Trade-off Between Performance and Explanation of Classification Models

Poorvi Rai, Han-Ping Lin, Xindi Li, Yifan Zhao
North Carolina State University

ABSTRACT

Supervised machine learning classification algorithms have been successfully applied to various domains. Software engineering is one of them. However, it could be sometimes difficult for machine learning algorithms to be explained to professional software developers. Additionally, better performed algorithms tend to be more difficult to explain. Therefore, there exists a trade-off between explanation and performance of the predictive models. In this paper, we present an empirical study of this trade-off with a well-designed questionnaire and the comparison of it with the model performance metrics. With four representative models, we studied their explanation to both machine-learning-familiar people and the others as well as the training performance of these models.

KEYWORDS

Explanation, performance, trade-off, classification

1 INTRODUCTION

Supervised learning classification algorithms have been successfully applied to various domains. Consequently, a variety of predictive models have been developed so far in order to support human decision making. When evaluating a predictive model, two sets of criteria are widely considered, performance and explanation. Although performance might be the most superior baseline in most circumstances, explanation should not be ignored, either, especially in the domains where the interference of professional knowledge is required. Performance and explanation generally have a trade-off relationship and this trade-off may be different for each domain.

Among these domains, software engineering has been a promising application area for machine learning. In order to deal with the increasing complexity of software development, machine learning predictive models have been applied to software defect detection[5], massive mobile software usage data prediction[15], device type prediction with software usage data[14]. While most researches in this area have been focusing on the performance of various learning algorithms, relatively less exploration has been done on the explanation. Additionally, the definition of explanation metrics of different models could vary significantly. That is also a reason why related research is still actively developing.

In this paper, we present a study concerning on the trade-off between explanation and performance of the predictive models. As the research material, we choose the dataset of six existing real-world configurable systems with different characteristics, sizes, implementation languages and configuration mechanisms. We select four machine learning models as our main research subjects: decision tree, naive Bayes, KNN and neural network. The study consists of two parts: explanation measurement and performance computation. For the state of research concerning explanation, we only found several related papers. Lipton 2016 proposed a qualitative assessment which is comprised of model transparency, component transparency and algorithmic transparency[17]. Chen 2018 presented a empirical comparison of frequently used algorithms in software analytics and introduced a new algorithm FFT which has achieved a considerably improvement on actionable explanation[3].

In this study, we set the following research questions:

RQ1: How should we measure the explanation of a model? Since there are not many studies assessing models explanation[12], we propose a research method using a questionnaire for both people that are familiar with machine learning algorithms and those are not to test the explanation and interpretability of the models. The test is short and easy to understand without too much professional knowledge towards the data mining area and focusing on the first impression of people to the models themselves and their learning results. However, for performance, there are many thoroughly verified metrics for models' performance, accuracy, precision, recall, f1 score, etc. Thus we will not discuss too much about the metrics of performance.

RQ2: Which model is the best in terms of the trade-off? We obtain the answer of this question by comparing the results of both explanation (people's response to the questionnaire) and performance (computed quantitative metrics).

This paper is organized as follows. Section 2 and 3 present the details of studies concerning explanation in other papers and our thoughts about potential approaches to quantify explanation. Section 4.1 presents the dataset we used for model training. Section 4.2 describes the models we selected theoretically. Section 4.3 presents the design of our explanation questionnaire. Section 5 presents the results and evaluation we obtained from the survey and dataset training. And section 6 and 7 cover the validity and conclusion.

2 BACKGROUND

Despite the absence of a definition, papers frequently make claims about the explainability of various models. From this, we might conclude that either: (i) the definition of explainability is universally agreed upon, but no one has managed to set it in writing, or (ii) the term is ill-defined, and thus claims regarding explainability of

Foundations of Software Science, Fall 2018, Computer Science

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$9999.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

various models may exhibit a quasi-scientific character. Our investigation of the literature suggests the latter to be the case. Both the motives for explainability and the technical descriptions of explainable models are diverse and occasionally discordant, suggesting that explainability refers to more than one concept.

