A Supervised Machine Learning (ML) model building methodology considering Interpretable Models, in addition to Accuracy

Zhumakhan Nazir*, Dinmukhamed Kaldykhanov*, Kozy-Korpesh Tolep*, and Supervisor: Jurn Gyu Park**

*Undergraduate Student at Nazabayev University, Computer Science, Kazakhstan
**Assistant professor at Department of Computer Science, SEDS, Nazarbayev University, Kazakhstan

***Index Terms*—kernel functions K_t, activation functions A_f, support vectors, interpretability and simulatability of ML models.**

I. Abstract

The key aspects in Machine Learning are data collection and model building. In data collection there are some techniques that enhances the quality of data. Modelling and analysing data are essential since it go through the built model and, consequently, affects to final result. Although, in model building part, the shape, construction, algorithms have a considerable impact on producing conclusive model. Besides high accuracy score of the model, there is another point to consider, the complexity of the algorithm. It might have very high score, but be very complex. The complexity of the model is vital in algorithm's running under the hood, in terms of hardware. In some areas, such as cosmology or medicine, the interpetability of the model comes first. The research work introduce the term simulatability of count (soc) for evaluating the complexity of model. For the purposes, some ML models were investigated and compared in terms of both soc and accuracy score over three different datasets. The Multi–layer Perceptron and support Vector Regression have highest accuracy score, but bigger soc values. While for Linear Model Tree its vice versa. The paper assess the model from both perspectives and propose the new model evaluation metric.

II. Introduction

Considering the **interpretability** of Machine Learning models is an importantly emerging and very challenging issue due to lack of appropriate quantitative metric(s) covering various interdisciplinary domains. The interpretable/explainable ML is an important issue on many academic, industrial and/or governmental machine learning projects. Accuracy of model's is important, however in highly error-sensitive areas such as medicine, cosmology or criminology interpretability of model comes into play. Any decisions made by model should be explained and verified easily. Also the knowledge learnt by models can be very useful if we are able to extract them. Slack et al. (2019) conducted user-study and concluded that if the number of operation to do prediction increases, interpretability of models decreases. Referring to his conclusion, in this research we assess models interpretability in terms of **simulatability** - a user's ability to run a model on a given input [1] and propose model-agnostic quantitative metric for it. Based on the proposed metric, in addition to accuracy, we will evaluate several supervised ML models.

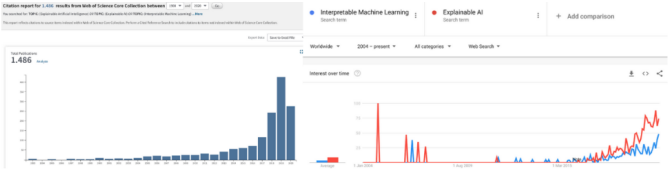

Fig. 1. Left: Citation count on "Interpretable ML or Explainable AI" on Web of Science. Right: "Interpretable ML and Explainable AI" from Google search trends. [2]

III. Related Works

Number of research works on interpretable ML and explainable AI increased exponentially in the last several years. Results in figure 1 are from the Web of Science and Google search from august 2020 [2]. Basically there are two main approaches to explain/interpret models: search the ways to explain existing models(e.g. devising metrics) or using interpretable models in the first place. Rudin (2019) argued that using interpretable models is preferred and they are able to replace complex back box models [3]. Google made its AI related best practices public and one of them was devoted to interpretability [4]. Farquad et al. (2010) extracted if-then based rules from Support Vector Machine with hybrid method: first fit data to SVM and get a reduced set of training data represented by support vectors and train another explainable model [5]. Doshi-Velez and Kim (2017) argued that by analysing human evaluation of interpretability we can build a proxy metric [6]. Slack et al. (2019) performed user study of simulatability and 'what if' local explainability for decision tree, logistic regression and neural network. As a result, the runtime operation count was proposed as a global interpretability metric [1].

