

# CS4641: Machine Learning

## Spring 2019

### PS1 Supervised Learning

Joel Ye

February 4, 2019

## 1 Introduction

The two datasets used are both publicly available, preferred for their manageable size and clean data. In particular, two datasets of medical measurements for classifying risk of heart disease and risk of diabetes were selected. A critical use of machine learning algorithms is to investigate biological and health domains, where requisite expertise is often prohibitive for data collection and overall throughput of research. Both problems are challenging (or the world would be a better place), but simple enough that these lightweight models can all perform non-trivially, allowing analysis of model configurations. The problem is also good for comparative analysis, as traditional diagnosis can be likened to decision trees or KNN, but other methods explored in this paper may perform better. These datasets are particularly useful for this analysis of model interpretability, as it is essential for any machine learning algorithm in the medical domain to be understood as best as possible.

To detail the two datasets, typical diagnostics are included. For heart disease, the provided information is cleansed down to age, sex, cholesterol, and various heart specific measurements for a total of 13 features. As per standard practice, the provided heart disease status (stage of progression) was collapsed to binary presence of disease. This dataset was pulled from UCI's repository[1], and contains only 303 instances, which causes overfitting that explains some of the noise soon presented.

The other dataset is the Pima Indians Diabetes dataset from Kaggle[2], which has 768 instances and 8 features, along with a binary label of diabetes presence. Its features include more diabetes centered diagnostics such as BMI, blood pressure, and number of pregnancies. The dataset elaborates that a number of NaN entires were filled by 0, but to avoid overengineering the features, this will be regarded as noise. Details of data can be found at cited links.

However, in both feature sets, true NaN values were replaced with average values per feature, and each feature was also independently centered and scaled to unit variance, commonly used to aid algorithms such as KNN as well as neural networks. For analysis, a (consistently) random 80-20 train-test split was used, stratified (class imbalances preserved) to prevent dataset bias interference.

Thus after engineering we have two medical binary classification tasks. Results vary largely due to the irreducible noise and small sample sizes, soon to be explained. This paper will examine performances of common algorithms, and then compare them to each other. The main metric used is error, complement of accuracy. A subset of figures were omitted for brevity.

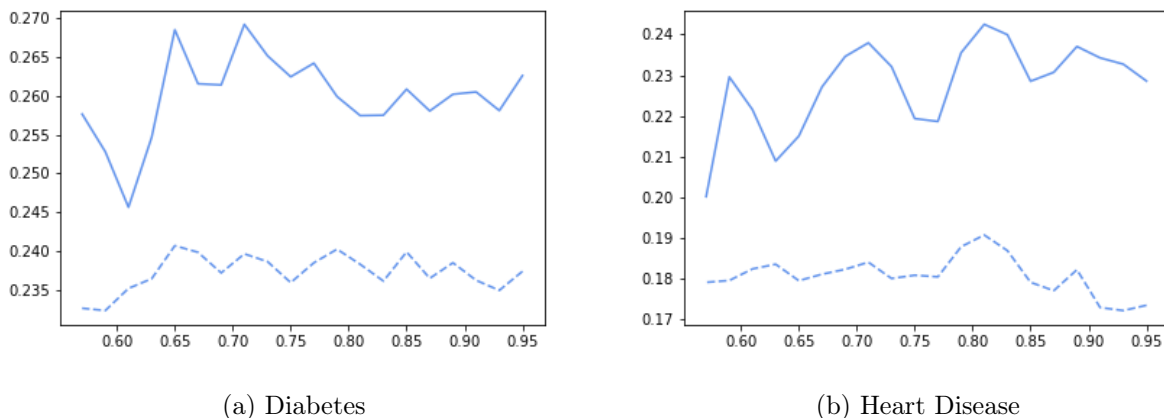


Figure 1: Decision Tree Learning Curves

## 2 Algorithms

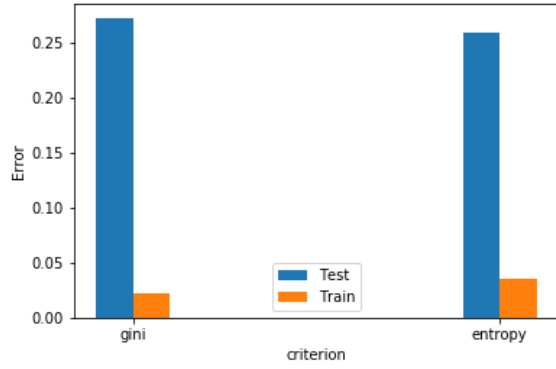
Supervised learning algorithms all share the common structure of prediction through labelled data. The five different supervised learning algorithms are as follows: decision trees, neural networks, boosting, support vector machines, and k-nearest neighbors. Implementations are from sklearn. All training curves were validated by averaging over 20 random splits, as cross validation is invalid on uneven ratios. Cross validation on generic analysis was omitted as trends were generally clear.

### 2.1 Decision Trees

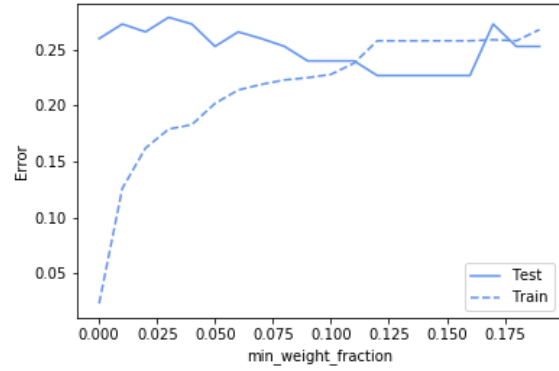
Decision trees predict by filtering input examples through several a tree of nodes that split based on feature values. Thus, each branch represents specific values of features, each node representing a decision point. Important hyperparameters mostly play on controlling tree structure, the other main one being the metric used for node creation. Sklearn offers gini impurity and information gain, but both are performance-wise similar (2a). On the training size learning curve (1), there is fairly consistent test error, which indicates the tree is not particularly biased but did not gain much insight into the problems either. When tweaking decision tree curves, trees performed best when reduced to extremely simple single digit leaves. There is not enough data in both datasets to make informed decisions on most of the features.