In Explaining Explanations[6], the authors evaluated explanation of models based on the four parameters of completeness compared to the original model, completeness as measured on a substitute task, ability to detect models with biases and human evaluation. Mori and Uchihira[11] performed a trade-off analysis between the accuracy and interpretability of different classification techniques with a scatter plot comparing relative ranks of accuracy with those of interpretability. The experiment was performed to show that the proposed new classification model, called superposed naive Bayes (SNB) can produce a balanced output that satisfies both accuracy and interpretability criteria. Di Chen's paper on actionable analytics[3] measures the FFT examples in terms of comprehensibility, which show that FFTs satisfy requirements raised by scientists for âĂIJeasily understandable at an expert levelâĂİ. But the measurements are based on the models themselves have nothing to do with how interpretative they are to the common public. Ryan Turner's Model Explanation System[13] proposes a new methodology for explaining the predictions of black box classifiers. In this system, the explanations are assumed to take the form of simple logical statements. All the above examples attempt to measure explainability of models and provide systems for the same, but are unable to quantify them or take the perspective of the model's users into account.

In Lipton's Mythos of Model Interpretability[17], we find parameters of interpretability such as simulatability, decomposability, algorithmic transparency, text explanation, visualization and explanation by example. These parameters give a theoretical understanding on how interpretability of a model can be measured but they cannot be quantified using any formulas or such. These tends to give an abstract knowledge on the topic of interpretability that cannot be actively used for further research. However, these parameters did help us formulate the questions for our survey, using which we have quantified the explanation of each of the classification model.

3 APPROACH

In recent years, classification models of supervised machine learning have been successfully applied to various domains, i.e., computer vision, speech recognition, information retrieval, marketing, finance, manufacturing, bioinformatics, and medical diagnosis[2] [4]. Consequently, a variety of predictive models have been developed so far in order to support or even replace human decision making in these domains. When building a predictive model, there are two important criteria: predictive accuracy and explainabiity[1]. Although accuracy is an essential property of a predictive model, explainabiity should also be taken into account particularly in the domains where the incorporation of expert knowledge into a predictive model is required. Predictive accuracy and explainabiity in general have a trade-off relationship, and their proper balance may be different for each domain.

To our knowledge there are few studies quantitatively measuring the trade-off between performance and explanation of classification models. So far, the studies have only commented on there existing a trade-off but haven't attempted at quantifying it for the classification models of machine learning. Our literature survey shows that the current research has only identified the parameters of explainability but been unable to correctly quantify them based on actual responses from people on whether they completely understand the working and generated output of the models.

The level of explainability of the models differs from person to person. Moreover, whether a person understands a classification model also depends on what background they have, their education and field of work. A survey is a data collection tool used to gather information about individuals. Hence we decided to do a survey that is taken by people from various different fields. One of the big benefits of using surveys is that they allow us to gather a large quantity of data relatively quickly and cheaply. We prepared the questionnaire of explainability assessment for our survey based on Lipton's criteria[17], which is comprised of model transparency (simulatability), component transparency (decomposability), and algorithmic transparency.

The survey starts with an explanation of all the classification models, to ensure that the people not familiar with the field of computer science have a basic understanding of the models. The first section of questions ask the participants to solve and answer some basic examples of the classification models. Based on the correctness of their answers we determine whether they truly understand the working of the models. The second section of the questionnaire provides the participants with the results generated when each model is run for a particular data set. We ask the participants to rate each result on a scale of 1 to 4 based on how well they understand it. The responses of the survey on a whole help us quantify the explainability of all the models.

4 METHODOLOGY

4.1 Dataset Description

The dataset[10] used to analyze the performance of the classification model has six existing real-world configurable systems with different characteristics: different sizes (42 thousand to 300 thousand lines of code, 192 to millions of configurations), different implementation languages (C, C++, and JAVA), and different configuration mechanisms (conditional compilation, configuration files, and command-line options). The dataset contains the whole population of each system, i.e., all configurations of each system and their performance measurements (the exception is SQLITE, for which the dataset contains 4,553 configurations for prediction modeling and 100 additional random configurations for prediction evaluation).

For each system, the performance has been measured using a standard benchmark, either delivered by its vendor (e.g., ORACLE 's standard benchmark for BERKELEY DB) or used widely in its application domain (e.g., AUTOBENCH and HTTPERF for the APACHE Web Server).

4.2 Model Description

4.2.1 Decision Tree

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

In decision analysis, a decision tree and the closely related influence diagram are used as a visual and analytical decision support tool, where the expected values (or expected utility) of competing alternatives are calculated.

A decision tree consists of three types of nodes:

1. Decision nodes - typically represented by squares
2. Chance nodes - typically represented by circles
3. End nodes - typically represented by triangles

Decision trees are commonly used in operations research and operations management[7]. If, in practice, decisions have to be taken online with no recall under incomplete knowledge, a decision tree should be paralleled by a probability model as a best choice model or online selection model algorithm. Another use of decision trees is as a descriptive means for calculating conditional probabilities.

