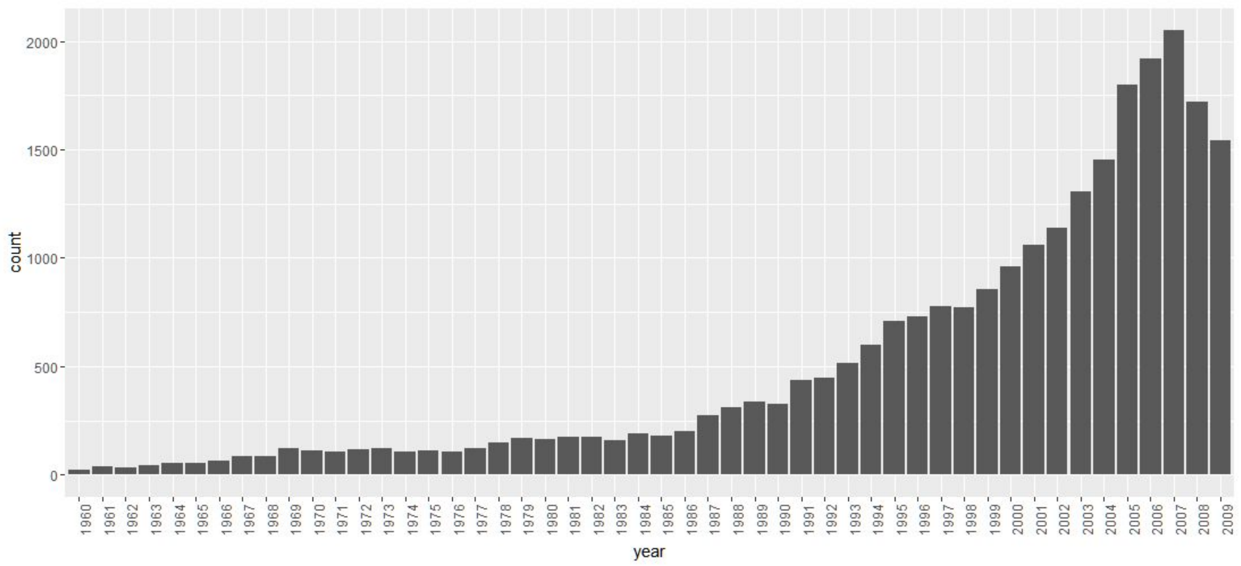
Neil Mahoney
Supervised Learning Approaches to Music Classification

The *million song dataset* is a collection of features describing one million musical recordings produced over the last century[1]. The dataset itself is comprised of several smaller datasets; one of which contains the release date of many audio tracks alongside a set of features quantifying the audio-spectral analysis of those tracks[2]. Both classification experiments performed for this analysis use the “prediction year” dataset as a source. One dataset was generated containing the audio features alongside a 5-year epoch in which the recording was produced, and another dataset for the decade in which it was produced. For the sake of brevity, these experiments will be referred to as the song-lustrum and song-decade classification, respectively, in this paper. The source dataset for both of these experiments gives release date at the granularity of an individual year and the distribution for that dataset can be seen below. Note that the total span of the dataset has been limited to recordings produced in the 50 year span between 1960 and 2009, though entries do exist in the source dataset outside of these bounds (in smaller quantities). After this filtering, 5% of the source dataset was sampled to bring the total down from 515K entries to about 25K, to be split between training and testing. This reduction in sample size makes our experiments much more computationally feasible. One final note is that the class segmentation in both experiments falls along decade lines: in the lustrum dataset classes would be 1960-1964, 1965-1969, and in the decade dataset 1960-1969, 1970-1979, etc.



1 http://labrosa.ee.columbia.edu/millionsong/
2 http://archive.ics.uci.edu/ml/datasets/YearPredictionMSD

Relevance and Challenges

Musical analysis and classification is of immense significance in the modern age. From a commercial and entertainment standpoint, music classification plays a central role in recommendation engines, exposing people to artists and music they are apt to appreciate and support. From an anthropological standpoint, we may use some of these tools to learn more about historical trends in music and speculate what drove them. In realms of art and psychology, we may connect cognitive and musical patterns otherwise unseen. Progress in musical classification means a deeper understanding of all the areas of society in which music touches. It has now been over 150 years since the earliest musical recordings were made, making it one of the richest datasets humanity has produced.

Alongside the promise, there are staggering hurdles. The act of extracting meaningful features from raw digital recordings is certainly one of them. The curators of the *million song dataset* are a group of researchers responsible for Echo Nest[3], a platform for musical analysis recently acquired by Spotify. The features present in these datasets describe an average of the timbral qualities of each corresponding audio track. The details[4] of these features include characteristics such a loudness, brightness, attack and other audio-spectral traits, with some level of dimensionality reduction applied. Since these features are unbounded, a preprocess step has been applied so that they are centered and scaled around zero. The result is that feature data is normalized before being passed into any training phases.