Other than the construction algorithm and input data, the main parameters are pruning parameters, i.e. factors and constants that control how opinionated the tree should be. Sklearn offers similar controls, particularly tweaked were max depth, max leaf nodes, and minimum leaf weight fraction. Both max depth and max leaf nodes restrict how expressive and biased the final tree can be, inversely related to train error. Minimum leaf weight fraction lower bounds how representative each leaf is, so higher values make trees less biased. Seeing 2b, 2c, 2d, indeed the training error curves vary as expected. For 2c, 2d, test error climbs as the model is more biased, and we see a good tradeoff of bias and variance, as the test error dips to a minimum of around 22%.

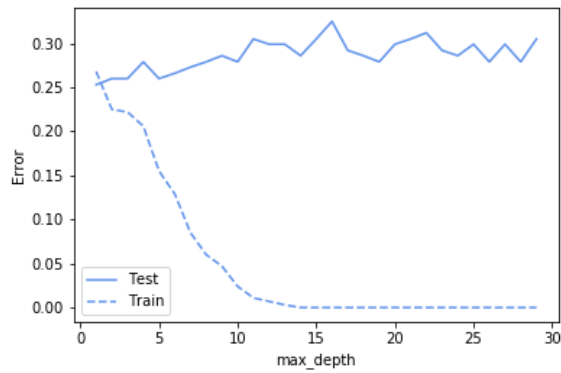
Note the relative performance of decision trees in 2e, 2f. Though the training curve looks similar, the test curve is noisier. This is indicative of the smaller dataset used for heart disease. Since trees fit to our training data is less likely to be generally good, each perturbation at construction time causes strong performance fluctuations. These pruning methods are few of many, and other methods such as post-pruning as opposed to growth restrictions would result in different curves.



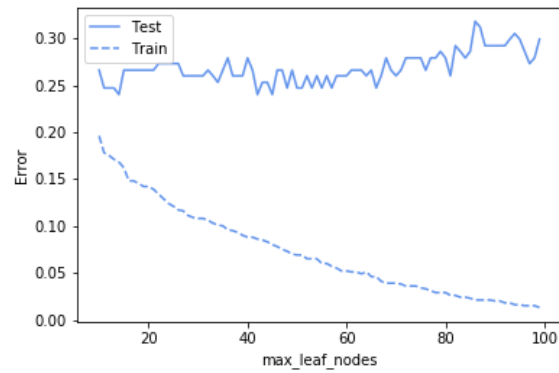
(a) Diabetes Error vs Training Criterion



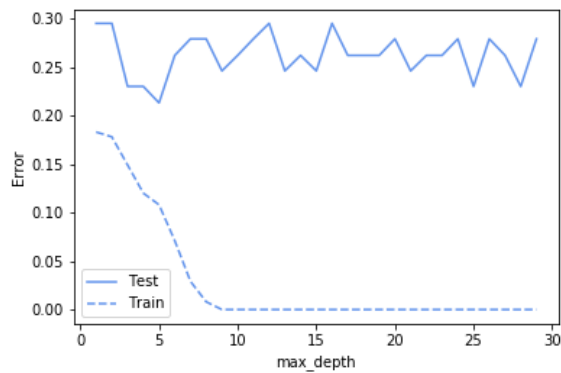
(b) Diabetes Minimum Leaf Fraction Weight



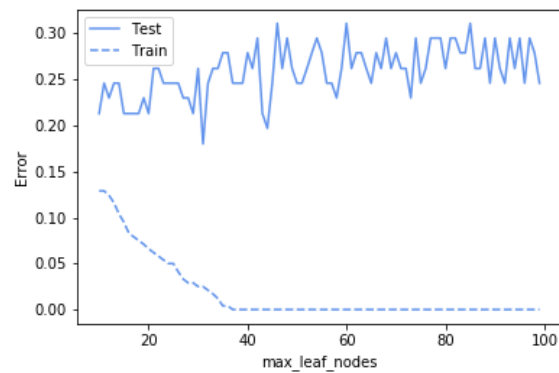
(c) Diabetes Error vs Training Criterion