IV. Methodology

To evaluate simulatability of ML models we propose a metric called Simulatability Operation Count (SOC) - number of mathematical operations performed during the inference(excluding operations to access memory). We assume that performing boolean operations, multiplication, division, addition, subtraction, exponentiation and trigonometric functions such as tanh cost the same, 1 unit. The SOC is based on the runtime operation count proposed by Slack et al. (2019) with operation count for memory access discarded, since while simulating the prediction users do not make a memory access explicitly and its dependent on underlying hardware. We selected three regression models with different architectures: Linear Model Tree (MT), Multi-layer Perceptron (MLP) and Support Vectors Regression (SVR). To evaluate, following performance datasets are chosen: Forest Fire (hard) [12], AutoMPG (medium) [13] and Servo (easy) [14].

First, SOC formula of the models will be found using the definition of the metric. Then each model will be trained with each dataset and compared in terms of accuracy. After finding best models in terms accuracy, we will try to improve interpretability (reduce SOC score) of the models by changing parameters that affect SOC. In general, it increases the error rate of models' predictions. In order to demonstrate such trade-off between the accuracy and interpretability we allow error rate to be increased by up to 10%. Basic definitions are given in TABLE I. For this project mainly scikit-learn framwork will be used [10].

Params.	Definition
P	number of features selected for prediction
D	number of maximum depth in Tree models
N_i	number of neurons in i-th layer
w_i^l	weight vector for i-th layer and l-th neuron
a_i^l	neuron value for i-th layer and l-th neuron
b_i^l	bias for i-th layer and l-th neuron
b	bias term
n	total number of layers
OC	operation count calculation formula
$\sigma(x)$	activation function
A_{sig}	sigmoid function
SV	number of support vectors
K_t	general name for kernel functions
K_{lin}	linear, $x^T x'$
K_{poly}	polynomial, $(\gamma x^T x' + c_0)^d$
K_{sig}	sigmoid, $\tanh(\gamma x^T x' + c_0)$
K_{rbf}	radial basis, $\exp(-\gamma \|\|x - x'\|\|^2)$

TABLE I
TABLE CAPTION

A. *Linear Model Tree*

1) About model: Model tree is a binary decision tree with linear regression models at the leaf nodes. The use of model and building is described in Quinlan's (1992) account of M5 design [7]. Construction of model tree is similar to decision tree, first according to features of the dataset initial tree is created. The branching criterion for the M5 model tree is rely on standard deviation of the values of features which reached a node as a measure of the error at that node. To calculate expected reduction of the error by testing every attribute at that node. The formula to calculate the standard deviation reduction (SDR) is,

$$SDR = sd(T) - \sum \frac{|T_i|}{|T|} sd(T_i) \tag{1}$$

where T is set of examples which reaches the node, T_i is the subset of examples that have the ith outcome of the potential set, and sd is the standard deviation [8]. Data in child nodes have lower standard deviation because of splitting process, that is why they are more pure. Then all possible splits are tested, and model chooses the one that gives maximum expected error reduction. After, tree is pruned, it merges lower subtrees into a one node to avoid overfitting (problem when model is too accurate, and it usually fails on test data). Then, smoothing process is used for compensation sharp discontinuities among neighbor linear models at leaves of the pruned tree. It occurred when model is used for prediction. Initially, leaf node is used to calculate the predicted value, next that value is filtered way it's back to root, by smoothing it at every node and combining it with the value which is predicted by linear model for that node.

$$p' = \frac{np + kq}{n + k}, \tag{2}$$

where p' represents prediction which passed to upper node, p is the prediction which received from lower node, n is number of training instances which reach the node below, and k is constant equal to 15 [7].

2) SOC of model: SOC for model tree is summation of SOC for tree and linear model at its leaves. SOC of tree is addition of arithmetic and boolean operations, boolean operations are performed to check if the node is a leaf node and for branching operation. Linear model at the leaves has the form $y = a_0 + a_1 x_1 + a_2 x_2$ if dataset has two features. Overall, SOC of model tree depends on maximum depth and amount of features of dataset. It can be described by following formula:

$$SOC = (2 * D + 1) + 2 * no.of features \tag{3}$$

3) Example: For instance, consider the model tree in Figure 2. It has 2 features X1 and X2. We run the model on input $x_1 = 5.5$ and $x_2 = 3$ (red circled). Blue lines represent another example of running models on inputs $x_1 = 4.5$ and $x_2 = 4.3$. The depth of the red circle is 3 while blue one has a depth of 2. There are 4 boolean operations to check branching condition and 3 to check if node is leaf node for the first input. While blue model requires 3 branching and 2 to leaf node checker boolean operations. As it can be noticed, number of operation count depends on depth.

