1. Classification problems

1.1 Wine Recognition (Data link)

The data of Wine Recognition is the results of a chemical analysis of wines grown in the same region in Italy by three different cultivators. There are thirteen different measurements taken for different constituents found in the three types of wine.

With 13 different statistics, we would like to predict which type a glass of wine belong to.

The data contains 13 numeric features and each data point belongs to one of 3 classes.

Following features are included:

- Alcohol
- Malic acid
- Ash
- Alcalinity of ash
- Magnesium
- Total phenols
- Flavanoids
- Nonflavanoid phenols
- Proanthocyanins
- Color intensity
- Hue
- OD280/OD315 of diluted wines
- Proline

1.2 Breast Cancer Wisconsin (Diagnostic) (Data link)

To diagnosis Breast Cancer with machine learning algorithm is a promising trend. A good machine learning model would help doctors to diagnosis Breast Cancer much faster and more accurate.

In the data, features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

The data contains 30 numeric features and each data point belongs to one of two types of breast cancer (i.e., WDBC-Malignant and WDBC-Benign).

Following features are included:

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness (perimeter^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension (“coastline approximation” - 1)

1.3. Metrics

For this project, since we have two multi-class classification problems, we will be using the overall accuracy as the main performance metrics.

Overall accuracy can be calculated as follow:

Overall accuracy = # of correct prediction / # of total prediction

1.4 Cross Validation

When measuring metrics, we will also be using n-fold cross validation, where n=5. The final overall performance is calculated as the average of model performance on each fold.

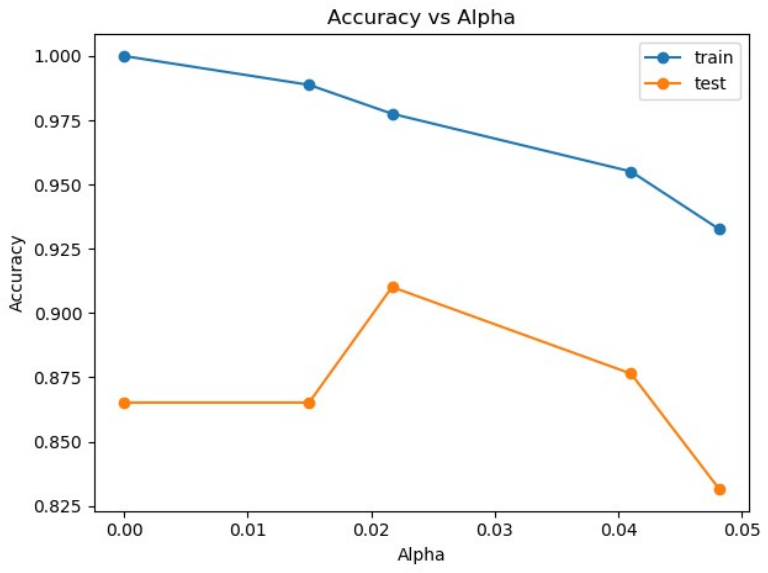
2. Modeling on Wine Recognition problem

2.1 Decision trees with pruning

The function we use to measure the quality of a split is Gini impurity.

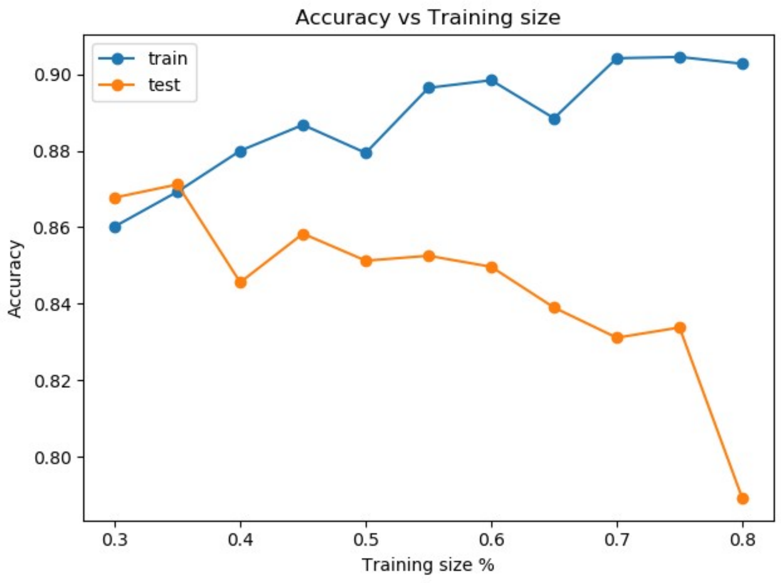
And we also use a post pruning method called cost complexity pruning. Minimal cost complexity pruning recursively finds the node with the "weakest link". The weakest link is characterized by an effective alpha, where the nodes with the smallest effective alpha are pruned first.

To fine tune the hyper parameter of alpha, here we trained models on various alpha with 50% training data size.



As we can see, alpha=0.0217 we have the best test performance. Hence we will use this value as our pruning method here.

Below is the graph of decision tree model performance on various training size. For example, 0.3 means we use 30% of the full data set as training set and 70% as test set.