In its initial state, the dataset presents a very challenging classification problem. There are fifty years a song could belong to in our timeframe. In the song-lustrum dataset, this translates to ten five-year periods, and in the song-decade dataset, five ten-year periods. Expectedly, the learning algorithms universally achieve better results assigning decade to a song rather than assigning more granular periods. With more extensive research, it would be worthwhile to compare these results to regression models with year as the output variable, and see if that yields more accurate results. Intuitively, music undergoes somewhat of an “era” effect where sound qualities are popularized during irregular spans of time that spill outside the neat boundaries which I attempt to capture here. For example, many sounds most associated with the 1960s really become popularized during the latter half of the decade[5] and expand into the 70s, and similarly with punk rock between the late 1970s and early 1980s. Another potential topic of further research is letting the music collectively define its milieu through unsupervised learning methods.

[3] http://the.echonest.com/
[4] http://developer.echonest.com/docs/v4/_static/AnalyzeDocumentation.pdf
[5] https://en.wikipedia.org/wiki/Revolver_(Beatles_album)

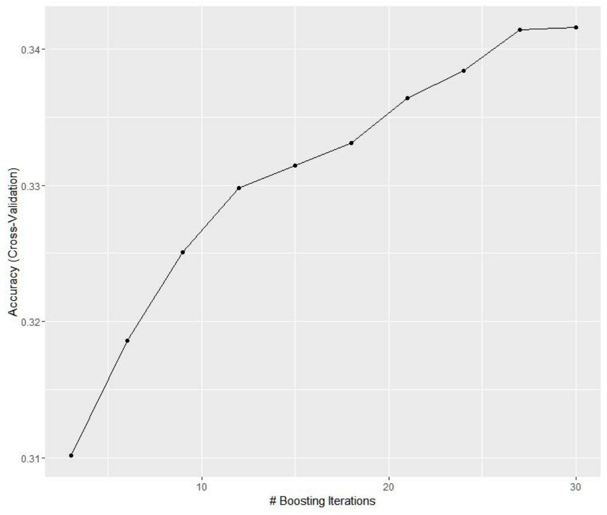
The most challenging aspect of this project was finding the balance between tractable results and computational resources. What you will see in the learning curves is that, across most learning algorithms, accuracy on test data never truly reaches a ceiling as the cumulative size of the datasets reach their maximum of 25K samples. Note that the source dataset has over a half-million samples, so the decision to limit data in experiments was not a product of natural sparsity but of computational demand. Even on a somewhat modern system (quad-core/8GB RAM) several of the models took hours to build. With higher portions of the source data included (50K and upward), that changed to days and often led to system-wide memory errors while building parallelized models in R code. What you see in the charts below was an attempt to arrive at meaningful results within a reasonable feedback cycle.

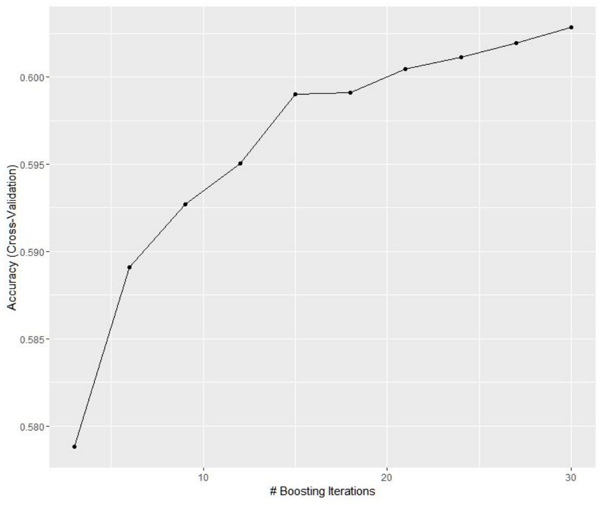
Parameter Selection

Before generating learning curves, experiments were run across a range of parameter values for those algorithms open to tuning. The optimal hyperparameters were extracted from these results and then used in building later models. More precisely, a series of models were built of increasing numbers of boosting trials for boosted decision trees, neighbor count for knn models, and hidden unit/decay value for neural network models. It was generally found that increasing all of these values led to higher accuracy values in their respective models. Accuracy was measured in respect to an average over the validation test sets, where 10-fold cross-validation was used train on 90% of the data and test on 10% in turn.

For all two-column charts that follow, the left contains results for the song-lustrum dataset and the right for the song-decade dataset.

