Bayesian Optimization for Predicting Rare Internal Failures in Manufacturing Processes

Abhinav Maurya
Department of Information Systems
H. J. Heinz III College
Carnegie Mellon University
Pittsburgh, PA - 15213
Email: ahmaurya@cmu.edu

***Abstract*—Modern manufacturing processes are highly instrumented to capture data at every step along the assembly lines. Such fine-grained data can help manufacturers perform quality control by predicting which manufactured products are at risk of being defective. A predictive analytics system that identifies internal failures in products before they are shipped can result in significant savings for the manufacturer and result in better customer satisfaction. However, predicting internal failures in manufactured products is a challenging machine learning task due to two reasons: (i) the rarity of such failures in modern manufacturing processes, and (ii) the failure of traditional machine learning algorithms to optimize non-convex metrics such as Matthew's Correlation Coefficient (MCC) used to measure performance on imbalanced datasets. In this paper, we present "ImbalancedBayesOpt", a meta-optimization algorithm that directly maximizes MCC by learning optimal weights for the positive and negative classes. We use Gradient Boosting Machine (GBM) as the base classifier for our meta-optimization algorithm due to its competitive performance on machine learning prediction tasks. Our quantitative evaluation suggests that "ImbalancedBayesOpt" can significantly improve the classification performance of base classifiers on severely imbalanced high-dimensional datasets.**

***Index Terms*—Bayesian Optimization, Gaussian Processes, Manufacturing Failure Detection, Imbalanced Binary Classification**

I. Introduction

Internet of Things (IoT) [1] has ushered in the age of ubiquitous intelligent goods and services. According to Gartner [2], IoT is estimated to comprise of 26 billion devices and to contribute 1.9 trillion USD to the global economy by 2020. This unprecedented increase in deployed sensing infrastructure and services can be attributed to a confluence of many factors such as:

- continued decline in the prices of sensors [3],
- non-digital "More-than-Moore" technologies [4], [5], [6] such as sensors, actuators, etc. which may not scale according to Moore's Law but provide diversified value addition to customers of semiconductor goods,
- rapid decline in the price of data storage and computation [7], [8] required to support advanced analytics.

The abundance of data collected using IoT necessitates the use of advanced analytics to mine useful insights and business value from the collected data. However, IoT data analytics brings its own set of challenges. The datasets collected using sensors are noisier and less interpretable than the structured datasets collected through human interaction on Internet websites.

The scale of data generated by IoT systems is another significant challenge for current state-of-the-art big data technologies. For example, General Electric (GE) estimates that each GE jet aircraft engine produces around one terabyte of data per flight. With around 20,000 planes and 5-10 flights per plane per day, the amount of data collected is staggering. Deriving insights from the vast amount of generated data requires creating high-performance data engineering systems and algorithms tuned to the specific type of sensor data being analyzed.

With the advent of Industrial Internet of Things (IIoT), manufacturing processes and product lifecycles are being continuously monitored at every step for any irregularities or failures [9], [10]. Jeff Immelt, CEO of General Electric, stated at GE's third "Mind+Machines" summit, that the corporation analyses around 50 million variables from 10 million sensors deployed in various scenarios such as railways, aviation, healthcare, and manufacturing [11]. According to Stefan Finkbeiner, CEO of Bosch Sensortec, Bosch globally produces around 4 million MEMS sensors on a daily basis, roughly 25 percent of which are used in the automotive sector [12]. Bosch has manufactured and sold a total of 7 billion MEMS sensors to date.

978-1-4673-9005-7/16/$31.00 ©2016 IEEE

Simply put, advanced analytics in the era of Industrial Internet poses many challenges such as:

- scale of collected data
- heterogeneity of data sources
- online and asynchronous generation of data by sensors
- faulty or dead sensors which lead to missing data issues

In this paper, our goal is to predict rare internal failures in manufacturing processes. There are two possible approaches to tackle this problem:

1) *Anomaly Detection:* Anomaly detection is an unsupervised machine learning task in which the goal is to detect if a datapoint is anomalous and unlikely to have been generated through the normal data-generating process. The data-generating process is learned from background data which signifies typical system behaviour and does not contain any anomalies. This method is best suited for detecting new anomalies that have not been seen before in the data.
2) *Binary Classification:* In this supervised machine learning task, we pose the task of predicting internal manufacturing failures as that of classifying datapoints into two classes. One of the classes indicates that no internal failure occurred, while the second class signifies that an internal failure was detected. Given examples of datapoints each marked with one of the classes, a binary classification algorithm can learn to predict the correct class for a datapoint. Typical binary classification algorithms assume that the number of datapoints in the two classes are roughly equal i.e. the classes are balanced. This assumption is violated when detecting internal manufacturing failures, since these occur very rarely in typical production workflows.

We adopt the second approach in order to utilize supervision when the dataset contains datapoints that are specifically marked as anomalous. In order to deal with the severe imbalance in the number of positive and negative datapoints, we choose to learn a weight parameter w that trades off between the losses incurred on the positive and negative datapoints. Specifically, we use a loss function which is the weighted sum of losses incurred over each datapoint, with the weight for positive datapoints being w and the weight for negative datapoints being $(1 - w)$. Many competitive classification algorithms such as Gradient Boosted Machines (GBMs) [13], Support Vector Machines (SVMs) [14], Decision Trees [15], Random Forests [16], Deep Feedforward Networks [17], etc. provide such linearly separable loss functions.

In section II, we present related literature and contrast our contributions with previous work. In section III, we describe a dataset on internal manufacturing failures that we use to evaluate our proposed method "ImbalancedBayesOpt", and metrics suitable for judging binary classification performance on imbalanced datasets. In section IV, we motivate and present our method "ImbalancedBayesOpt" that directly optimizes Matthew's Correlation Coefficient (MCC) by using Gradient Boosting Machine (GBM) as a base classifier. In section V, we describe our experimental setup for reproducibility of the experiments and present results for "ImbalancedBayesOpt" as well as the simpler approaches of "GBM" and "ImbalancedGridOpt". We finally end the paper with a discussion of the results and conclusions in section VI.