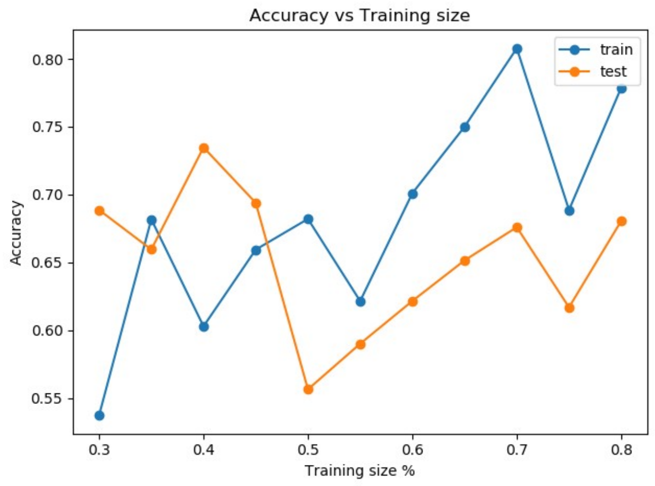
As training size increases, accuracy of training set also increases. On the contrary, accuracy of test set is decreasing. There is a clear performance gap between training set and test set when training size is larger than 70%. Hence, when training size is larger than 70%, our model is overfitting.

The best performance on test set we get from decision tree model is 87.12% accuracy.

2.2 Neural networks

Here we use a simple neural network with 5 hidden layers, each layer contains 100 units. The activation function is ReLU, learning rate is 1e-5.

Below is the graph of Neural networks model performance on various training size. For example, 0.3 means we use 30% of the full data set as training set and 70% as test set.



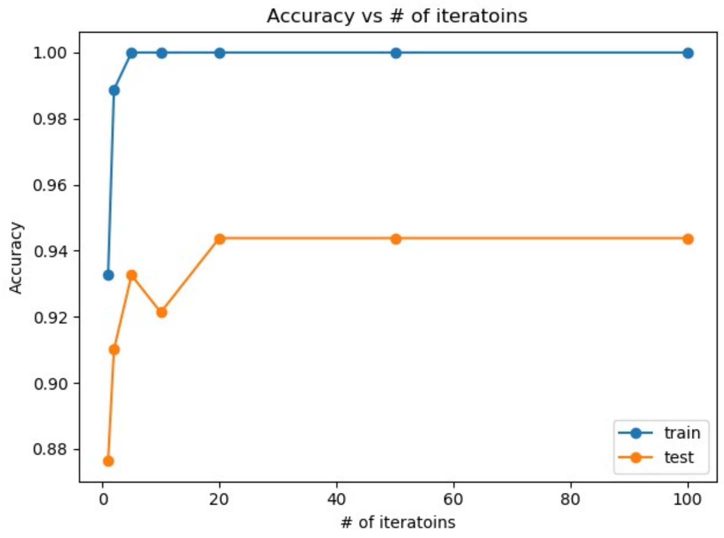
As training size increases, our training and test performance is both increasing.

The best performance we get from neural network is 73.48% accuracy.

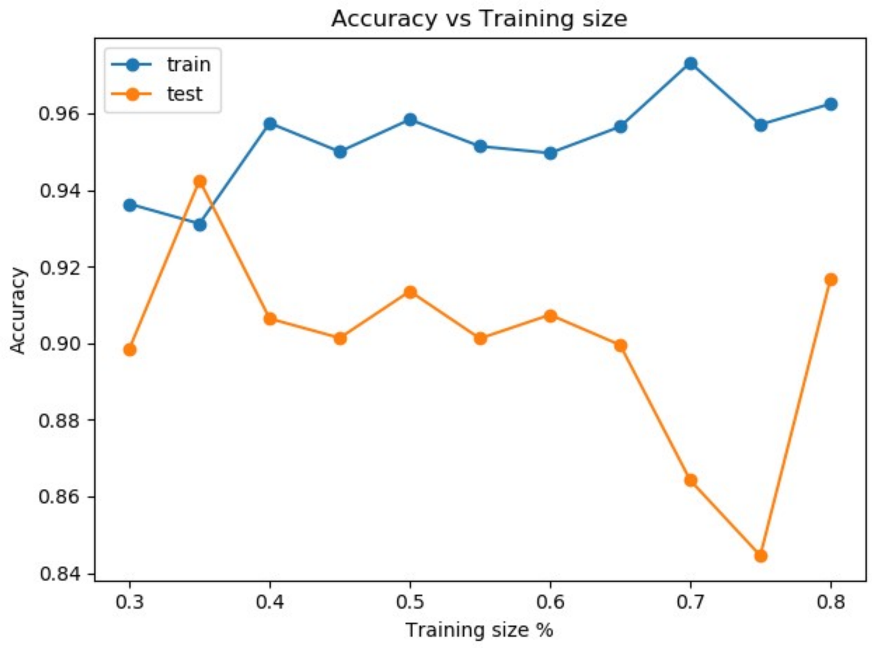
2.3 Boosting

For boosting we use the Gradient Boosting Decision Tree. One of the hyper parameters we need to fine tune is the number of estimator, or you can say number of iterations we need.

