# Comparing Musical and Lyrical Features to Build the Optimal Song Release Year Regression Model

**Michael Kronovet, Tanishq Kancharla**

## 1 Introduction

Existing software like the popular app Shazam is able to take audio from a song and find the song itself by comparing it against songs in a central database. However there are major problems with this software which we hope to alleviate with our machine learning algorithm. Shazam is able to identify the precise song by matching the sounds to songs in their database, so they need an incredibly large database to store all of these different songs, which is no doubt expensive and difficult to maintain and update. In addition, if the song is not within Shazam's database, then they are unable to identify the song, so they may struggle identifying more obscure songs. We propose building a model that does not require a massive amount of data storage and can predict the release year of any song just from the song's musical features and lyrics. We get our data for the musical features from a 10,000 song subset of the Million Song Dataset, and we get our lyrical features from the Musixmatch API.

## 2 Background

For our midway report, we assessed the accuracy/error of various base classifiers and regressors for predicting song release year. We extracted and processed the musical features and fit many base classifiers just using these features which were trained to minimize the MSE. We tested both regression and classification models on the data, where the regression models attempted to classify the exact year a song was released while the classifiers were built to predict the decade the song was released. We found that from the various models that we fit, linear lasso model performed the best of regression, and the random forest classifier performed the best of our classification models. The lasso regression model had a MSE of 122.999 whereas a naive regressor always predicting 1997 had a MSE of 142.830 (this particular year minimizes the MSE). After further investigation it seemed that the lasso model predicts about every release year between 1970 and 2005. This could be due to the database containing mostly songs from this period, but this is definitely not ideal for our model. The random forest classifier also seemed to predict mostly decades between 1980 and 2000. As a result, the classifier only does slightly better than if you were to naively guess that every song was released in the 2000s.

For the final project, we decided to treat the year prediction solely as a regression problem rather than a classification problem. To see why, an example song released in 1980 is likely of similar style as a song released in 1979. Therefore, it makes more sense to weight the error of a prediction of 1979 for a song released in 1980 much lower than a prediction of 2010 (but in classification of release decade, the errors would be equal).

## 3 Related Work

We often referred to the Million Song Dataset website for information about data features, extraction, and other resources.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

We also used the paper, "Music Genre Classification with the Million Song Dataset" by Dawen Liang, Haijie Gu, and Brendan O'Connor [1] as a reference for working with both lyrical/text features and music features. In this paper, the students fit a model from the Million Song Dataset using both lyrics and musical features to classify genre. In order to combine this data, they first fit a logistic regression model to just the lyrics data in order to get the probabilities that each song was a specific genre and then input those probabilities as a feature in the dataset to replace the lyrics information. This technique for combining lyrics and music information greatly restricts the data and the models that could be fit for several reasons (one being that it assumes no relationship between the individual lyrics and the other musical features), so we decided to include the actual lyrics data when fitting our models.

The paper "Lyric Text Mining in Music Mood Classification" by X. Hu, J. Downie, and A. Ehmann [2] compares and contrasts music mood classification models trained on audio features vs. lyric text features. Their support vector machine model gave us valuable information that we incorporated into our own models. For example, they performed their own cleaning on the bag of words data to make sure each song had associated words that were representative of the song. We utilized some of their cleaning methods for our own data by removing stop words such as "the", "of", "a", etc.

