SurfOpt: A New Surface Method for Optimizing the Classification of Imbalanced Dataset

André Rodrigo da Silva, Leonardo M. Rodrigues and Luciana de Oliveira Rech
Department of Informatics and Statistics (INE), Federal University of Santa Catarina (UFSC)
Rua Delfino Conti, s/n, Trindade, Cx.P. 476, CEP: 88040-900, Florianópolis, Brazil

Keywords: Machine Learning, Skewed Classes, Imbalanced Datasets, Binary Classification.

Abstract: Imbalanced classes constitute very complex machine learning classification problems, particularly if there are not many examples for training, in which case most algorithms fail to learn discriminant characteristics, and tend to completely ignore the minority class in favour of the model overall accuracy. Datasets with imbalanced classes are common in several machine learning applications, such as sales forecasting and fraud detection. Current strategies for dealing with imbalanced classes rely on manipulation of the datasets as a means of improving classification performance. Instead of optimizing classification boundaries based on some measure of distance to points, this work directly optimizes the decision surface, essentially turning a classification problem into a regression problem. We demonstrate that our approach is competitive in comparison to other classification algorithms for imbalanced classes, in addition to achieving different properties.

1 INTRODUCTION

In many scenarios of practical application of machine learning, such as sales forecasting (Syam and Sharma, 2018), epidemic prevention (Guo et al., 2017), fraud detection (Carneiro et al., 2017), and disease evolution (Zhao et al., 2017), the subpopulation (or class of interest) may consist of a derisory portion of the observed events, as in the analogy "needle in a haystack", but even harder than that. The events observed as data points may be indistinguishable from one another for prediction purposes. For example, the next customer to close a deal with an online business may have many, if not all, characteristics of a customer who has not closed a deal in the past.

As it is not always feasible to sample all the relevant characteristics of a population, the discriminating characteristics of a population are commonly neglected. This causes the classifier's performance to fall due to the fact that the data points began to appear closer and show more similarity, with overlapping class distributions, which makes decision boundaries subject to uncertainty. This work addresses typical problems in this type of scenario.

Although binary classification problems are not hard classification problems alone, characteristics on the data points and intricacies of the problem can add further complexity. Typical examples of such characteristics are classes being severely skewed, not linearly separable, the set of features being of both categorical and continuous variables, and the observations being of sparse nature. Even algorithms that result in complex non-linear classification models will tend to stop classifying examples as being of the minority class altogether under skewness to minimize classification error.

The accuracy metric is the most popular measure of a classifier's performance since accuracy suffices as a metric for classification with balanced classes. For datasets with imbalanced classes, accuracy is misleading – as it is biased to the majority class – and insensitive to changes in the quality of classification of the minority class. For example, sales prediction is a scenario where accuracy limitations come into play. In this case, it is essential to know how many times buying customers (i.e. the minority class) were correctly predicted and to quantify to which fraction of the total buying customers events it corresponds. A model can be right every time it predicts a buying customer, but if its predictions of non-buying customers are not also perfect, it can still be misclassifying most of the buying customers.

Cross-validation of binary classifiers results in four primary metrics, true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). From these metrics, it becomes possible to com-

Rodrigo da Silva, A., Rodrigues, L. and Rech, L.
SurfOpt: A New Surface Method for Optimizing the Classification of Imbalanced Dataset.
DOI: 10.5220/0007407107240730
In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART 2019)*, pages 724-730
ISBN: 978-989-758-350-6

pose more advanced metrics, such as specificity or also called True Negative Rate (TNR), which measures the rate of negative examples being predicted correctly versus the total negative cases. Another useful metric is the Negative Predictive Value (NPV), which is the ratio between the number of correctly predicted negative cases and the total number of predicted negative cases. When dealing with skewed data, the accuracy metric of a classifier tends to be very stable and unable to represent changes of specificity or negative predictive value (assuming minority class as negative). Recent years have been of growing interest on better overall metrics to evaluate a classifier that can represent changes in the prediction of the minority class, such as the F-measure (F1) (Lipton et al., 2014) and the Matthews Correlation Coefficient (MCC) (Boughorbel et al., 2017).