(d) Diabetes Minimum Leaf Fraction Weight



(e) Heart Disease Error vs Training Criterion



(f) Heart Disease Minimum Leaf Fraction Weight

Figure 2: Decision Tree Metrics

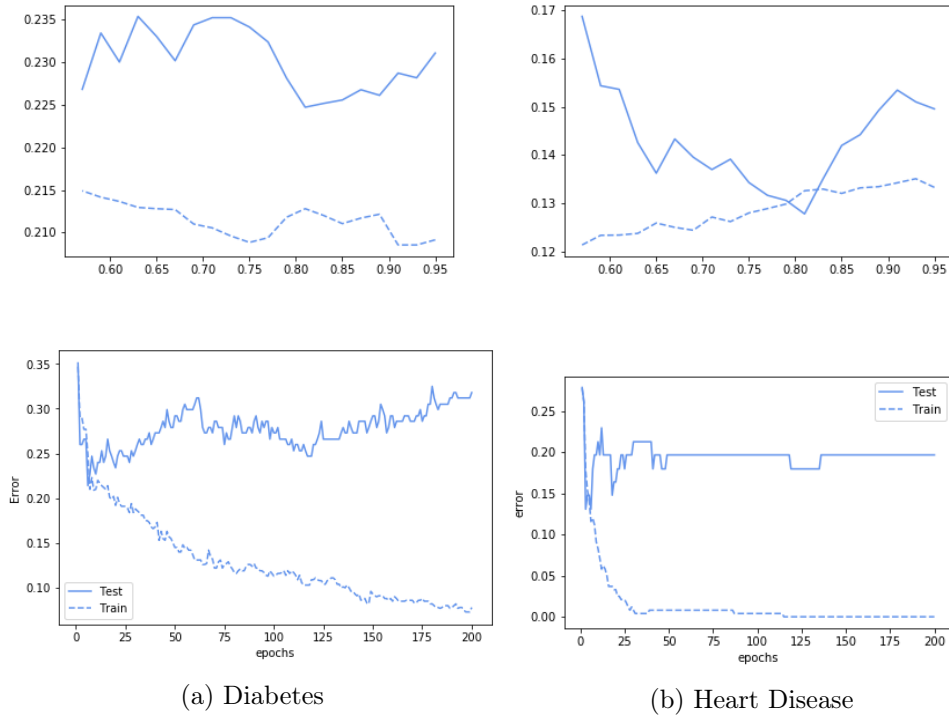


Figure 3: Neural Network Learning Curves

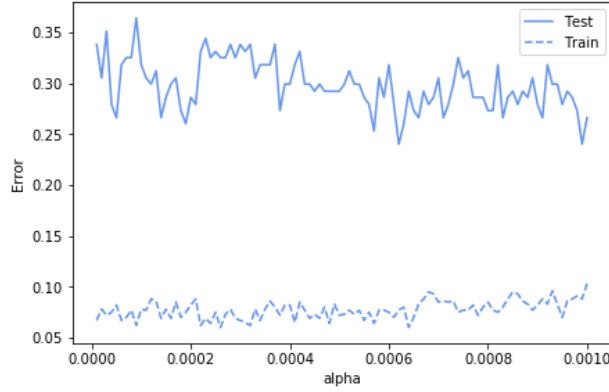
## 2.2 Neural Networks

Neural networks are the most heavyweight algorithm in this analysis, and in their general form should be able to get very competitive results. For this analysis, sklearn’s multilayer perceptron was used, meaning that the input data is fed into a series of fully connected hidden layers, each activated by a nonlinear function; weights are updated via backpropagation to improve hypotheses. There are many hyperparameters to tweak with neural networks, but the ones experimented with were architecture, activation function, iterations, and alpha, a regularization term.

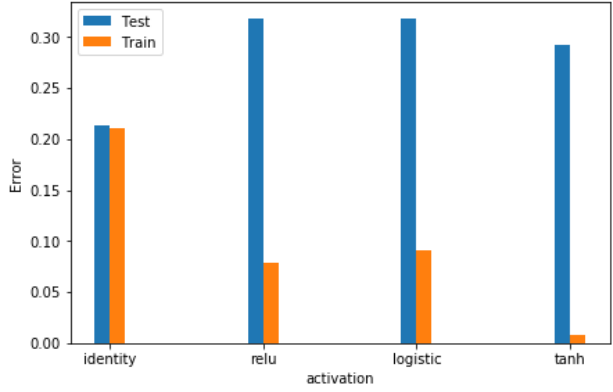
For most of the curves the net ran until convergence. Unlike with decision trees, there are some downward trends in the test errors. Regularization  $\alpha$  was already much higher than typically used, and architectures were also very simple. Given this, on heart disease we have the proper rising train error as the data becomes harder to fit with the restricted hypotheses, and lowering test error, where the rise at the end can be attributed to insufficient data. Likewise, test and train error curves are extremely close on diabetes, but data is insufficient for trends to emerge.

Looking at error vs iterations (which ran on different hyperparameters), we see heart disease should be a simpler problem as the network converges to perfect accuracy, but even after 200 epochs, the diabetes train set did not converge (note that sklearn’s convergence tolerance was dramatically reduced for this experiment). In both cases, we still have a good example of how training convergence means overfitting, as this validation experiment should we should have stopped in around 10 epochs for both problems, retaining simple hypotheses. Note that at convergence the test error flatlines as well, as the weight updates are infinitesimal.

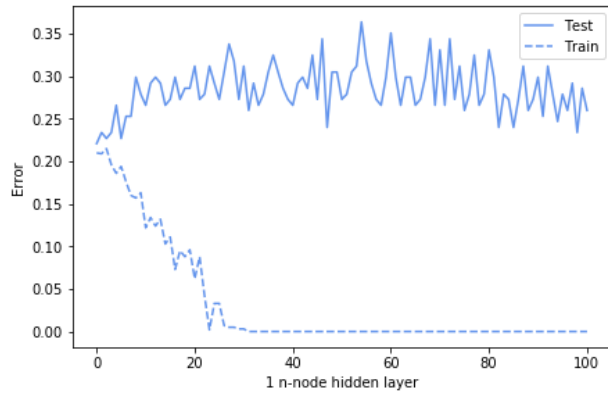
Analyzing the neural network metrics shed light on the nature of the problems. For example, per 4a, the most consistent activation function is actually the linear identity map, implying a simple linear combination of the inputs yields the best hypothesis on the diabetes dataset. More