# 4 Methods

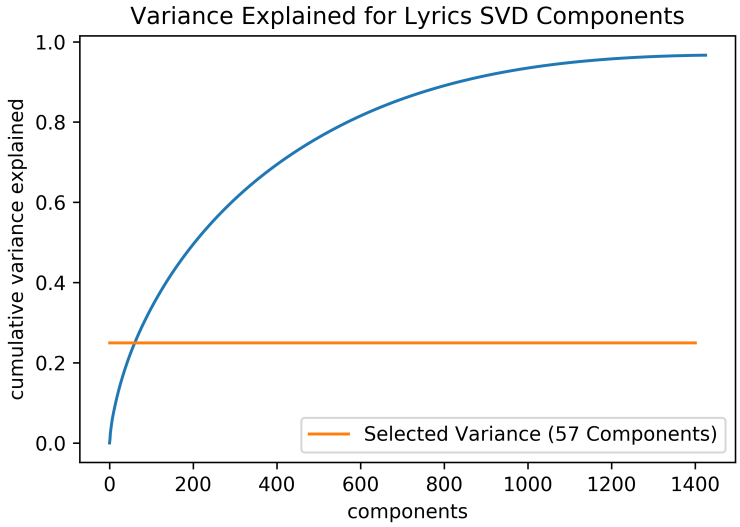
## 4.1 Data Processing and Retrieval

The 10,000 song subset of the Million Song Dataset (MSD) stored the data for each song in an HDF5 file which we scripted through to convert to a pandas Dataframe. We removed every song without its year recorded, dropped features that were unrelated to the musical qualities of the songs, removed confidence values for song features, and took the weighted average of the 12 dimensional timbre and pitch vectors for each segment in the song. The transformed pitch values ended up being highly correlated with each of the corresponding transformed timbre values, but both seemed to provide useful information so we retained them. In addition, when a song did not have information for a certain variable, we input the mean of that feature so that we did not throw out too much data.

The Million Song Dataset came with a text file that mapped every MSD ID to a corresponding Musixmatch ID for each song. We scripted through this text file and listed each Musixmatch ID for each song. We then created several accounts with the Musixmatch website and built a script to iterate through our songs and call the Musixmatch API, which stored the lyrics for 2536 of our songs. Unfortunately, we only had a free trial of the API, which only allowed us to scrape the first 30 percent of the lyrics from each song.

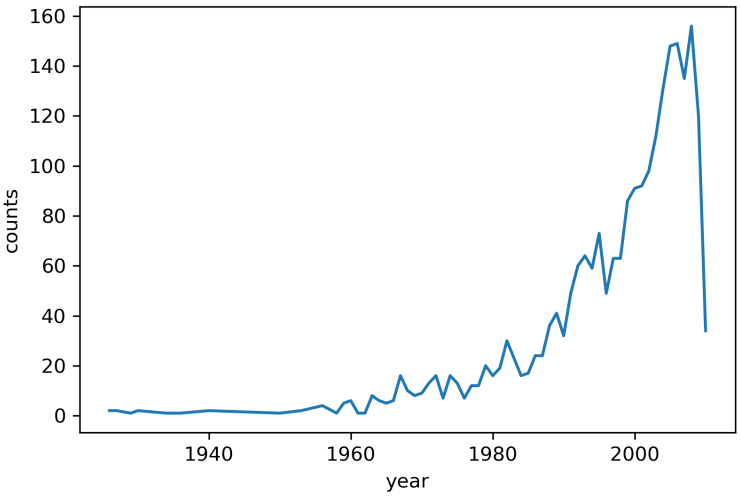
Once we had the lyrics in text form for each song, we transformed the lyrics into a bag of words representation where we removed stop words, words that occurred too many times, and words that occurred too rarely. Next we performed a tf-idf transformation on the vectorized lyrics data that would scale the words that occur less frequently to have more importance than more common words. After applying this transformation, we extracted the singular value decomposition of the lyrics and retained 58 components which explained 25 percent of the variance in the transformed lyrics. Retaining this amount of the variance allowed us to retain a considerable amount of information from the lyrics data while not diluting our musical features. This is shown in the first graph on the next page.

Our final data set consisted of 2536 samples with 31 musical features and 58 lyrical features. The musical features were: the song duration (1), when the end of fade-in for the song occurs (2), key (3), loudness (4), mode (5), tempo (6), time signature (7), 12 values representing the timbre throughout the song (8-19), and 12 values representing the pitch throughout the song (19-31). The lyrical features were just the top 58 principal components after performing the bag of words, tf-idf, and singular value decomposition transformations on the actual song lyrics.




### 4.2 Data Skew and WAMSE

Our final data set of 2536 samples ended up being very skewed, with the majority of songs being released in 2000 or above.




Distribution of Data

In order to account for this skew, we decided to define our own weight-adjusted metric to judge a model that lets us see how well the model does at predicting values outside 2000. Therefore, an ideal model would not only have a low mean squared error, but also a low weight adjusted mean squared error. We define this weight-adjusted mean squared error (WAMSE) below.

$$\text{WAMSE} = \frac{1}{\sum_j^n w_j} \sum_i^n w_i (y_i - m(x_i))^2, \text{ where } w_i \propto \frac{1}{\text{inclusion probability of } y_i}$$

This would allow us to get more varied predictions and avoid having our regressor stuck predicting years around 2000, since values that show up much less frequently are weighted more heavily.