This paper presents an algorithm that handles classifications tasks with data imbalance and evaluates its classification behaviour. The main contributions of this work are the following:

- The novel SurfOpt algorithm, which is able to handle properly data imbalance;
- A parameter-based strategy that allows to directly optimize the classification surface;
- A three-metric diagnostic approach for performance issues on classifiers;

The next sections of this paper are organised as follows. Section 2 presents the related work. Section 3 includes essential concepts for the development of this work. Section 4 introduces the SurfOpt algorithm, which is the main contribution of this work. Section 5 presents an evaluation of the proposed algorithm. Section 7 concludes the paper and presents suggestions for future work.

2 RELATED WORK

This section presents the work related to this research, including those considered essential in the areas of machine learning classification, support vector machines, and boosting-and-bagging algorithms.

2.1 Machine Learning Classification

A classification task is a problem where a population (e.g. flowers) needs to be discriminated in its different classes (e.g. iris setosa, versicolor, virginica), whereas automated classification consists of learning a function that maps input features (i.e. individual characteristics) to outputs (i.e. class labels).

2.2 Support Vector Machines (SVM)

Linear separable classes can be easily classified by a model generated by the SVM algorithm, which works by finding an $n-1$ dimensional hyperplane that separates classes in a n dimensional space, maximising the margin between the hyperplane and each set of points. This algorithm is convex, meaning that it always find the optimal solution. Not linearly separable classes can also be efficiently classified with SVM by using a kernel. The kernel-trick projects points in a higher dimensional space where they are linearly separable (Boser et al., 1992; Hofmann, 2006). When used in a two-dimensional space, SVM is equal to dividing points with a straight line.

2.3 Boosting and Bagging

Adaboost and Adabag algorithms are ensemble methods that work by iteratively improving classification results with weak classifiers. The ensemble model of an Adaboost algorithm elects to which class every observation belongs by using a weighted vote strategy, where the weights of each classifier votes are learned during training. Adabag uses a simple majority vote strategy, and classifiers added to the ensemble in every iteration are independent of the previous classifiers, opposing Adaboost, which derives classifiers from previous iterations (Alfaro et al., 2013; Schapire, 2013).

2.4 Oversampling and Undersampling

Recent work for imbalanced dataset classification use non-algorithmic solutions, such as oversampling, undersampling or a combination of both strategies (Haixiang et al., 2017). Oversampling approaches artificially increase the number of data-points of the minority class. Undersampling techniques consist of removing data-points of the majority class to balance the dataset. These techniques manipulate the dataset as a mean to improve the classifier performance.

2.5 Surface Optimization

Algorithms for classification generally optimize a classification boundary based on some distance-to-point metric (i.e. large margin classifiers). These algorithms showed satisfactory convergence properties and great accuracy in many practical applications. The weakness of this kind of algorithm lies in its sensibility to outliers, and the fact they optimize for individual points and not for the entire set, making classifiers subject to bias found in data.

SurfOpt optimizes decision surface directly, being able to learn trade-off regions for classification, and being resilient to outliers as the influence of individual data points diminishes as the number of data points grow. We show that even maximum margin classifiers with polynomial kernels such as SVM are unable to capture trade-off regions on imbalanced data.

3 BACKGROUND

In this section, we will shortly review two concepts that are important to the SurfOpt algorithm formulation and evaluation.

3.1 Matthews Correlation Coefficient

Equation (1) presents the Matthews Correlation Coefficient (MCC), which can be undefined if any of the measures TP, TN, FP, TN are equal to zero. The addition of an infinitesimal positive constant to the denominator of the formula solves that problem. The MCC equation outputs in the interval $[1,-1]$ if the MCC of a classifier is 1, which is perfectly correct. On the other hand, an MCC of -1 means a perfectly incorrect classifier.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP) \times (TP+FN) \times (TN+FP) \times (TN+FN)}} \quad (1)$$

3.2 Convex hull

The convex hull of a set of points is the smallest convex set that contains the points (Barber et al., 1996). The points on a convex hull define the vertices of a polygon that contains the entire set. Some implementations of the convex hull run on linear expected time (Devroye and Toussaint, 1981). In this work, we show how to avoid finding convex hull of sets and distance-to-point calculations by optimising a curve equation, and a transformation of the data-points.