Below is the graph for relationship between # of iterations and model performance. As we can see, as # of iteration increases, model accuracy is also getting better. But it tends to saturate when # of iteration is large enough, which means we will not gain any benefit when # of iteration is larger than 100.



Below is the graph of GBDT model performance on various training size. For example, 0.3 means we use 30% of the full data set as training set and 70% as test set.



As training size increases, accuracy of training set also increases. On the contrary, accuracy of test set is decreasing. There is a clear performance gap between training set and test set when training size is larger than 70%. Hence, when training size is larger than 70%, our model is overfitting.

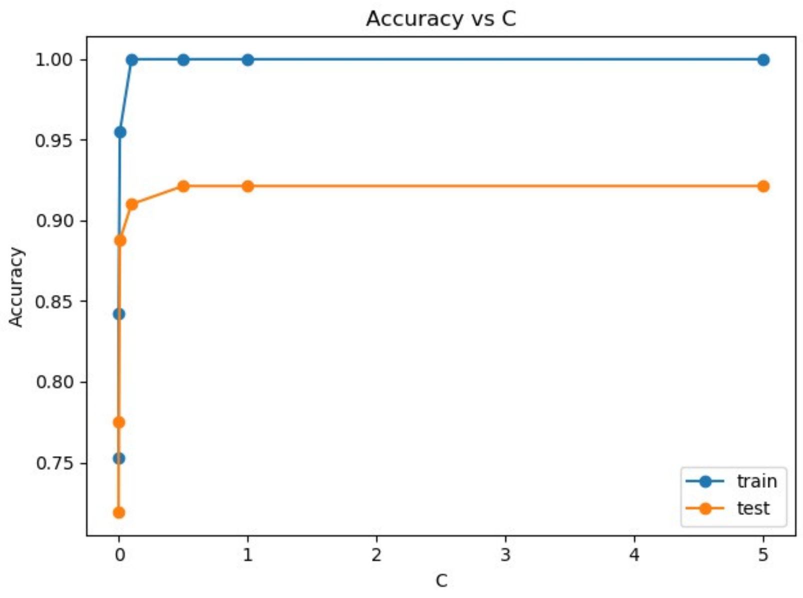
The best performance we get from GBDT model is 94.25% accuracy.

2.4 Support Vector Machines

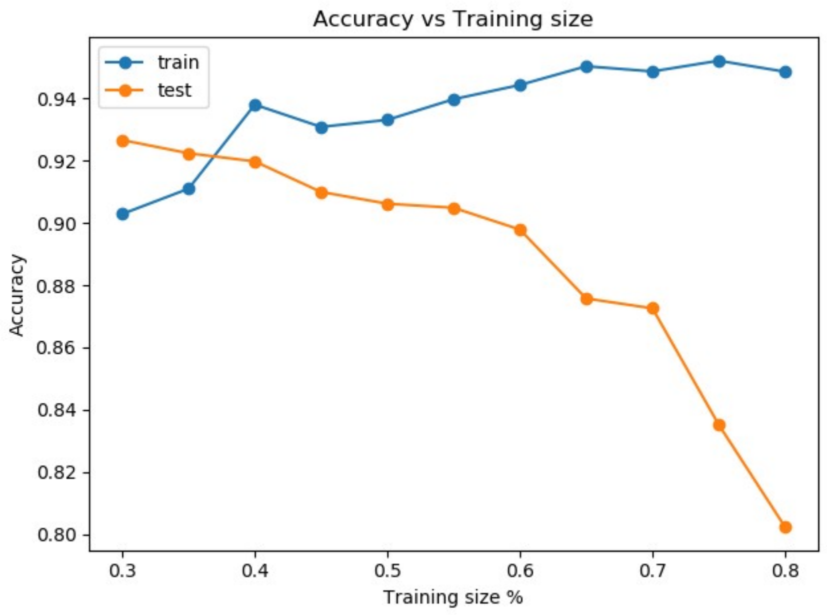
First, we will try SVM with a linear kernel.

One of the hyper parameter to tune for SVM is the Regularization parameter C. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

Below is the graph of model performance against C.



Clearly, as C increase from 0 to 5, model performance peaks at around C value of 1.



Below is the graph of SVM linear kernel model performance on various training size. For example, 0.3 means we use 30% of the full data set as training set and 70% as test set.

