Predicting Style and Quality of *Vinho Verde* from its Physical Properties

Jonathan Calles
Dept. of Bioengineering, Stanford University Stanford, CA, USA
jecalles@stanford.edu

***Abstract*—Classifying and rating wine is generally done manually by highly trained experts. More broadly, predicting subjective quality from objective, physical measurements remains a difficult and well studied problem. In this work, I trained classifiers to identify the style of a given wine, as well as regressors to predict its quality, from 11 empirical physicochemical measurements. I was able to achieve a classification accuracy of 99.6% and to predict wine quality within 0.437 points out of 10 on average.**

I. Introduction

Wine is frequently considered to be delicious. It is also extremely popular. According to the International Organisation of Vine and Wine (OIV), the globe produced a staggering 263 mhL in 2019 (the hilariously presented and official units "megahectalitres" can be converted using 1 mhL = 10^8 L Given this enormous market, it is unsurprising that a significant amount of effort has been invested in determining what factors contribute to the subjective quality of wine [1]–[4].

Some of this effort attempts to move beyond our reliance on human experts, modelling wine preference directly from the physical and chemical properties of the wine. Some work has gone into making objective, quantitative "electric tongues", or multisensor potentiometric arrays used to quantify gustatory sensation [5], [6]. Other work has focused more on quantifying certain traits of wine. These are features such as "savoriness", or having notes of "dried fruit" or "pepper" [7], or perhaps "balance" or "flavor intensity" or "astringency" [8], all from objective physical measurements. However, the most difficult, and possibly most fraught goal is to model the overall quality of wine with no additional information other than what can be objectively measured.

In 2009, a group of researchers from Minho in the north of Portugal published a standard dataset in the field of subjective quality prediction and of wine characterization. They collated physical measurements from over 6000 examples of a local, young wine called *vinho verde* [9], as well as solicited a panel of three experts to judge the quality of these wines on a ten point scale. Their best performing model, an SVM regressor (see Methods) was able to predict quality within 0.46 points of the judged value for red wine, and within 0.45 points for white. However, they were only able to achieve this result by manually separating wines by style and training models on each style individually. A true "robo-sommelier" must be able to describe, classify, and rate!

In this project, I sought to build models that both classify wine by style and rate wines by quality. Using Random Forest models, I was able to successfully perform both tasks, outperforming the best models by Cortez and colleagues without separating the dataset by style.

II. Methods

All computation was done in Python 3.7 using the Anaconda distribution. I used Scikit-Learn for all data preprocessing and machine learning algorithms [10], Pandas [11] and Numpy [12] for manipulating data, and Seaborn [13] and Matplotlib [14] to visualize results.

A. Dataset and Features

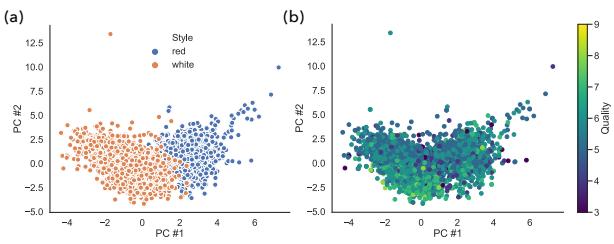

Fig. 1. Training data projected onto the first two principle components. Color represents wine style (left) and quality (right).

The dataset considered here includes 6497 examples of *vinho verde*. The dataset is unevenly split between two styles: 75% of examples are of white wines (4898) and 25% are of reds (1599). Each example is described by 11 physicochemical features, all continuously distributed: fixed acidity, volatile acidity, citric acid concentration, residual sugar content, chlorides concentrations, free sulfur dioxide content, total sulfur dioxide content, density, pH, sulphates concentrations, and alcohol content. Each example is also given two labels: its style (red or white), and its subjective rating (integers between 0 and 10) as determined by a panel of three sommeliers. I sampled 25% of the dataset at random (1625) to produce a test set, which I witheld until test time. The remaining data (4872) was randomly split into the training set (3654, 80%) and dev set (1218, 20%).

I preprocessed raw features by standardizing values to zero mean and unit variance. I also performed dimensionality reduction on the data using Principle Component Analysis (PCA) (Fig 1). Given n principle components, PCA maps

data from $\mathbb{R}^m$ to the subspace $S \subset \mathbb{R}^n$ that preserves the most variance in the data. The number of principle components used for each learning algorithm was set as a model-specific hyperparameter.