If we consider each leaf node where linear model has the form $y = a_0 + a_1 x_1 + a_2 x_2$. That is why, every leaf node has the same amount of operations. As it can be seen, there are 4 mathematical operations (2 summation and 2 multiplication). SOC for linear model at the leaf node depends on the number of features of dataset.

B. *Multilayer Perceptron*

1) About model: A multilayer perceptron (MLP) is a feed-forward artificial neural network that generates a set of outputs

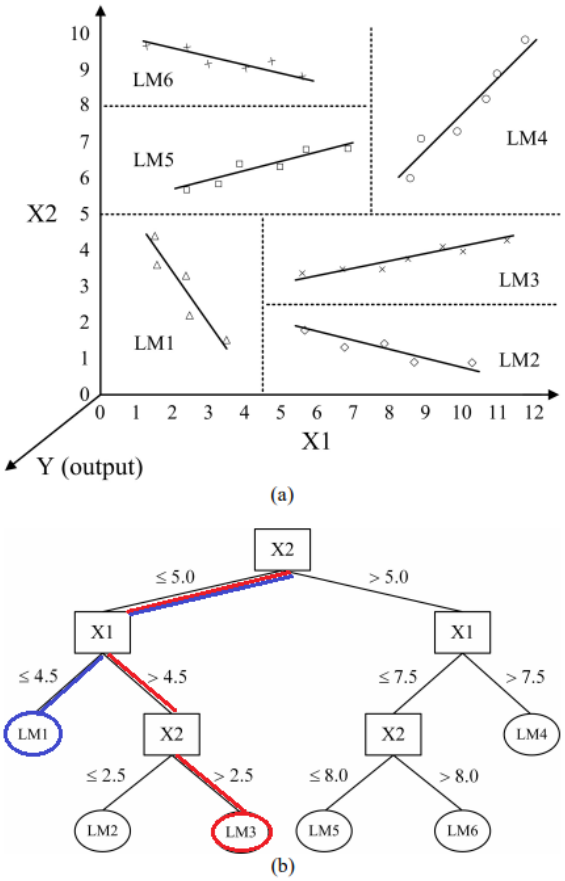

Fig. 2. A model tree where the answer when model runs on the input (x_1 = 5.5, x_2= 2) which is circled red and the blue model runs on the input (x_1 = 4.5, x_2= 4.3) [11]

from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input and output layers. MLP uses backpropogation for training the network [9]. The inputs are pushed forward through the MLP by taking the dot product of the input with the weights that exist between the input layer and the hidden layer, and the bias term has to be added ($w * a + b$). This dot product yields a value at the hidden layer [9].

MLPs utilize activation functions at each of their calculated layers. There are many activation functions to discuss: rectified linear units (ReLU), sigmoid function, tanh. Model pushes the calculated output at the current layer through any of these activation functions,

$$a_i = \sigma(w_i * a_{i-1} + b_i) \tag{1}$$

For the multiple layers repeat steps two and three until the output layer is reached. A simple MLP with a two hidden layer can be represented graphically as follows:

2) SOC of model: Formally, a multiple-hidden-layer MLP is a function f: $R^D \Rightarrow R^L$, where D is the size of input vector and L is the size of the output vector $\sigma(x)$. In the Fig. 1, suppose activation function was applied for the layer 2. The equation for a_2^1 would be as follows,

$$a_2^1 = \sigma(a_1^1 * w_1^1 + a_1^2 * w_1^2 + b_2^1) \tag{2}$$

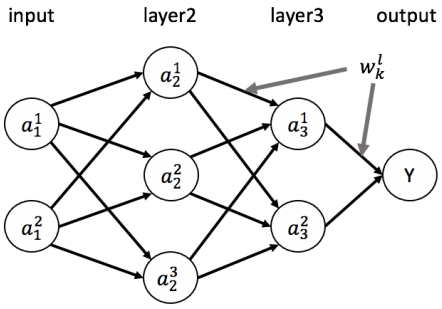

Fig. 3. MLP weight vector.