Decision trees, influence diagrams, utility functions, and other decision analysis tools and methods are taught to undergraduate students in schools of business, health economics, and public health, and are examples of operations research or management science methods.

The decision tree elements are shown in Figure 1. Drawn from left to right, a decision tree has only burst nodes (splitting paths) but no sink nodes (converging paths). Therefore, used manually, they can grow very big and are then often hard to draw fully by hand. Traditionally, decision trees have been created manually - as the aside example shows - although increasingly, specialized software is employed.

4.2.2 K-nearest Neighbors

KNN(K-Nearest Neighbors) algorithm is one of the simplest classification algorithm and it is one of the most used learning algorithms. Basically, KNN is a non-parametric, lazy learning algorithm. When we say a technique is non-parametric , it means that it does not make any assumptions on the underlying data distribution.

Although KNN algorithm can be used in classification and regression, in our project, we only focus on the classification. So for the KNN classification process, according to the following figure1, we want to predict the class for X_u, and what we need to do is find a set include the most nearest neighbor of X_u, and so that according to the classification result of these neighbors, we can successfully predict the class of X_u. According to this figure, ω_1,ω_2,ω_3 represents different classes. And you can easily see that for the 5 nearest neighbors, 4 belongs to RED group(ω_1) and only 1 belongs to GREEN group(ω_3), obviously X_u will be classify to group RED, because its nearest neighbors mostly belongs to group RED.
More specifically, There are two important concepts in the above example. One is the method to calculate the distance between X_u and other point. By default, the knn() function employs Euclidean distance which can be calculated with the following equation:

$$d(p_1, p_2) = \sqrt{\sum_d (p_1^d - p_2^d)^2} \tag{1}$$

This equation(1) is generalized to cover any dimension. For 2D space, the equation reduces to the following:

$$d(p_1, p_2) = \sqrt{(p_1^1 - p_2^1)^2 + (p_1^2 - p_2^2)^2} \tag{2}$$

A main advantage of the KNN algorithm is that it performs well with multi-modal2 classes because the basis of its decision is based on a small neighborhood of similar objects. Therefore, even if the target class is multi-modal, the algorithm can still lead to good accuracy. However a major disadvantage of the KNN algorithm is that it uses all the features equally in computing for similarities. This can lead to classification errors, especially when there is only a small subset of features that are useful for classification.

4.2.3 Naive Bayes

Naive Bayes is a simple, yet effective and commonly-used, machine learning classifier. It is a probabilistic classifier that makes classifications using the Maximum A Posterior decision rule in a Bayesian setting. It can also be represented using a very simple Bayesian network. Naive Bayes classifiers have been especially popular for text classification, and are a traditional solution for problems such as spam detection.

The goal of any probabilistic classifier is, with features x_0 through x_n and classes c_0 through c_k, to determine the probability of the features occurring in each class, and to return the most likely class. Therefore, for each class, we want to be able to calculate $P(c_i|x_0, ..., x_n)$.

In order to do this, we use Bayes rule. Recall that Bayes rule is the following:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3}$$

In the context of classification, you can replace A with a class, c_i, and B with our set of features, x_0 through x_n. Since P(B) serves as normalization, and we are usually unable to calculate $P(x_0, ..., x_n)$, we can simply ignore that term, and instead just state that $P(c_i|x_0, ..., x_n) \propto P(x_0, ..., x_n|c_i) * P(c_i)$, where $\propto$ means "is proportional to". $P(c_i)$ is simple to calculate; it is just the proportion of the data-set that falls in class i. $P(x_0, ..., x_n|c_i)$ is more difficult to compute. In order to simplify its computation, we make the assumption that x_0 through x_n are conditionally independent given c_i, which allows us to say that $P(x_0, ..., x_n|c_i) = P(x_0|c_i) * P(x_1|c_i) * ... * P(x_n|c_i)$. This assumption is most likely not true–hence the name naive Bayes