II. Related Work

When performing binary classification on imbalanced datasets, accuracy is not the right metric to optimize. For example, in a dataset with 99% negative datapoints, it is trivial to get 99% accuracy by using a degenerate model that always outputs 0. Therefore, other metrics have been devised to measure the performance of classifiers on imbalanced datasets:-

- Precision (P): $\frac{TP}{TP+FP}$
- Recall (R): $\frac{TP}{TP+FN}$
- F_1-Score (F_1): $\frac{2 \cdot P \cdot R}{P+R}$
- Matthew's Correlation Coefficient (MCC): $\frac{TP*TN-FP*FN}{\sqrt{(TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)}}$

The above metrics provide a much better idea of a classifier's performance on imbalanced datasets. In this paper, we focus on optimizing MCC by correcting for class imbalance. However, the method can be used for optimizing other metrics such as precision and recall as well.

There is considerable prior work in optimizing the non-convex metric of interest directly rather than optimizing accuracy as most machine learning classification algorithms do. Much of this work focuses on optimizing precision while maintaining some minimum recall, or optimizing recall while maintaining precision above a certain threshold, or maximizing the mean average precision over queries in information retrieval. There is no general framework for directly optimizing any non-convex metric specified by the domain expert for binary classification on imbalanced datasets. Our work provides a general Gaussian process-based approach to optimizing any non-convex binary classification metric specified by the domain expert as suitable for judging the performance of binary classifiers on imbalanced datasets from the domain.

We describe our method with respect to optimizing Matthew's Correlation Coefficient (MCC). However, our approach is more general and can be easily applied to optimizing other metrics such as precision, recall, Area Under Curve (AUC), etc. without any significant methodological change to the approach.

Prior work in [13] is a good off-the-shelf classifier to use for binary classification. It has been known to provide competitive results on many public data science competitions. However, the statistical model in [13] optimizes on prediction accuracy and performs poorly on imbalanced datasets. Prior research in [18], [19] provides a framework for optimizing precision. However, these works are limited to linear hypothesis classifiers, and require considerable changes and computational overhead to extend them to the non-linear case and to new metrics such as MCC. In [20], the authors also try to maximize the sum of precision P and recall R such that $R > P$ by varying classification threshold. However, our focus is on optimizing MCC rather than precision or recall. Additionally, the optimization procedure still optimizes accuracy rather than the non-convex metric in question. Hence, this method will perform poorly on imbalanced datasets unlike our method that directly optimizes for MCC.

Prior work in [21] maximizes recall (quantity of returned results) subject to a minimum precision (quality of returned results). Also, they do not directly optimize for MCC but rather still optimize for accuracy, and then choose the threshold that provides highest recall at a given precision. Using the same approach to get highest precision at a given recall is still different from our method, because in our method, we directly optimize for MCC. [22], [23] optimize related metric of AUC, rather than optimizing MCC as in our case.

Prior work in [24] uses leading indicators for building static classification rules but does not build a data-driven statistical model. Building such models shows that the optimally informative features and decision rules can be different in different manufacturing units. Hence, a static decision rule based on a single set of leading indicators cannot be universally optimal for detecting internal failures across manufacturing units. Prior work in [25] finds residuals between predictions and actual values to detect changes in incoming data distribution and target being predicted. However, it detects anomalies, but does not predict them. It also does not identify a classification model suitable for the task of detecting rare internal manufacturing failures.

III. Dataset

The dataset used to evaluate the methods described in this paper was provided by Bosch as a part of a data science competition[1] hosted on Kaggle. Each datapoint in the dataset represents measurements of manufactured parts as they move through Bosch's production lines. Each datapoint includes a unique "Id" that was assigned to the manufactured part. The binary target variable indicates if the part will fail quality control, with a value of 1 indicating failure.

The dataset consists of a large number of anonymized features, where each feature is specific to a production line and a station on the line. For example, L2_S95_F3456 is the 3456^{th} anonymous feature measured during the product manufacturing process on line 2 and station 95.

The massive dataset was provided in the form of three types of files:

- Numerical: This type of file contains real-valued features.
- Categorical: This type of file records discrete-valued features.
- Date: This type of file records the timestamp for when each numeric or categorical feature was recorded.

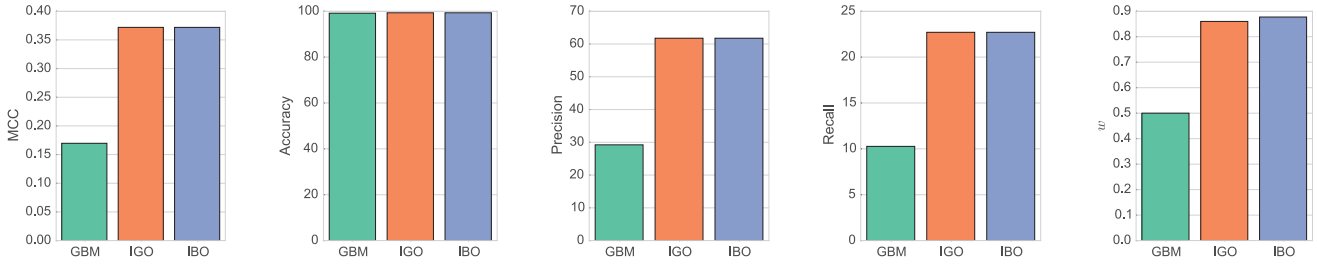
There are two files per type: first for the training dataset, and second for the test dataset. Hence, the final input data files are as follows:-

- train_numeric.csv - the training set numeric features, which contains the 'Response' variable.
- test_numeric.csv - the test set numeric features, which contains the Ids for which the 'Response' variable must be predicted.
- train_categorical.csv - the training set categorical features
- test_categorical.csv - the test set categorical features
- train_date.csv - the training set date features
- test_date.csv - the test set date features

We load the three training data files - train_numeric.csv, train_categorical.csv, and train_date.csv - and join the resulting data tables on the 'Id' field to obtain a single training data table *train_data*. We similarly join the data tables obtained from the three test data files - test_numeric.csv, test_categorical.csv, and test_date.csv - to obtain a single test data table *test_data*. Thus, *train_data* and *test_data* have identical fields, except that *train_data* also includes the 'Response' field. We convert the categorical variables into dummy binary features using one-hot encoding. We do not perform any further feature preprocessing or feature selection, except setting any missing values to 0.