The neuron has input size 2, hence there is only 2 multiplication and 2 add operations. For instance, for the neuron a_3^1 there is 3 multiplication and 3 additions followed by operation count for activation function. Therefore, the equation 2 has 2 multiplication and 2 addition through activation function; the number of operations is $2(mult.) + 2(add) + OC(\sigma(x))$. The TABLE II describes more on activation functions.

Function type	Formula	SOC
Linear	g(x) = x	1
Logistic (Sigmoid)	g(x) = 1 / (1+e^{-}z)	3
RELU	g(x) = max(x, 0)	1
Tanh	g(x) = (e$^2z - 1)/(e^2x + 1)$	7

TABLE II
ACTIVATION FUNCTIONS SOC

3) Example: In the Fig 1. to calculate second layer outputs the eq. 1 has to be used 3 times as MLP has 3 neurons in the layer. The operation count would be calculated for all (three) layers. Typically, assume one layer has D input vector size and L output vector size. The equation for in-layers operation count would be as follows,

$$OC(D \rightarrow L) = (2 * D + OC(\sigma(x))) * L \tag{3}$$

A number of additions and a number of multiplications are both equal to input size. In order to find out total operation count of the MLP, the equation 3 might be used for all layers except output layer. Note that in the regression last layer does not go through activation function, hence, it might be calculated separately. Going back to the Fig 1. the total operation count is only the summation of the layer operations ($layer2$, $layer3$ and $output\ layer$).

$$OC_{a_1 \rightarrow a_2} = (2 * N_{layer1} + OC(\sigma(x))) * N_{layer2} \tag{4}$$

$$OC_{a_2 \rightarrow a_3} = (2 * N_{layer2} + OC(\sigma(x))) * N_{layer3} \tag{5}$$

$$OC_{a_3 \rightarrow a_4} = 2 * N_{layer3} * N_{layer4} \tag{6}$$

Assume the network uses Sigmoid function for the activation and the OC for sigmoid is 3. By using above equations, the final SOC of the network is,

$$SOC = (2 * 2 + 3) * 3 + (2 * 3 + 3) * 2 + 2 * 2 * 1$$

$$SOC = 21 + 18 + 4 = 43$$

Overall formula for SOC over network :

$$SOC = \Big[\sum_{i=2}^{n-1}(2{*}N_{i-1}{+}OC(\sigma(x))){*}N_i\Big]{+}2{*}N_{n-1}{*}N_n \quad (4)$$

C. *Support Vector Regression*

1) About model: Support vector regression (SVR) is a supervised learning method for regression. As a part of SVM, to make a decision it uses subset of training dataset called support vectors, which are found during the training. Prediction made by:

$$\sum\nolimits_{i\in SV}(\alpha_i - \alpha_i{*})K_t(x_i, x') + b \quad (4)$$

where $\alpha_i - \alpha_i*$ - difference of dual coefficients, x_i - i^{th} support vector, x' - input data and b - bias [10].

2) SOC of model: SOC depends on type of kernel function. In scikit-learn, difference of dual coefficients can be accessed by *dual_coef_* attribute which holds the result of difference. Looking at formula (4), SOC of SVR can be generalized as:

$$\begin{gathered} SOC(SVR) = SV + SV \times SOC(K_t) + \\ SV - 1 + 1 = SV \times \big(SOC(K_t) + 2\big) \end{gathered} \quad (5)$$

where first SV term is for number of multiplications done with the output of a kernel function, and each time kernel function is computed, thus $SV \times SOC(K_t)$. $SV - 1$ comes from number of additions performed in summation operator and last 1 is for adding bias term.

