Comparison of Machine Learning Algorithms

Anshu Ranjan Soham Talukdar Arvindh Seshadhri
Mithun Paul Kingshuk Mukherjee

April 28, 2016

Abstract

This work is the outcome of the project completed as part of CAP6610 Machine Learning at University of Florida in Spring 2016. In this article various Machine Learning techniques, are implemented. The implementations are done on 7 data sets from the UCI repository and experimental results are graphically compared to each other.

1 Introduction

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data. In this work various Machine Learning techniques, like Random Forest, Support Vector Machines, Deep Learning, Naive Bayes, Bagging, Boosting etc are implemented.

2 Datasets

We used 7 datasets in total from UCI Repository[4][5]:

1. Breast Cancer[2]: Each instance has one of 2 possible classes: benign or malignant. There are 10 attributes and 699 rows.
2. Forest Type Mapping[3]: It constins 27 attributes and four class labels: b, s, m and o. In total it has 699 feature vectors.
3. Optical Handwritten Recognition[4]: It has 5620 instances with 64 attributes. The class labels are the numbers 0 to 9.
4. Iris[4]: 3 classes of 50 instances each, where each class refers to a type of Iris plant. Class labels are Iris Setosa, Iris Versicolour and Iris Virginica.
5. Magic[4]: This is one of the largest dataset that we have used. It consists of 19020 features vectors, 10 attributes and 2 class labels: gamma(g) and hadron(h).

6. Tic Tac Toe[4]: 958 instances with 9 attributes and two class labels: positive and negative. It encodes the complete set of possible board configuration for the game of Tic-Tac-Toe.

7. Letter Recognition[4]: This has been the most challenging dataset because it has 26 class labels: A to Z, and had the largest number of feature vectors. It contains 16 attributes with 20000 instances.

This is standard dataset used for teaching courses of machine learning.

3 Support Vector Machines

In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier[1].

3.1 Description of the classifier

The API that we used to run SVM is libsvm from [6][7]. The programming environment that was used was Matlab.

3.1.1 Data Format

In this section we mention the format of data that is needed for libsvm to understand. Most of the data sets in the UCI repository were not in this format. Hence code was written to convert data into the right format. Libsvm uses the so called "sparse" format where zero values do not need to be stored. Hence a data with attributes 1 0 2 0 is represented as 1:1 3:2.
The format of training and testing data file is:

```
<label> <index1>:<value1> <index2>:<value2> ...
```

Each line contains an instance and is ended by a newline character. For classification, $< label >$ is an integer indicating the class label (multi-class is supported). For regression, $< label >$ is the target value which can be any real number. For one-class SVM, it's not used so can be any number. The pair $< index >:< value >$ gives a feature (attribute) $value :< index >$ is an integer starting from 1 and $< value >$ is a real number. The only exception is the precomputed kernel, where $< index >$ starts from 0; see the section of precomputed kernels. Indices must be in ASCENDING order. Labels in the testing file are only used to calculate accuracy or errors. If they are unknown, just fill the first column with any numbers.

3.2 Description of the Algorithm

In this section we describe various steps taken to implement the libsvm.

3.2.1 Data Input

First the preprocessed data is read in using the libsvmread command.
libsvmread - This command reads the data set from the given data set and converts it to two vectors, label and instances. A sample code is as given below:

```
[heart_scale_label, heart_scale_inst] = libsvmread(fullfile(dirData,
'HandWritingRecognition.combined'));
```

Once the data is read from the data sets, the corresponding percentages for testing and training is taken as input from the user. Then the data is segregated as based on this input.

3.2.2 Training

The training of the data is done using the svmtrain command provided by libsvm.
svmtrain - This command reads the data set from the given data set and converts it to two vectors, label and instances. A sample code is as given below:

```
cv = svmtrain(trainLabel, trainData, param);
```

parameters - There are a few main parameters provided as input to svmtrain.

- $-c$ This is the cost. This sets the parameter C of C-SVC, epsilon-SVR, and nu-SVR. Default value is 1.
- $-g$ This sets the gamma in kernel function. The default value is 1/num_features.
- $-q$- This says that the command svmtrain has to be run in quiet mode i.e no outputs.
- $-v$- This says that the command svmtrain has to use K-fold cross validation mode. The value of k is provided as input.