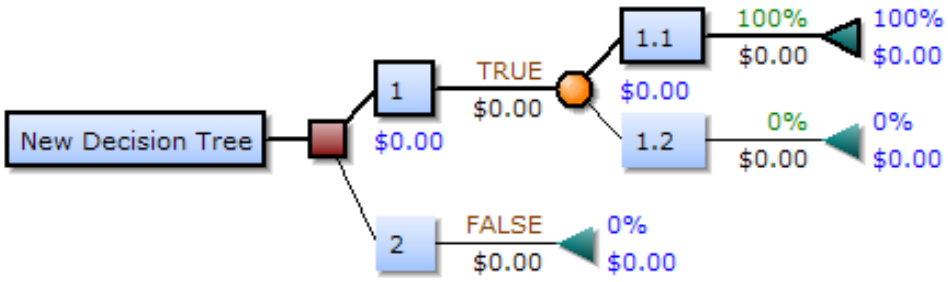

Figure 1: Decision tree elements

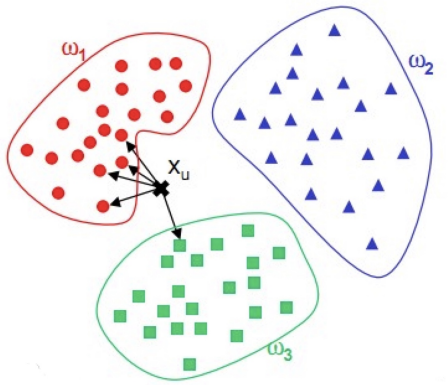

Figure 2: KNN(k = 5)

classifier, but the classifier nonetheless performs well in most situations. Therefore, our final representation of class probability is the following:

$$P(c_i|x_0,...,x_n) \propto P(x_0,...,x_n|c_i)$$
$$\propto P(c_i)\prod_{j=1}^{n} P(x_j|c_i) \quad (4)$$

Calculating the individual $P(x_j|c_i)$ terms will depend on what distribution your features follow. In the context of text classification, where features may be word counts, features may follow a multinomial distribution. In other cases, where features are continuous, they may follow a Gaussian distribution.

Note that there is very little explicit training in Naive Bayes compared to other common classification methods. The only work that must be done before prediction is finding the parameters for the features 'individual probability distributions, which can typically be done quickly and deterministically. This means that Naive Bayes classifiers can perform well even with high-dimensional data points and/or a large number of data points.

Now that we have a way to estimate the probability of a given data point falling in a certain class, we need to be able to use this to produce classifications. Naive Bayes handles this in a very simple manner; simply pick the c_i that has the largest probability given the data point's features.

This is referred to as the Maximum A Posteriori decision rule. This is because, referring back to our formulation of Bayes rule, we only use the P(B|A) and P(A) terms, which are the likelihood and prior terms, respectively. If we only used P(B|A), the likelihood, we would be using a Maximum Likelihood decision rule.

4.2.4 *Neural Network*

Artificial neural networks are one of the main tools used in machine learning. As the "neural" part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

While neural networks (also called "perceptrons") have been around since the 1940s, it is only in the last several decades where they have become a major part of artificial intelligence. This is due to the arrival of a technique called "backpropagation," which allows networks to adjust their hidden layers of neurons in situations where the outcome doesn't match what the creator is hoping for âĂŤ like a network designed to recognize dogs, which misidentifies a cat, for example.

Another important advance has been the arrival of deep learning neural networks, in which different layers of a multilayer network extract different features until it can recognize what it is looking for.

For a basic idea of how a deep learning neural network learns, imagine a factory line. After the raw materials (the data set) are input, they are then passed down the conveyer belt, with each subsequent stop or layer extracting a different set of high-level features. If the network is intended to recognize an object, the first layer might analyze the brightness of its pixels. The diagram is shown in Figure 3. The next layer could then identify any edges in the image, based

