A Comparative Analysis of Hyperopt as Against Other Approaches for Hyper-Parameter Optimization of XGBoost

Sayan Putatunda
VMware Software India pvt. Ltd.
Kalyani Vista, JP Nagar
Bangalore, India
+91 080 4044 0000
sayanp@iima.ac.in

Kiran Rama
VMware Software India pvt. Ltd.
Kalyani Vista, JP Nagar
Bangalore, India
+91 080 4044 0001
efpm04013@iiml.ac.in

ABSTRACT
The impact of Hyper-Parameter optimization on the performance of a machine learning algorithm has been proved both theoretically and empirically by many studies reported in the literature. It is a tedious and a time-consuming task if one goes for Manual Search. Some of the common approaches to address this include Grid search and Random search. Another alternative is performing the Bayesian optimization using the Hyperopt library in Python. In this paper, we tune the hyperparameters of XGBoost algorithm on six real world datasets using Hyperopt, Random search and Grid Search. We then compare the performances of each of these three techniques for hyperparameter optimization using both accuracy and time taken. We find that the Hyperopt performs better than the Grid search and Random search approaches taking into account both accuracy and time. We conclude that Bayesian optimization using Hyperopt is the most efficient technique for hyperparameter optimization.

CCS Concepts
• Computing methodologies→Supervised learning by classification

Keywords
Bayesian Optimization; Grid Search; Random Search; XGBoost; Machine Learning; Hyper-Parameter Optimization

1. INTRODUCTION
Any machine learning algorithm such as Deep neural networks, gradient boosting models, etc. comprises various hyperparameters that requires tuning for the optimal performance of the algorithm. Different algorithms have different types of hyperparameters to tune as described in Table 1 where some of the major hyperparameters for Extreme gradient boosting (XGBoost), Support vector machines (SVM), Random forests (RF) and Deep neural networks are mentioned. As we can see there are many hyperparameters for some algorithms and this makes the problem of hyperparameter optimization even more complicated. There have been studies that have shown that tuning hyperparameters improves model performances and have compared several approaches [1], [2].

SPML '18, November 28–30, 2018, Shanghai, China
© 2018 ACM. ISBN 978-1-4503-6605-2/18/11...$15.00
DOI: https://doi.org/10.1145/3297067.3297080

Hyperparameter optimization is the science of tuning the hyperparameters of these algorithms to get the optimal performance [3]. One of the most obvious ways for performing hyperparameter optimization is by doing it manually. But given the fact that there are many settings and possibilities, this strategy doesn't scale. So, in the literature there have been many techniques proposed for hyperparameter optimization, which will be discussed in detail in Section 2. Some of the well-known methods are Grid search, Random search and Bayesian optimization.

Table 1. Algorithms along with their hyperparameters

Algorithm	**Hyperparameters**
Decision tree	min_samples_split
	max_depth
	min_samples_leaf
XGBoost	n_estimators
	learning_rate
	colsample_by_tree
	reg_lambda
	max_depth
	subsample
C-SVM	C (Soft margin constant)
	gamma
	kernel
Random forest	max_features
	max_depth
	min_samples_leaf
Neural network	number of hidden layers
	number of hidden nodes
	dropout
	activation function
	learning_rate
	momentum
	epochs
	batch size

Extreme gradient boosting i.e. XGBoost is a very popular and powerful data mining method for solving large scale machine learning problem and have performed better than many other advanced machine learning algorithms when applied to structured datasets [4]. The reason for choosing XGBoost for this paper is that it is a highly accurate method, has many parameters and it can be simulated into Random forest, which is another very powerful algorithm. It is one of the widely used method for solving various business problems and so it is useful to study ways by which the performance of the XGBoost algorithm can be improved further. A well-tuned XGBoost can achieve state of art prediction accuracy compared to a not properly configured XGBoost. However, there are many hyperparameters in XGBoost that one need to tune (as described in Table 1), which would be a very time-consuming exercise if one does it manually.