A sample code is as given below:

```
param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
```

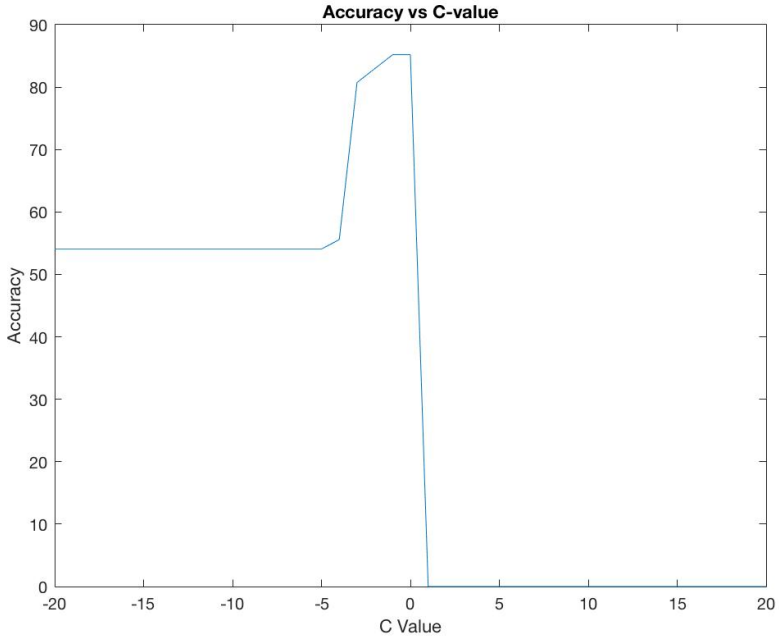


Figure 1: SVM Varying C value Phase 1

3.2.3 Multi Step Parameter Accuracy

To arrive at an optimum value of C and Gamma, a multi step cross validation methodology was used. First the C and Gamma values are varied from -n to +n (Example:-20 to +20). For each value of the combination of C and Gamma picked, the corresponding accuracy is stored. If the generated accuracy is better than the previous one, that accuracy is picked up. This is phase 1. In phase 2, the best C and Gamma values picked from Phase 1 is used to determine the boundaries of the variation. The stepsize is also reduced by half. Again the svmtrain is run to find the best C and Gamma values. The same is repeated in Phase 3 also. Thus at the end of Phase 3, micro accurate values of C and Gamma are arrived at. This become the input parameters for the Testing run.

All the 3 phases, along with the variation of accuracy, are shown in figures below. For example in 1 variation of C in Phase 1 is shown. It can be seen that C value peaks between -5 and 0. In figure 2 the range is expanded and it is shown that C value peaks around -0.5. In figure 3 the range is further magnified and picks out the best C value at -0.5. A similar exercise is done for Gamma value also.

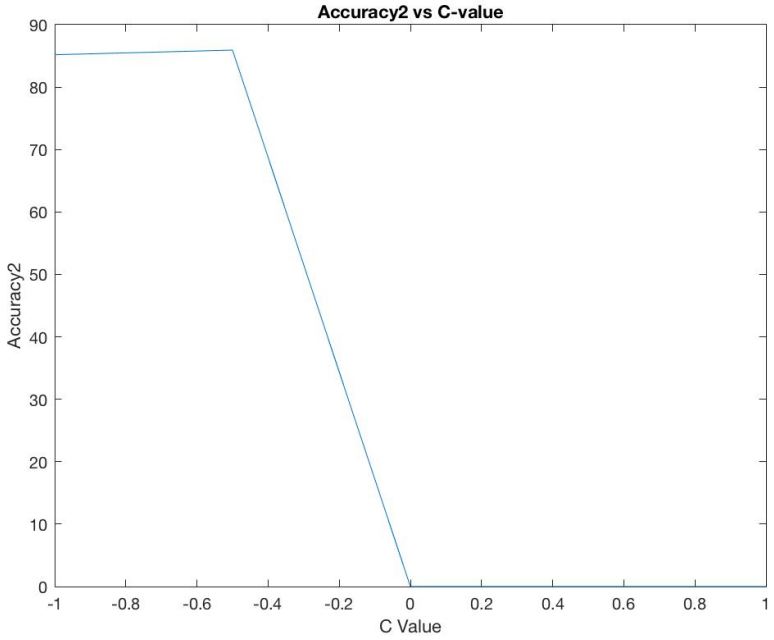


Figure 2: SVM Varying C value Phase 2

3.2.4 Testing

In the testing phase, the above picked best parameters for C and Gamma are provided as input to the svmtrain for one more time. The testdata and testlabel, created in the data section is also provided as input to the function. The model thus derived as the output of svmtrain is provided as input to the testing function svmpredict. **svmpredict** - This command takes the test data and test label as input, along with the model arrived at using the svmtrain command above. A sample code is as given below:

```
[predict_label, accuracy, prob_values] = svmpredict(testLabel,
testData, bestModel, '-b 1');
```