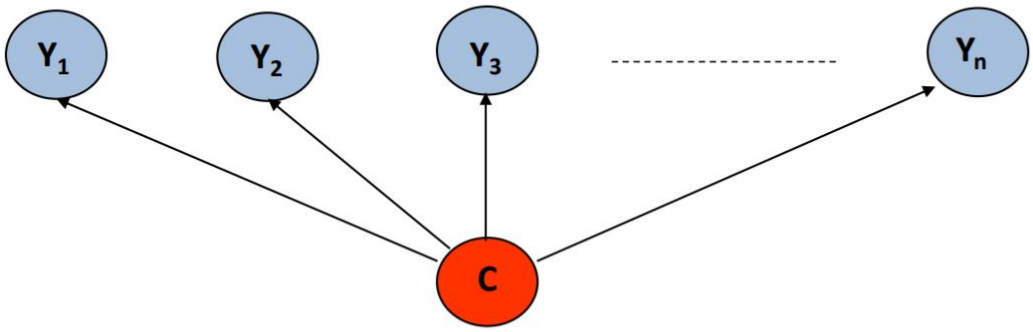

Figure 3: Probabilistic Graph of Naive Bayes

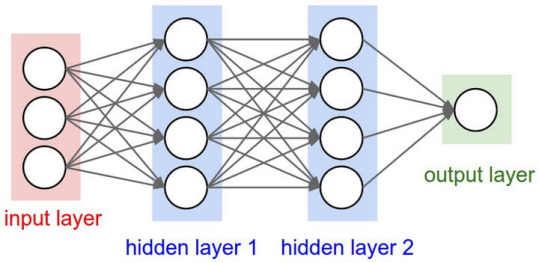

Figure 4: Artificial Neural Network

on lines of similar pixels. After this, another layer may recognize textures and shapes, and so on. By the time the fourth or fifth layer is reached, the deep learning net will have created complex feature detectors. It can figure out that certain image elements (such as a pair of eyes, a nose, and a mouth) are commonly found together.

Once this is done, the researchers who have trained the network can give labels to the output, and then use backpropagation to correct any mistakes which have been made. After a while, the network can carry out its own classification tasks without needing humans to help every time.

There are multiple types of neural network, each of which come with their own specific use cases and levels of complexity. The most basic type of neural net is something called a feedforward neural network, in which information travels in only one direction from input to output.

A more widely used type of network is the recurrent neural network, in which data can flow in multiple directions. These neural networks possess greater learning abilities and are widely employed for more complex tasks such as learning handwriting or language recognition.

There are also convolutional neural networks[16], Boltzmann machine networks[8], Hopfield networks[9], and a variety of others. Picking the right network for your task depends on the data you have to train it with, and the specific application you have in mind. In some cases, it may be desirable to use multiple approaches, such as would be the case with a challenging task like voice recognition.

On a technical level, one of the bigger challenges is the amount of time it takes to train networks, which can require a considerable amount of compute power for more complex tasks. The biggest issue, however, is that neural networks are "black boxes," in which the user feeds in data and receives answers. They can fine-tune the answers, but they do not have access to the exact decision making process.

This is a problem a number of researchers are actively working on, but it will only become more pressing as artificial neural networks play a bigger and bigger role in our lives.

4.3 Questionnaire Design

The main goal of this questionnaire is to see the comprehensibility of Decision Trees, KNN, Naive Bayes and Neural Net algorithms. In the first section of the questionnaire, some brief description of those four algorithms are given here so that participants especially who do not have any background of computer science or machine learning, could have some basic understanding about the algorithms. It is shown in Figure 5.

We prepare two types of questions for tests.

In the first section we have four simple examples which check if participants can use what they learn from the algorithm description to figure out the correct output. For example, Figure 6 is the KNN question. A simple example of KNN is given by an image showing the neighbors of the test case. And the participants should classify the test case based on the appropriate neighbors in the image. And for the decision tree, we use a resulting diagram of a

Decision Trees:

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree has three entities, namely decision nodes which represents the features, branches which represent a decision rule and leaves which represent the decisions or the final outcomes. The algorithm is as follows:
1.compute the entropy for data-set
2.for every attribute/feature:
1.calculate entropy for all categorical values
2.take average information entropy for the current attribute
3.calculate gain for the current attribute
3. pick the highest gain attribute.
4. Repeat until we get the tree we desired.

KNN:
1. Compute a distance value between the item to be classified and every item in the training data-set
2. Pick the k closest data points (the items with the k lowest distances)
3. Conduct a "majority vote" among those data points — the dominating classification in that pool is decided as the final classification

Naive Bayes:

This model should give the conditional probabilities of each attribute level (range for numerical ones) conditioned on the class label after the training. For a test case, it just computes the joint probabilities of given attributes values and each class label and choose the largest one as the classification output since under the naive assumption of Bayes theorem, all attributes should be restrictively independent from one another.

Neural Net:

In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called 'edges'. Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

Figure 5: Algorithm description

decision tree model and asked the participants to make decision according to the given tree diagram, i.e., to find the appropriate decision path for the given test case. For the model of naive Bayes and neural network, it is relatively harder to present the training output than the former two. Thus we designed some theoretical questions related to its statistical background more. Specifically, the question for naive Bayes is asking participants to distinguish the correct joint probability (equivalent to corresponding conditional probability). And question for neural network is regarding to the computational process of the information flow and backpropagation during the model training.

In the second section, we give participants harder questions. During the training process of our own dataset, we generated one output for each of the four algorithms. Then we present those outputs in this section to ask participants to score them based on how explainable (easy to understand) they appear to be. For example, the output of Decision Tree is shown in Figure 7. We believe after getting enough number of responses, we can analyze the results and get a straightforward conclusion about the comprehensibility and explainability of the four algorithms.