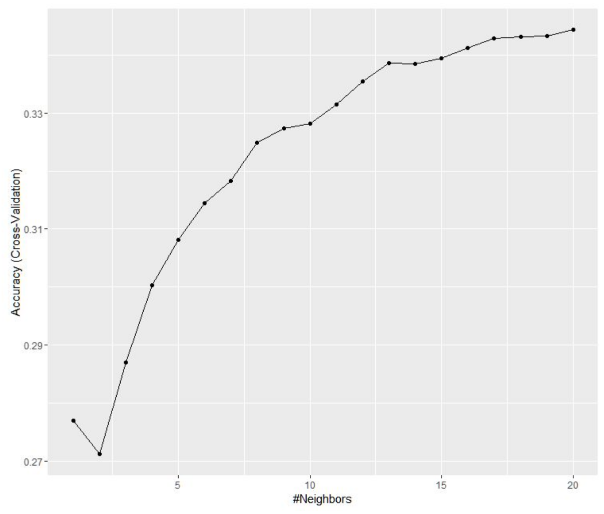
Boosting by trials

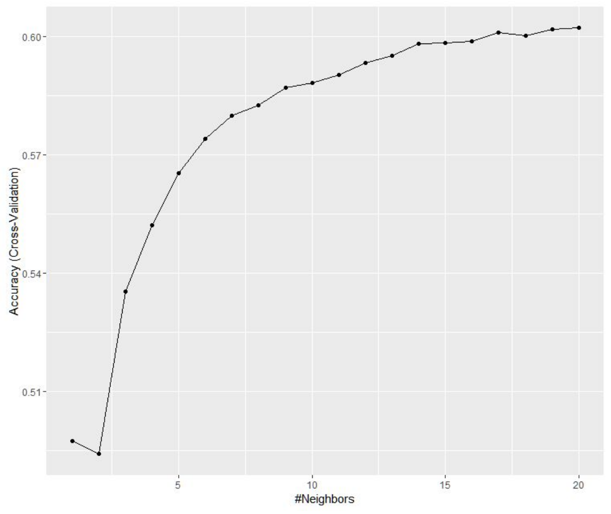




Increasing boosting trials showed similar model results across each dataset. Significant rates of accuracy improvement occur while increasing trials up to ten, and continued, though more gradual, improvement occurs up to 30 trials. The final hyperparameter value selected was 28 trials for song-lustrum and 30 trials for song-decade. The accuracy tops out at just over 34% and 60.5%, respectively. It is worthwhile to note at this point that the probability of a classifier selecting the correct time-class by pure chance is 10% in song-lustrum and 20% in song-decade, so in both datasets boosting achieves about three-times better accuracy than chance as the number of trials nears 30.

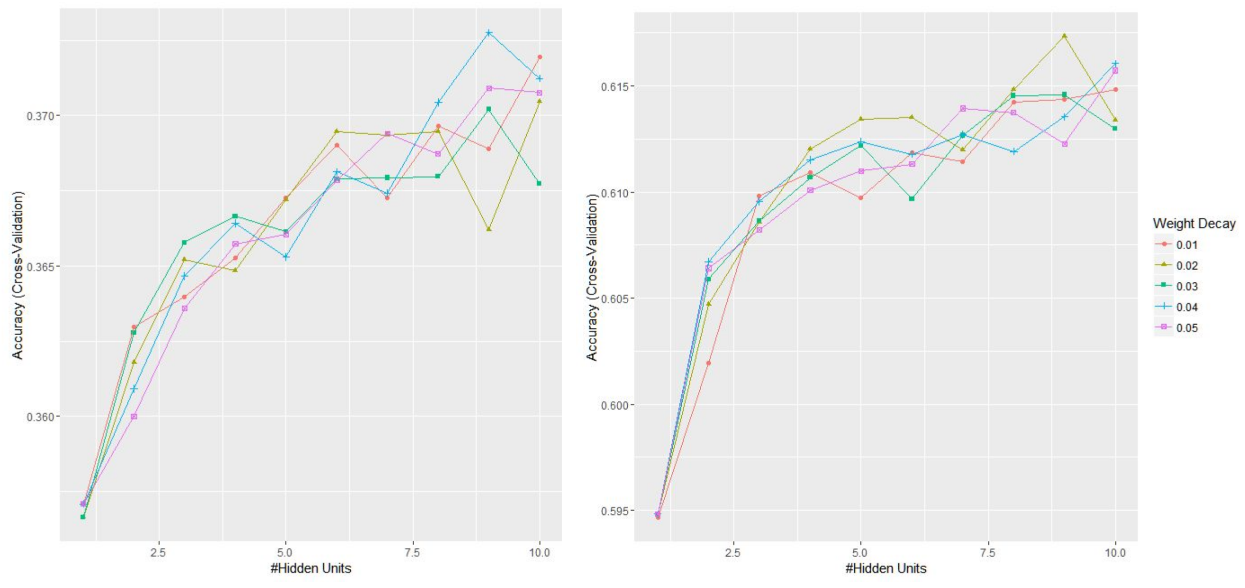
<u>KNN by neighbors</u>





Similarly, we see continued gains in accuracy as the number of neighbors are increased while building KNN models iteratively. Performance begins tapering around a value of k=8 and continues slow upward gains to a maximum of k=20 in what resembles logarithmic growth. A value of k=20 was selected for further experimentation with KNN. Logically, it would make sense for these datasets to require high k-values due to its size and complexity. The more neighbors taken into account, the more context the learner has for making a classification, up until the point where so many neighbors are involved the decision cannot be swayed in any coherent direction..