3.2.5 One versus Rest

For many data sets provided (Eg:Forest Mapping Dataset, Handwriting Recognition Dataset), the data was multiclass. SVM is a binary classification model. The question then was, how can we use SVM in the multi-class scenario? A simple strategy is to do binary classification 1 pair at a time. Here we used one-versus-rest approach. The original codes (svmtrain and svmpredict) from

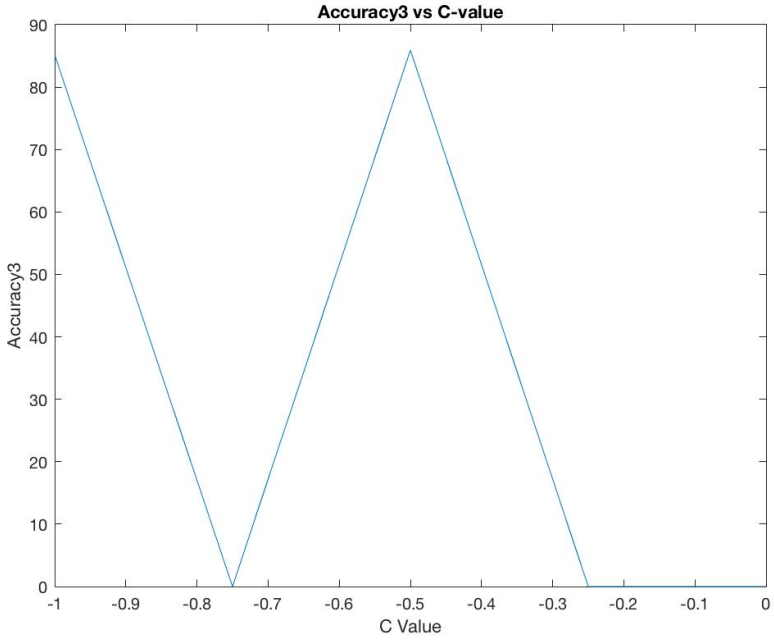


Figure 3: SVM Varying C value Phase 3

the libsvm package itself can be used to do the job by making a "wrapper code" to call the original code one pair at a time.

```
bestParam = ['-c ', num2str(2^bestLog2c), ' -g ', num2str(2^bestLog2g)];
model = ovrtrain(trainLabel, trainData, bestParam);
[predict_label, accuracy, prob_values] = ovrpredict(testLabel,
testData, model);
```

For k-fold cross validation the below command was used:

```
cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
cv = get_cv_ac(trainLabel, trainData, cmd, 3);
```

3.2.6 Plots and Dimensionality Reduction

In most of the given data sets the feature vectors were more than 2. In that case representation in a 2 dimensional plot was difficult. To overcome this, the feature vectors were reduced to 2 dimensions using a command called mdscale, provided by matlab. This performs nonmetric multidimensional scaling on the n-by-n dissimilarity matrix D, and returns Y, a configuration of n points (rows)

Training Percentage	Testing Percentage	Accuracy
10	90	96.54
20	80	97.65
30	70	98.33
40	60	98.29
50	50	98.53

Table 1: SVM Classification accuracies of Breast Cancer Data set

in p dimensions (columns). The Euclidean distances between points in Y approximate a monotonic transformation of the corresponding dissimilarities in D. By default, mdscale uses Kruskal's normalized stress1 criterion. A sample code is as given below:

```
distanceMatrix = pdist(heart_scale_inst,'euclidean');
newCoor = mdscale(distanceMatrix,2);
```

However in certain data sets like Breast cancer data set, mdscale was refusing to converge stating that the points in the configuration are co-located. This was overcome by adding a 'criterion' 'sstress'. This Squared stress normalizes with the sum of 4th powers of the inter-point distance leading to faster convergence.

Once the data was reduced to 2 dimensions, this data was plotted as a Scatter plot.