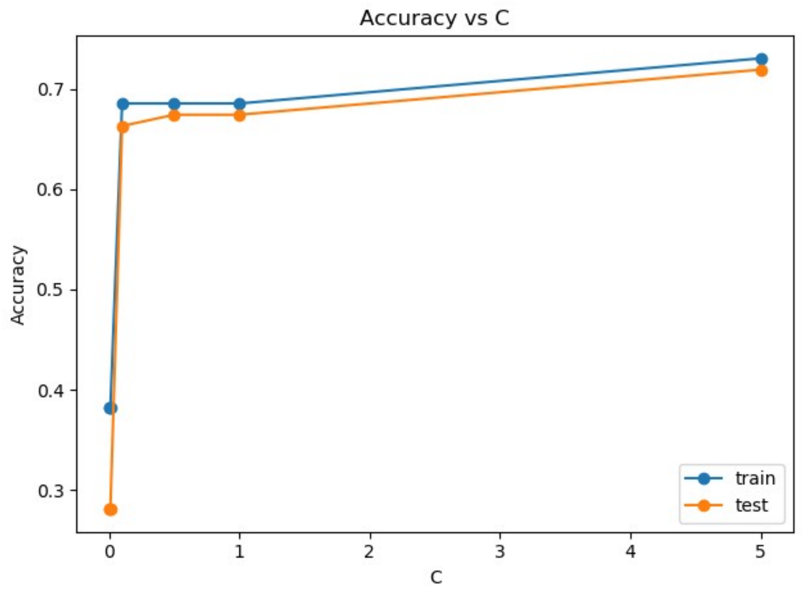
As training size increases, accuracy of training set also increases. On the contrary, accuracy of test set is decreasing. There is a clear performance gap between training set and test set when training size is larger than 70%. Hence, when training size is larger than 70%, our model is overfitting.

The best performance we get from SVM model is 92.67% accuracy.

Second, we will try SVM with a radial basis function kernel. It is computed as follow:

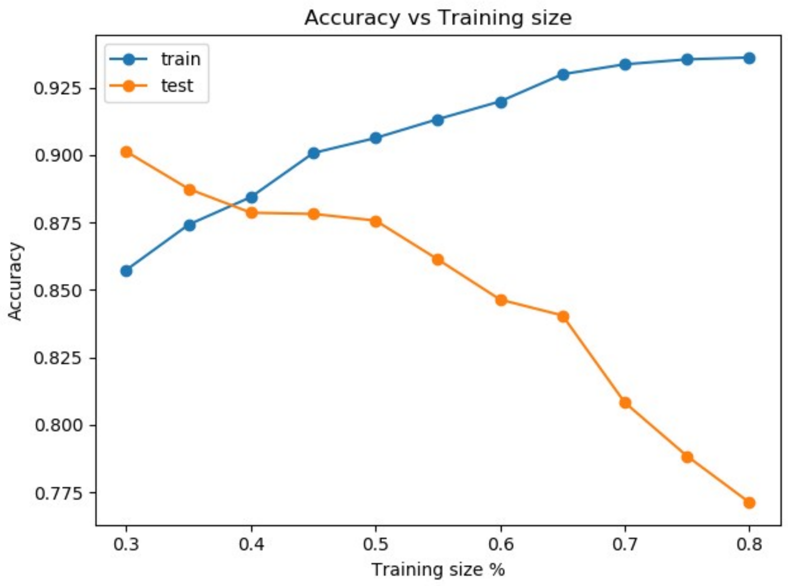
$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

Below is the graph of model performance against C.



Clearly, as C increase from 0 to 1, model performance peaks at around C value of 1.

Below is the graph of SVM linear kernel model performance on various training size. For example, 0.3 means we use 30% of the full data set as training set and 70% as test set.

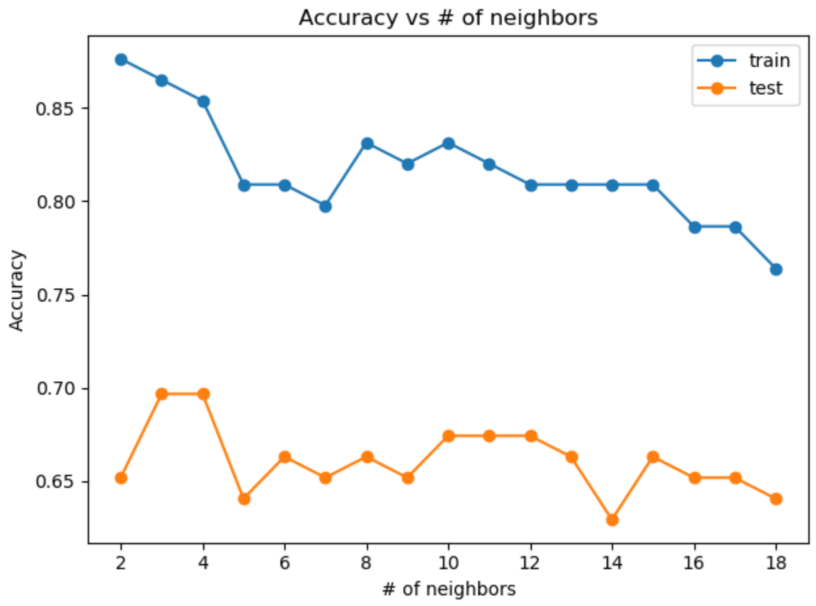


As training size increases, accuracy of training set also increases. On the contrary, accuracy of test set is decreasing. There is a clear performance gap between training set and test set when training size is larger than 70%. Hence, when training size is larger than 70%, our model is overfitting.