In this paper, we compare the performance of XGBoost for classification problems on six real world datasets by applying Bayesian optimization for hyperparameter optimization using Hyperopt (please refer to Section 4.1) and comparing it with other approaches such as Grid search and Random search. We take both accuracy and time taken into factor for evaluation. This has not been attempted earlier in the literature to the best of our knowledge. The contribution of this paper is that it empirically proves that the Bayesian optimization method using Hyperopt is a very effective method for Hyperparameter optimization of XGBoost algorithm and performs better than other widely used approaches Grid search and Random search.

2. BACKGROUND AND RELATED WORK

Grid search is one of the most commonly used methods for Hyperparameter optimization. It is a type of brute-force technique. The idea here is to search through a manually defined subset of hyperparameters for the concerned ML algorithm [5]. So the search space is defined and the goal is to optimize the performance of the ML algorithms. This is a very simple method but with the increase in hyperparameters, the computational costs increase exponentially [5]. Some of the major advantages of the Grid search method are its ease of implementation and parallelism. Moreover, in low dimensional spaces i.e. 1-D or 2-D, Grid search is a very reliable method [3].

In case of Random Search, the configuration space is defined by using a generative process for drawing random samples and the hyperparameter assignments are drawn from this process and evaluated [1]. Bergstra & Bengio [3] showed that Random search has similar advantages when compared to that of Grid Search, but in high dimensional space Random search is a far more efficient method than Grid search. In some studies such as [3], it is found that generally Random search performs better than the Grid search.

Bayesian Optimization is a kind of “black box” optimization technique [6]. It is a very powerful tool for optimizing objective functions that slow or costly to evaluate [7], [8]. It generally requires a prior function and an acquisition function. The way it operates as explained in Mockus et al. [9] is that, first we choose a prior measure over the objective space. Then the prior and the likelihood are combined to get a Posterior measure over the objective function given some observation and then the posterior decides the next evaluation in accordance to the loss function.

A Gaussian Process (GP) is a popular choice for a prior function [6]. However, for the choice of acquisition function, there is a trade-off between exploration and exploitation [6]. Some of the popular choices are Expected Improvement (EI) [9] and Entropy Search (ES) [10]. Sequential Model-Based Global Optimization (SMBO) is a formalism of Bayesian optimization and it is very useful when the fitness function is costly to evaluate [7]. Here the model-based algorithms approximate the expensive fitness function f using a surrogate function, which is cheaper to evaluate [1]. The goal of the SMBO algorithm is to optimize the surrogate function and get the point x* that maximizes the surrogate. This point is then proposed where the true function f should be evaluated [1]. Some of the advantages of the SMBO algorithm include its ability to handle parallel evaluation of the function f and its ability to handle high dimensional data effectively [11].

An alternative to the Gaussian Process (GP) approach (where we model $p(y|x)$) is the Tree-structured Parzen Estimator (TPE) approach that models $p(x|y)$ and $p(y)$ [1]. The TPE method accomplishes the task of modeling $p(x|y)$ by using non-parametric densities instead of the distributions of the configuration prior. It basically replaces the prior distributions such as uniform or log-uniform with a variation of Gaussian mixture [1]. It then explores the search space using different observations in the non-parametric densities to get the best parameters.

Hyperopt is a python library that enables researchers and practitioners to implement Bayesian optimization or SMBO (see Section 4.1 for more details). Hyperopt has been used for hyper-parameter optimization of deep neural networks and convolutional neural networks [12]. Bergstra et al. [13] explored Hyperopt for SVM and RF and found it to be the best performer.

Some of the other approaches for hyperparameter optimization include evaluating the algorithms performance by using small and random subsets of fixed sizes that are chosen manually for multiple training runs [14]. A new cross-validation design was proposed by Krueger et al. [15] that on a growing subset of data, sequentially tests a fixed set of configurations and discards the poorly-performing configurations early. Some of the other heuristic approaches for hyper-parameter optimization reported in the literature are genetic algorithms [16], swarm optimization [17] and coupled simulated annealing [18].

3. DATA

We have taken six publicly available datasets from the UCI machine learning repository [19]. The details pertaining to each of these datasets is given below and in Table 2, we describe the number of attributes and features in each of the six datasets. .