5 EVALUATION

5.1 Result from the questionnaire

We selected 21 people from both CS and non-CS students to do our questionnaire. In the first part of our questionnaire asking about algorithm correctness: we got 19 correct answer, 1 wrong answer, 1 not sure with total 90.5% correctness on Decision Tree; 17 correct answer, 4 wrong answer, no not sure with total 81% correctness on KNN; 12 correct answer, 7 wrong answer, 2 not sure with total 57 % correctness on Naive Bayes; 4 correct answer, 13 wrong answer, 4 not sure with total only correctness 19% on Neural Network. The statistics in shown in figure 9 and figure 10.

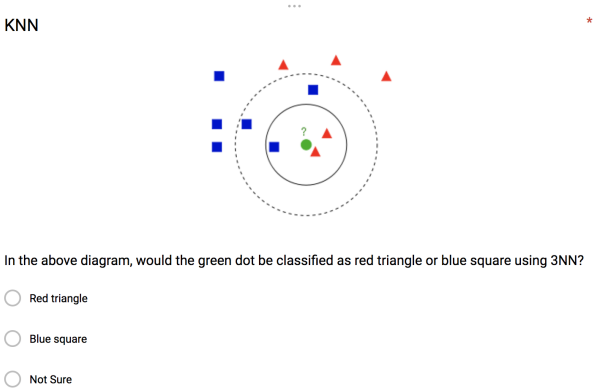

Figure 6: KNN Question

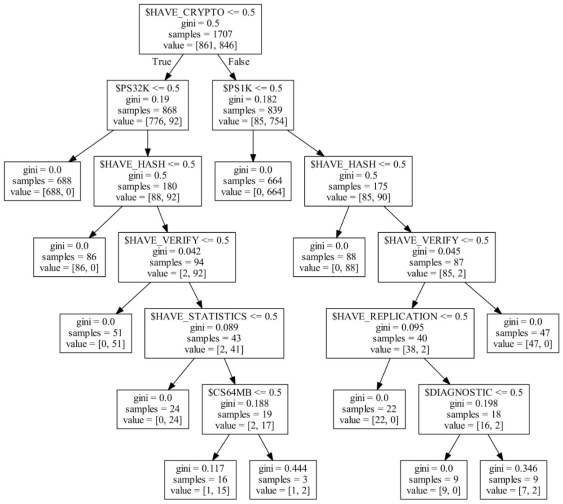

Figure 7: Decision Tree Output

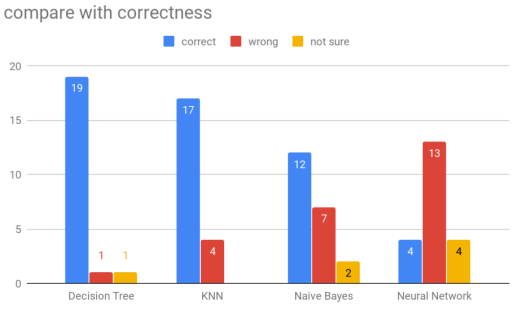

Figure 8: evaluation_correctness_distribution

For the second part of our questionnaire asking about how explainable of each algorithm: for Decision Tree, there are 1 response to score_1, 1 response to score_2, 5 response to score_3, 11 response