3.3 Experimental Results

Classification was done on 6 different data sets. In this section we describe each of the classification experiments in each of the data sets with the corresponding results and plots.

3.3.1 Breast Cancer Dataset

In this experiment SVM classification was done against the Wisconsin breast cancer data set [8][9][10]. This data set had 699 samples spread across 10 attributes. The attributes were: Sample code number, Clump Thickness,Uniformity of Cell Size,Uniformity of Cell Shape,Marginal Adhesion,Single Epithelial Cell Size,Bare Nuclei,Bland Chromatin,Normal Nucleoli and Mitoses. The two classes were benign(denoted by 2) and malignant(denoted by 4).

The experiment ran for 123 seconds, across 5 combinations of testing and training data percentages. The accuracies varied between 96.54% to 98.54%.Various exact accuracies can be found in Table 1 A scatter plot of the thus classified breast cancer data can be found in figure 4.Here as can be seen from various colors that the data has been classified into either of the aforementioned 2 classes. Note that unfilled circles denote the training data.

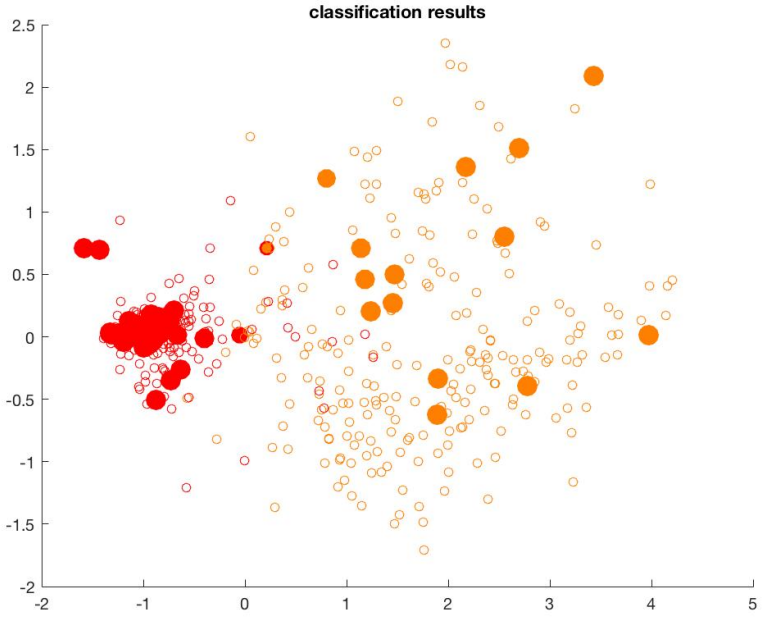


Figure 4: SVM Classified Breast Cancer Data

3.4 Forest Mapping Dataset

In this experiment SVM classification was done against the Wisconsin breast cancer data set [11][12].

This data set contains training and testing data from a remote sensing study which mapped different forest types based on their spectral characteristics at visible-to-near infrared wavelengths, using ASTER satellite imagery. The output (forest type map) can be used to identify and/or quantify the ecosystem services (e.g. carbon storage, erosion protection) provided by the forest.

This data set had 523 samples spread across 28 attributes. The attributes are as follows: b1 - b9: ASTER image bands containing spectral information in the green, red, and near infrared wavelengths for three dates (Sept. 26, 2010; March 19, 2011; May 08, 2011. pred_minus_obs_S_b1 - pred_minus_obs_S_b9: Predicted spectral values (based on spatial interpolation) minus actual spectral values for the 's' class (b1-b9). pred_minus_obs_H_b1 - pred_minus_obs_H_b9: Predicted spectral values (based on spatial interpolation) minus actual spectral values for the 'h' class (b1-b9).

This data set has 4 classes denoted as below: Class: 's' ('Sugi' forest), 'h' ('Hinoki' forest), 'd' ('Mixed deciduous' forest), 'o' ('Other' non-forest land)

The experiment ran for 23 seconds, across 5 combinations of testing and training data percentages. The accuracies varied between 94.53% to 98.43%.Var-