<u>ANN by hidden units and decay</u>



The two hyperparameters under optimization for neural network modeling are hidden unit count and decay value. Results show a similar tapering effect as the number of hidden units is increased, and similar behavior under the different decay values chosen. Final values were chosen for both datasets as 10 hidden units and .04 decay. Individual tests were run with weights above this range and no performance increase was observed.

With an idea of how hyperparameters were selected, we can continue to further investigation into our individual learning algorithms.

Decision Trees

The specific decision tree algorithm used is the C5.0 algorithm implemented in the R package C50[6]. C50 generates a very large tree for the lustrum data set (top) and not-as-large tree for the decade dataset (bottom). I would speculate that the decision trees are so large because a) our classification problem is inherently difficult and b) our 12 attributes are all continuous numerical values.

```
Decision Tree
----------------
Size      Errors
3061 5135(27.3%)   <<
```

[6] https://cran.r-project.org/web/packages/C50/index.html

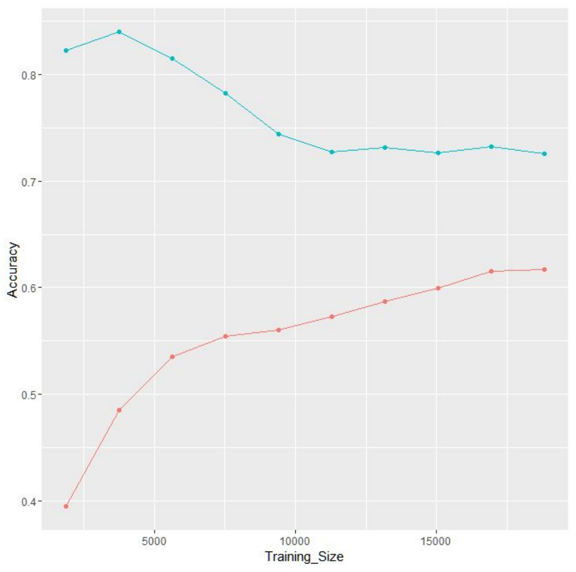
```
      (a)   (b)   (c)   (d)   (e)   (f)   (g)   (h)   (i)   (j)    <-classified as
     ----  ----  ----  ----  ----  ----  ----  ----  ----  ----
       85           1     1     1           5     6    13    16    (a): class [1960,1965)
        8   201     3     2     4     6    10    16    37    25    (b): class [1965,1970)
        7     7   267     7     7    13    20    21    31    38    (c): class [1970,1975)
        6    16    14   316     8    13    25    36    31    50    (d): class [1975,1980)
        5    11    17    17   404    32    21    27    46    48    (e): class [1980,1985)
       11     4    14    16    24   643    35    56    72    57    (f): class [1985,1990)
        6    21    24    32    28    51  1189    97   106   184    (g): class [1990,1995)
       11     8    15    21    41    73    95  1850   252   551    (h): class [1995,2000)
        8    24    32    32    27    54    77   202  3065  1081    (i): class [2000,2005)
        8    24    23    23    32    51    74   183   547  5670    (j): class [2005,2010)

   Attribute usage:
   100.00% X1
   100.00% X2
    98.92% X3
    81.62% X6
    66.58% X4
    66.33% X5
    66.26% X7
    64.69% X9
    63.56% X12
    56.63% X11
    55.39% X10
    43.13% X8
```

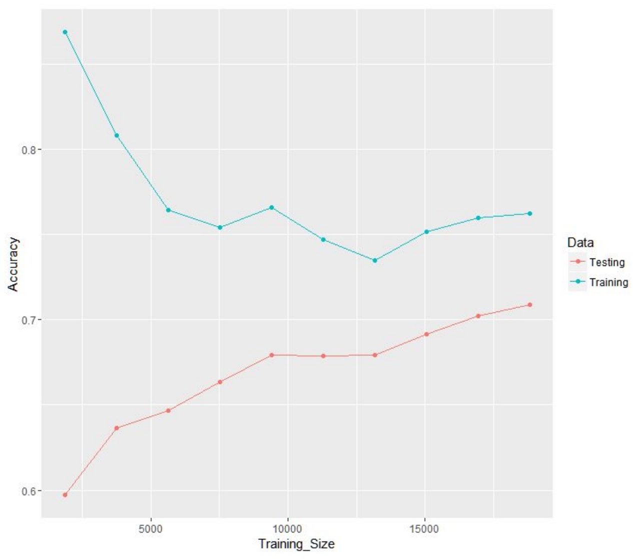
```
     Decision Tree
     ----------------
     Size      Errors
     1237 4359(23.2%)   <<


      (a)   (b)   (c)   (d)   (e)    <-classified as
     ----  ----  ----  ----  ----
      205     8    13    46   162    (a): class [1.96e+03,1.97e+03)
       15   426    33   112   312    (b): class [1.97e+03,1.98e+03)
       13    38   825   221   478    (c): class [1.98e+03,1.99e+03)
       15    58   155  2537  1912    (d): class [1.99e+03,2e+03)
       12    35    94   627 10473    (e): class [2e+03,2.01e+03)

   Attribute usage:
   100.00% X1
    97.73% X3
    93.48% X2
    70.12% X6
    54.07% X5
    48.95% X11
    46.87% X7
    46.61% X4
    42.20% X9
    29.89% X12
    29.49% X8
    20.82% X10
```