Car evaluation dataset: This dataset comprises 6 attributes and 1728 number of observations [20]. The target variable is acceptability of the car. We have modified the target variable into a binary class. We will refer to this dataset as CARS for the rest of this paper.

Bank marketing dataset: This dataset has 20 features and 45211 number of observations. This dataset is related to a direct marketing campaign conducted by a Portuguese bank and was first used in Moro et al. [21]. The various attributes include age, education, loan taken by customer, credit default, details regarding campaign and social and economic context information. The target variable is a binary class variable that represents whether a customer has subscribed to a term deposit. We will refer to this dataset as BANK for the rest of this paper.

Diabetic retinopathy dataset: In this dataset the features are extracted from images and the target variable signifies whether a particular case contains signs of Diabetic retinopathy [22]. This dataset contains 20 attributes and 1151 number of observations.

We will refer to this dataset as DIABETIC for the rest of this paper.

Credit card default dataset: This dataset was used in Yeh and Lin [23] to predict the probability of credit card default by customers of a Taiwanese bank. This dataset contains 24 attributes and 30000 number of observations. We will refer to this dataset as DEFAULT for the rest of this paper.

Absenteeism at work dataset: This dataset was used in Martiniano et al. [24] to predict the probability of absenteeism at work. This dataset contains 21 attributes and 740 number of observations. In this paper, our target variable is whether a person is regular at work or not, which is derived from their absent hours i.e. if absent hours is zero then the person is regular to office. The various features include reason for absence, transportation expense, work load, age, social drinker/smoker, etc. We will refer to this dataset as ABSENT for the rest of this paper.

Breast cancer dataset: This dataset is publicly available in the UCI machine learning repository [19] and was first used in Michalski et al. [25]. This dataset contains 9 attributes and 286 number of observations. The target variable contains 201 instances of no-recurrence of cancer vs. 85 instances of recurrence of cancer. The various features include age range, menopause, degree of malignancy, etc. We will refer to this dataset as CANCER for the rest of this paper.

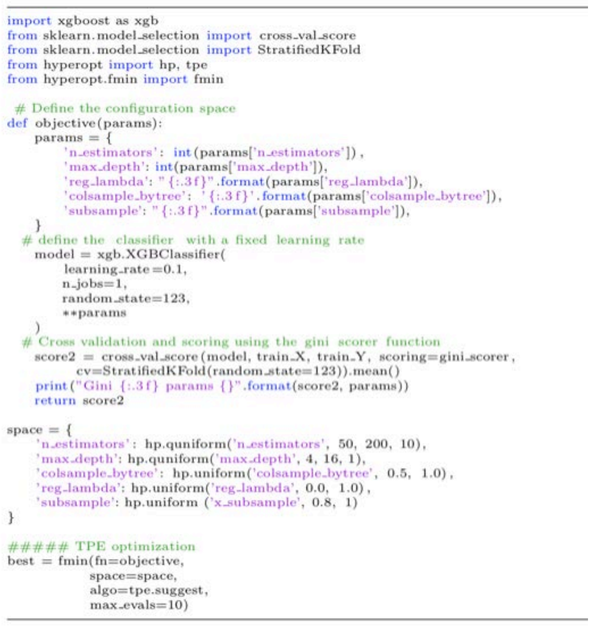
Table 2. Dataset Description

Dataset	Number of attributes	Number of observations
CARS	6	1728
BANK	20	45211
DIABETIC	20	1151
DEFAULT	24	30000
ABSENT	21	740
CANCER	9	286