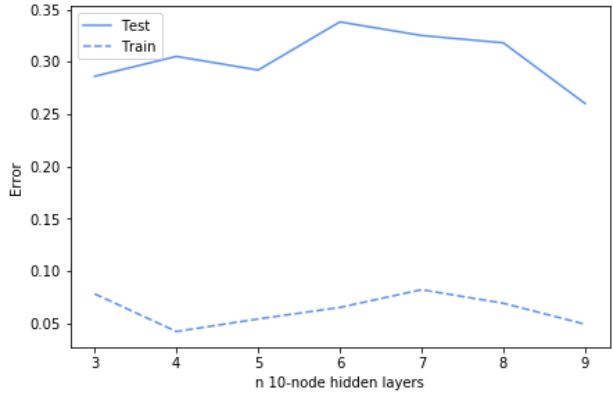
(a) Diabetes  $\alpha$



(b) Diabetes Activation Function



(c) Diabetes, 1 hidden layer



(d) Diabetes, n hidden layers

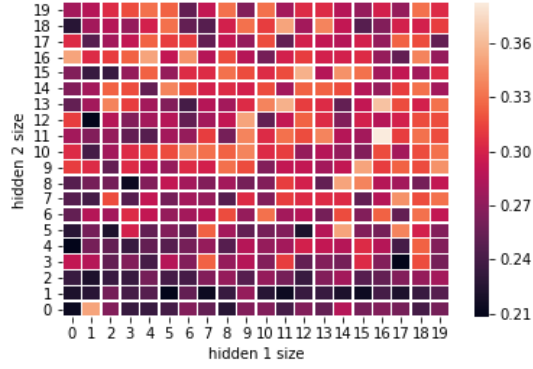
regularization slightly lowers error over time, but neither width nor depth of hidden layers has any noticeable impact on the error. A 2 hidden layer analysis reveals some interesting properties. On the training side, the top right region show the threshold for perfectly overfit networks. Comparing this with heart disease, we see it's immediately possible to overfit heart disease, on top of almost patternless 5c, implying the network derived a general heuristic but cannot improve much further than that. From this we would probably expect to see similar limits in performance across other algorithms. In 5a we note the slight improvement in test error when the second hidden layer size is small, again validating that simpler hypotheses work better. These parameters generally all outperform the other architectures of 4c, 4d.

Quickly noting the training time, important for more complicated use cases of neural nets. 6a,6b shows approximately linear time complexity with respect to epochs, fitting our simply linear network.

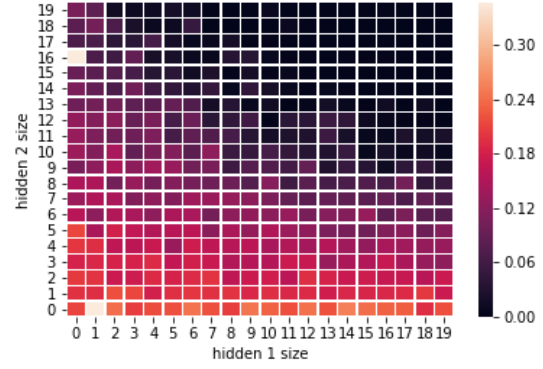
## 2.3 Boosting

Boosting is a meta-learning technique that capitalizes on individual weak learners to build a collective algorithm that predicts well. The ada-boosting algorithm from sklearn was used. Typically the input weak learners are expected to do only slightly better than random, and the meta-learning delegates classification to the individual learners, weighted by the learners' track records. This method might be expected to perform better because it is more resistant to overfitting.

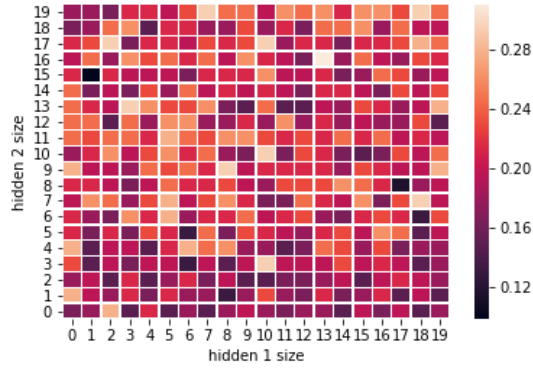
The learning curves 7a, 7b do not indicate that boosting models have any difficulty adapting