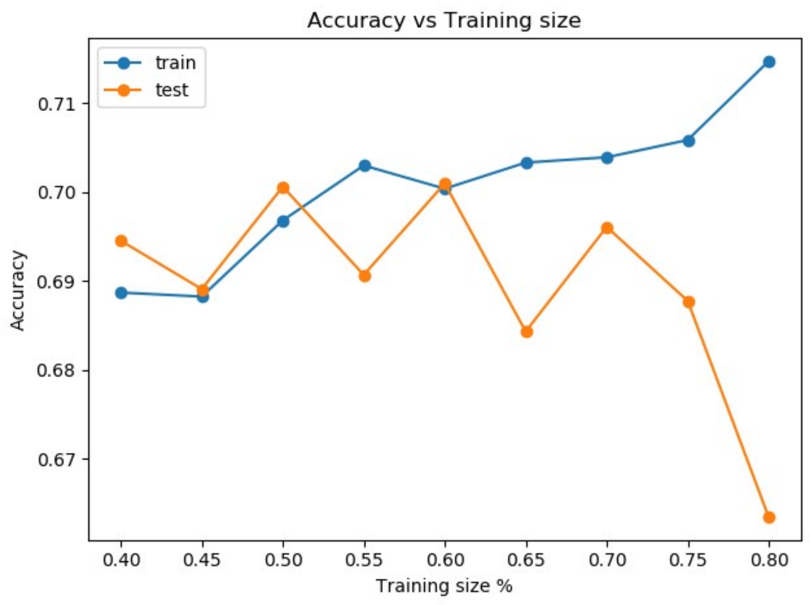
The best performance we get from SVM model is 90.13% accuracy.

2.5 k-nearest neighbors

For this method, we need to determine how many neighbors we need to make a prediction on current data point. Below is the graph of # of neighbors against accuracy.



As we can see, k=3 is the best option.



As training size increases, accuracy of training is increasing but not test set. When training size is larger than 70%, the model is highly likely overfitting.

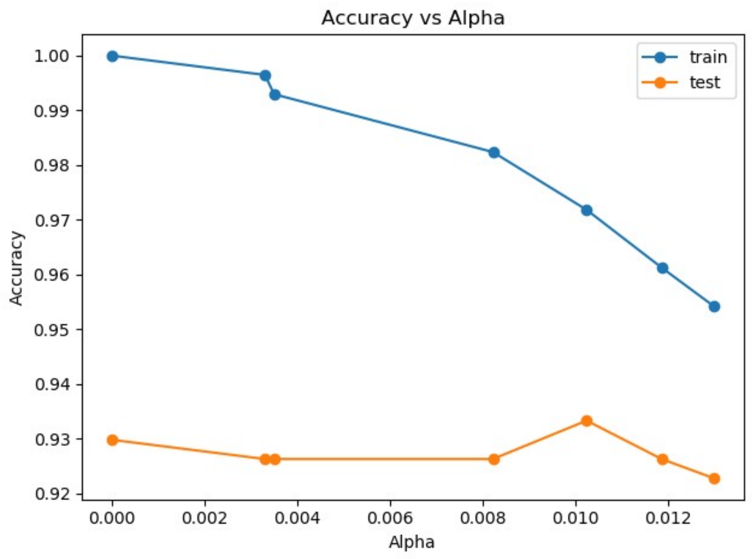
The best performance we get from knn model is 70.09% accuracy.

3. Modeling on Breast Cancer problem

3.1 Decision trees with pruning

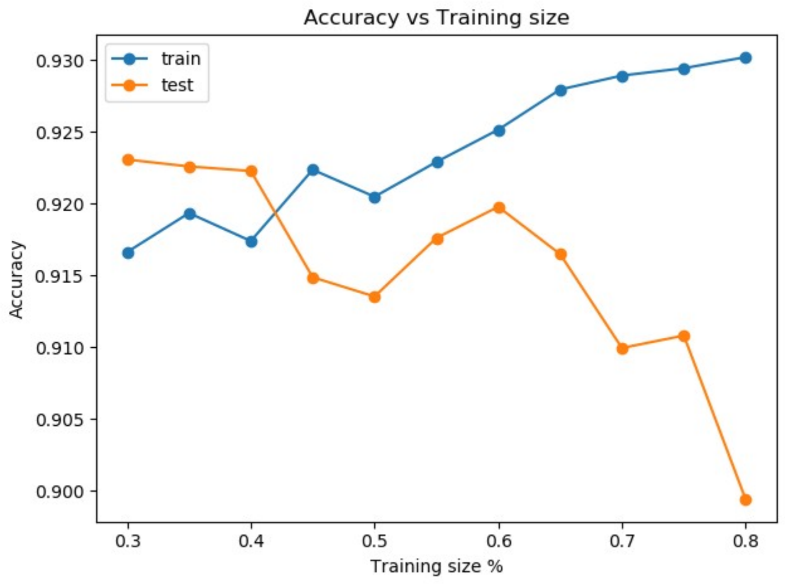
As before, we will be using the same decision tree model with same pruning method as in win recognition problem.

To fine tune the hyper parameter of alpha, here we trained models on various alpha with 50% training data size.



As we can see, alpha=0.010 we have the best test performance. Hence we will use value as our pruning method.

Below is the graph of decision tree model performance on various training size. For example, 0.3 means we use 30% of the full data set as training set and 70% as test set.