Several interesting facts can be gleaned while looking at these sample decision tree results. First, attribute importance generally decreases according to their original ordering (X1 => X12). This, I believe, occurs because the authors of this data have already performed some level of dimensionality reduction on the audio-spectral attributes, and in doing so isolated the most significant attributes at the front of the attribute list in their original dataset. Notice that X1 is viewed as by far the most significant predictor in both models, and that the three most important attributes in both models belong to the set {X1, X2, X3}. We can abstractedly imagine what physical properties of sound most indicate the time period it was produced in. Perhaps frequency separation, loudness, and range all represent such predictors, as they are the most prone to advances in recording technology. Recording components (ie microphones) are highly specialized and adept at capturing only a specific frequency range. Listen to recordings from the 1950s, for example, and you will find that the sound exists entirely in the mid-range, by standards of modern recordings, and the human ear. The post-hoc analysis of these recordings may also be revealing the history of music recording technology.

Looking at the confusion matrices, there are expected and unexpected qualities to it. An expected trait is that when samples are misclassified, they are more apt to receive a classification closer to the time period of their true classification, meaning the decision trees may often be wrong, but typically they are not *very* wrong. Again, maybe this just means the problem is better suited to regression techniques, or we need a more sophisticated way of generating classes than along artificial decade lines. An unexpected trait of the matrices is that misclassifications (regardless of correct class) often get assigned a class representing a time period closer to the present (see misclassifications under labels *i* and *j* in song-lustrum and *e* in song-decade). One potential explanation for this is that there are simply many more samples present from more recent years (see distribution chart of page 1). It would worthwhile training on a version of the data with a uniform distribution by year and seeing if this property holds.

Below you will see the learning curves for decision trees, showing increases in test accuracy as 10% of the total dataset is iteratively included. For this and all other experiments, 75% of the data was used for training (with cross validation) and 25% as a test/holdout set.





For decision trees, we see test accuracy increase from .4 to .6 for song-lustrum, and .6 to .7 for song-decade. It's clear that with a large number of classes and samples, an abundance of data is essential for achieving satisfactory results.

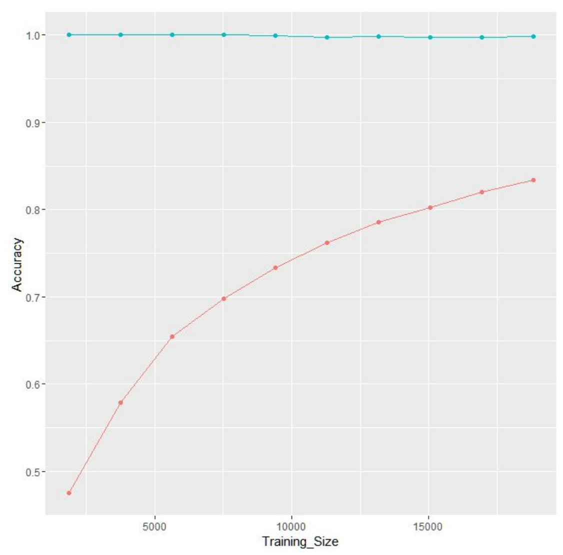
Another interesting fact is that training accuracy decreases (error increases) as sample size grows. This is an indicator of the *variance* of the models being generated; as the training encounters new scenarios (ie classes), it lacks the knowledge/experience to deal with them. On the other hand, test accuracy maintains a steady upward trend so it appears the *bias* of model is not such a problem.

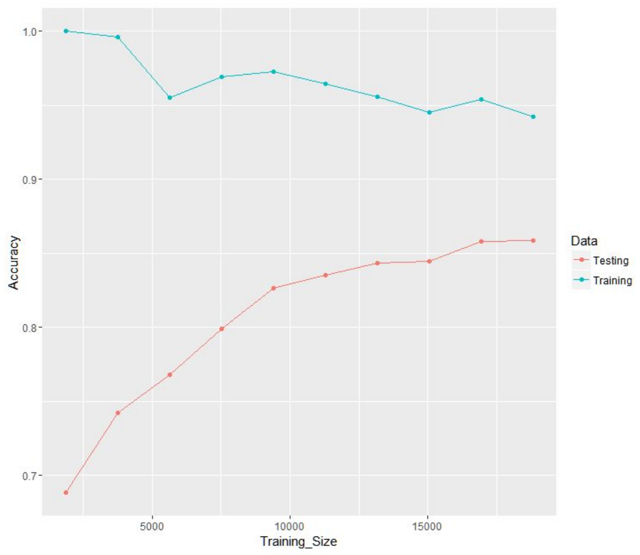
Note that the model is still being generated by a cross-validation process on the 75% of data that excludes the test set, which will be the same process for all models that follow. Also, pruning is built into the algorithm which C5.0 is based off of[7] and no effort was made to modify its defined behavior (global pruning of branches which do not aid performance).