Lets find SOC for different types of kernel functions used in scikit learn, also given in TABLE I.

3) Linear kernel function:

$$SOC(K_{lin}) = P + P - 1 = 2P - 1 \quad (6)$$

P multiplications and $P - 1$ additions

4) Polynomial kernel function:

$$SOC(K_{poly}) = 2P - 1 + 1 + 1 + 1 = 2P + 2 \quad (7)$$

$2P - 1$ for dot product and additional 3 operations for γ, c_0 and power d

5) Sigmoid kernel function:

$$SOC(K_{sig}) = 2P - 1 + 1 + 1 + 1 = 2P + 2 \quad (8)$$

$2P - 1$ for dot product, and 1 for each of γ, c_0 and $\tanh$

6) Radial Basis kernel function:

$$SOC(K_{rbf}) = 3P - 1 + 1 + 1 = 3P + 1 \quad (9)$$

$3P - 1$ operations for computing squared Euclidean distance and 1 multiplication with -γ and 1 for exponentiating.

Overall results are given in table below:

Kernel type	SOC fomula
Linear	SV $\times(2P+1)$
Polynomial	SV $\times(2P+4)$
Sigmoid	SV $\times(2P+4)$
RBF	SV $\times(3P+3)$

TABLE III
SOC FORMULA FOR SVR

V. EXPERIMENTAL RESULTS

1) Forest Fire: A complex dataset has 517 samples and each has 12 features and target variable area described TABLE IV.

Feature	Description
area	in ha, 0 means less that 1ha/100=100m^2
X,Y	x and y coordinates of place of fire
month,date	catergorical value from jan. to dec and mon. to sun corresponsingly
temp,wind, rain	meteorological data
RH	relative humididty
FFMC,DMC,DC, ISI	components of Fire Weather Index (FWI) of the Canadian system

TABLE IV
FOREST FIRE FEATURES

2) AutpMPG: Auto MPG dataset has 398 instances and 8 features including predicting attribute "mpg"(miles per gallon). "The data concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of 3 multivalued discrete and 5 continuous attributes." (Quinlan, 1993). Detailed feature descriptions are given in TABLE V

Feature	Description
mpg	miles per gallon, continuous output variable
model year	version of a car.
cylinders	power unit of engine
displacement	measure of the cylinder volume.
horsepower	power of engine produces.
weight	weight of car.
acceleration	is calculates amount of time taken for car to reach a velocity of 60 miles per hour.

TABLE V
AUTO MPG FEATURES

3) Servo: The dataset has 167 instances and 5 features including target variable. All values of the dataset are discrete, and target variable (labeled 'class') is continuous in [0.13,7.1] and it is a raise time of servomechanism. Feature are given in TABLE VI

Feature	Description
motor	A,B,C,D,E
screw	A,B,C,D,E
pgain	3,4,5,6
vgain	1,2,3,4,5
class	0.13 to 7.10

TABLE VI
SERVO FEATURES

In all datasets categorical variables converted to numerical using OrdinalEncoder and normilized by StandardScaler. Then LocalOutlierFactor with default n_neighbours=20 and z-score

are used to remove outliers from dataset. 10%, 8% and 9% of dataset removed in a result from Forest Fire, Auto MPG and Servo respectively. Features with Varianve Inflation Factor (VIF) greater than 10 counted as collinear with the rest of features and dropped iteratively starting from highest one (O'brien, 2007). Such collinearity is observed in Auto MPG dataset only, as a result *displacement* and *horsepower* were removed from it. GirdSearchCV was used to select best model in terms of accuracy. Strategy as follows: we select few set of hyper-parameters with wide range, if optimal hyper-parameters lie on the edge of range, we shift range so that optimal point lie in the middle. As we get close to optimal point, the range is squeezed to check more point around the optimal one. To validate results, we selected available research experiments with lowest error rate for these datasets and results of default linear regression and decision tree from scikit-learn [10] as a control model. Results are in TABLE VII.