4. EXPERIMENTAL RESULTS

4.1 System Setup and Implementation

We used Python 3.6 with Hyperopt and Scikit-learn libraries for our experiments on a system with 16 GB RAM, 2.9 GHz Intel Core i7, 2133 MHz LPDDR3 and Mac OS. Hyperopt is a python library that enables researchers and practitioners to implement Bayesian optimization or SMBO (that we discussed in Section 2) for hyperparameter optimization of different machine learning algorithms [12]. It was primarily developed for research on Computer vision [11] and Deep learning [1]. In Hyperopt, we first define the configuration space and then use the fmin driver to optimize as shown in Figure 1. Hyperopt provides many options for the different types distributions when defining the configuration space. Some of the prominent ones include uniform, log uniform, normal and many more. It also allows using "quantized" continuous distributions, which are suitable for discrete variables with respect to which the objective function is "smooth" (Bergstra et al., 2013). Some of the examples of these "quantized" continuous distributions are quniform, qnormal and qlognormal. One can refer to Bergstra et al. [12] for a detailed discussion on the choice of various distributions allowed in Hyperopt.



```
import xgboost as xgb
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from hyperopt import hp, tpe
from hyperopt.fmin import fmin

 # Define the configuration space
def objective(params):
    params = {
        'n_estimators': int(params['n_estimators']) ,
        'max_depth': int(params['max_depth']),
        'reg_lambda': "{:.3f}".format(params['reg_lambda']),
        'colsample_bytree': '{:.3f}'.format(params['colsample_bytree']),
        'subsample': "{:.3f}".format(params['subsample']),
    }
  # define the classifier with a fixed learning rate
    model = xgb.XGBClassifier(
        learning_rate=0.1,
        n_jobs=1,
        random_state=123,
        **params
    )
  # Cross validation and scoring using the gini scorer function
    score2 = cross_val_score(model, train_X, train_Y, scoring=gini_scorer,
          cv=StratifiedKFold(random_state=123)).mean()
    print("Gini {:.3f} params {}".format(score2, params))
    return score2

space = {
    'n_estimators': hp.quniform('n_estimators', 50, 200, 10),
    'max_depth': hp.quniform('max_depth', 4, 16, 1),
    'colsample_bytree': hp.uniform('colsample_bytree', 0.5, 1.0),
    'reg_lambda': hp.uniform('reg_lambda', 0.0, 1.0),
    'subsample': hp.uniform ('x_subsample', 0.8, 1)
}

##### TPE optimization
best = fmin(fn=objective,
            space=space,
            algo=tpe.suggest,
            max_evals=10)
```

Figure 1. The implementation of Hyperopt in Python.

As shown in Figure 1, we define the search space for various hyperparameters of XGBoost such as n_estimators, max_depth, colsample_by_tree, reg_lambda and subsample. As it can be clearly seen in the above code snippet that each of the hyperparameters are defined within a range of values in the form of a distribution and this is where Hyperopt is significantly different to other approaches such as Grid search and Random search. We use Stratified K- fold cross validation (with K=3) and compute the mean Gini score (which we will discuss in Section 4.2). And finally, we identify the optimum hyper- parameters. Similarly, for Grid search and Random search hyperparameter optimization, we perform the same experiment with similar ranges for all the hyperparameters using the GridSearchCV and RandomizedSearchCV (with number of iterations = 10) functions of the Scikit-learn library.

4.2 Evaluation Metric

In this paper, we use the mean Gini score as an evaluation metric. The Gini index is one of the most commonly used metric by economists and sociologists due to its clear economic interpretation [26]. The Gini index is summary index in the Lorentz curve [27] that is used in analysis of a variety of scientific problems [28]. The connection between Lorentz curve and ROC was first pointed out by Lee and Hsiao [29] and it is defined as follows:

$$Gini = 2*AUC-1 \qquad (1)$$

Where, AUC= Area under the curve. As mentioned earlier, we compute the mean Gini score after applying the Stratified K-fold cross validation. For comparison purposes, the method with a higher mean Gini score is a better performer. We also consider the time taken by each of the three methods viz. Hyperopt, Grid search and Random search to generate the best parameters.

4.3 Results

In Table 3 we describe the results of our experiments i.e. the time taken (in Seconds) to generate the best parameters using the three different methods and the mean Gini score with the obtained best hyperparameters (as described in Table 4) for the XGBoost algorithm on each of the six datasets. We see that for each of the six datasets, the Hyperopt approach has a consistently higher mean Gini score when compared to Grid search and Random search. Also, the time taken by Hyperopt is close to the fastest method i.e. Random search whereas the Grid search method is the slowest.