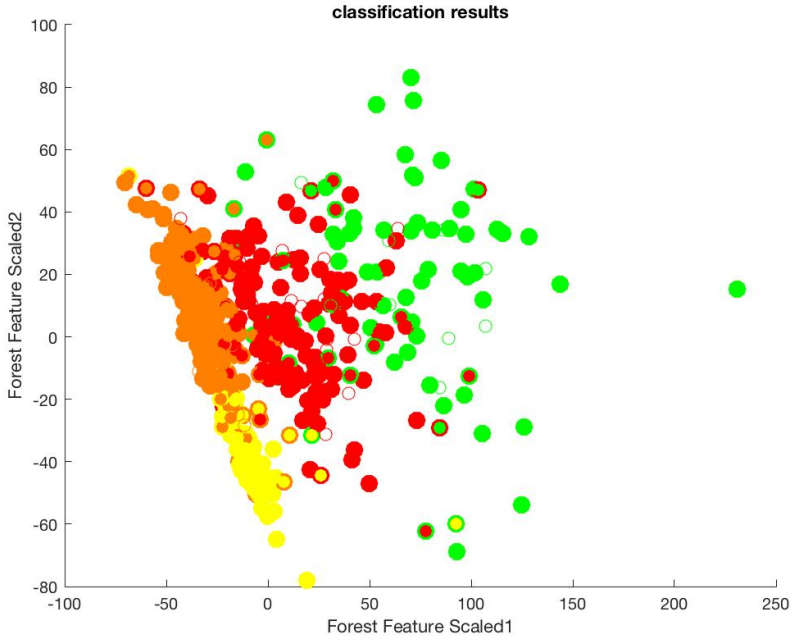


Figure 5: SVM Classified Forest Mapping Data

ious exact accuracies can be found in Table 2 A scatter plot of the thus classified forest map data can be found in figure 5. Here it can be seen from various colors that the data has been classified into either of the aforementioned 4 classes.Note that unfilled circles denote the training data.

3.5 Handwriting Recognition Dataset

In this experiment SVM classification was done against the Optical Handwriting recognition data set [13][14][15].

This data set used preprocessing programs made available by NIST to extract normalized bitmaps of handwritten digits from a preprinted form. From a total

Training Percentage	Testing Percentage	Accuracy
10	90	94.636
20	80	95.43
30	70	96.65
40	60	97.62
50	50	98.43

Table 2: SVM Classification accuracies of Forest Mapping Data set

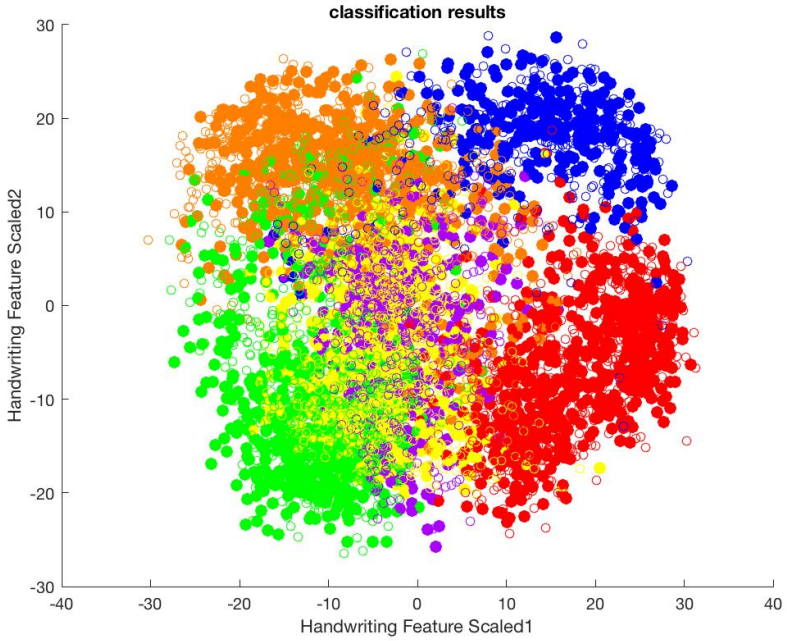


Figure 6: SVM Classified Handwriting Recognition Data

of 43 people, 30 contributed to the training set and different 13 to the test set. 32x32 bitmaps are divided into nonoverlapping blocks of 4x4 and the number of on pixels are counted in each block. This generates an input matrix of 8x8 where each element is an integer in the range 0..16. This reduces dimensionality and gives invariance to small distortions.

This data set had 5620 samples spread across 16 attributes. All input attributes are integers in the range 0..16. The last attribute is the class code 0..9 i.e there are 9 classes for classification.