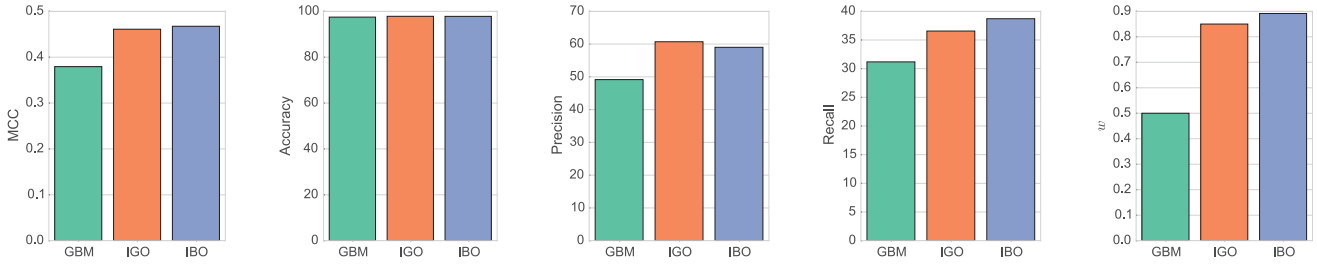
The two classes in the training dataset are highly imbalanced. From the summary statistics table (I), we see that the ratio of positive to negative datapoints in the training dataset is 0.0058, which makes the classification task extremely challenging.

[1] https://www.kaggle.com/c/bosch-production-line-performance



(a) MCC (b) Accuracy (c) Precision (d) Recall (e) Chosen w

Fig. 1: Figures showing results on the *Unbiased_10* dataset for the three algorithms - GBM, IGO, and IBO. Figures (a-d) show the metrics of MCC, Accuracy, Precision, and Recall respectively. Figure (e) shows the optimal value of w selected by IGO and IBO against the default value of $w = 0.5$ for GBM.



(a) MCC (b) Accuracy (c) Precision (d) Recall (e) Chosen w

Fig. 2: Figures showing results on the *Unbiased_1* dataset for the three algorithms - GBM, IGO, and IBO. Figures (a-d) show the metrics of MCC, Accuracy, Precision, and Recall respectively. Figure (e) shows the optimal value of w selected by IGO and IBO against the default value of $w = 0.5$ for GBM.

Statistic	Value
Total Number of Datapoints	1183747
Total Number of Features	6747
Number of Positive Datapoints	6879
Number of Negative Datapoints	1176868
Number of Train Datapoints	946999
Number of Test Datapoints	236748
Ratio of +ve/-ve Datapoints	0.0058451755

TABLE I: Dataset Summary Statistics

IV. Methodology

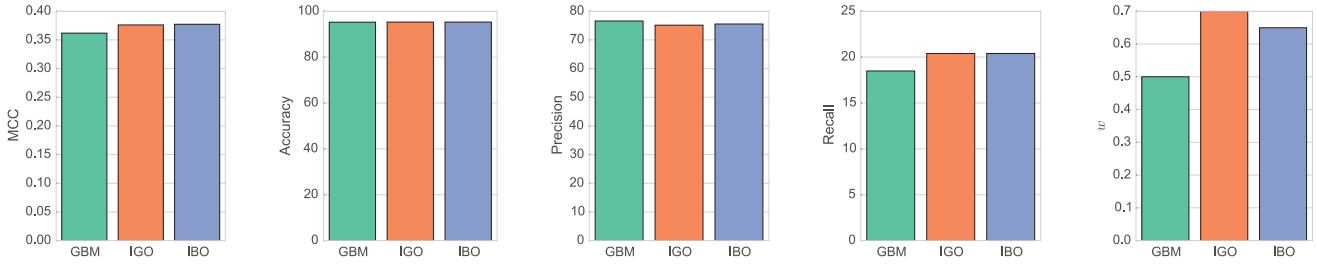
In this section, we describe our methods "ImbalancedGridOpt" (IGO) and "ImbalancedBayesOpt" (IBO) for optimizing MCC through hyperparameter search on a uniform grid and using a Gaussian process [26], [27] respectively.

A. Gradient Boosting Machine (GBM)

We use the Gradient Boosting Machine (GBM) classifier [13] to obtain our prediction models. We use the `scikit-learn` package in Python to learn the model. GBM is known to provide competitive performance compared to other commonly used, off-the-shelf classifiers such as Decision Trees [15], Random Forests [16], K-Nearest Neighbor Classification [28], Support Vector Machines [14], Logistic Regression [29], and Naive Bayes [29]. We refer to this approach of classification as "GBM".

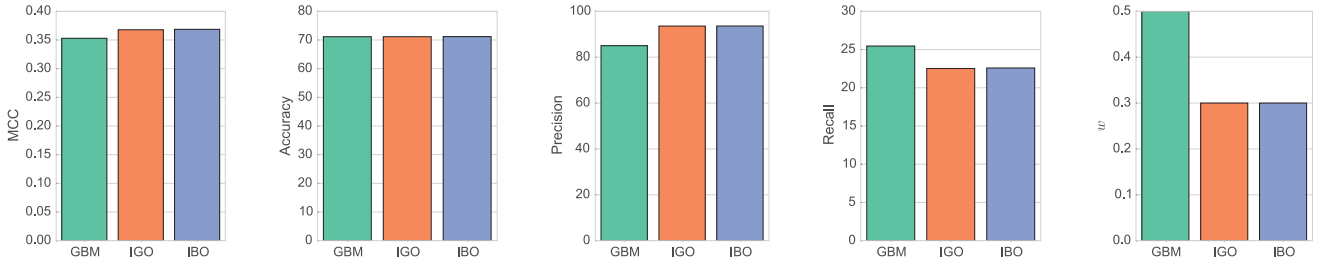
B. Optimizing MCC on Imbalanced Datasets

Our research question is to detect if a manufactured part suffers from internal defects, based on sensor measurements from the assembly lines during the manufacturing process. Since internal defect rates using modern manufacturing processes are low, the resulting dataset is often highly imbalanced, and accuracy is therefore a poor metric to optimize for actionability. It is more important to have high quality predictions in the top-k predictions (ranked by probability of belonging to the positive class). Such a goal is not captured by the traditional machine learning metric of accuracy. For example, in a dataset with 99% negative datapoints, it is trivial to get 99% accuracy by using a degenerate model that always outputs 0.