4 SurfOpt: OPTIMISING CLASSIFICATION SURFACE

When classifying data-points of populations that overlap one another, a non-smooth threshold for the decision boundary such as a single straight line will fail to capture the region of best classification trade-off. To best capture the trade-off region, we envisioned a weak classifier with a non-linear decision boundary. Our classifier boundary is a curve, defined by the function below:

$$f(a) = a^2 * exp(\omega) \quad (2)$$

The curve defined by Equation (2) is convenient as it never assumes negative values. For large positive values of ω, it approximates a vertical line and, for small negative values of ω, it approximates a horizontal line. From Equation (2), we learn the parameter ω that define the width of the arc. To classify data-points outside of the curve, it is necessary to do a kernel transformation. Thus, concomitantly, we learn a transformation to the points. The parameter θ defines the angle of rotation to apply to every point $P(a,b)$ in relation to the origin point $P(0,0)$.

$$b = f(a) \quad (3)$$

$$a' = a\ \cos(\theta) - b\ \sin(\theta) \quad (4)$$

$$b' = a\ \sin(\theta) + b\ \cos(\theta) \quad (5)$$

In addition to the rotation transformation, we add the parameters c and d to shift every point a and b coordinates in relation to the origin point $P(0,0)$, as depicted in Figure 1.

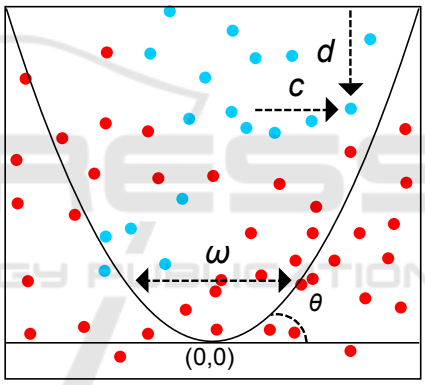

Figure 1: Curve boundary and transformations to the points.

$$k_a(a) = a\ \cos(\theta) - b\ \sin(\theta) + c \quad (6)$$

$$k_a(b) = a\ \sin(\theta) + b\ \cos(\theta) + d \quad (7)$$

Equations (6) and (7) transform the x-axis and the y-axis of the points on the plane, respectively. Once a curve decision boundary is set and the points transformed, we then use the Heaviside step function as the decision function of our algorithm.

$$g(a) = k_b(a) - f(a) \quad (8)$$

$$H(a) = \begin{cases} 0, & \text{if } g(a) < 0 \\ 1, & \text{if } g(a) \geq 0 \end{cases} \quad (9)$$

First, Equation 8 determines if a point is below or above (i.e. inside) the curve after the transformation. Then, if it is inside, Equation 9 labels the point as being of the minority class – denoted by 1 (one) – or the majority class – denoted by 0 (zero).

The area defined by the curve within the range of the data-points is an ellipse-segment, every point inside of it is predicted to belong to the minority class. The algorithm evaluates the classifier performance and yields a value $x \in [0,1]$ for classifier accuracy. In order to maximise x, we use the logistic loss, which can be defined as follows:

$$Cost(x,y) = \begin{cases} -log(h_\theta(x)), & \text{if y=1} \\ -log(1-h_\theta(x)), & \text{if y=0} \end{cases} \quad (10)$$

In Equation (10) $h_\theta(x)$ denotes the hypothesis function with parameters θ with input x, our hypothesis function are (6,7), we assume y to be always equal to 1 since it is the maximum value for the accuracy metric (perfect classifier) that we want to achieve. With Equation 10, we are able to calculate the gradient descent and penalise the parameters of our hypothesis function every iteration.

We calculate the gradient of the parameters with the sigmoid function, i.e. $f(x) = \frac{1}{1+e^{-x}}$, to minimize the error. We experimented different gradient descent algorithms such as Adam(Kingma and Ba, 2015) and Adagrad(Duchi et al., 2011), AMSgrad (Reddi et al., 2018) showed the best results. The AMSgrad proceeds to update the transformations parameters, i.e. Equations (6) and (7). Usually, the gradient descent algorithm would repeat the optimization steps until the max-number of iterations or the final performance is reached. In our algorithm, we use a restart routine whenever TNR and TPR metrics are below a threshold, as a way to avoid over-fitting of the accuracy metric. When the algorithm stops, we select the curve which resulted in the best MCC on the training set.