Table 3. **Mean Gini with the best parameters and the time taken to identify the best parameters with each hyper-parameter optimization techniques for the six datasets**

Dataset	Method	Mean Gini	Time (in s)
CARS	Hyperopt	0.8445	3.35
	Random Search	0.7428	3.1
	Grid Search	0.7726	139.52
BANK	Hyperopt	0.7518	28.3
	Random Search	0.7114	14.13
	Grid Search	0.6742	569.74
DIABETIC	Hyperopt	0.5055	5.45
	Random Search	0.4906	2.87
	Grid Search	0.4707	289.98
DEFAULT	Hyperopt	0.5454	180.29
	Random Search	0.5003	115.88
	Grid Search	0.5258	3586.23
ABSENT	Hyperopt	0.9739	2.09
	Random Search	0.8710	1.68
	Grid Search	0.8593	141.15
CANCER	Hyperopt	0.2497	1.70
	Random Search	0.1961	1.05
	Grid Search	0.1670	98.54

Table 4. **The best parameters with each hyper-parameter optimization techniques for the six datasets**

Dataset	Parameters	Optimal value for each Methods		
		Hyperopt	Random Search	Grid Search
CARS	colsample_by_tree	0.9106	0.3072	0.5
	max_depth	6	16	3
	n_estimators	70	50	50
	reg_lambda	0.3798	0	0.05
	subsample	0.922	0.4692	1
BANK	colsample_by_tree	0.7644	0.2995	0.25
	max_depth	9	9	16
	n_estimators	70	200	50
	reg_lambda	0.9835	0.9096	0.1
	subsample	0.8857	0.41	0.9
DIABETIC	colsample_by_tree	0.5603	0.6249	0.25
	max_depth	15	16	3
	n_estimators	180	50	50
	reg_lambda	0.0838	0.8037	0.1
	subsample	0.8271	0.5357	0.9
DEFAULT	colsample_by_tree	0.9864	0.3709	0.5
	max_depth	7	16	16
	n_estimators	110	50	50
	reg_lambda	0.395	0.1968	0.5
	subsample	0.9772	0.3009	0.9
ABSENT	colsample_by_tree	0.5029	0.9897	1
	max_depth	6	6	3
	n_estimators	70	150	50
	reg_lambda	0.9742	0.3433	0.1
	subsample	0.9508	0.4753	1
CANCER	colsample_by_tree	0.7098	0.3072	1
	max_depth	16	6	12
	n_estimators	70	200	200
	reg_lambda	0.2206	0.4384	0.5
	subsample	0.8694	0.7926	1

5. CONCLUSION

In this paper, we perform hyperparameter optimization of the XGBoost algorithm on six real world datasets using Hyperopt, Random search and Grid search. We then compare the performances of each of these three techniques using both accuracy and time taken. We find that the Hyperopt performs better than the Grid search and Random search approaches since it has a higher mean Gini score that indicates higher prediction accuracy. In terms of time taken, the Grid search method is the most time consuming whereas the Random search method is the fastest followed closely by Hyperopt. Thus, taking into account both accuracy and time taken, hyperparameter optimization using Hyperopt is the best approach.

6. ACKNOWLEDGMENTS

Our thanks to VMware for giving us the required infrastructure and support for this work. We also thank M. Zwitter and M. Soklic for making the breast cancer dataset publicly available.

7. REFERENCES

[1] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl, “Algorithms for Hyper-Parameter Optimization,” *NIPS*, pp. 1–9, 2011.

[2] J. Snoek, H. Larochelle, and R. P. Adams, “Practical

bayesian optimization of machine learning algorithms," in *Proc. of NIPS*, 2012.

[3] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[4] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[5] R. Schaer, H. Muller, and A. Depeursinge, "Optimized Distributed Hyperparameter Search and Simulation for Lung Texture Classification in CT Using Hadoop," *J. Imaging*, vol. 2, no. 19, pp. 1–20, 2016.

[6] A. F. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, "Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017)*, 2017, pp. 528–536.

[7] I. Dewancker, M. McCourt, and S. Clark, "Bayesian Optimization Primer," *SigOpt*, 2016.