(a) MCC (b) Accuracy (c) Precision (d) Recall (e) Chosen w

Fig. 3: Figures showing results on the *Biased_10* dataset for the three algorithms - GBM, IGO, and IBO. Figures (a-d) show the metrics of MCC, Accuracy, Precision, and Recall respectively. Figure (e) shows the optimal value of w selected by IGO and IBO against the default value of $w = 0.5$ for GBM.



(a) MCC (b) Accuracy (c) Precision (d) Recall (e) Chosen w

Fig. 4: Figures showing results on the *Biased_1* dataset for the three algorithms - GBM, IGO, and IBO. Figures (a-d) show the metrics of MCC, Accuracy, Precision, and Recall respectively. Figure (e) shows the optimal value of w selected by IGO and IBO against the default value of $w = 0.5$ for GBM.

Notation	Description
N	Total Number of Datapoints
TP	True Positives
TN	True Negatives
FP	False Positives
FN	False Negatives
w	weight of positive datapoints
$(1-w)$	weight of negative datapoints
$g(w)$	weighted loss function
$M(w)$	classification model learned with weight w
$MCC(w)$	MCC of classification model learned with weight w
$loss(y_i, \hat{y}_i)$	one of various losses between y_i and $\hat{y}_i$ such as 0-1 loss, hinge loss, etc.

TABLE II: Table of Notation

We wish to maximize the evaluation metric of Matthew's Correlation Coefficient (MCC). However, optimizing for MCC is computationally difficult since it is a structured, non-convex function that depends on predictions for the entire dataset and cannot be linearly decomposed over individual datapoints [30].

Most machine learning classifiers optimize for the simpler, linearly decomposable metric of accuracy as follows:

$$\underset{M(w)}{\text{maximize}} \quad (TP + TN) \tag{1}$$

$$\therefore \quad \underset{M(w)}{\text{maximize}} \quad (N - FP - FN) \tag{2}$$

$$\therefore \quad \underset{M(w)}{\text{minimize}} \quad (FP + FN) \tag{3}$$

$$\therefore \quad \underset{M(w)}{\text{minimize}} \quad \sum_{i=1}^{N} loss(y_i, \hat{y}_i) \tag{4}$$

We optimize a modified parameterized objective function $g(w)$ with parameter w such that parameter balances the trade-off between losses incurred on the datapoints belonging to the positive and negative classes. The model obtained by solving the parameterized optimization problem is denoted by $M(w)$. By varying the parameter w, we can correct the imbalance between losses incurred on the datapoints belonging to the

positive and negative classes.

$$\begin{aligned} M(w) = &\ \text{argmin}\ g(w) \\ = &\ \text{argmin}\ \{w \cdot FP + (1-w) \cdot FN\} \end{aligned} \tag{5}$$

where w is a parameter between $(0, 1)$.

$M(w)$ in equation (5) is a machine learning classification model obtained by minimizing the weighted loss function $g(w) = \{w \cdot FP + (1-w) \cdot FN\}$. We can calculate its Matthew's Correlation Coefficient $MCC(w)$ on a held-out validation dataset. Thus, we have a technique for providing a weight w, optimizing the weighted loss function $g(w)$ on the training dataset to obtain a classification model $M(w)$, and receiving the evaluation metric $MCC(w)$ on the validation dataset. By varying w, we search over a large space of machine learning models for a model M^* that directly optimizes $MCC(w)$ as follows:-

$$M^* = \underset{M(w)}{\text{argmax}} \quad MCC(w) \tag{6}$$

We note that w cannot be exactly equal to either 0 or 1 because these values lead to a trivial loss over datapoints of just one class and therefore a trivial baseline output since the loss in this case can be best minimized by outputting 1 if $w = 0$ and 0 if $w = 1$. Our goal now is to search over the parameter w to obtain a statistical model M^* that provides maximum MCC.

1) ImbalancedGridOpt (IGO): A naive but easy to implement way to optimize MCC over the parameter w is to search uniformly over $w \in (0, 1)$. We used a uniformly spaced grid of 100 values in $(0, 1)$ to optimize $g(w)$. We call this search procedure "ImbalancedGridOpt", since it optimizes our metric of interest MCC on an imbalanced dataset using a grid search over the parameter w.

2) ImbalancedBayesOpt (IBO): Using a meta-optimization technique like Bayesian optimization, we can find the parameter w such that parametrized algorithm returns the highest MCC. Bayesian optimization [26], [31] based on Gaussian processes [27] is good for exploration versus exploitation trade-off in black-box optimization, and focuses on areas of parameter space w that have higher chances of attaining maximum objective value $MCC(w)$. As a result, we chose Bayesian optimization as our meta-optimization algorithm in addition to a simpler baseline "ImbalancedGridOpt" of naively searching over a uniform grid $w \in (0, 1)$. We call the Gaussian process-based Bayesian optimization of MCC as "ImbalancedBayesOpt" (IBO). We initialize the GP-based meta-optimization procedure with the objective value in optimization (7) at 10 uniformly spaced values of $w \in (0, 1)$. We then allow the GP to perform 10 iterative queries of the objective function before it returns us the value of w it considers optimal for maximizing the objective function in (7). We then use the returned optimal value of w to train a model using both the training and held-out data.