4.1 SurfOpt: Algorithm

This section presents the logic used in SurfOpt approach. Algorithm 1 describes the procedures performed for this optimising the classification surface.

Algorithm 1: SurfOpt.

S	data points;
C_i	points on the curve;
E_i	current evaluation of the iteration i;
L	labels of points;
c	x-offset of the curve origin;
d	y-offset of the curve origin;
θ	rotation angle;
ω	parameter, from Equation (2);
N	true negative rate threshold;
P	true positive rate threshold;
M	maximum iterations for the gradient descent;
r	resolution, number of points sampled in the curve;
α	learning rate;

function SAMPLECURVEPOINTS(S, R, c, d, θ, ω)
Sample R points from a parabola within some range beyond the data points, transformed to the parameters c, d, θ, ω with Equations (6) and (7).
return C
end function

procedure RESTARTCURVEPOSITION(c,d,θ,b)
sets c (x-offset) as a random value within $[1,-1]$ times the sample standard deviation in x-axis;
sets d (y-offset) as a random value within $[1,-1]$ times the sample standard deviation in y-axis;
sets θ (rotation) with a random value within $[2,4] \times \pi$;
resets ω to its original value;
end procedure

function CLASSIFIEREVALUATION(S, C_i)
Transforms data-points: every point inside the arc of the curve defined by Equation (2) is predicted to be of the minority class. Otherwise, it predicts the point to belong to the majority class.
return E_i
end function

function LEARNCURVE(S, L, N, P, M, b, R)
$\omega_0 = \omega$
RESTARTCURVEPOSITION(c, d, θ, ω)
for i in 1 to M **do**
C = SAMPLECURVEPOINTS(S, R, c, d, θ, ω)
E_i = CLASSIFIEREVALUATION(S, C_i)
Calculate error e with Equation (10)
Saves curve and it's evaluation
if TNR $< N$ or TPR $< P$ **then**
RESTARTCURVEPOSITION(c, d, θ, ω_0)
else
for j in 1 to $|C|$ **do**
Δc += $\frac{\partial}{\partial c}\left[sigmoid(x'_j)\right]$
Δd += $\frac{\partial}{\partial d}\left[sigmoid(y'_j)\right]$
$\Delta\omega$ += $\frac{\partial}{\partial \omega}\left[sigmoid(x'_j)\right] + \frac{\partial}{\partial \omega}\left[sigmoid(y'_j)\right]$
$\Delta\theta$ += $\frac{\partial}{\partial \theta}\left[sigmoid(x'_j)\right] + \frac{\partial}{\partial \theta}\left[sigmoid(y'_j)\right]$
end for
$grad_c = error \times \Delta c$
$grad_d = error \times \Delta d$
$grad_\omega = error \times \Delta\omega$
$grad_\theta = error \times \Delta\theta$
Update c, d, ω, θ
end if
end for
end function

5 EVALUATION

This section presents the evaluation of the SurfOpt algorithm, including the metrics analyzed, the experiments executed and the results obtained for this work.

5.1 Skewed Dataset Classification

One of the pervasive challenges of imbalanced classification is to evaluate models that create reliable predictions, and the classification tasks, under those circumstances, have at least three common pitfalls:

I **True Positive Rate (TPR) Detrimental**: the classifier inflates the number of misclassification of the majority class, raising the number of correct predictions of the negative class. This creates more false negatives than true negatives, but that is perceived as a positive effect on some metrics due to the fact they treat TPR and TNR as equally important;

II **Negative Predictive Value (NPV) detrimental**: this is the rate of correct negative predicted events over all negative predictions. When classes are severely skewed, the overall metrics can show reasonable values. The true negative rate may be high meaning most negative examples on the dataset are being correctly classified, but the classifier may be wrong most of the times it predicts the negative class with just a minor effect on the overall metric (i.e. accuracy), ignoring class imbalance;

III **True Negative Rate (TNR) Detrimental, or the safe bet**: this is the inverse situation of NPV Detrimental, where the classifier may display even a perfect metric of negative predictive value. Also, it can be always right when it predicts the minority class. Still, overall metrics may point a useful classifier, but it seldom predicts the minority class, leading to a very low true negative rate.