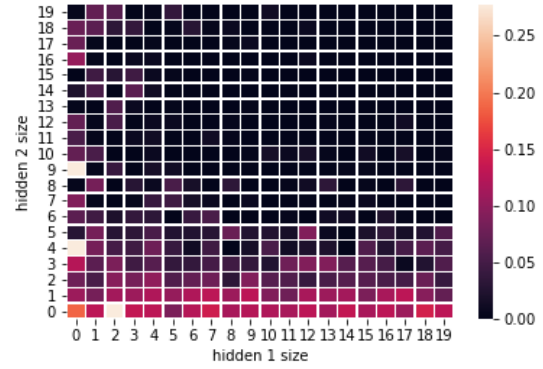
(a) Diabetes, 2 layers test error



(b) Diabetes, 2 layers training error

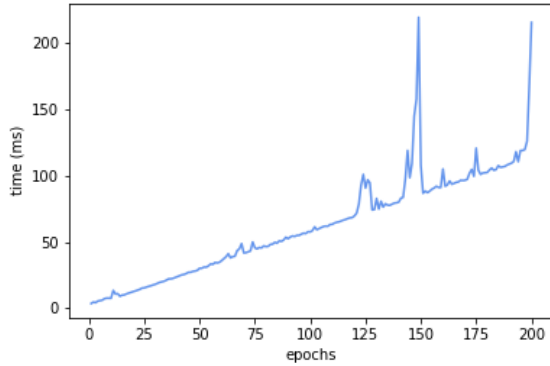


(c) Heart Disease, 2 layers test error

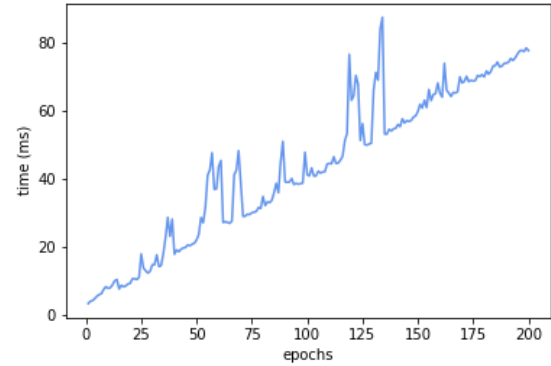


(d) Heart Disease, 2 layers training error

Figure 5: Neural Network 2d error



(a) Diabetes



(b) Heart Disease

Figure 6: Neural Network Train Time

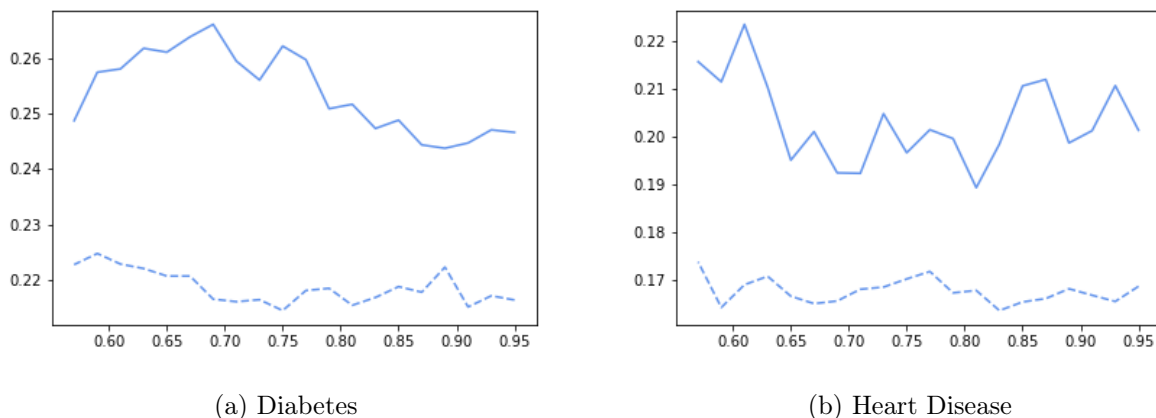


Figure 7: Boosting Learning Curves

to more training data, and noting that performance is only slightly better than decision trees<sup>(1)</sup>, we can conclude the data is easy to learn simple patterns with. Since boosting implicitly fights overfitting, test error has slight downwards trends, though we note the same noise in heart disease that occurred with trees.

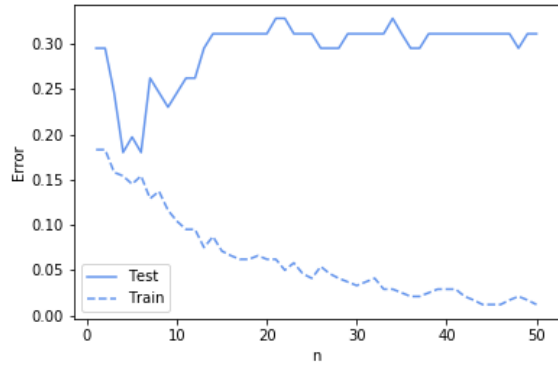
The hyperparameters for adaboosting investigated are number of learners ( $n$ ), base classifiers, and max depth (when using decision trees). Note the inconsistency between 8b and 8c is due to a difference in boosting algorithms (SAMME was used for the latter, a discrete boosting algorithm over SAMME.R, a continuous one). The default one used for experimentation is more consistent and typically gets lower test error, thus better to benchmark (our validation) with. As number of learners increase in 8a, test error sinks briefly and then rises back up, seen before in plots of analogous bias increasing hyperparameters. For reference, number of estimators can be seen as number of iterations of the boosting algorithm. Max depth was only kept from 1 to 4, to prevent the weak learners from overfitting data, and there is at least some value in allowing more than a decision stump as a learner. There is no conclusive winner for which base classifier does better in general (which is why it is a hyperparameter to be tweaked).

## 2.4 Support Vector Machines

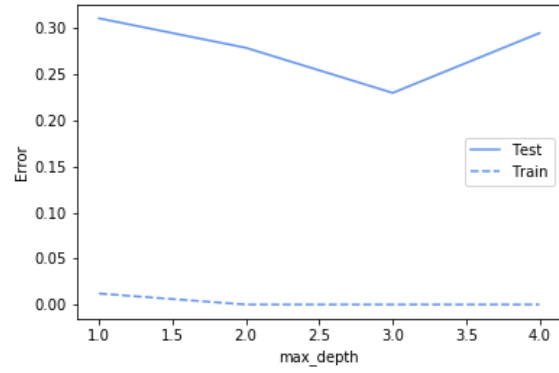
SVMs are known to be good models for data that is approximately linearly separable in some dimension. It is mainly useful for tasks with a few important features, due to curse of dimensionality. Though both tasks here are conveniently binary, SVM generalizes, in one vs one scheme in sklearn SVC, which is the model used. SVM finds the feature space in which the data is linearly separable via the kernel trick, so the kernel used is an important hyperparameter. These kernels themselves are tuneable, and on top of this the main hyperparameter of interest is the regulatory term  $C$ , which controls error penalty.

First considering the learning curves, we see generally the same trends as with neural nets. With a smaller train set, the diabetes problem suffers from uninformative, inseparable spaces, i.e. there is simply not enough data to create a meaningful boundary given all the noise. However, not much data is needed to construct a good approximation of the ideal boundary. Since the regularization used was high (1), the test error declines as SVMs see more train data.