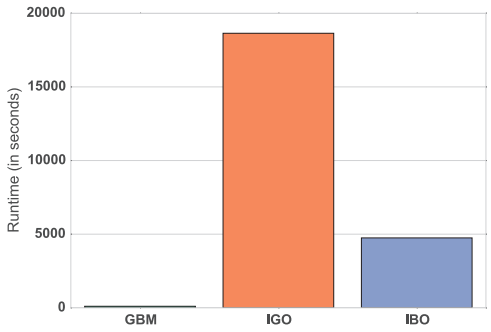


Fig. 9: Runtimes of the three algorithms.

3) Incorporating Constraints in Bayesian Optimization: In the current problem of optimizing MCC, we have not placed any constraints on other metrics like AUC, precision, recall, accuracy, etc. However, such constraints can be easily accommodated in the Bayesian Optimization framework by relaxing the constraint using hinge loss[2] and adding this additional loss to the objective function. For example, if we have a constraint such as $Recall(w) >= r_0$, the violation incurred $\{r_0 - Recall(w)\}$ can be incorporated using hinge loss as follows:

$$M^* = \underset{M(w)}{\text{argmax}} \quad MCC(w) - \eta \cdot max\{0, r_0 - Recall(w)\} \tag{7}$$

Compared to the optimization problem in (6), the optimization objective in (7) converts the constraint into a hinge loss. When $Recall(w) \geq r_0$, $\{r_0 - Recall(w)\} \leq 0$ and therefore the hinge loss is inactive. When $Recall(w) < r_0$, $\{r_0 - Recall(w)\} > 0$ and the hinge loss drives the total optimization objective down so that the particular value of w is not chosen as optimal even if it maximizes $MCC(w)$. The multiplier of the hinge loss η is set to a sufficiently high value (e.g. $\eta = 100$) so that the constraint is satisfied and its violation incurs a huge but continuous loss.

C. Experimental Setup

For the two meta-optimization procedures "ImbalancedGridOpt" (IGO) and "ImbalancedBayesOpt" (IBO), we continue to use boosted classification trees [13] so that the results of our meta-optimization of predictive precision could be

[2]http://research.microsoft.com/en-us/um/people/manik/projects/trade-off/hinge.html

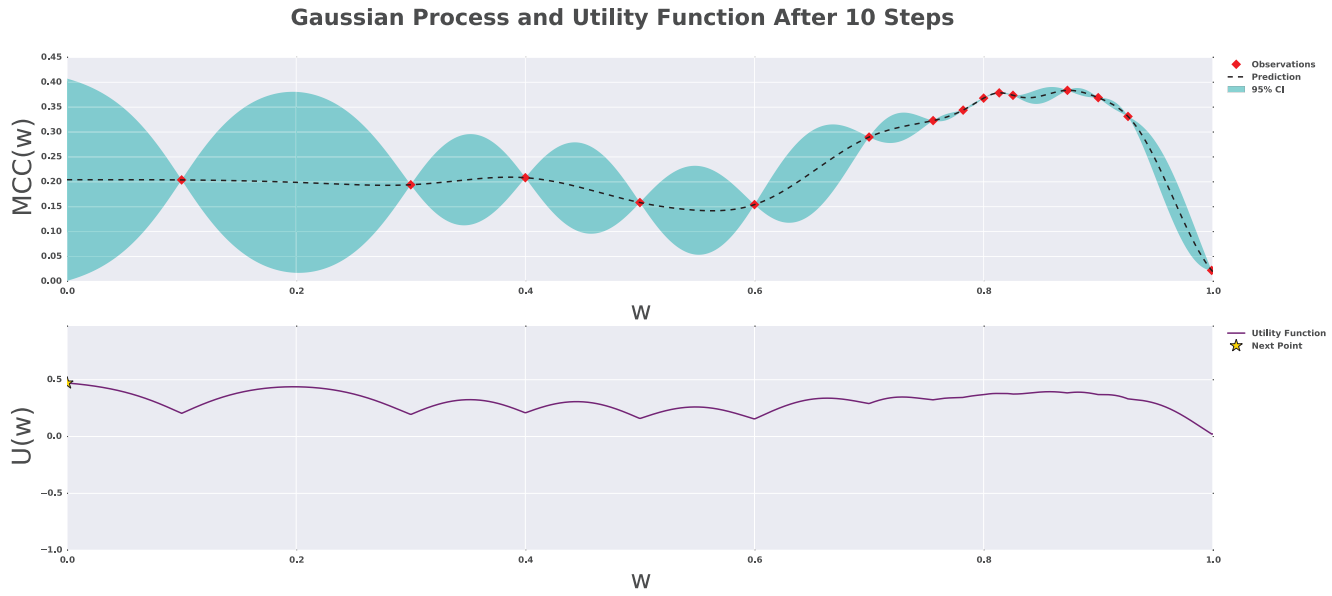


Fig. 5: Gaussian Process posterior and UCB utility function at the end of "ImbalancedBayesOpt" algorithm on the *Unbiased_10* dataset.

compared to the "GBM" technique we introduced earlier. We note that the weighted optimization procedure we propose can be performed using any optimization-based machine learning model whose loss function is linearly separable over the training datapoints. Examples of such classification models are Gradient Boosted Machines (GBMs) [13], Support Vector Machines (SVMs) [14], Decision Trees [15], Random Forests [16], Deep Feedforward Networks [17], etc. Thus, our meta-optimization technique to maximize MCC can use a wide variety of underlying machine learning algorithms.

We do not tune hyperparameters of GBM when using it as an independent classifier, or as a base classifier for "ImbalancedGridOpt" and "ImbalancedBayesOpt". We expect hyperparameter tuning will improve the results of all three - GBM, IGO, and IBO. We leave this investigation to a future version of our research.

The summary statistics of the Bosch manufacturing failures dataset are provided in table (I). Since it is a massive dataset, we subsampled the dataset to obtain smaller datasets that could be used to train the model quicker than using the entire dataset. Such subsampling is even more important in the meta-optimization algorithms "ImbalancedGridOpt" and "ImbalancedBayesOpt" since these algorithms fit the model numerous times to obtain the optimal value of w.