### 4.3 Initial Model Fitting

We tested KNN regression, linear ridge regression, linear lasso regression, SVM regression, decision tree regression, and AdaBoost regression and found that an AdaBoost regressor consistently performed the best. After more testing, we saw that using an exponential loss function did the best job of minimizing the WAMSE, while retaining a low MSE.

### 4.4 The AdaBoost Regressor

Our AdaBoost regressor is based off of AdaBoost.R2 described in Drucker's 1997 paper [5], and implemented in sklearn. The following procedure is employed for the regressor using exponential loss.

Initially, each training sample is assigned a weight $w_i = 1, i = 1, ..., N$.

Repeat the following while the average loss $\overline{L}$ defined below is less than 0.5.

1. The probability that training sample $i$ is in the training set is $p_i = \frac{w_i}{\sum w_i}$ where the summation is over all members of the training set. Pick $N_1$ samples (with replacement) to form the training set.
2. Construct a regression machine $t$ from that training set (our regression machine is a decision tree regressor). Each machine makes a hypothesis $h_t : x \to y$. Then pass every member of the training set through this machine to obtain a prediction $y_i^{(p)}(x_i), i = 1, ..., N_1$.
3. Calculate a loss for each training sample, where we use exponential loss

$$D = \sup |y_i^{(p)}(x_i) - y_i|, i = 1, ..., N_1$$

$$L_i = 1 - \exp[\frac{-|y_i^{(p)}(x_i) - y_i|}{D}]$$

4. Calculate an average loss $\overline{L} = \sum_{i=1}^{N_1} L_i p_i$
5. Form $\beta = \frac{\overline{L}}{1-\overline{L}}$. $\beta$ is a measure of confidence of in the predictor. Low $\beta$ means high confidence in the prediction.
6. Update the weights $w_i \to w_i \cdot \beta^{1-L_i}$. The smaller the loss, the more the weight is reduced making the probability smaller that this pattern will be picked as a member of the training set for the next machine in the ensemble.
7. For a particular input $x_i$, each of the $T$ machines makes a prediction $h_t, t = 1, ..., T$. Obtain the cumulative prediction $h_f$ using the T predictors:

$$h_f = \inf\{y \in Y : \sum_{t:h_t \leq y} \log(\frac{1}{\beta_t})\}$$

   This is the weighted median. Equivalently, each machine $h_t$ has a prediction $y_i^{(t)}$ on the $i$th pattern and an associated $\beta_t$. For pattern $i$ the predictions are relabeled such that for the pattern $i$, we have:

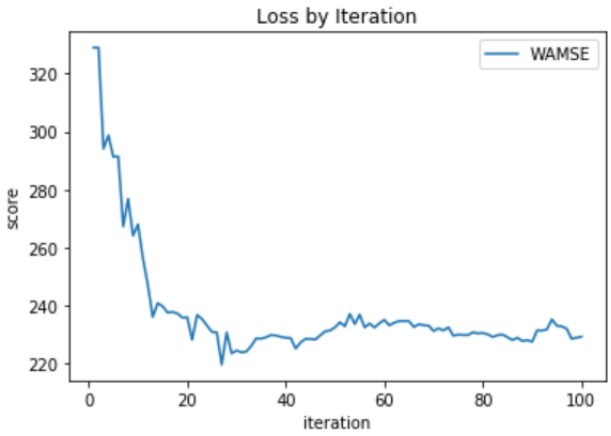
$$y_i^{(1)} \leq y_i^{(2)} \leq ... \leq y_i^{(t)}$$

   (retain the association of the $\beta_t$ with it's $y_i^{(t)}$). Then sum the $\log(\frac{1}{\beta_t})$ until we reach the smallest $t$ so that the inequality is satisfied. The prediction from that machine $t$ we take as the ensemble prediction. If the $\beta_t$ were all equal, this would be the median.

Intuitively, the effect of varying the weight $w_i$ to give more emphasis to "difficult" examples means that each subsequent machine has a disproportionately harder set of examples to

train on. Thus, the average loss tends to increase as we iterate through the algorithm and ultimately the bound on $\overline{L}$ is not satisfied and the algorithm terminates.