To diagnose such problems of imbalanced classifiers, we propose the joint observation of a set of three metrics: accuracy, negative predictive value, and true negative rate. Issues are evidenced in Figure 2.

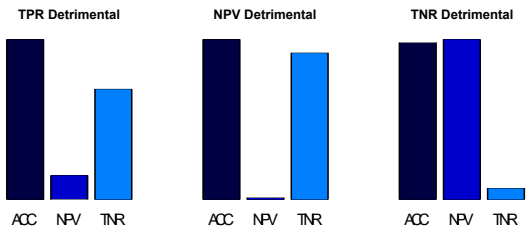

Figure 2: Challenges of imbalanced classification.

5.2 Experiments

We used random sampling to get 5.000 data points in Euclidean space from two normal distributions (A, B), with unitary standard deviations. The samples of distribution *A* consist of the majority class which contains the most data points (97.5%). The samples of distribution *B* belong to the minority class (2.5%). The distributions share all characteristics but the mean values of *x* and *y* coordinates, the barycenters of each class, have approximately unitary distance from each other. This means that the classes overlap in space, as seen in real datasets. We repeated this 500 times to make 500 synthetic datasets. For cross-validation, every dataset was split between a training (70%) and a validation (30%) set.

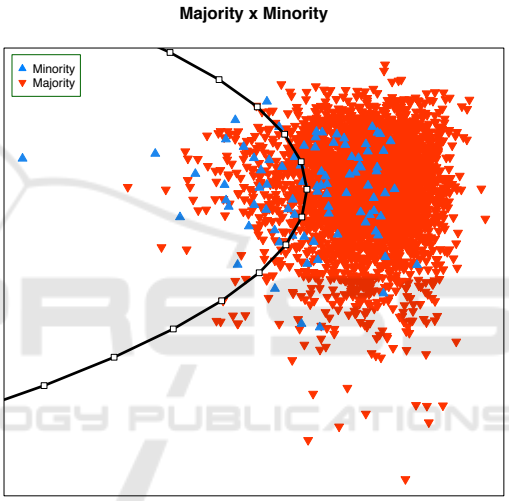

Figure 3: Two populations of data points overlapping one another, with an approximately unitary distance between their barycenters. 97.5% of points belongs to the majority class, 2.5% to the minority class. A curve decision surface learned with the SurfOpt Algorithm separates these classes.

5.3 Results

Four classification algorithms were run. The SurfOpt algorithm was programmed in the R programming language. For the SVM algorithm, we used the e1071 package (Dimitriadou et al., 2006). For Adabag and Adaboost, we used the package Adabag (Alfaro et al., 2013). All the parameters were found with grid-search. For SurfOpt, we used $\omega = 0$, *TNR thresold* $=$ 0.35, *TPR thresold* $= 0.10$, *max iterations* $= 2000$, *resolution* $= 20$. For the AMSmgrad optimization algorithm, we used $\alpha = 0.9$, $\beta = 0.9$, $\gamma = 0.999$. For the SVM, we used a radial kernel *cost* $= 1000$, $\gamma = 0.1$. For the experiments with Adaboost, we used *mfinal* $= 1$ and the Breiman learning coefficient. For Adabag, we used *mfinal* $= 1$, *max depth* $= 70$.

Table 1: Mean results for each algorithm.

Algorithm	ACC	NPV	TNR	MCC
SurfOpt	88.5%	11%	38.4%	0.156
SVM	97.5%	39.8%	0.7%	0.0409
Adaboost	97.3%	34.5%	7.4%	0.148
Adabag	97.3%	32.8%	8.5%	0.154

Table 1 shows the mean results for 500 synthetic datasets. Note that the SurfOpt algorithm attains comparable performance in terms of mean Matthews Correlation Coefficient (MCC) to the ensemble methods of Adabag and Adaboost. Support Vector Machines shows little classification value in terms of mean MCC. The low mean MCC metric for the SVM algorithm models can be explained by the fact it is unable to deal with trade-off regions even with the radial kernel, as it optimizes for distance to points. As expected, the skewness made the SVM models biased to the class with most points. Although SurfOpt algorithm shows a similar mean MCC to Adaboost and Adabag ensemble methods, in our algorithm, the mean True Negative Rate (TNR) is higher than the ensemble methods, meaning that it correctly predicted a higher ratio of the negative class data points. With the observation of those three metrics, it can be identified the SurfOpt algorithm as a True positive rate (TPR) detrimental algorithm, as it trades overall accuracy to attain a better classification of the minority class. SVM, Adaboost and Adabag classifiers are True Negative Rate (TNR) detrimental, as they only will predict the minority class if it is a very safe bet.