We perform two types of subsampling:

- *Unbiased:* In this subsampling approach, we subsample both the positive and negative datapoints using the same subsample rate s.
- *Biased:* In biased subsampling, we retain all datapoints of the positive class, since it is much rarer compared to the negative class. Datapoints of the negative class are subsampled at a subsample rate of s.

For each of the above subsampling schemes, we used two values of the subsample rate s - 0.01 and 0.1, indicating that 1% and 10% of the datapoints were subsampled respectively. Thus, we have 4 combinations of subsampling:

- *Unbiased_1:* Datapoints of both classes are subsampled at a rate of 1%.
- *Unbiased_10:* Datapoints of both classes are subsampled at a rate of 10%.
- *Biased_1:* Datapoints of negative class are subsampled at a rate of 1%. All positive datapoints are included in the subsampled dataset, being much rarer than datapoints of the negative class.
- *Biased_10:* Datapoints of negative class are subsampled at a rate of 10%. All positive datapoints are included in the subsampled dataset.

Each of the three algorithms - GBM, IGO, and IBO - was then applied to each subsampled dataset, and metrics such as MCC, Accuracy, Precision, and Recall were measured for each algorithm on each subsampled dataset to perform a comprehensive evaluation.

V. Results

Figures 1, 2, 3, and 4 show the results for the subsampled datasets *Unbiased_10*, *Unbiased_1*, *Biased_10*, and *Biased_1*

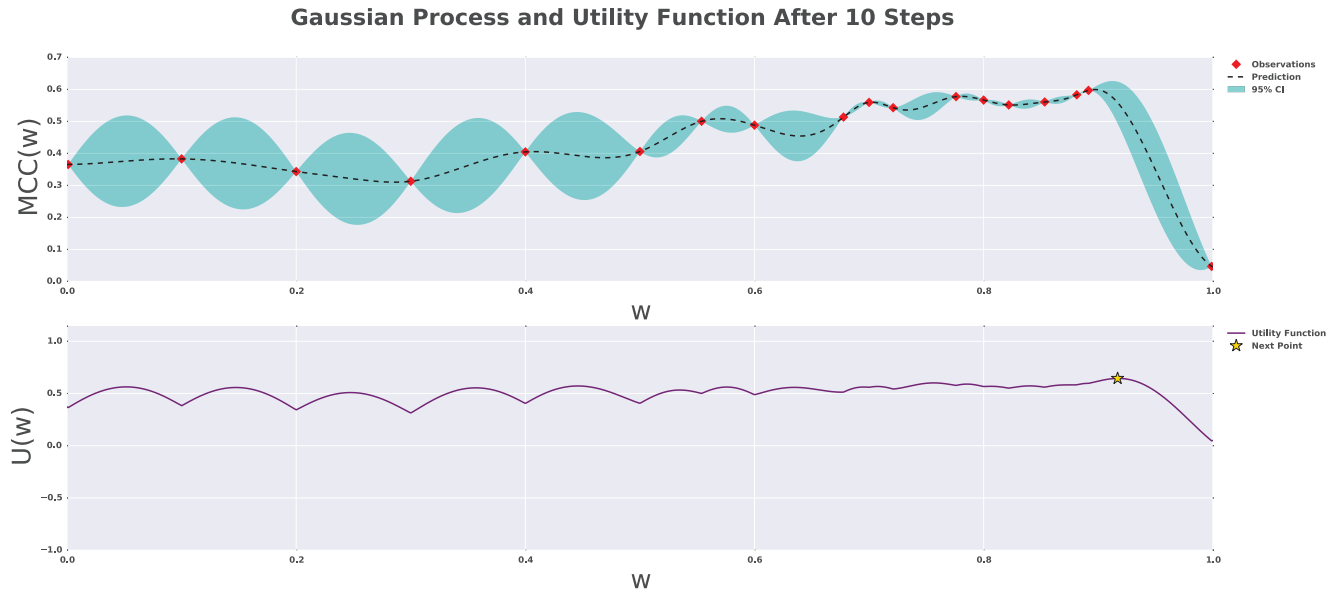


Fig. 6: Gaussian Process posterior and UCB utility function at the end of "ImbalancedBayesOpt" algorithm on the *Unbiased_1* dataset.

respectively. Each figure shows four metrics - MCC, accuracy, precision, and recall - in sub-figures (a-d) respectively. Often, optimizing on one metric can lead to deteriorated performance of other metrics. We observe that precision and recall follow the same trend as MCC, since all three are dependent on the performance of the classifier in identifying positive datapoints. Interestingly, accuracy does not deteriorate for IGO and IBO, indicating that tuning the classifier to better recognize rare positive datapoints does not have a severe deteriorating effect on its performance in recognizing datapoints of the majority negative class. The fifth sub-figure shows the optimal value of w chosen by "ImbalancedGridOpt" (IGO) and "ImbalancedBayesOpt" (IBO) against the default value of $w = 0.5$ for GBM. When $w = 0.5$, the losses for both classes are weighted equally ($w = 1 - w = 0.5$), and hence this value corresponds to the default "GBM" classifier.

In the case of the unbiased datasets where the proportion of positive to negative datapoints is maintained as in the original dataset, we observe that optimizing $MCC(w)$ by searching for a corresponding optimal w performs much better than "GBM". The increase in negative datapoints in the *Unbiased_10* dataset compared to the *Unbiased_1* dataset is much higher than the corresponding increase in number of positive datapoints, since the latter are much rarer and a very small fraction of the total dataset. As a result of this increased difference between the number of negative and positive datapoints in the *Unbiased_10* dataset, "GBM" - which optimizes the unweighted sum of losses over all datapoints - fares much poorly on the *Unbiased_10* dataset compared to the *Unbiased_1* dataset.