All model specific hyperparameters were determined with a grid search over the hyperparameter space using 5-fold cross validation on the training set. Models were visually checked by testing on the dev set. Once hyperparameters were set, all models were retrained on the combined train and dev sets before final testing.

B. Learning algorithms

I used the following algorithms in this work: Linear Regression, Logistic Regression, Naive Bayes, Gaussian Discriminant Analysis (GDA), Support Vector Machines (SVM), Random Forests, and AdaBoosting. I describe the working principles of each algorithm below.

Linear Regression seeks to fit a predictor for $y^{(i)}$ that is strictly linear in $x^{(i)}$ [15], which can be formalized as †he following optimization problem:

$$\min_{\theta} \frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)}) \right)^2, \tag{1}$$

where $h_\theta(x) = \theta^T x$ is the decision function and $\theta \in \mathbb{R}^d$ are the parameters of the model.

Linear Regression can be extended to binary classification problems through Logistic Regression [16]. In Logistic Regression, the decision function is altered slightly to map its outputs between 0 and 1: $h_\theta(x) = \frac{1}{1+\exp\{-\theta^T x\}}$.

Naive Bayes is a generative algorithm that can be used for binary and multiclass classification [17]. The Multivariate Gaussian Naive Bayes algorithm assumes that $x_j^{(i)}$ are conditionally independent given $y^{(i)}$. With d features per example and k classes, we have $kd + k - 1)$ parameters to estimate:

$$\begin{aligned} \phi_{j|y=l} &= p(x_j | y = l),\ l = 1, \ldots, k, \\ \phi_{y=l} &= p(y = l),\ l = 1, \ldots, k-1, \end{aligned} \tag{2}$$

which can be estimated by maximizing the likelihood (or corresponding log likelihood) of these parameters:

$$\begin{aligned} \mathcal{L}(\phi) &= \prod_{i=1}^{n} p(x^{(i)}, y^{(i)}; \phi_y, \phi_{j|y}) \\ &= \prod_{i=1}^{n} \frac{\left(\prod_{j=1}^{d} p(x_j^{(i)} | y^{(i)}, \phi_{j|y}) \right) p(y^{(i)} | \phi_y)}{p(x^{(i)} | \phi_y, \phi_{j|y})}. \end{aligned} \tag{3}$$

Because the quality labels are discrete, it is possible to use Naive Bayes for quality prediction as well.

GDA is another generative classification algorithm [18]. If there are k classes, GDA assumes that each example $x^{(i)}$ is drawn from one of k Gaussian distributions depending on $y^{(i)}$, that is $x^{(i)}|y = l \sim \mathcal{N}(\mu_l, \Sigma)$. GDA also assumes that the data is homoscedastic, meaning all classes share the same covariance matrix. GDA therefore fits $kd + d^2 + k - 1$ parameters:

$$\begin{aligned} \mu_i &\in \mathbb{R}^d \\ \Sigma &\in \mathbb{R}^{d \times d} \\ \phi_{y=l} &= p(y = l),\ l = 1, \ldots, k-1, \end{aligned} \tag{4}$$

which are then fit by maximizing the likelihood $\mathcal{L}(\mu, \Sigma, \phi_y) = \prod_{i=1}^{n} p(x^{(i)}|y^{(i)}, \mu, \Sigma) p(y^{(i)}|\phi_y)$.

SVMs are powerful algorithms that utilize the kernel trick to implicitly map data to a higher dimensional space, where nonlinear decision boundaries in the original space become linear in the implicit space [19]. SVMs used for classification seek to solve the following primal optimization problem:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^t w + C \sum_{i=1}^{n} \xi \\ \text{s.t.} \quad & y_i \cdot \hat{y}_i \leq \xi \\ & \xi \geq 0,\ i = 1, \ldots, n, \end{aligned} \tag{5}$$

where $\hat{y}_i = w^T \phi(x_i) + b$. Often, the Lagrangian Dual to this problem is solved instead for computational simplicity.