The experiment ran for 6474 seconds, across 5 combinations of testing and training data percentages. The accuracies varied between 96.47% to 99.79%.Various exact accuracies can be found in Table 3 A scatter plot of the thus classified

Training Percentage	Testing Percentage	Accuracy
10	90	96.47
20	80	98.24
30	70	98.29
40	60	99.76
50	50	99.79

Table 3: SVM Classification accuracies of Handwriting recognition Data set

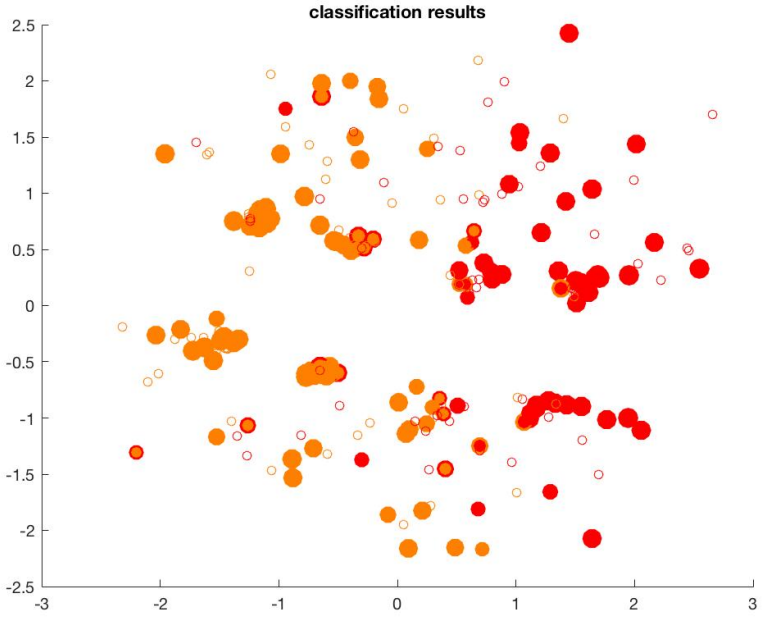


Figure 7: SVM Classified Heart Disease Data

forest map data can be found in figure 6. Here it can be seen from various colors that the data has been classified into either of the aforementioned 9 classes.Note that unfilled circles denote the training data.

3.6 Heart Disease Dataset

In this experiment SVM classification was done against the Statlog Heart Disease data set [16].

This data set had 270 samples spread across 13 attributes. Following are those attributes:age,sex,chest pain type (4 values), resting blood pressure ,serum cholestoral in mg/dl, fasting blood sugar ¿ 120 mg/dl ,resting electrocardiographic results (values 0,1,2) , maximum heart rate achieved, exercise induced angina , oldpeak = ST depression induced by exercise relative to rest ,the slope of the peak exercise ST segment , number of major vessels (0-3) colored by flourosopy ,thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

There are 2 classes for classification-prescence or abscence of heart disease.

The experiment ran for 8 seconds, across 5 combinations of testing and training data percentages. The accuracies varied between 86.06% to 89.02%.Various exact accuracies can be found in Table 4 A scatter plot of the thus classified forest map data can be found in figure 7. Here it can be seen from various colors that the data has been classified into either of the aforementioned 2 classes.Note

Training Percentage	Testing Percentage	Accuracy
10	90	86.06
20	80	86.17
30	70	87.89
40	60	87.11
50	50	89.02

Table 4: SVM Classification accuracies of Heart Disease Data set

that unfilled circles denote the training data.

4 Random Forest

Random Forest is one of the top performers in the paper mentioned in the project description [17]. We first went through online materials to get an understanding of the learning algorithm. We espcially found a youtube video very helpful [18].

4.1 Description of the classifier

Following are the details we found about Random Forest:

Description - A learning algorithm which constructs a forest of trees in a randomized manner (also called ensemble classifier). Each tree makes a prediction of label, and the label which is predicted by the maximum number of trees is selected.

Parameters - There are two main parameters:

- *mtry* - Number of attributes to be randomly selected at each node to perform the split.
- *ntree* - Total number of trees formed.