We now look at the results on the biased datasets where all positive datapoints are included in the subsampled but the negative datapoints are subsampled at a certain rate. Figures 3 and 4 show results for the biased datasets *Biased_10* and *Biased_1* respectively. We notice that IGO and IBO improve the MCC metric but not by much. This can be expected since the extremely biased subsampling of the dataset (sampling positive datapoints with a probability of 1.0) corrects the severe imbalance in the training datasets. We make two observations with regard to the results on the datasets created through biased subsampling:

1) IBO and IGO perform remarkably well in the case of severely imbalanced datasets. Their performance on the *Unbiased_10* and *Unbiased_1* datasets can be taken as indicative of their performance on the entire dataset, since all three datasets have the same rare fraction of positive and negative datapoints.
2) Performance on the biased datasets indicates that if there are sufficient datapoints of the anomalous positive class for the classification model to capture and characterize the anomalous signals, then weighting mechanisms like IGO and IBO are not required to get good anomaly detection performance. However, getting a really high fraction of anomalous datapoints might often not be possible in many real-world scenarios. The *Biased_10* and *Biased_1* datasets have an unusually high fraction

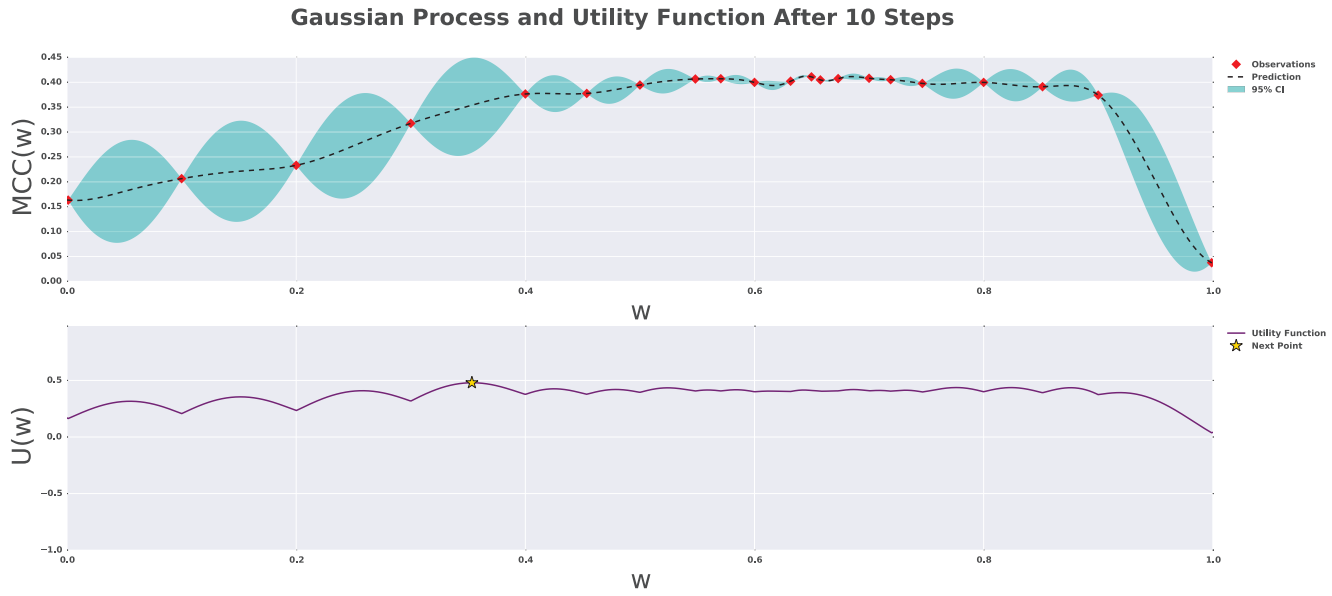


Fig. 7: Gaussian Process posterior and UCB utility function at the end of "ImbalancedBayesOpt" algorithm on the *Biased_10* dataset.

of positive anomalous datapoints due to the artifact of biased subsampling.

Figures 5, 6, 7, and 8 show the final Gaussian process posterior and the utility function for the subsampled datasets *Unbiased_10*, *Unbiased_1*, *Biased_10*, and *Biased_1* respectively. The Gaussian process posterior shows the mean and variance functions over the domain of $w = (0, 1)$. The utility function is used for deciding which value of w will be chosen next for evaluating $MCC(w)$. In our implementation, we use the Upper Confidence Bound (UCB) utility function [31] to iteratively select the next w at which to evaluate $MCC(w)$.We notice that the Gaussian Process spends significant time in the domain of w that has the potential to return high values of $MCC(w)$.

Finally, we distinguish between "ImbalancedGridOpt" (IGO) and "ImbalancedBayesOpt" (IBO) on the basis of runtime shown in figure (9). IGO evaluates $MCC(w)$ at 100 uniformly spaced values of w. "ImbalancedBayesOpt" is initialized with MCC values at 10 uniformly sampled values of w. It uses these 10 $MCC(w)$ values to build a GP posterior. On the basis of this uniformly initialized posterior, "ImbalancedBayesOpt" iteratively samples 10 more w values, building a classifier using the sampled weight w and obtaining $MCC(w)$ for the built classifier. After building each classifier and evaluating $MCC(w)$ for the classifier, the Gaussian Process updates the posterior using the $(w, MCC(w))$ pair of values [26]. Since "ImbalancedBayesOpt" builds 20 classifiers and "ImbalancedGridOpt" builds 100 classifiers, "ImbalancedBayesOpt" is 5 times faster than "ImbalancedGridOpt" while providing comparable values of the attained MCC.

VI. Conclusions

We presented a framework for optimizing non-convex metrics such as Matthew's Correlation Coefficient (MCC) that are more suited for measuring classification performance on imbalanced datasets. We investigated its performance on four subsampled datasets where we varied the subsampling rate as well as the subsampling bias between the positive and negative classes. We found that our proposed method "ImbalancedBayesOpt" performs better than "GBM" in scenarios where the dataset is severely imbalanced. It performs comparably to a naive grid search algorithm "ImbalancedGridOpt", while providing a considerable speedup from "ImbalancedGridOpt" under our experimental setup.

References

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A Survey. *Computer networks*, 54(15):2787–2805, 2010.

[2] Peter Middleton, Peter Kjeldsen, and Jim Tully. Forecast: The Internet of Things, Worldwide, 2013. *Gartner Research*, 2013.