SVMs can be extended to regression problems by altering the optimization problem used for SVM classification [20]. We introduce the idea of a hypertube of radius ϵ centered along the decision boundary, inside of which points are given no penalty. We then use examples that lie on the surface of this hypertube as our support vectors:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i^{low} + \xi_i^{up} \\ \text{s.t.} \quad & y_i - \hat{y}_i \leq \epsilon + \xi_i^{up} \\ & \hat{y}_i - y_i \leq \epsilon + \xi_i^{low} \\ & \xi_i^{low},\ \xi_i^{up} \geq 0, /i = 1, \ldots, n, \end{aligned} \tag{6}$$

where again $\hat{y}_i = w^T \phi(x_i) + b$. Again, often the Lagrangian Dual of this problem is solved instead of the primal problem.

The kernel used for the SVM algorithm is extremely important for model performance. In this work, I used a radial basis function, or Gaussian Kernel: $K(x_1, x_2) = \exp\{\gamma ||x_1 - x_2||_2^2\}$, where γ describes the length scale of the kernel.

Random Forests are an ensemble method for classification and regression that can use any model as a base estimator, although frequently uses decision trees [21]. The idea is to train many weak learners on different subsets of the data, then pool their estimates. This creates a model that is more accurate than any of its constituent models. Random forests avoid overfitting with two tricks. First, each model is trained on a subset of the training data generated by random sampling with replacement ("bootstrapping"). Second, each model only considers a randomly selected subset of possible features instead of training on the whole feature space. See Algorithm 1 in Appendix for pseudocode.

AdaBoost is another ensemble method, also frequently used on decision trees, that can be applied to classification or regression [22]. At each step in training, a new weak estimator

is trained on weighted data, where the importance of each is example is set by how difficult the ensemble finds classifying that point. The ensemble model then returns a prediction that is a weighted average of the individual predictions of each constitutive model, weighted by the accuracy of each model on the training set. See Algorithm 2 in Appendix for pseudocode.

III. Results

A. Classifying Wine by Style

I use the following metrics to assess the performance of my wine style classifiers: accuracy, balanced accuracy, F score, Cohen's Kappa, and the area under the curve (AUC) of the Receiver Operator Characteristic (ROC) curve. Accuracy is simply defined as the proportion of correctly classified examples. Balanced accuracy weighs correct and incorrect classifications by the proportion of the dataset corresponding to each class, which is more indicative than accuracy in unevenly distributed datasets.

F score represents a summary statistic giving the harmonic mean of the precision and the recall of a model [23]. Cohen's Kappa is a metric that captures how much a predictor's agreement with a variable is due to chance, given by $\kappa = \frac{p_o - p_e}{1 - p_e}$, where p_0 is the accuracy of the model and p_e is the likelihood of guessing correctly at random [24]. The ROC curve describes the false positive rate of a model versus its accuracy [25] (Fig 2a). The ROC AUC summarizes the curve in one metric.

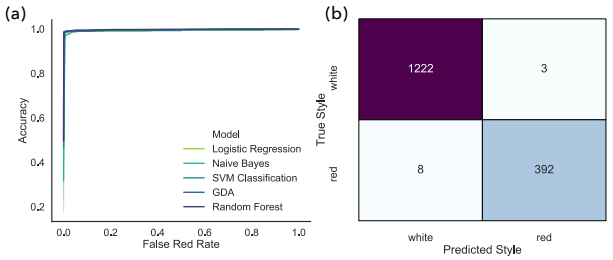

Fig. 2. ROC curves (left) for style classifiers. Confusion matrix (right) for Random Forest style classifier.

All tested classifiers performed very well (Fig. 2, Table 1). Most models used all features in the dataset. The exception was the Naive Bayes model, which performed optimally with only 2 principle components.

One benefit of using Random Forests is that they easily allow one to estimate feature importance using feature permutation [21]. Since each model in the ensemble is trained only on a subset of the data, they can be cross validated with the remaining data (called "out of bag" scoring, or OOB). By randomly scrambling the values of a particular feature and retraining the model, we can assess how much the OOB score changes compared to baseline and use this as a measurement of feature importance (Fig. 4). Using this method, we can see that the strongest predictors of wine style are total sulfur dioxide content and residual sugar content.