Decision Trees with Boosting

The same decision tree algorithm (C5.0) is used in the boosting experiments which follow. The only difference is the hyperparameter corresponding to boosting trials is set higher than one, based on previous optimization experiments specific to each dataset. Here are the boosted results:

[7] https://en.wikipedia.org/wiki/C4.5_algorithm

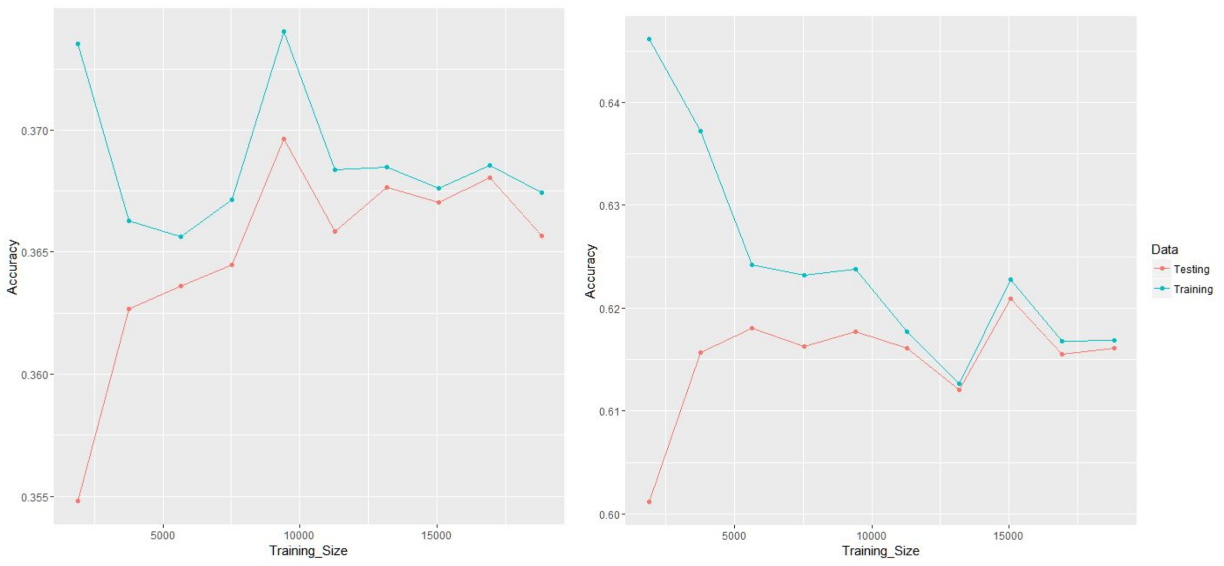




Of all the experiments completed, boosted decision trees demonstrated the best results. On both datasets, boosting is able to achieve an accuracy over 80%, better than non-boosted trees, and significantly better than neural nets, svm, and knn. What's particular striking is performance on the song-lustrum dataset, far surpassing any other algorithm's ability to deal with the challenging classification problem. Instinctively, the advantage which boosted trees has over other algorithms is that it repeatedly focuses on the most challenging classifications. What this might mean in our datasets is that the samples which fall along the fault lines of the classes (the songs released at the start/end of a decade or lustrum) will receive more attention than others. It might not be difficult to gauge that a song produced in the 1990s was not produced in the 1970s, but whether or not it was produced in the 1980s would be a harder question to answer. By focussing on learning the attributes which separate narrow bands musical time, it could achieve success on the samples around those bands. It also raises the question again whether we want a learner to navigate around those (artificial) epochs rather than something more organic.

Neural Networks

From the standpoint of having observed several promising modeling algorithms, we now turn to neural networks. At first glance, the learning charts show that neural nets reach a performance plateau before the upper bound of the datasets are reached (the first algorithm so far to do so). Looking more closely at the y-axis scale, we observe that even in the area where accuracy is increasing, roughly the first 50% of data inclusion, that increase for both datasets is less than one-fifth of one percent. Compared to boosted decision trees, where including an additional 10% of the data would yield a 10% improvement in test accuracy, this seems lackluster.