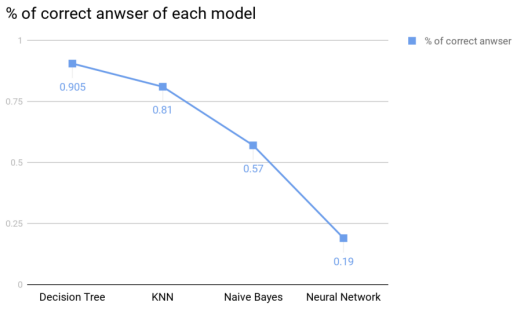

Figure 9: evaluation_correctness_average score

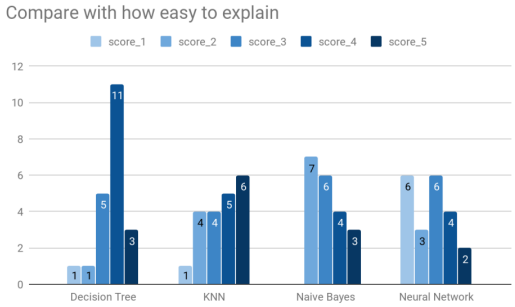

Figure 10: evaluation_explain_distribution

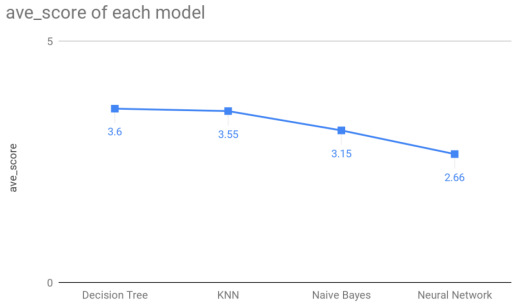

Figure 11: evaluation_explain_average score

to score_4, 3 response to score_5; for KNN there are 1 response to score_1, 4 response to score_2, 4 response to score_3, 5 response to score_4, 6 response to score_5; for Naive Bayes, there are no response to score_1, 7 response to score_2, 6 response to score_3, 4 response to score_4, 3 response to score_5; for Neural Network there are 6 response to score_1, 3 response to score_2, 6 response to score_3, 4 response to score_4, 2 response to score_5. Overall, we got average of 3.6 score on Decision Tree, 3.55 score on KNN, 3.15 score on Naive Bayes, and 2.66 score on Neural Network. The statistics is shown in figure 11 and figure 12.

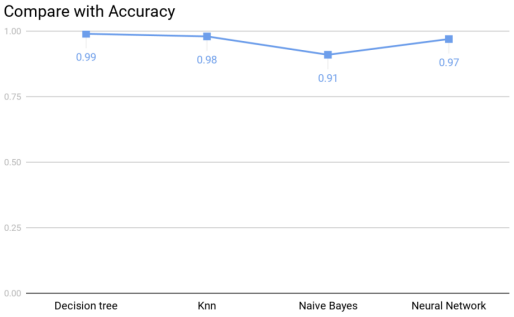

Figure 12: evaluation_accuracy_score

5.2 Result from the performance

We used the BDBCAll dataset from the Seacraft to test accuracy of each model. The results are 99% on Decision Tree, 98% on KNN, 91% on Naive Bayes, and 97% on Neural Network. The statistics is shown in figure 13.

5.3 Analysis of the results

For the response from the correctness of each algorithm, we could find that the Decision Tree model got the highest correctness (90%) comparing to other models. For KNN, most people could got the right answer too(81%). But for NaÃŕve Bayes and Neural Network, the correctness started to decrease, and more people get confused with the question. Naive Bayes got 57% correctness, while Neural Network got only 19% correctness, and more than half of our participants even chose the wrong answer instead of chose âĂIJnot sureâĂİ. It led to the conclusion that the logic and algorithm for Neural Network was very difficult for people to understand fully in a short amount of time. People might get the easy concept after reading the description, but still be unable to recognize the full process of how Neural Network trained the weights. On the other hand, people could understand how decision tree worked, and was able to choose the correct answer in a short time even if for a non-CS person. We could also get the same conclusion from the second part of our questionnaire. For Decision Tree and KNN, we got high score of 3.6 and 3.55, which led the fact that people thought these two models were easier to explain for them. But for Naive Bayes the score was only 3.15, which was more complex for them to explain for the example we have them. And Neural Network only got 2.66 score, which was the most difficult to explain comparing to the other models. It turns out that those weights of the neurons in the Neural Network make it hard for people to understand how it computes the weights, what the weights means, and why it works like that.

For the results in comparing accuracy of each model, all models perform good result of higher than 90%. We can conclude that there is no significant difference between these four models using our dataset from SeaCraft. As all the models perform the same high accuracy, we propose using Decision Tree, that is the highest correctness score and also the highest explainable score from our previous questionnaire.

6 THREATS OF VALIDITY

This study has limitations that must not be ignored when interpreting its results.

6.1 Sampling of Participants

The proportion of participants who are not familiar with machine learning topics in the survey is not sufficient to have an effective statistical assertion. After all, the property of explanation is more user-oriented rather than developer-oriented. The importance of explanation for users may further increase in a situation where the model is used for some critical decision making. Therefore, it is definite that we should include more user-like people as the participants of this survey and we should have increased the number of this group of participants in the survey.

6.2 Complexity of Survey Questionnaire

The statement of some questions are not clear enough and may cause some participants' confusion. For example, the question for KNN should have described each component of the output clearly to prevent participants' confusion. Also, the first section of the questionnaire which was used to ascertain whether the participants truly understood the working of the different models were relatively easy. This was done so that people with no knowledge or background in Machine Learning could be comfortable in taking the survey. However, to improve the results of our study we could have added or increased the complexity of the questions in the survey. We could have added real life samples of the models as used in the industry today to produce real time data for our study.