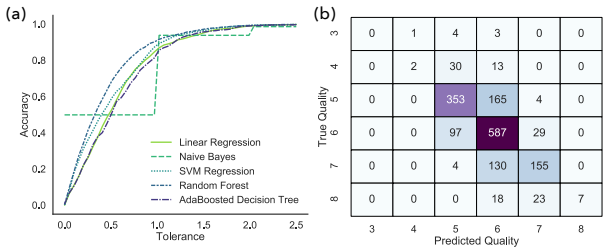

Fig. 3. REC curves (left) for quality regressors. Confusion matrix (right) for Random Forest quality regressor.

B. Predicting Wine Quality

Predicting wine quality is a regression task, however the true labels are discrete. This means that some multiclass classification metrics can be used to assess these models if we allow continuous prediction to be "correct" within some tolerance value. Accuracy and balanced accuracy were used as defined above, with a tolerance of 0.5 (round continuous predictions to the nearest whole number). Mean absolute error (MAE) represents the average absolute deviation of a model's predictions from the true value. R^2 represents the average squared magnitude over all the residuals of the model ($R^2 = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$).

The Regression Error Characteristic (REC) curve is meant to generalize the ROC curve to regression tasks. It analyzes the accuracy of a model as we adjust the tolerance within which we call a prediction accurate [26], (Fig. 3a). The REC AUC similarly summarizes the REC curve.

To begin, the REC curve for the Naive Bayes model looks like a step function. This makes sense: the Naive Bayes algorithm will only output integer predictions, thus its accuracy only goes up when the tolerance is increased by an integer amount. Worth noting, Naive Bayes is the best performing model with a tolerance of 1. This case is not interesting, because this corresponds to the case where a good wine (6) is classified somewhere between mediocre (5) and great (7).

AdaBoosted Decision Trees performed very poorly (Table 2). This may be due to the prevalence of extreme outliers in the data, which AdaBoosting tends to overfit to. This also explains why SVMs and Random Forests perform so well on this dataset, given that both of these models are naturally robust to outliers. In fact, the Random Forest with 100 estimators was able to outperform the best published models from Cortez *et al.* 2009 with an MAE of 0.44 without separating the data into styles and training separate models on each style.

Analyzing feature importances for the Random Forest quality regressor shows that the sommeliers who assessed these wines care quite a lot about alcohol content, above and beyond any other feature. These feature importances also disagree with those from Cortez and colleagues for sulphates (which they hypothesized correlated with yeast health during fermentation), and for citric acid and residual sugar (which they hypothesized was some measure of how vinegary the wine is). My model seemed to care more about total sulfur

TABLE I
WINE STYLE CLASSIFIERS PERFORMANCES ON TEST SET.

	Accuracy	Balanced Acc.	F Score	Cohen's Kappa	ROC AUC	# PC Used
Logistic Regression	99.38%	99.00%	0.9874	0.9668	0.9942	11
Naive Bayes	98.58%	97.55%	0.9708	0.9237	0.9928	2
SVM	99.57%	99.13%	0.9912	0.9768	0.9971	11
GDA	99.63%	99.25%	0.9924	0.9801	0.9956	11
Random Forest	99.32%	98.88%	0.9862	0.9635	0.9977	11

dioxide content, which is known to affect microbial growth [27].

IV. DISCUSSION

I was able to train several classifiers that accurately separated wines by style. However, this task is relatively trivial. Even visually, one can draw a separating hyperplane in a two dimensional subspace that reasonably classifies the data (Fig 1). More interestingly, I was able to train several regressors that reasonably predicted the quality of a given wine, improving on the models from the original paper.

Working on this project has lead me to believe that we will not converge on a good assessment of general wine quality. I believe that the subjective experience of even trained sommeliers has a high enough variance that we will only be able to determine general principles of quality rather than objective predictors. Even feature importances to determining wine quality vary qualitatively across studies [2], [9] (Fig. 4).

I believe that the most interesting direction for this field of research to pursue is the prediction of "qualities" of a given wine, not the "quality" of such wines. As mentioned previously in the Introduction, there is a body of research on this topic suggesting its feasibility [7], [8]. More importantly, I believe this endeavor is more theoretically sound. It is reasonable to suspect that what makes a wine "jammy" may be the total concentration of a family of organic molecules, or that the "persistency" of the wine's flavor may result from the relative ratio of volatile flavorants to water soluble ones, or more broadly that any of these subjective qualities has some underlying physics and chemistry to them. ML aided qualitative labelling may therefore inform consumer preferances, or suggest new wines based on previous preferences.