Training over this AdaBoost regressor gives us the following loss progression over iterations for WAMSE:

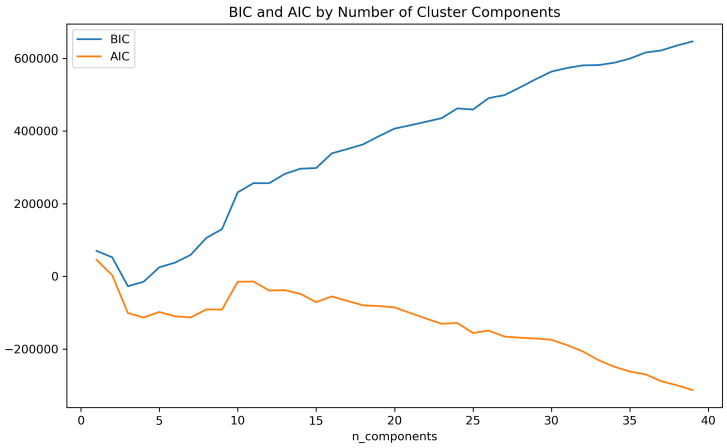



As we can see, the WAMSE is generally decreasing as we train the AdaBoost regressor and add more base regressors to the ensemble. We used this training loss information to aid in the hyperparameter tuning of our AdaBoost model.

### 4.5 Lyrics Feature Testing

In order to test whether the lyrics data provided useful features for our predictions, we fit our AdaBoost model to use the principal components from the lyrics. We also fit an AdaBoost model with the entire data set consisting of both the musical and lyrical factors.

In addition, we ran a Gaussian mixture model on the entire data set (including both musical and lyrical features) and predicted the cluster for each point in our data which we input as a feature in our data set. We chose 9 clusters as the optimal cluster number for the Gaussian mixture model in order to minimize the BIC and maximize the AIC criterion.




We then fit a model using only the musical features and the predicted GMM clusters as an additional feature to see if the interaction between lyrics and musical feature would provide useful information for predicting the song year.

### 4.6 Tuning Hyperparameters

After we had settled on the optimal regressor (AdaBoost with exponential loss), we decided to tune our model to find the optimal hyperparameters for the AdaBoost model. We segmented our data into a training, testing, and validation set and fit a variety of AdaBoost models with different hyperparameters to the training set, evaluated the models using the testing set, and then reported the actual accuracy of the model on the validation set. We varied the base regressor models (trees, different forms of linear regression, etc.), the number of base regressors, and the learning rate. We tuned the models to minimize both MSE and the WAMSE to find two optimal models (one for each).

The optimal model tuned to minimize MSE was an AdaBoost regressor with 75 decision trees with a maximum depth of 6 for each, exponential loss, and a learning rate of 1.

The optimal model tuned to minimize WAMSE was an AdaBoost regressor with 50 decision trees with a maximum depth of 3 for each, exponential loss, and a learning rate of 1.

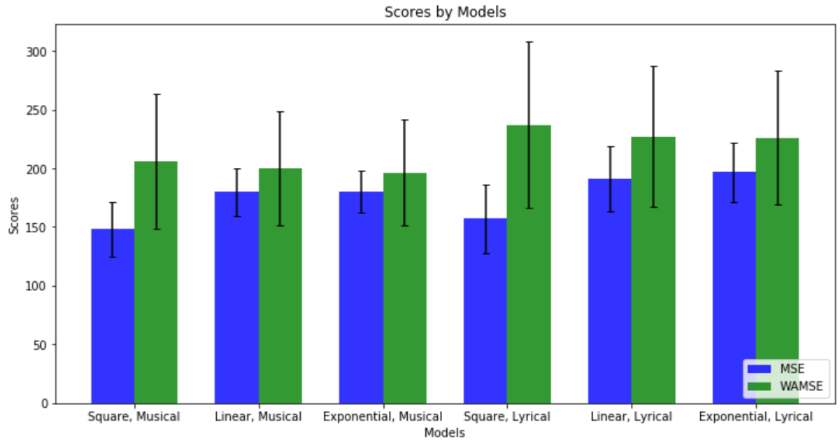
## 5 Results

### 5.1 Testing Models

First, we tested many different models to see which model would do best in this setting, using a 4:1 training to testing ratio in our dataset:

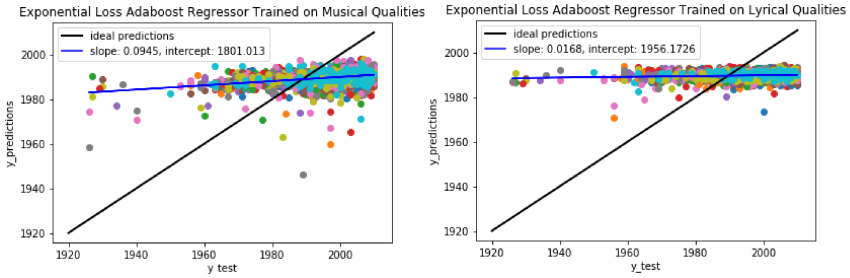
| Model | Train MSE | Test MSE | Test WAMSE |
|---|---|---|---|
| KNN Regression | 118.18 | 136.818 | 460.19 |
| Linear Ridge Regression | 112.31 | 119.788 | 392.027 |
| Linear Lasso Regression | 112.673 | 120.022 | 398.161 |
| SVM Regression | 138.33 | 164.436 | 721.734 |
| Decision Tree Regressor | 0 | 258.524 | 556.575 |
| AdaBoost Regressor | 153.052 | 185.64 | 339.606 |

Our best model, as defined by our metrics described above, was an AdaBoost regressor with exponential loss trained solely on musical features. The following are results for all six (linear, square, exponential tested with musical or lyrical features) models tested on the same 20-fold cross-validation sets of musical features:




As is clear, the AdaBoost regressors trained only on musical features outperform the regressors trained on lyrical features in terms of both MSE and WAMSE. We also note that the AdaBoost regressor with exponential loss seems to do best overall. To get a better sense of how our error is distributed, we created plots of y predictions vs y trues. Here we show such plots for exponential loss models trained on musical features vs. lyrical features. The black line is our ideal predictions (y true = y prediction), and the blue line represents our

line of best fit for our predictions. The colors represent different folds in our training (and are thus unimportant).




As you can see, neither are actually too close to our ideal line. However, in terms of slope, the musical features model does about 10 times better than the lyrical features model. The line of best fit for the lyrical models is nearly horizontal and intersects the ideal line around 1997, which is exactly how the naive regressor behaves. This tells us that the model is minimizing MSE solely by predicting 1997, nearly independent of what the lyrical features actually are. Therefore, our analysis shows that the lyrical features are nearly independent of the song release year.

In order to test if there is useful information regarding the interaction between lyrical and musical features, we run a hypothesis test on the MSEs from the model with just musical features and the model trained on musical featues and the GMM clustering formed from the musical and lyrical features. We find that, assuming the true MSEs of the two models are the same, there is a 68.1 percent chance of getting a difference in the MSEs as high as we observed. Therefore, we can conclude that the GMM clustering feature is not really helping the model's predictive ability for song release year.

From our analysis, we can firmly state that the lyrical features were not helpful in any form for predicting song release year, and they seemed to have no relation to the musical features as far as for their effect on song release year.

### 5.2 Model Tuning

After this analysis, we tuned the hyperparameters of our AdaBoost regressor to minimize MSE and WAMSE as described earlier. The results of our best models compared to the naive regressor are:

| Model | Validation Set MSE | Validation Set WAMSE |
|---|---|---|
| Naive Regressor | 142.83 | 1538.4 |
| AdaBoost Regressor tuned to MSE | 130.167 | 410.8 |
| AdaBoost Regressor tuned to WAMSE | 169.613 | 346.601 |