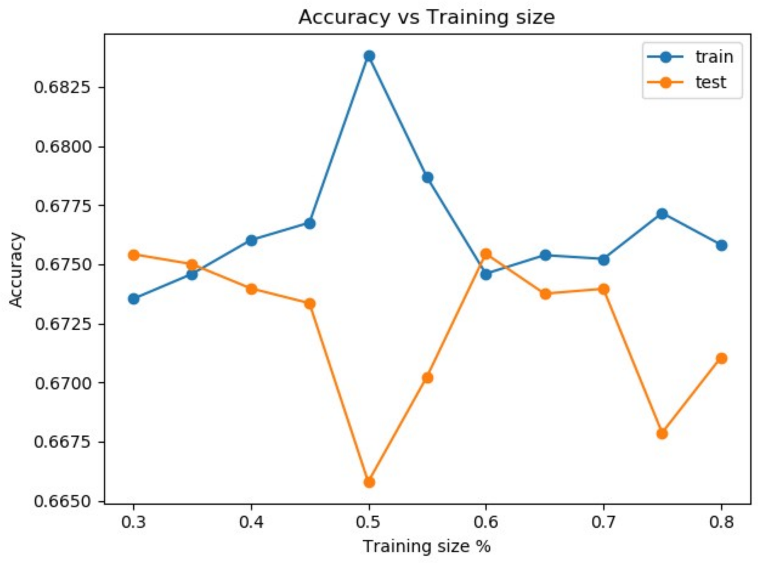


As training size increases, accuracy of training set also increases. On the contrary, accuracy of test set is decreasing. There is a clear performance gap between training set and test set when training size is larger than 70%. Hence, when training size is larger than 70%, our model is overfitting.

The best performance on test set we get from decision tree model is 92.31% accuracy.

3.2 Neural networks

Here we use a simple neural network with 5 hidden layers, each layer contains 100 units. The activation function is ReLU, learning rate is 1e-5.



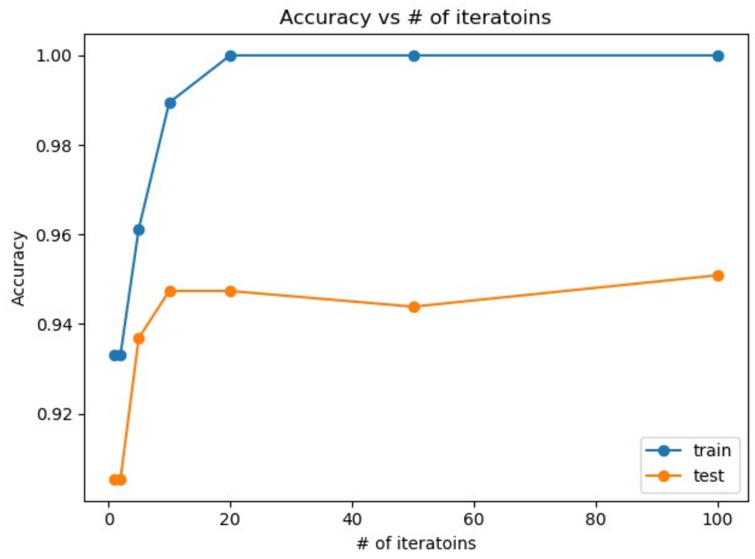
Below is the graph of Neural networks model performance on various training size. For example, 0.3 means we use 30% of the full data set as training set and 70% as test set.

There is a clear performance gap between training set and test set when training size is larger than 70%. Hence, when training size is larger than 70%, our model is overfitting.

Overall accuracy on test set is 67.54%.

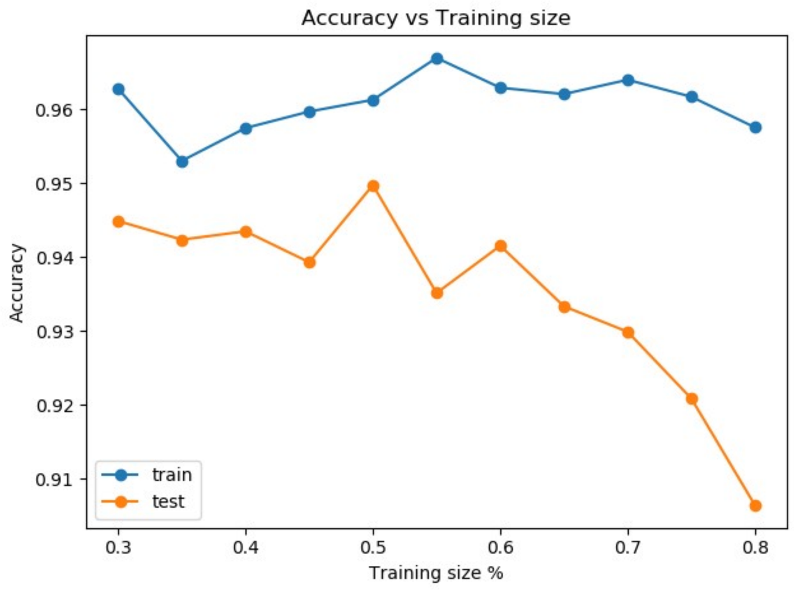
3.3 Boosting

For boosting we use the Gradient Boosting Decision Tree. One of the hyper parameters we need to fine tune is the number of estimator, or you can say number of iterations we need.