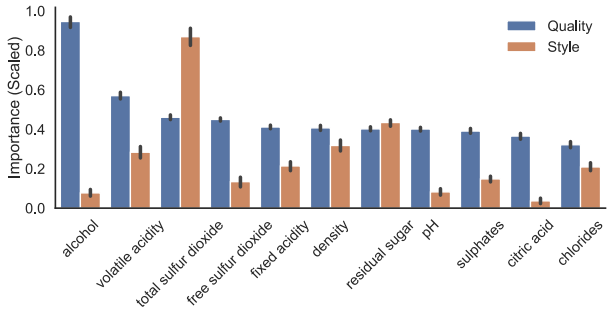

Fig. 4. Feature importances for Random Forest models trained to classify wine style (orange) or predict wine quality (blue). Importances were scaled within task for ease of visualization.

If I could do anything to improve on this project, I would seek to expand the feature space and assign a new class of labels for each wine corresponding to wine "qualities". In terms of new features, I would seek to add spectroscopic measurements to this dataset, which are inexpensive and simple to perform. These could include color (measured by peak absorbance, or by absorbances at some fixed wavelengths), opacity, protein content, refractive index, and more. In an ideal world, I would extend this analysis further with label free Mass Spectrometry, studying the entire chemical composition of the wine. This is expensive, time consuming, and overkill, but would most definitely give a more complete understanding of the chemistry of each wine. In terms of new labels, I would hire a set of sommeliers to rank each wine along a set of subjective axes, each describing from 0 to 10 how much the wine can be said to have that quality (e.g., "this wine is a 6 on dryness"). I would then use the expanded feature space to try and model these new labels.

REFERENCES

[1] J. Aleixandre-Tudó, I. Alvarez, M. García, V. Lizama, and J. Aleixandre, "Application of multivariate regression methods to predict sensory quality of red wines," *Czech Journal of Food Sciences*, vol. 33, pp. 217–227, jun 2016.

[2] Y. Gupta, "Selection of important features and predicting wine quality using machine learning techniques," *Procedia Computer Science*, vol. 125, pp. 305–312, jan 2018.

[3] S. Tempere, S. Pérès, A. F. Espinoza, P. Darriet, E. Giraud-Héraud, and A. Pons, "Consumer preferences for different red wine styles and repeated exposure effects," *Food Quality and Preference*, vol. 73, pp. 110–116, apr 2019.

[4] X. Chu, Y. Li, Y. Xie, D. Tian, and W. Mu, "Regional difference analyzing and prediction model building for Chinese wine consumers' sensory preference," *British Food Journal*, vol. ahead-of-p, oct 2019.

[5] A. Legin, A. Rudnitskaya, L. Lvova, Y. Vlasov, C. Di Natale, and A. D'Amico, "Evaluation of Italian wine by the electronic tongue: recognition, quantitative analysis and correlation with human sensory perception," *Analytica Chimica Acta*, vol. 484, pp. 33–44, may 2003.

[6] D. Kirsanov, O. Mednova, V. Vietoris, P. A. Kilmartin, and A. Legin, "Towards reliable estimation of an "electronic tongue" predictive ability from PLS regression models in wine analysis," *Talanta*, vol. 90, pp. 109–116, feb 2012.

[7] J. Niimi, O. Tomic, T. Næs, D. W. Jeffery, S. E. Bastian, and P. K. Boss, "Application of sequential and orthogonalised-partial least squares (SO-PLS) regression to predict sensory properties of Cabernet Sauvignon wines from grape chemical composition," *Food Chemistry*, vol. 256, pp. 195–202, aug 2018.

[8] J. A. Cayuela, B. Puertas, and E. Cantos-Villar, "Assessing wine sensory attributes using Vis/NIR," *European Food Research and Technology*, vol. 243, pp. 941–953, jun 2017.

TABLE II
WINE QUALITY REGRESSORS PERFORMANCES ON TEST SET

	Accuracy	Balanced Acc.	MAE	R^2	REC AUC	# PC Used
Linear Regression	52.43%	24.38%	0.5688	0.2878	1.933	11
Naive Bayes	49.85%	28.16%	0.5797	-0.0311	1.931	9
SVM	58.22%	29.08%	0.5077	0.4004	1.994	11
Random Forest	67.94%	37.10%	0.4374	0.5202	2.063	11
AdaBoost	49.91%	21.99%	0.5876	0.2572	1.913	10

[9] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, pp. 547–553, nov 2009.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in {P}ython," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference* (S. van der Walt and J. Millman, eds.), pp. 51–56, 2010.