Models\$Datasets$	Servo	Auto MPG	Forest Fire
LMT	0.132 & 19	2.01 & 13	7.180 & 27
MLP	**0.081** & 2840	**1.773** & 910	5.576 & 159
SVR	0.227 & 1185	1.830 & 6264	**5.147** & 17550
Linear Reg.	0.863 & 8	2.304 & 12	6.723 & 24
Decision Tree	0.168 & 15	2.524 & 13	8.453 & 3
Other references	0.220 & - [17]	2.020 & - [18]	6.334 & - [19]

TABLE VII
ACCURACY BASED RESULTS (MEAN ABSOLUTE ERROR & SOC)

Models\$Datasets$	Servo	Auto MPG	Forest Fire
LMT	**17**	**13**	**23**
MLP	638	50	25
SVR	660	795	726

TABLE VIII
SOC BASED RESULTS

Now it is the main part of this research. Interpretability of the models will be improved based on their SOC formula. Results are in TABLE VIII

4) LMT: The SOC for model tree depends on number of features and depth of the tree. To reduce SOC depth and number of features should be reduced. For Auto-Mpg and Forest Fire most accurate model was model tree with maximum depth of 1, since complex trees mostly overfit for unseen data. Thus, to optimize model for mentioned datasets only number of features was decreased. Features were dropped according to it's importance using SelectKBest with f_regression score - correlation level between the feature and the target. For Forest Fire 2 features (DC and ISI) dropped which resulted in optimal SOC and accuracy. However, removing least important features for Auto-Mpg increased error limit more than 10%, thus SOC is unchanged. For Servo, dropping least significant feature highly increase the accuracy, but maximum depth for most accurate model was 5, giving opportunity to decrease it for SOC optimization.

5) MLP: From the Fig. 3 and TABLE II, there are two main parameters that affects to SOC: the neural network structure (hidden layer size) and activation function. Accuracy based modelling was done by using Mean Absolute Error as a scoring metric. The point is finding appropriate amount of layer for neural network using default value for neurons. Afterwards, search out the number of neurons for each layer (same for all layers) until reaching minimum error. For SOC based modelling, since each neuron performs activation function, the function with lowest operation tested. Also, the least influence for SOC change is the number of input features. Hence, it is tested lastly and SelectKBest has been used for feature reduction.

6) SVR: As shown in TABLE III, SOC of SVR depends on a kernel function, a number of features and a number of support vectors. Only RBF kernel is considered for Servo and Auto MPG since other kernels exceed the accuracy limit (10%). For Forest Fire, all kernels can be used, but only linear kernel is selected since it results in lowest SOC by definition. Number of features reduced using SelectKBest. Number of support vectors can be reduced by tuning hyper-parameters (C and *gamma*) or by NuSVR - another equivalent implementation of SVR which takes an additional argument *nu* (lower bound of the fraction of support vectors to be selected). SVR selects too many training instances as support vectors if dataset is hard (95% for Forest Fire), so NuSVR will be used as a tool that reduces number of support vectors of most accurate model. This resulted in 30%, 85% and 53% decrease of number of support vector from Servo, Auto MPG and Forest Fire datasets respectively.

As it was initially predicted SVR has better results in terms of accuracy comparing to other models (TABLE VII), except Servo dataset where it has poorest performance amongst the three models. For Servo dataset, performance of MLP is almost three times better than for SVR, which resulted in rise of SOC for two times. It can be concluded that, for some small datasets like Servo, MLP can have smaller mean absolute error by increasing number of neurons. Also, LMT has almost two times smaller loss than SVR. As a result, for smaller datasets LMT and MLP have better performance.