Below is the graph for relationship between # of iterations and model performance. As we can see, as # of iteration increases, model accuracy is also getting better. But it tends to saturate when # of iteration is large enough, which means we won't gain any benefit when # of iteration is larger than 100.

Below is the graph of GBDT model performance on various training size. For example, 0.3 means we use 30% of the full data set as training set and 70% as test set.



As training size increases, accuracy of training set also increases. On the contrary, accuracy of test set is decreasing. There is a clear performance gap between training set and test set when training size is larger than 70%. Hence, when training size is larger than 70%, our model is overfitting.

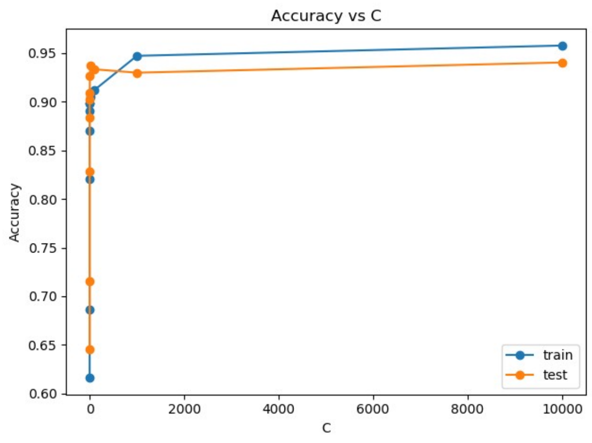
The best performance we get from GBDT model is 94.97% accuracy.

3.4 Support Vector Machines

First, we will try SVM with a polynomial kernel. It is calculated as follow:

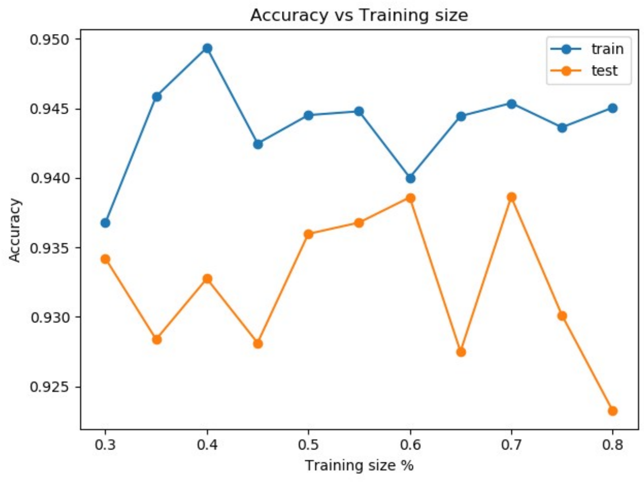
$$K(x,y) = (x^\mathsf{T} y + c)^d$$

Below is the graph of model performance against C.



Clearly, as C increase from 0 to 10000, model performance peaks at around C value of 10000.

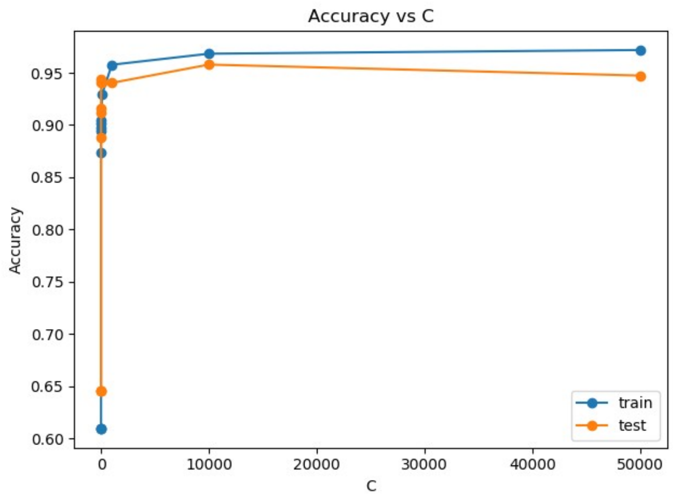
Below is the graph of SVM linear kernel model performance on various training size. For example, 0.3 means we use 30% of the full data set as training set and 70% as test set.



As training size increases, accuracy of training set also increases. On the contrary, accuracy of test set is decreasing. There is a clear performance gap between training set and test set when training size is larger than 70%. Hence, when training size is larger than 70%, our model is overfitting.

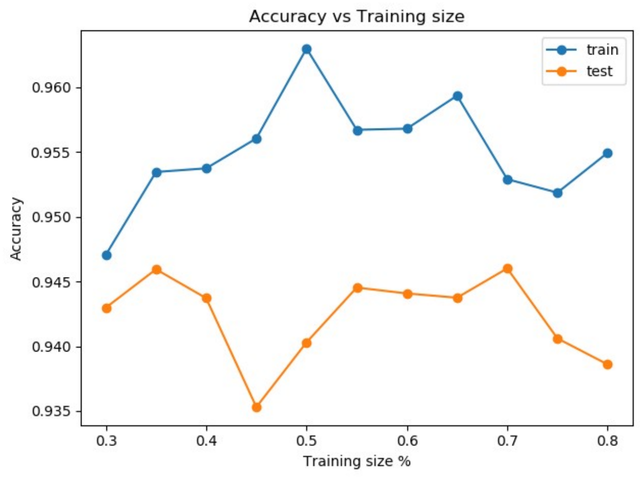
The best performance we get from SVM model is 92.67% accuracy.