[12] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in Science & Engineering*, vol. 13, pp. 22–30, mar 2011.

[13] M. Waskom, O. Botvinnik, P. Hobson, J. B. Cole, Y. Halchenko, S. Hoyer, A. Miles, T. Augspurger, T. Yarkoni, T. Megies, L. P. Coelho, D. Wehner, Cynddl, E. Ziegler, Diego0020, Y. V. Zaytsev, T. Hoppe, S. Seabold, P. Cloud, M. Koskinen, K. Meyer, A. Qalieh, and D. Allan, "seaborn: v0.5.0 (November 2014)," nov 2014.

[14] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[15] A. M. Legendre, *Nouvelles méthodes pour la détermination des orbites des comètes*. Nineteenth Century Collections Online (NCCO): Science, Technology, and Medicine: 1780-1925, F. Didot, 1805.

[16] D. R. Cox, "The Regression Analysis of Binary Sequences," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 20, no. 2, pp. 215–242, 1958.

[17] D. J. Hand and K. Yu, "Idiot's Bayes: Not So Stupid after All?," *International Statistical Review / Revue Internationale de Statistique*, vol. 69, no. 3, pp. 385–398, 2001.

[18] R. A. FISHER, "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[19] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, sep 1995.

[20] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," 2004.

[21] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, oct 2001.

[22] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, pp. 119–139, aug 1997.

[23] C. Van Rijsbergen, "Foundation of Evaluation," *Journal of Documentation*, vol. 30, pp. 365–373, jan 1974.

[24] J. Cohen, "A Coefficient of Agreement for Nominal Scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[25] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, pp. 861–874, jun 2006.

[26] J. Bi and K. P. Bennett, "Regression Error Characteristic Curves," *ICML 2013*, 2003.

[27] M. I. S. U. Moroney, "Total Sulfur Dioxide - Why it Matters, Too!," 2018.

V. APPENDIX

Algorithm 1 Random Forest

procedure RF.FIT(X, y, BaseModel, $n_{models}, n_{features}$)

  for $i = 1$ to n_{models} **do**

    $X_i, y_i \leftarrow$ sample X, y with replacement

    $\phi_i \leftarrow n_{features}$ from X_i at random

    $M_i \leftarrow$ BaseModel.FIT($\phi_i(X_i), y_i$)

  end for

  return $\{M_i, i = 1, \ldots, n_{models}\}$

end procedure

procedure RF.PREDICT(X, Models)

  for M_i in Models **do**

    $\hat{y}_i = M_i$.PREDICT(X)

  end for

  $\hat{y} =$ mean($\hat{y}_i$)

  return $\hat{y}$

end procedure

Algorithm 2 AdaBoost

procedure ADA.FIT(X, y, BaseModel, n_{models})
    initialize example weights $w^{(1)} = \frac{1}{n}$
    for $i = 1$ to n_{models} **do**
        $M^{(i)} \leftarrow$ BaseModel.FIT($X, y, w^{(i)}$)
        $\hat{y}^{(i)} \leftarrow M^{(i)}$.PREDICT($X, y$)
        $\epsilon^{(i)} \leftarrow$ ERROR($y, \hat{y}^{(i)}$)
        $\alpha^{(i)} \leftarrow \frac{1}{2} \log \frac{1-\epsilon_i}{\epsilon_i}$
        update example weights $w^{(i)}$ based on M_i error
    end for

    Models $\leftarrow \{(M^{(i)}, \alpha^{(i)}), i = 1, \ldots, n_{models}\}$
    return Models
end procedure

procedure ADA.PREDICT(X, Models)

    for $M^{(i)}, \alpha^{(i)}$ in Models **do**
        $\hat{y}_i = M^{(i)}$.PREDICT(X)
        $\tilde{y}_i \leftarrow \alpha^{(i)} y_i$ ▷ weigh predictions by model weight
    end for

    $\hat{y} =$ mean($\tilde{y}_i$)

    return $\hat{y}$
end procedure