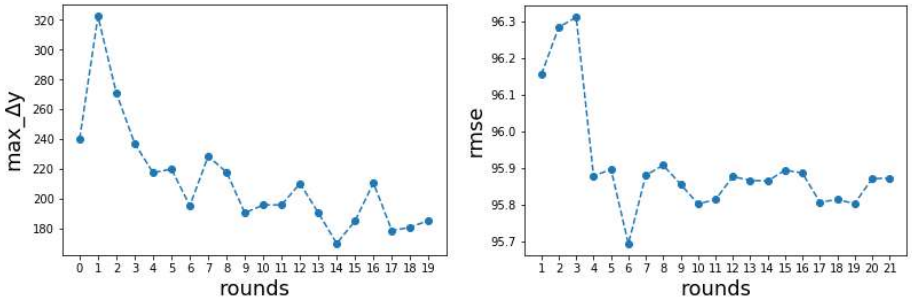
Fig. 3: Decision space. Figure(a) shows the decision boundary of neural network, Figure(b) shows the initial decision boundary of decision tree and the blue point marks the point at which the difference of the neural network function and decision tree function is maximum.and Figure(c) shows the decision boundary of decision tree after training with neural network and the blue point shows the maxima found for that round of training.



(a) (b)

Fig. 4: The change in the maximum difference between the functions and the accuracy of decision tree at each training round. Figure (a) shows the maximum difference between the functions of 2 models at each round and Figure(b) shows the percentage error of decision tree at each round.

is estimated via a validation set as 8. We run our method for 20 rounds. For each round, we run Bayesian optimization with 60 iterations (approximately 12 iterations per feature) to recommend a new point. And we sample 30 new data around the recommended one (within a radius of $\delta = 0.01$). Along with the original training data, we retrain the decision tree with these 30 samples. We show how the maximum difference (Δy) between the functions of neural network and decision tree changes at each round in Figure 5 (a). We can see it is decreasing with the number of rounds, that is to say, the function trained by decision tree would be close to that of neural network. The improvement in the prediction of the decision tree is illustrated by the decreasing trend in the RMSE at each iteration as shown in Figure 5(b). The initial RMSE of test data in decision tree is 96.3 and becomes 95.7 after 20 rounds. We also extracted rules from the decision tree we have obtained finally. We show an example of a subtree and rules extracted from it in Figure 6 and Table 1 respectively.



(a) (b)

Fig. 5: The change in the maximum difference between the functions and the accuracy of decision tree at each training round. Figure(a) shows the maximum difference between the functions of 2 models at each round and Figure(b) shows the RMSE of decision tree at each round.

Rule 1 (Blue)	**If** between 1 am and 5 am, **then** number of bikes rented is 15.92
Rule 2 (Red)	**If** between 12 midnight and 1 am, **and if** spring or summer **then** number of bikes rented is 38.34, **if** fall **or** winter then 22.24
Rule 3 (Green)	**If** after 5 am and temperature less than 13 Celsius **then** number of bikes rented is 13.68

Table 1: Rules Extracted from the subtree in Figure 6.

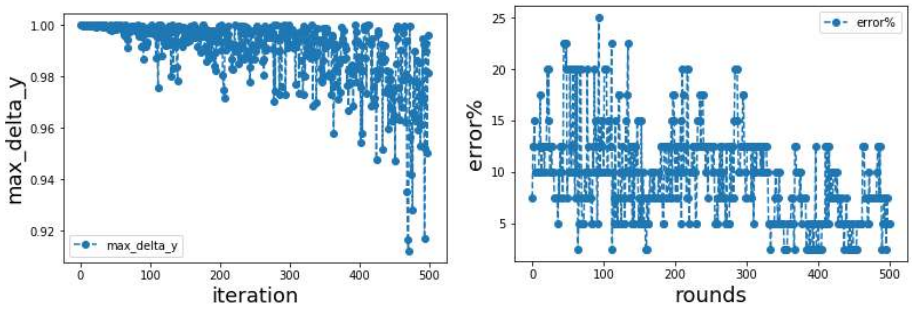
Fig. 6: The examples of the extracted rules from a subtree learnt from bike sharing data. We explain the rules in table 1.