Second, SVM with a radial basis function kernel. Below is graph of model performance against C.



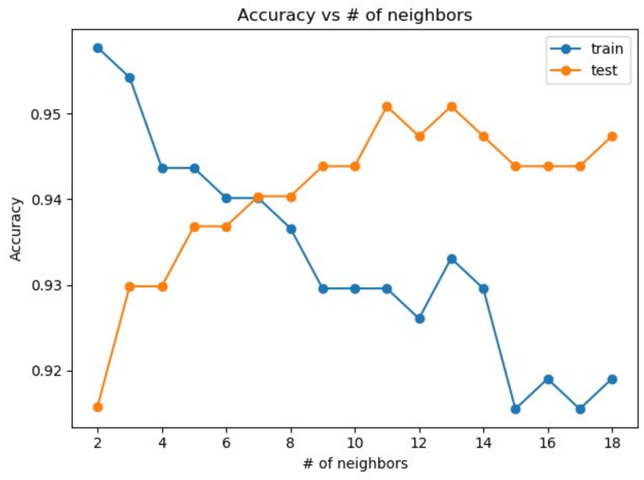
Clearly, as C increase from 0 to 1, model performance peaks at around C value of 10000.

Below is the graph of SVM linear kernel model performance on various training size. For example, 0.3 means we use 30% of the full data set as training set and 70% as test set.

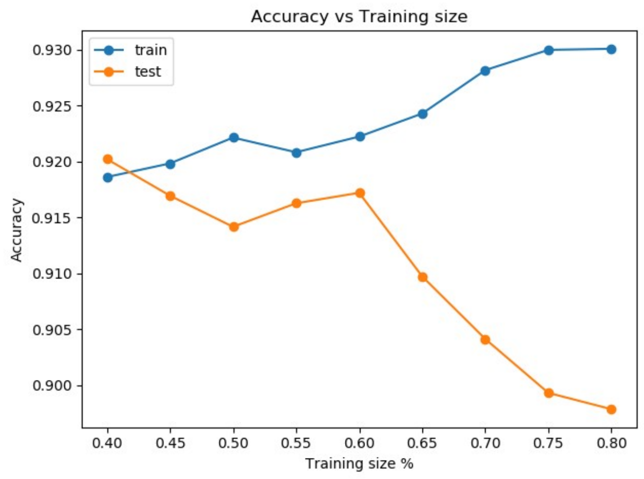


As training size increases, accuracy of both train and test set are both stable, there is no clear trend of increasing or decreasing. But as we can anticipate, performance on training set is always better than that on test set. The best performance we get from SVM model is 94.60% accuracy.

3.5 k-nearest neighbors



For this method, we need to determine how many neighbors we need to make a prediction on current data point. Below is the graph of # of neighbors against accuracy. k=11 is the best option.



As training size increases, accuracy of training is increasing but not test set. When training size is larger than 70%, the model is highly likely overfitting.

The best performance we get from knn model is 92.09% accuracy.

4. Model comparison

Below is the comparison of overall accuracy for each algorithm on each problem.

	Wine Recognition	Breast Cancer Diagnosis
Decision Tree	87.12%	92.31%
Neural Works	73.48%	67.54%
Gradient Boosting Decision Tree	94.25%	94.97%
SVM with linear kernel	92.67%	/
SVM with polynomial kernel	/	92.67%
SVM with rbf kernel	90.13%	94.60%
KNN	70.09%	92.09%

Performance wise, we can gain best performance from boosting. It is powerful and out performed the other models, because of its ensemble nature. Ensemble model tend to be more generalized, which is a great way to lower model variance and increase robustness.

As for time of training algorithm, Neural Networks, Boosting and SVM require much longer time, while simple decision tree and knn model is fast.

4.1 Decision Tree

To improve performance on Decision Tree, we need to combine it with bagging or boosting method so that it can be balanced between underfitting and overfitting.

In our project, we used Gradient Boosting to gain better performance from this type of model.

Decision tree is a powerful and interpret-able tool when we need to look at how our model is predicting the data, ratehr than a black box such as neural networks.

4.2 Neural Networks

To improve performance on Neural Networks, we need to construct a network with much more hidden layers and much more unit. But this way will be costly on training time.

As an non linear model, Neural Networks is capable of handling various tasks.

4.3 Boosting

As shown, to tune the number of iterations in boosting model is the way to improve performance.

4.4 SVM

As we have shown above, to fine tune the hyper parameter C in SVM model is the way to improve its performance.

We also tried different kernels (i.e., linear, polynomial and radial basis function), and there is no global optimal kernel, we have to test each kernel on each data set and each problem.

Linear kernel is more suitable for the Wine Recognition, and RBF kernel is better for Breast Cancer Diagnosis.

4.5 KNN

As we have shown above, to fine tune the hyper parameter k in KNN model is the way to improve its performance. KNN is tend to overfit model, hence we need to be careful with the choise of k.