[3] Elgar Fleisch and Christian Tellkamp. The business value of ubiquitous computing technologies. In *Ubiquitous and pervasive commerce*, pages 93–113. Springer, 2006.

[4] Chris Mack. The Multiple Lives of Moore's Law. *IEEE Spectrum*, 52(4):31–31, 2015.

[5] Wolfgang Arden, Michel Brillouët, Patrick Cogez, Mart Graef, Bert Huizing, and Reinhard Mahnkopf. "More-than-Moore" White Paper. 2010.

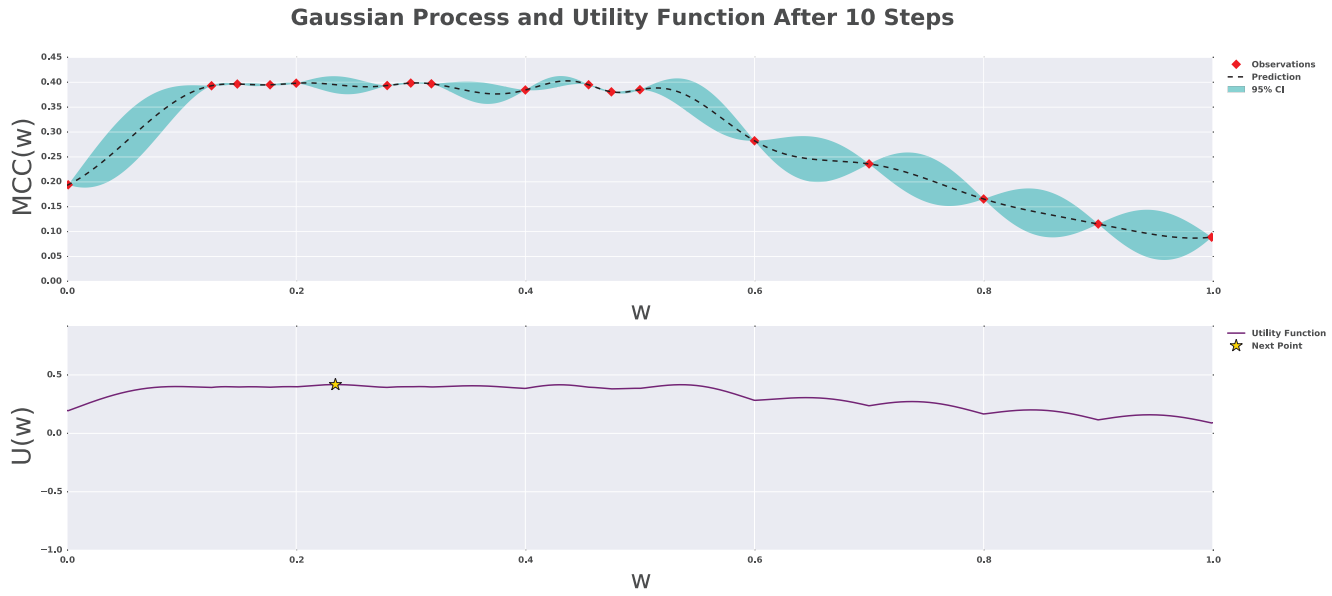


Fig. 8: Gaussian Process posterior and UCB utility function at the end of "ImbalancedBayesOpt" algorithm on the *Biased_1* dataset.

[6] Andrew B Kahng. Scaling: More than Moore's law. *IEEE Design and Test of Computers*, 27(3):86–87, 2010.

[7] Jim Gray and Prashant Shenoy. Rules of thumb in Data Engineering. In *Data Engineering, 2000. Proceedings. 16th International Conference on*, pages 3–10. IEEE, 2000.

[8] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of Cloud Computing. *Communications of the ACM*, 53(4):50–58, 2010.

[9] Li Da Xu, Wu He, and Shancang Li. Internet of Things in Industries: A Survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243, 2014.

[10] Marek Obitko, Václav Jirkovskỳ, and Jan Bezdíček. Big data challenges in industrial automation. In *International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, pages 305–316. Springer, 2013.

[11] Heather Clancy. "How GE generates USD 1 billion from data". http://fortune.com/2014/10/10/ge-data-robotics-sensors/, 2014. Online; accessed September 28, 2016.

[12] Wang Ying. Bosch Sensortec to focus on IoT and Wearables. http://europe.chinadaily.com.cn/business/2016-04/09/content_24392834.htm, 2016. Online; accessed September 28, 2016.

[13] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[14] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. 2001.

[15] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[16] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[17] Yoshua Bengio, Ian J Goodfellow, and Aaron Courville. Deep Learning. *An MIT Press book in preparation. Draft chapters available at http://www.iro.umontreal.ca/ bengioy/dlbook*, 2015.

[18] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 271–278. ACM, 2007.

[19] David MJ Tax, Marco Loog, and Robert PW Duin. Optimal mean-precision classifier. In *International Workshop on Multiple Classifier Systems*, pages 72–81. Springer, 2009.

[20] Xuchang Zou, Raffaella Settimi, Jane Cleland-Huang, and Chuan Duan. Thresholding strategy in requirements trace retrieval. Citeseer.

[21] Ljubomir Buturovic, Mike Wong, Grace W Tang, Russ B Altman, and Dragutin Petkovic. High Precision Prediction of Functional Sites in Protein Structures. *PloS one*, 9(3):e91240, 2014.

[22] Toon Calders and Szymon Jaroszewicz. Efficient AUC optimization for classification. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 42–53. Springer, 2007.

[23] Corinna Cortes and Mehryar Mohri. AUC optimization versus error rate minimization. 2004.

[24] J. Keyes and C. Litty. Systems and methods associated with targeted leading indicators, November 13 2003. US Patent App. 10/063,663.

[25] J.Z. Shan. Detecting change in data, November 16 2010. US Patent 7,836,111.

[26] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Slgorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

[27] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[28] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[29] A Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. 2002.

[30] Brian W Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[31] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proceedings of the 27th International Conference on Machine Learning*, number EPFL-CONF-161305. Omnipress, 2010.