Experiments with IRIS dataset

This data has been used for our classification task, where there are 4 attributes ('sepal length', 'sepal width', 'petal length', 'petal width') and 3 classes of iris flower. We use the same percentage for training data and test data as the bike sharing data. The neural network here is a 1 hidden layer with 10 neurons. We set the depth of decision tree as 3. We run 30 rounds and 60 Bayesian optimization iterations at each round. Since only the class label is available in the classification problem, we use the difference of the class probability from neural network and decision tree as the objective value. Other settings are similar to the bike sharing dataset.

We run our method for 500 rounds. For each round, we run Bayesian optimization with 150 iterations to recommend a new point. As per the suggestion of BO, we again sample 50 new data points around the recommended one (within a radius of $\delta = 0.001$). Along with the original training data, we retrain the decision tree with these additional 50 points. We show how the maximum difference (Δy) between the functions of neural network and decision tree changes at each round in Figure 7 (a). We can see that after a few round of training, the maximum difference between the decision of the neural network and the decision tree has reduced to a low value. In Figure 7 (b), we show the decrease in the error percentage of the decision tree. It demonstrates that the iterative training of the decision tree reduces the difference in the predictions about a data point. Also, it implies that the decision made by the surrogate are more similar to the neural network model.

Figure 8 and Table 2 show some rules that have been learned using our decision tree for iris data.



(a) (b)

Fig. 7: The change in the maximum difference between the functions and the accuracy of decision tree at each training round. Figure(a) shows the maximum difference between the functions of 2 models at each round and Figure(b) shows percentage error of decision tree at each round.

Fig. 8: The examples of the extracted rules from a subtree learnt from iris dataset. We explain the rules in table 2.

Rule 1 (blue)	**If** petal length is between 4.75 and 2.85, **then** Iris Verisicolor
Rule 2 (Red)	**If** petal length less than 2.85 and width less than 1.55 **then** Iris setosa
Rule 2 (Green)	**If** petal length greater than 4.75 and petal width greater than 1.7 **then** Iris Verginica

Table 2: Rules Extracted from the subtree in Figure 8.

4 Conclusion

We presented a new model-agnostic framework for explaining black-box machine learning models. This mechanism is capable of extracting rules of internal working of any black-box machine learning models. We use a decision tree to mimic the black-box model through an efficient training scheme devised by Bayesian global optimization. We applied our proposed framework on two problems: a classification model built using iris dataset and a regression model built for bike sharing dataset. Empirical results demonstrate the usefulness of the proposed framework.

Acknowledgement

"This research was partially funded by the Australian Government through the Australian Research Council (ARC). Prof Venkatesh is the recipient of an ARC Australian Laureate Fellowship (FL170100006)."

References

1. Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss*, volume 14, pages 23–26, 2014.
2. Dylan I Ballard and Amar S Naik. Algorithms, artificial intelligence, and joint conduct. *Antitrust Chronicle*, 2:29, 2017.
3. Christopher Bishop. *Pattern Recognition and Machine Learning*, volume 16, pages 461–517. 01 2006.
4. Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.
5. Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1721–1730. ACM, 2015.
6. Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017.
7. Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence. In *Transparent Data Mining for Big and Small Data*, pages 71–94. Springer, 2017.
8. Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.
9. Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813*, 2016.
10. David Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2017.

11. Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer, 2016.
12. Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. pages 3128–3137, 2015.
13. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.
14. Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.
15. Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. pages 1135–1144, 2016.
16. Christopher K Williams and Carl Edward Rasmussen. Gaussian processes for machine learning. *the MIT Press*, 2(3):4, 2006.
17. Qian Yu, Jingen Liu, Hui Cheng, Ajay Divakaran, and Harpreet Sawhney. Multimedia event recounting with concept based representation. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 1073–1076. ACM, 2012.