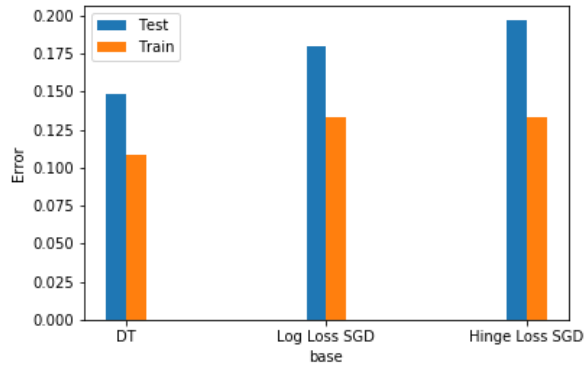
The depicted metrics in 10 are kernel used, gamma, a hyperparameter of rbf kernel, and  $C$ , the aforementioned penalty term. Both gamma and  $C$  are shown on log axis, as their values typically



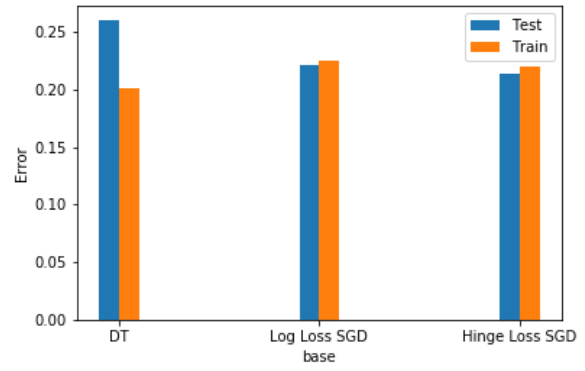
(a) Heart Disease n



(b) Heart Disease max depth



(c) Heart Disease base classifier



(d) Diabetes base classifier

Figure 8: Boosting Metrics



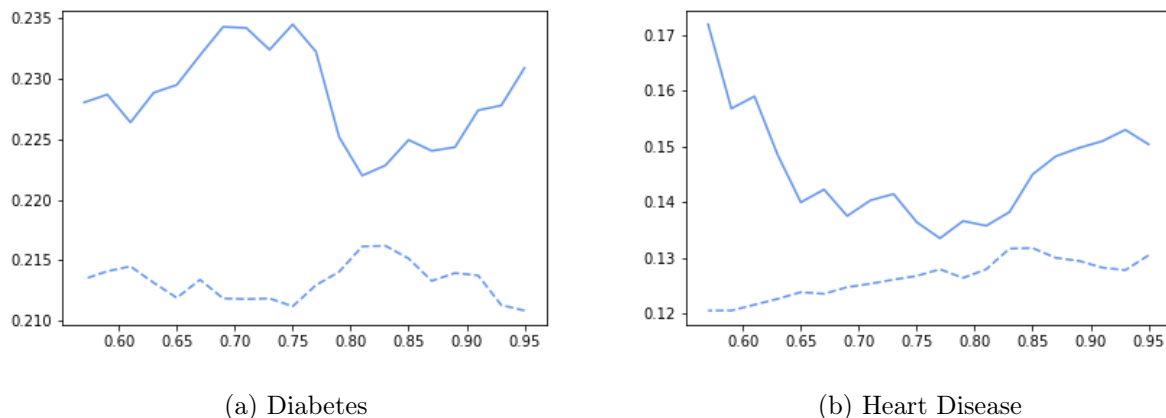


Figure 9: SVM Learning Curves

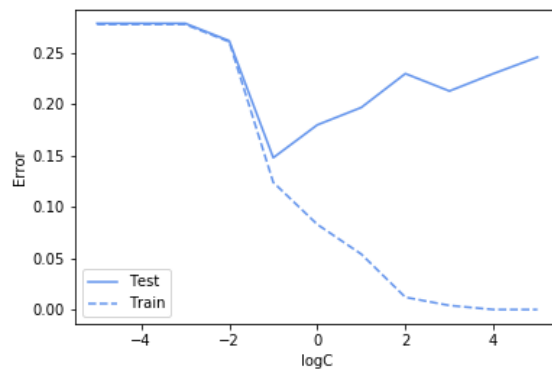
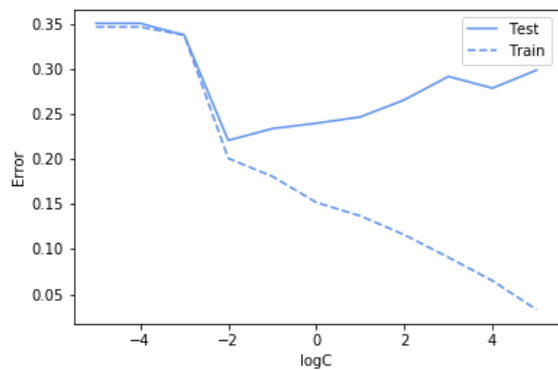
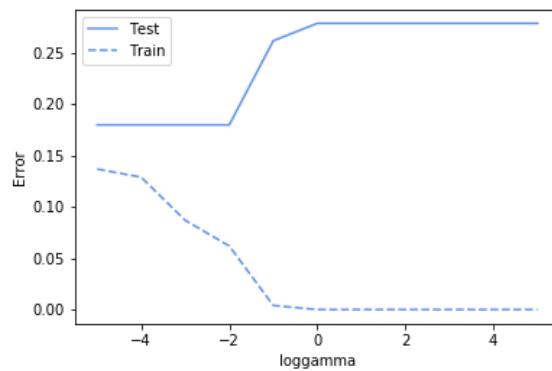
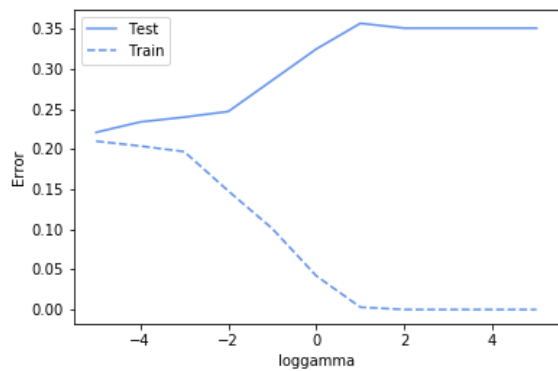
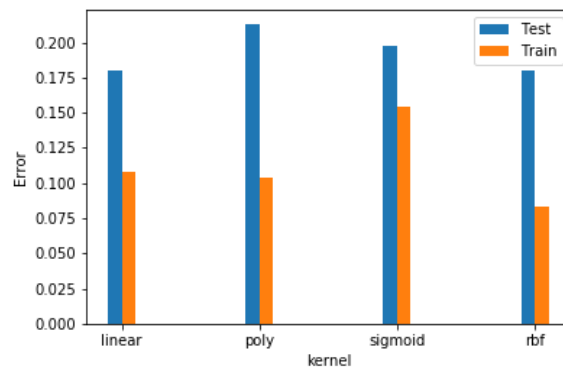
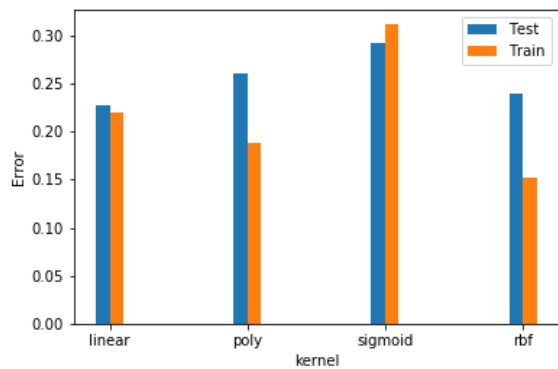
scale as such. From the kernel graphs, we note the same conclusion as from neural nets using linear activation, in that a linear kernel gets best test time results. However, rbf (which stands for radial basis function), it's notable that test error is only slightly worse than linear, and the data is fit significantly better. Gamma is a parameter that tweaks the influence radius of local data, where low gamma means each point has large radius. High gamma means local influence is emphasized, causing the overfitting seen. The regulatory term displays another typical bias variance tradeoff.