## 6 Discussion and Analysis

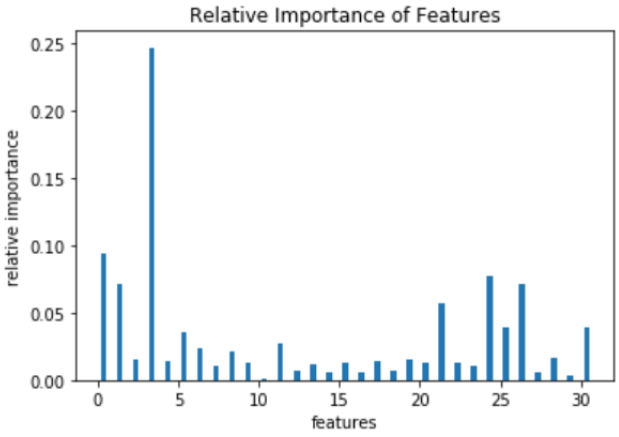
From our analysis of the data we can conclude that lyrics do not seem to be an accurate predictor of the year a song was released, and that music features are slightly better but still not very useful. Our models performed the best when only being trained on the musical features, and the interaction term between lyrics and music did not help very much. We also saw no trend in the ability of lyrics or the interaction between lyrics and music to predict the year a song was released. Although we were limited in only able to attain the first 30 percent of the lyrics from our song, this still seems like a large enough amount to be able to deduce something from the lyrics if they were a significant predictor. We were also limited in our lack of data which made us restrict the number of principal components we used with the lyrics. Fitting our model on the entire Million Song Dataset and being able to get the entirety of the lyrics data for each song would have definitively allowed us to better

access our model's validity and our major claims throughout this paper. The musical features had a slight predictive ability, but they were still not very accurate when using them to predict song release year. Being able to use the entire MSD would have allowed us to fit models using more complex transformations of the musical features, which could have resulted in better models. Although, we were still able to outperform the naive classifier and some of our base models using the musical features and our tuned WAMSE and MSE AdaBoost models. Obviously our method would not beat Shazam in its actually accuracy, but for songs that are very obscure and can't be predicted normal, we seem to be better than randomly guessing at predicting the year a song was released.

Based on the fundamental way AdaBoost works, it makes sense that it would inherently the best model for the task of minimizing the WAMSE in this problem setting. This is because when fitting the base classifiers, the AdaBoost algorithm provides greater weight to songs that were predicted more incorrectly so that future regressors in the ensemble are forced to learn the training examples that are harder to learn (i.e. the songs outside 2000). Therefore, this regressor would not get stuck evaluating every point between 1970-2000 like other models, which led to it having a significantly lower WAMSE from most other models we tested.

### 6.1 Best Predictive Features

We determine feature importance for our AdaBoost regressor using sklearn's default metric, which bases feature importance on how the base decision trees use each variable in its splits. For example, loudness and song duration were consistently used by the base regressors as the first splits in the decision trees, indicating that they were very important features.




The most important features for predicting year were shown consistently to be the loudness and song duration, while the song key was the least important feature. It was surprising to see that key was the least important feature in our data set for predicting the song year, because we expected that songs with similar years would have related style and thus similar keys and time signatures. It was also very interesting to see that loudness and song duration were the most important factors, because from personal experience, songs during different time periods don't seem to be distinctly different lengths or loudness. It is also possible that we neglected loudness because when listening to music we can change the volume to whatever we like. It is possible that the importance of loudness as a predictor of song release year could represent a flaw in data collection for the MSD since songs that were recorded a long time ago could have just been recorded using worse quality equipment, resulting in lower volumes for the recordings. These features could also possibly be related to the different underlying genres of the music. More analysis on loudness, song duration and the progression of genre could give very interesting insight into how these features are truly correlated with song release year.

# 7 References and Acknowledgments

## 7.1 Acknowledgments

We would like to thank Cosma Shalizi for his advice in tackling some of the problems we encountered while analyzing the data set.

## 7.2 References

1. Liang, Dawen, Haijie Gu, and Brendan O'Connor. "Music genre classification with the million song dataset." Machine Learning Department, CMU (2011).
2. Hu, Xiao, J. Stephen Downie, and Andreas F. Ehmann. "Lyric text mining in music mood classification." American music 183.5,049 (2009): 2-209.
3. Martineau, Justin, AND Finin, Tim. "Delta TFIDF: An Improved Feature Space for Sentiment Analysis" International AAAI Conference on Web and Social Media (2009): n. pag. Web. 11 Mar. 2019
4. Yu, Lei, and Huan Liu. "Feature selection for high-dimensional data: A fast correlation-based filter solution." Proceedings of the 20th international conference on machine learning (ICML-03). 2003.
5. Singhi, Abhishek, and Daniel G. Brown. "Hit song detection using lyric features alone." Proceedings of International Society for Music Information Retrieval (2014).
6. Drucker, Harris. "Improving regressors using boosting techniques." ICML. Vol. 97. 1997.