Given the black-box nature of neural networks, it's challenging to surmise why exactly this is the case, but one can speculate. Maybe there is a fundamental incompatibility between neural networks and the problem. Maybe it would have been a worthwhile experiment to test different neural net implementations other than the one used[8]. The implementation at hand includes only the simplest type of neural network, with one single hidden layer and no back-propagation. The only parameters capable of being tuned are the number of hidden units and decay rate, both of which were optimized in initial experiments. It is a very real possibility that complexity of the data overwhelms the simple neural net, resulting in a high-bias model with low accuracy. Note that accuracy is still substantially better than chance, so learning is happening, though it quickly hits a wall. A corollary fact is that the learning curve only takes about 6 minutes to generate (including training time) on the song-lustrum dataset, less than I would anticipate for general nn-training, which adds to the suspicion that the modeling is simplistic.

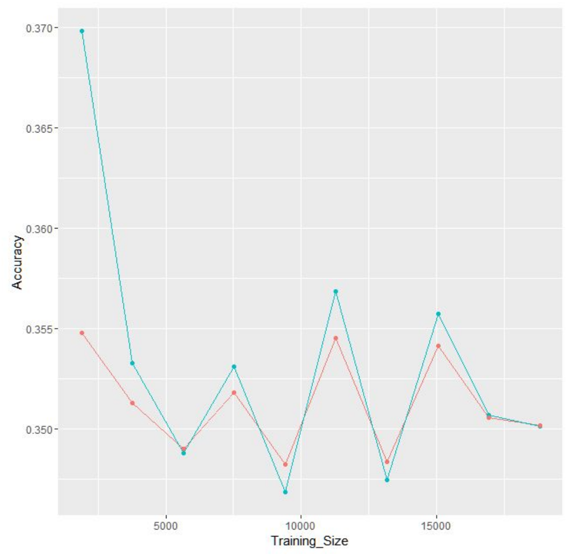
Support Vector Machines

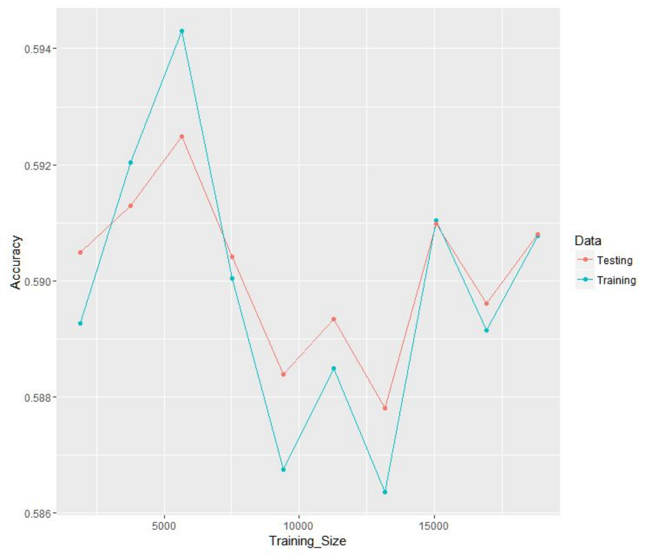
SVM performance is somewhat akin to neural networks, and lagging that of boosted and non-boosted decision trees. Two kernels were selected for these experiments, a linear and radial basis function kernel. These are the two slowest learning procedures of those explored here. Generating a learning curve on the song-lustrum dataset with linear and RBF kernels takes 10 minutes 2.5 hours by wall-clock time, respectively.

[8] https://cran.r-project.org/web/packages/nnet/index.html

Linear Kernel

Unlike all other learning methods, accuracy/error stays flat as more data is added with a linear kernel. Considering why this might be the case, it may be true that the data at hand is just not linearly separable. I imagine the high feature and class count contributes to this, as well as the regression-like nature of the problem. Imagine trying to linearly separate the points on a line itself. It may be able to find a pivot point at which it divides half the line from the other half, but it would never be able to isolate a particular point (ie year/lustrum/decade). It would consequently eliminate one large segment of the data which it sees as most “wrong”, but leave another inaccurate segment within the selected region itself.