6.3 Rate of Correctness

The rate of correctness of the questions are excessively high, hence the results of different models are less distinguishable. This is due to the fact that the percentage of people from Computer Science background taking the survey was greater than than that of people from other fields taking the survey. Also, since the questions were relatively easy, especially the ones' in the first section, it improved the rate of correctness. This may have slightly biased and affected the results of trade-off between explanation and performance of the different models.

6.4 More Dataset For Evaluating the Models

In this project, we choose one dataset from the SeaCraft to measure the accuracy of each model. But using only one dataset may cause a bias problem, because the dataset we picked may perfectly fitted on one specific model, but badly fitted on other models. Also, Neural Network is good at predicting large amount of data, but if our dataset is too small, we can not evaluate the true power of Neural Network. In order to prevent the bias problem, more dataset should have been used to calculate the average accuracy among all the models. Using this approach should obtain more convincing results.

7 CONCLUSION

7.1 RQ1

How should we measure the explanation of a model?

There may not be an ubiquitous quantitative metric for model's explanation. But our proposed questionnaire has provided a semi-quantitative metric for explanation based on people's perception of the models. Since the property of explanation itself orients how easy for people to understand the model and its resulting implications, the proposed questionnaire could be considered as a metric for explanation.

7.2 RQ2

Which model is the best in terms of the trade-off?

We propose using Decision Tree, because all four models perform with high accuracy, but Decision Tree gains the highest correctness score and highest explainablity score in our questionnaire. The reason that there is no obvious trade-off between accuracy and explainability may be due to our dataset being too small, so it is easy for all four models to predict.

Although not concrete, the results of our study give an idea as to how the trade-off between explainabilty and performance of different models can be quantitatively measured. Our future work directions include improving the quality and complexity of the questions in the survey, including more people with backgrounds in different fields and utilizing a bigger dataset to better evaluate the performance of the models.

REFERENCES

[1] Freitas AA. 2014. Comprehensible classification models: a position paper. In *ACM SIGKDD Explorations Newsletter*. ACM, 15(1):1âĂŞ10.
[2] Bishop CM. 2006. Pattern recognition and machine learning. In *Springer*.
[3] Rahul Krishna Tim Menzies Di Chen, Wei Fu. 2018. Applications of Psychological Science for Actionable Analytics. In *FSE*.
[4] Friedman J Hastie T, Tibshirani R. 2009. The elements of statistical learning. In *2nd edn. Springer*.
[5] Qiao Huang. 2017. Supervised vs Unsupervised Models: A Holistic Look at Effort-Aware Just-in-Time Defect Prediction. In *IEEE International Conference on Software Maintenance and Evolution*.
[6] Ben Z. Yuan Ayesha Bajwa Michael Specter Leilani H. Gilpin, David Bau and Lalana Kagal. [n. d.]. Explaining Explanations: An Approach toEvaluating Interpretability of Machine Learning.
[7] R. Dan Reid Nada R. Sanders. 2012. Operations Management: An Integrated Approach, 5th Edition. John Wiley & Sons.
[8] Geoffery Hinton Ruslan Salakhutdinov, Andriy Mnih. 2007. Restricted Boltzmann machines for collaborative filtering. In *International Conference on Machine Learning*.
[9] Wan Ahmad Tajuddin Wan Abdullah Saratha Sathasivam. 2008. Logic Learning in Hopfield Networks. In *arXiv:0804.4075*.
[10] Norbert Siegmund. 2015. CPM. (March 2015). https://doi.org/10.5281/zenodo.322483
[11] Naoshi Uchihira Toshiki Mori. 2018. Balancing the trade-off between accuracyand interpretability in software defect prediction. In *Empir Software Eng*.
[12] Ryan Turner. 2015. A Model Explanation System. In *IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING*.
[13] Ryan Turner. 2016. A MODEL EXPLANATION SYSTEM. In *IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING*.
[14] Tao Xie XuanLu. 2016. PRADA: Prioritizing Android Devices for Apps by Mining Large-Scale Usage Data. In *ACM International Conference on Software Engineering*.
[15] Tao Xie Xuanzhe Liu. 2017. Understanding Diverse Usage Patterns from Large-Scale Appstore-Service Profiles. In *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*.
[16] Yoshua Bengio Patrick Haffner Yann LeCunn, Leon Bottou. 1998. Gradient-Based Learning Applied to Document Recognition. In *IEEE*.
[17] Lipton ZC. 2016. The mythos of model interpretability. In: 2016 ICML workshop on human interpretability in machine learning. WHI.