## 2.5 K-Nearest Neighbors

K nearest neighbors classifies each input using the labels of the input's closest K neighbors. KNN is unique for its nonparametric nature, causing constant traintime but linear runtime. The parameters considered are k, p, and weighting function. k is the number of neighbors to consider when taking the classification vote. On the low end, this encourages overfitting, matching exactly nearest neighbor, while on the high end, it encourages underfitting, considering only the majority class. The weighting function can intervene in this process, as it decides how the votes of each neighbor is weighted, (if at all). Thus more distant neighbors have reduced influence, this should help with smoothing our hypothesis. p is the exponent on the Minkowski distance (where p=2 implies Euclidean distance). Higher values of p make Minkowski distance tend toward Manhattan distance, which encourages listening to neighbors that are more similar on fewer axes (but it's not too clear how this affects error).

Examining the learning curve, we see this is the first learner that has visible training error rise as training size increases. KNN's simple method causes bias error, which will rise as more varied data is introduced into the training set. Test error is mostly consistent, as it essentially approximates the upper bound on overall noise in the dataset, which does not change as our sparse feature space becomes slightly more dense.

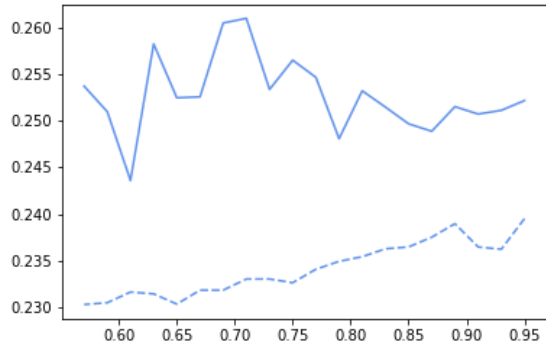
Looking at the effects of each metric, weighting turns out to be extremely unimportant, with no effects in either dataset, though its interesting that using distance as weight causes immediate overfitting and zero training error. This is likely due to the 7-8 dimensional input features. The k graph shows progression towards perfect fitting of train and suggests a model error limit of about 16%. The effect of p on diabetes data is similar.