6 FUTURE WORK

We expect to investigate the properties of an ensemble approach to the SurfOpt algorithm, using sets of ununiform curves with complementary optimization criteria, to compensate for individual classifier's disadvantages. Other studies may also test SurfOpt performance regarding oversampling and undersampling approaches, explore the classifier properties with noisy data, and find a generalization to n-dimensional spaces.

7 CONCLUSION

In this paper, we have introduced SurfOpt algorithm. It brings together classification performance with the benefits of optimizing a classification surface. Opposed to common algorithms, SurfOpt algorithm can better classify the minority class of a binary set even under severe skewness. We devised a diagnostic to classification performance for imbalanced data based on three basic metrics, as an effort to study classification under skewness. The source code of this work is being made avaliable online (Da Silva, 2018).

ACKNOWLEDGMENT

The authors would like to thank the financial support from CAPES/Brazil and CNPq/Brazil, which was offered through the project Abys (401364/2014-3).

REFERENCES

Alfaro, E., Gamez, M., Garcia, N., et al. (2013). Adabag: An R Package For Classification with Boosting and Bagging. *Journal of Statistical Software*, 54(2):1–35.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The Quickhull Algorithm For Convex Hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the 5th annual workshop on Computational learning theory*, pages 144–152. ACM.

Boughorbel, S., Jarray, F., and El-Anbari, M. (2017). Optimal Classifier for Imbalanced Data Using Matthews Correlation Coefficient Metric. *PloS one*, 12(6):e0177678.

Carneiro, N., Figueira, G., and Costa, M. (2017). A Data Mining Based System For Credit-Card Fraud Detection In E-tail. *Decision Support Systems*, 95:91–101.

Da Silva, A. R. (2018). Surfopt. https://github.com/andreblumenau/SurfOpt.

Devroye, L. and Toussaint, G. T. (1981). A Note on Linear Expected Time Algorithms for Finding Convex Hulls. *Computing*, 26(4):361–366.

Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D., Weingessel, A., and Leisch, M. F. (2006). The e1071 Package. *Misc Functions of Department of Statistics (e1071), TU Wien*.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Guo, P., Liu, T., Zhang, Q., Wang, L., Xiao, J., Zhang, Q., Luo, G., Li, Z., He, J., Zhang, Y., et al. (2017). Developing a Dengue Forecast Model using Machine Learning: A Case Study in China. *PLoS neglected tropical diseases*, 11(10):e0005973.

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., and Bing, G. (2017). Learning from Class-imbalanced Data: Review of Methods and Applications. *Expert Systems with Applications*, 73:220–239.

Hofmann, M. (2006). Support Vector Machines—Kernels and the Kernel Trick. *Notes*, 26.

Kingma, D. P. and Ba, J. (2015). Adam: A Method For Stochastic Optimization. *3rd International Conference for Learning Representations*.

Lipton, Z. C., Elkan, C., and Naryanaswamy, B. (2014). Optimal Thresholding of Classifiers To Maximize F1 Measure. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–239. Springer.

Reddi, S. J., Kale, S., and Kumar, S. (2018). On the Convergence of Adam and Beyond. *6th International Conference on Learning Representations (ICLR)*.

Schapire, R. E. (2013). Explaining Adaboost. In *Empirical inference*, pages 37–52. Springer.

Syam, N. and Sharma, A. (2018). Waiting for a Sales Renaissance in the Fourth Industrial Revolution: Machine Learning and Artificial Intelligence in Sales Research and Practice. *Industrial Marketing Management*, 69:135–146.

Zhao, Y., Healy, B. C., Rotstein, D., Guttmann, C. R., Bakshi, R., Weiner, H. L., Brodley, C. E., and Chitnis, T. (2017). Exploration of Machine Learning Techniques In Predicting Multiple Sclerosis Disease Course. *PloS one*, 12(4):e0174866.