Algorithm - Here is a small description of the algorithm which runs behind the scenes when we use a random forest library. Let us say we have a dataset of 1000 rows and 20 columns. We select $ntree = 500$ and $mtry = 4$. To make each tree (500 in total) it follows these steps:

1. Create a random sample, S, of size 1000 (= size of dataset) of the dataset using replacement.
2. Select 4 (= *mtry*) attributes, A1, A2, A3 and A4 at random from 20.
3. Chose the best among these 4 attributes to divide the dataset using gini index. Let that attribute be A2
4. Split S using A2 to get two disjoint subsets of data in S, say S1 and S2.
5. Repeat steps 2 to 4 until we are down to 1 feature vector in a node

Given a new test set, each tree is traversed based upon the set of attributes chosen at each node. The label of the leaf node thus reached is returned as the

predicted label. The label predicted by maximum number of tree is given as the result.

4.2 Experimentation

4.2.1 randomForest library in R

We found a handy library for experimenting on this classifier in R called randomForest[19]. The typical usage of this package is:

$$randomForest(X, data, mtry, ntee);$$

'X' is the formula which describes the model to be fitted. 'data' is the Data frame where on which the model will be trained. 'mtry' and 'ntree' is identical to the parameters explained in previous subsection.

4.2.2 Experimental Setup

As in [17], we used a constant number of trees for each dataset: 1024, and mtry was calibrated among the values: 1, 2, 4, 6, 8, 12, 16, 20, 22, 24, 26. The range may vary depending upon the total number of attributes.

To prevent overfitting, we calibrated using two strategies:

Bootstrapping: Bootstrapping [20] takes a random sample with replacement. The random sample is the same size as the original data set. Samples may be selected more than once and each sample has a 63.2% chance of showing up at least once. Some samples wont be selected and these samples will be used to predict performance. The process is repeated multiple times (say 30 to 100).

K-fold cross validation [20]: Here, we randomly split the data into K distinct blocks of roughly equal size.

1. We leave out the first block of data and fit a model.
2. This model is used to predict the held-out block
3. We continue this process until weve predicted all K heldout blocks The final performance is based on the hold-out predictions

K is usually taken to be 5 or 10 and leave one out crossvalidation has each sample as a block

The implementation of both techniques are done using the following guideline. [20]:

Algorithm for calibration of parameters.

```
Define sets of model parameter values to evaluate;
for each parameter set do
    for each resampling iteration do
        Hold out specific samples ;
```

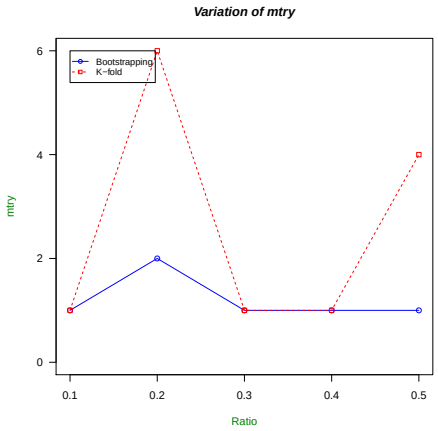


Figure 8: Variation of parameter *mtry* for Bootstrapping and K-fold validation for Breast Cancer Dataset

```
        Fit the model on the remainder;
        Predict the hold out samples;
    end
Calculate the average performance across hold out predictions
end
Determine the optimal parameter set;
```

4.2.3 Experimental Results

[t]

We attempted various measures to understand and analyze the model generated by randomForest package. As can be noted in Figure 8, there was no observable relationship between the values of *mtry* obtained from the two techniques i. e. Bootstrapping and K-fold validation. The optimal value chosen by the two varied for certain instances and were same for others.

Dataset	Running time (min)
Breast Cancer	3
Forest Type Mapping	9
Optical Handwritten	120
Iris	1
Tic Tac Toe	47
Letter	40
Magic	223

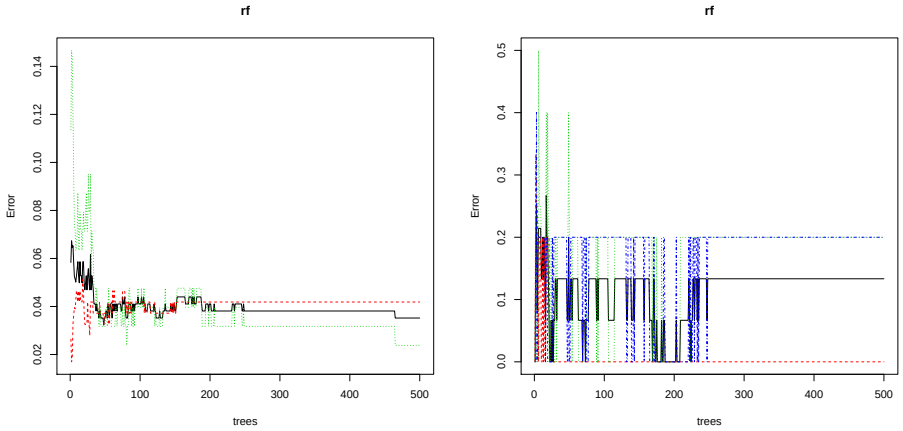
Table 5: Running time for different datasets. Note that for Letter and Optical Handwritten dataset, we run only for one ratio, 10:90 because it took too long for other ratios.