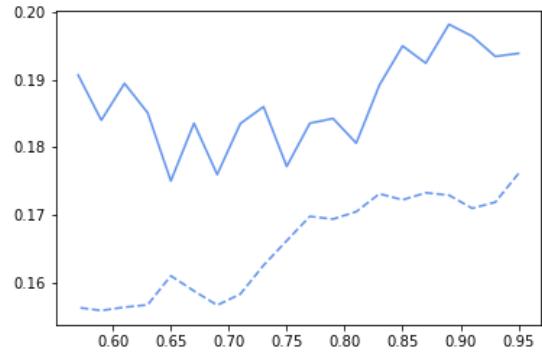
(a) Diabetes

(b) Heart Disease

Figure 10: SVM Metrics

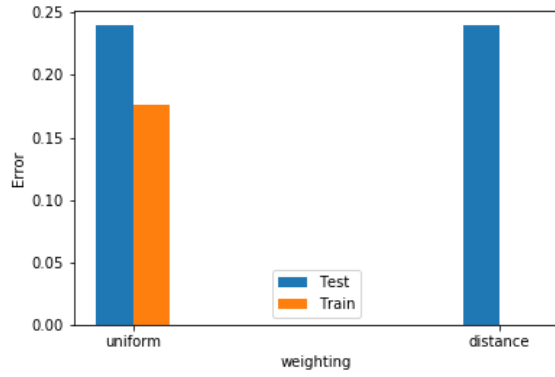


(a) Diabetes

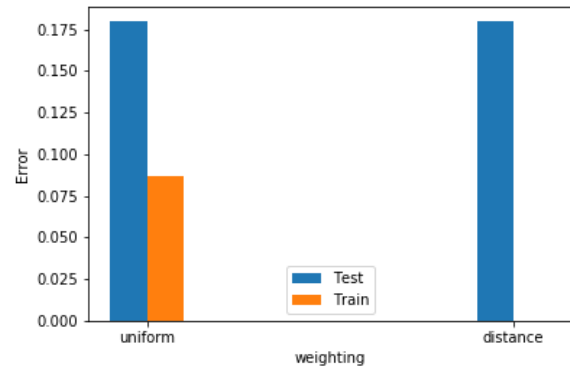


(b) Heart Disease

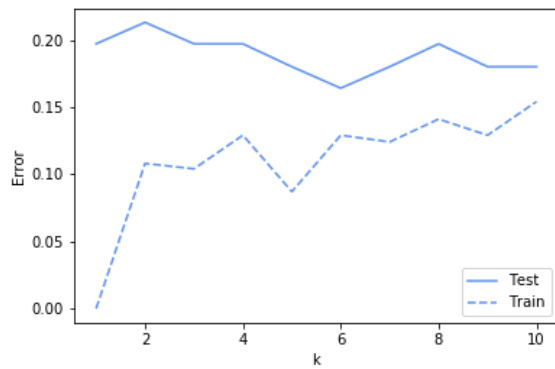
Figure 11: KNN Learning Curves



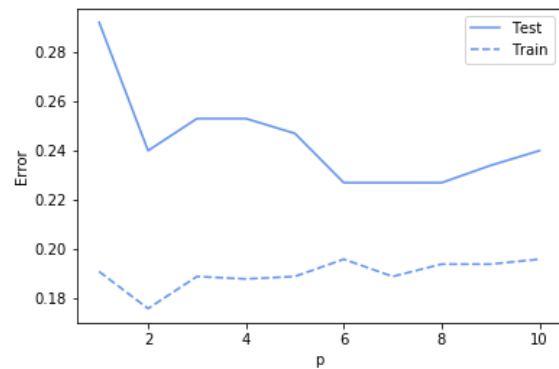
(a) Diabetes Weighting



(b) Heart Disease Weighting



(c) Heart Disease k



(d) Diabetes p

Figure 12: KNN Metrics

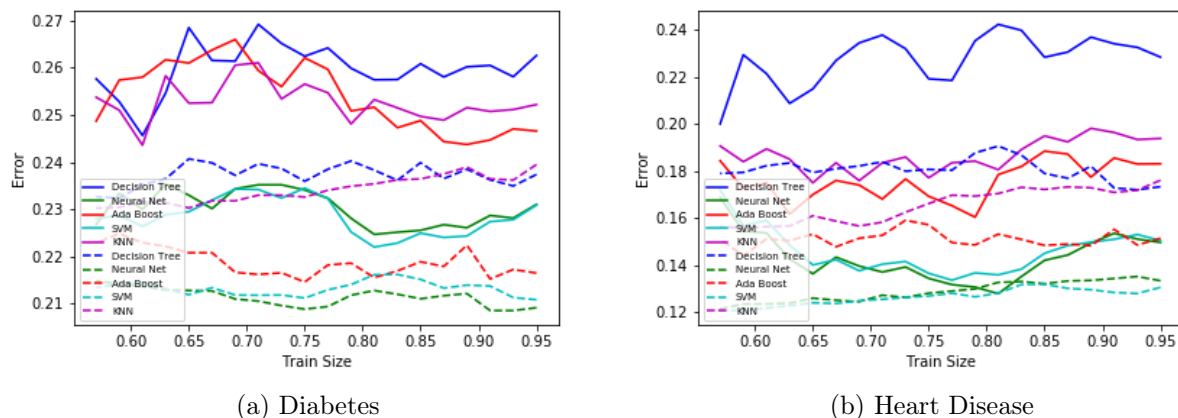


Figure 13: Learning Curves Summary

### 3 Comparison of Algorithms and Conclusion

Individual analyses help illuminate the effects of hyperparameters, while comparative analysis will help detail quirks of the dataset. In general it is difficult to see the effects of train set size on the error. There is high variance in the dataset as none of the models visibly converges to a single error. The irreducible error in the dataset is about 22 percent for diabetes, and 14 percent for heart disease, and these could stem from lack of suitable hypotheses, but more likely are caused by the inherent dataset noise, and lack of true hard rules in reality. In the medical domain, there are absurdly many factors. As the graphs demonstrate, SVM models pull out slightly ahead of the others, while overly simplistic decision trees, KNN, and even boosting suffer at test time. However, all the models get fairly similar performance, within 5-10% ranges for test error, as they were all optimized to minimize variance. This is why the testing and training errors are fairly closely aligned, especially easily regularized structures such as SVM and neural net. In most scenarios, the neural net bias is high due to its natural flexibility, but having tweaked the regularization parameter to excessively high even that variance gap can be reduced.

Notice that overall our error "curves" lay mostly flat. In some problems we expect training error to rise with set size in properly regularized models (as noise seeps into the train set), but in this case our dataset is too small to see this pattern. In the heart disease dataset, we have at least 10 continuous features, with only 200 something training examples. By curse of dimensionality, even if the features were binary, the models have to guess for many inputs. The slight uptick in test at the end of every curve is due to noise in the tiny test set of 30 instances or so. Relative performance of the models between the datasets is consistent which makes sense considering the similarity of the domains. The general best performer is a toss-up between SVMs and neural nets, which also makes sense because the nets investigated were essentially linear classifiers as well.

### References

- [1] Andras Janosi, William Steinbrunn, Matthias Pfisterer, Robert Detrano, *Heart Disease Dataset*. 1988.
- [2] Pima Indians Diabetes Database  
<https://www.kaggle.com/uciml/pima-indians-diabetes-database/home>