In terms of SOC, Model Tree is the best candidate for all three datasets as it has from nearly 6 times (comparing to MLP for Forest Fire) to 650 times (comparing to SVR for Forest Fire) better results. Consequently for smaller datasets MLP is more accurate, as for large ones SVR; but their SOC increases too which is noticebly high for SVR. However, after optimization of SOC in TABLE VIII considerable decrease can be noticed in MLP when running Auto MPG, approximately 18 times and SVR running Forest Fire, nearly 24 times. Overall, LMT has better SOC values, but in comparison with MLP results are better 37 times for Servo, and 1.08 times for Forest Fire. The observation of abnormal behavior depends on optimization technique. As it was mentioned SOC of MLP depends on number of neuron layers and amount of input features, so for Forest Fire MLP could safely decrease number of neurons to one and amount of features to 8. As, for LMT which has best model with max_depth of 1 could safely decrease number of features to 10. Thus, different behavior can be noticed for each dataset. Comparing with other models like Linear Regression and Decision Tree, large differences can be seen, especially for Forest Fire where Model tree has SOC of 27 and Decision Tree has 3. Mainly, the reason is SOC of Decision Tree mainly depends on max_depth which is 1. For linear regression, SOC

is equal to 2*number of features. Lowest value for it can be spotted for Servo which has 4 independent features.

VI. Conclusion

In this research, we introduced model agnostic inpterpretability metric (SOC) which easily can be implemented for any other models. LMT and MLP has much less SOC than SVR and LMT is the most interpretable model, while their accuracies are close. As Rudin (2019) argued, this an example of a case when more simpler models can replace complex or black box models. In addition, we observed that SOC is positively correlated to computational and memory resource consumption of the models. This research has demonstrated how model can be optimized in terms of interpretability, power and memory resources by slightly allowing error rate to increase (10% in this research). Such trade off can be useful to build automatic preliminary predictors in error-sensitive tasks (interpretable) and to deploy ML models in resource limited embedded hardwares (resource efficient).

References

[1] Slack, D., Friedler, S. A., Scheidegger, C., Roy, C. D. (2019). Assessing the Local Interpretability of Machine Learning Models. arXiv preprint arXiv:1902.03501.

[2] Molnar, C., Casalicchio, G., Bischl, B. (2020). Interpretable Machine Learning–A Brief History, State-of-the-Art and Challenges. arXiv preprint arXiv:2010.09337.

[3] Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence, 1(5), 206-215.

[4] Carvalho, D. V., Pereira, E. M., Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. Electronics, 8(8), 832.

[5] Farquad, M. A. H., Ravi, V., Raju, S. B. (2010). Support vector regression based hybrid rule extraction methods for forecasting. Expert Systems with Applications, 37(8), 5577-5589.

[6] Doshi-Velez, F., and Kim, B. 2017. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608.

[7] Quinlan, J. R. (1992). "Learning with continuous classes." Proc., 5thAustralian Joint Conf. on Artificial Intelligence, Adams Sterling,eds., World Scientific, Singapore, 343–348

[8] Pal, M., 2006. M5 model tree for land cover classification. Int. J. Remote Sens., 27: 825-831. DOI:10.1080/01431160500256531

[9] Deep learning: Multilayer Perceptron. Gulcehre C. (2015). http://deeplearning.net/tutorial/mlp.html

[10] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[11] Ditthakit, P., Chinnarasri, C. (2012). Estimation of Pan Coefficient using M5 Model Tree. American Journal of Environmental Sciences. 8. 95-103.

[12] Cortez, P., Morais, A. D. J. R. (2007). A data mining approach to predict forest fires using meteorological data.

[13] Auto Mpg dataset. Link: http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data-original

[14] Servo dataset (1993). Link: https://archive.ics.uci.edu/ml/datasets/Servo

[15] Quinlan, J.. "Combining Instance-Based and Model-Based Learning." ICML (1993).

[16] O'brien, R. M. (2007). A caution regarding rules of thumb for variance inflation factors. Quality quantity, 41(5), 673-690.

[17] Cortez, P., Embrechts, M. J. (2013). Using sensitivity analysis and visualization techniques to open black box data mining models. Information Sciences, 225, 1-17. page 17.

[18] Birnbaum, L. A. (Ed.). (2014). Machine Learning Proceedings 1993: Proceedings of the Tenth International Conference on Machine Learning, University of Massachusetts, Amherst, June 27-29, 1993. Morgan Kaufmann. page 240.

[19] Stanford-Moore, A., Moore, B. Wildfire Burn Area Prediction. page 5.