[8] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A. Castellanos, "Active policy learning for robot planning and exploration under uncertainty," in *Robotics: Science and Systems*, 2007, no. 321–328.

[9] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," in *Towards Global Optimization*, vol. 2, L. C. W. Dixon and G. P. Szego, Eds. New York: North Holland, 1978, pp. 117–129.

[10] P. Hennig and C. Schuler, "Entropy search for information-efficient global optimization," *JMLR*, vol. 1, 2012.

[11] J. Bergstra, D. Yamins, and D. D. Cox, "Making a Science of Model Search : Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures," in *Presented at the 30th International Conference on Machine Learning (ICML 2013), Atlanta, Gerorgia, June 16 – 21, 2013. In JMLR Workshop and Conference Proceedings 28 (1)*, 2013, pp. 115–123.

[12] J. Bergstra, D. Yamins, and D. D. Cox, " Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms," in *PROC. OF THE 12th PYTHON IN SCIENCE CONF. (SCIPY 2013)*, 2013.

[13] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: a Python library for model selection and hyperparameter optimization," *Comput. Sci. Discov.*, vol. 8, 2015.

[14] T. Nickson, M. A. Osborne, S. Reece, and S. Roberts, "Automated machine learning on big data using stochastic algorithm tuning," *CoRR*, 2014.

[15] T. Krueger, D. Panknin, and M. Braun, "Fast cross-validation via sequential testing," *JMLR*, 2015.

[16] J.-T. Tsai, J.-H. Chou, and T.-K. Liu, "Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm," *IEEE Trans. Neural Networks*, vol. 17, no. 1, pp. 69–80, 2006.

[17] S.-W. Lin, K.-C. Ying, S.-C. Chen, and Z.-J. Lee, "Particle swarm optimization for parameter determination and feature selection of support vector machines," *Expert Syst. Appl.*, vol. 35, no. 4, pp. 1817–1824, 2008.

[18] S. X. Souza, J. A. K. Suykens, J. Vandewalle, and D. Bolle, "Coupled simulated annealing," *IEEE Trans. Syst. Man, Cybern. Part B*, vol. 40, no. 2, pp. 320–335, 2010.

[19] D. Dheeru and E. Karra Taniskidou, "UCI Machine Learning Repository." 2017.

[20] M. Bohanec and V. Rajkovic, "Expert system for decision making," *Sistemica*, vol. 1, no. 1, pp. 145–157, 1990.

[21] S. Moro, P. Cortez, and P. Rita, "A Data-Driven Approach to Predict the Success of Bank Telemarketing," *Decis. Support Syst.*, vol. 62, pp. 22–31, 2014.

[22] B. Antal and A. Hajdu, "An ensemble-based system for automatic screening of diabetic retinopathy," *Knowledge-Based Syst.*, vol. 60, pp. 20–27, 2014.

[23] I. C. Yeh and C. H. Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 2473–2480, 2009.

[24] A. Martiniano, R. P. Ferreira, R. J. Sassi, and C. Affonso, "Application of a neuro fuzzy network in prediction of absenteeism at work," in *Information Systems and Technologies (CISTI), 7th Iberian Conference*, 2012, pp. 1–4.

[25] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, "The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains," in *Proceedings of the Fifth National Conference on Artificial Intelligence*, 1986, pp. 1041–1045.

[26] S. Mirzaei, G. Borzadaran, and M. Amini, "A Comparative Study of the Gini Coecient Estimators Based on the Linearization and U-Statistics Methods," *Rev. Colomb. Estad.*, vol. 40, no. 2, pp. 205–221, 2017.

[27] M. O. Lorenz, "Methods of measuring the concentration of wealth," *Q. Publ. Am. Stat. Assoc.*, vol. 9, no. 70, pp. 209–219, 1905.

[28] J. Bi, *Sensory Discrimination Tests and Measurements*. John wiley and sons, 2015.

[29] W. C. Lee and C. K. Hsiao, "Alternative summary indices for the receiver operating characteristic (ROC) curve," *Epidemiology*, vol. 7, pp. 605–611, 1996.