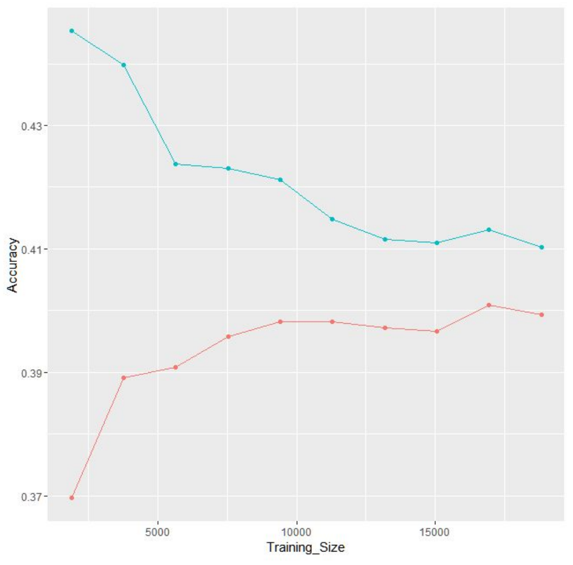


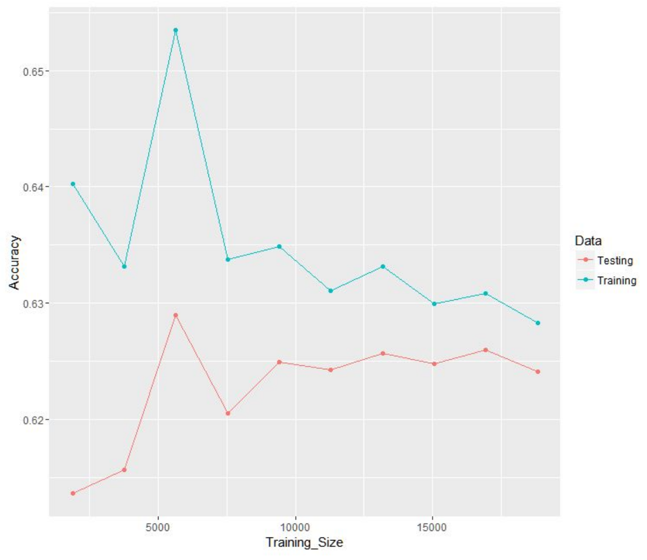


RBF Kernel

In contrast, the accuracy increases as data is added with an RBF kernel, though overall it also does not reach the levels of accuracy as in decision tree methods. An RBF kernel differs in that it may be trying to transform the feature space in such a way that it is linearly separable. Since we see better and more consistent performance as more data is added, one could speculate that it finds success there. This flexibility could be one of the reasons RBF is a default kernel choice for some SVM packages[9]. The cost incurred in this exploration means that SVM with RBF is by far the slowest algorithm tested with this data, and really became the bottleneck on the decision of how much data to include for all algorithms to learn on. With little to gain in performance measures, I would rethink the decision to include this kernel again unless all non-linear kernels exhibit similar performance traits.

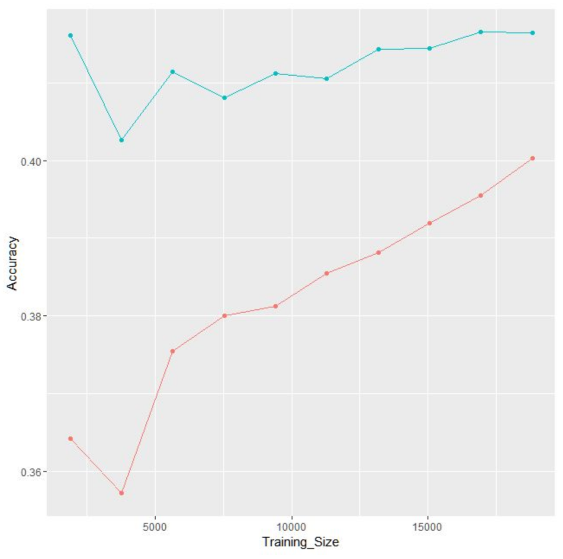
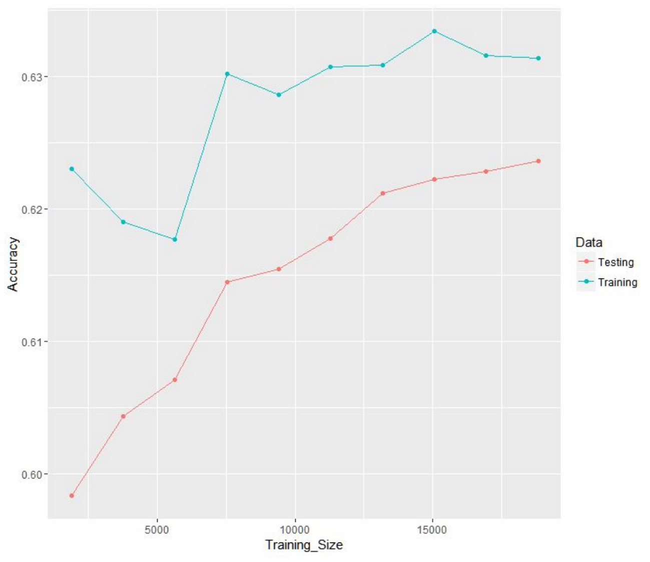
[9] https://www.rdocumentation.org/packages/kernlab/versions/0.9-24/topics/ksvm?





KNN

KNN demonstrates the steadily increasing accuracy that we have seen in decision tree methods, though the rate of improvement is lower. Overall is does not manage to best the results from neural networks or SVMs. However, It gives me more confidence that with the inclusion of more data, it would. KNN also takes far less time to produce such a learning curve, generating the below result sets within a matter of seconds. Of course, the counterpoint to this nicety is that KNN must retain all of the data in order to make future predictions where the previous models do not. KNN is also vulnerable to the “fault line” problem where a sample’s neighbors in feature-space might pull it somewhat arbitrarily between several potential classes, and there is not any mechanism to correct that behavior.


Accuracy
Training_Size

Accuracy
Training_Size
Data
Testing
Training