Figure 9 and 10 shows the plots between number of trees and misclassification error for all the datasets. The different dashed curves represent the error for each class, while one black curve represent the out of bag error rate. As expected, the errors were high when the number of tree were low and decreased as the number of tree grew. They generally seem to become stable after some point. Interestingly for each case a certain class tend to maintain their error rate relative to other classes and this behavior is more apparent as the number of trees grew.

Table 12 shows the time taken to run each dataset. This time is the sum of time taken by bootstrapping and k-fold validation techniques (both are in one for loop). The 'Letter' and 'Optical Handwritten' datasets had 26 and 10 classes respectively. We were able to manage to make it run only for one ratio i.e. 10:90 of training:testing. As expected, it was observed that the increase in number of classes caused an increase in running time exponentially. The other factor was the number of rows in training sets.

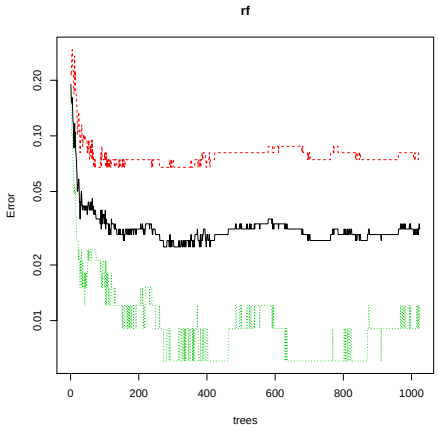
Figure 11 and 12 shows the importance of different variables for each dataset in the model built up by randomForest. We considered only one model for each dataset (Note that we built different models for different ratios and different techniques). We cross checked this with feature set of Iris, and found that the fourth column is indeed the most evident determinant of the class. We think that this feature is very useful in real life examples. This should be helpful in making feature selection. It should also give the user a good idea about the nature of dataset.

Figure 13 and 14 shows the most important reading i.e. misclassification error for each dataset. We get different accuracies for different ratio of testing and training sizes and for the two techniques i.e. bootstrapping and k fold validation. We observed there is not much difference in the measure of accuracy in the two techniques in most cases. In general, it can said that k fold validation gives slightly better results, as can be seen in the case of Forest Type Mapping. The increase in the size of training data either had no or some improvement in accuracy. This improvement is evident in the Tic Tac Toe dataset.

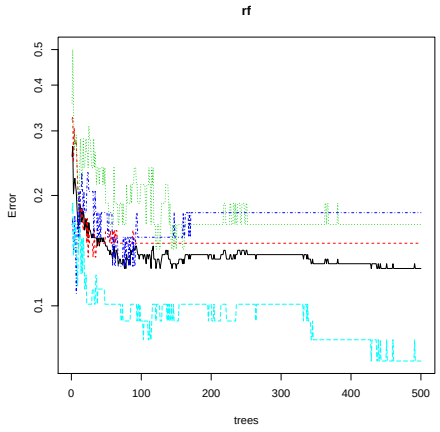


(a) Breast Cancer

(b) Iris

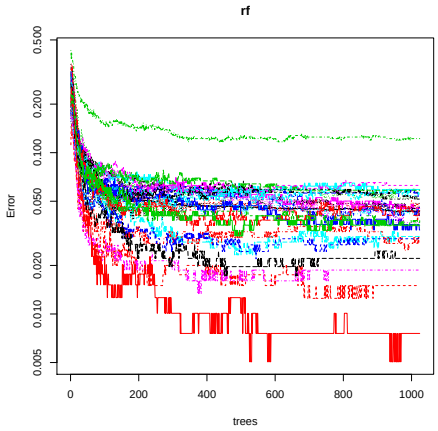


(c) Tic Tac Toe

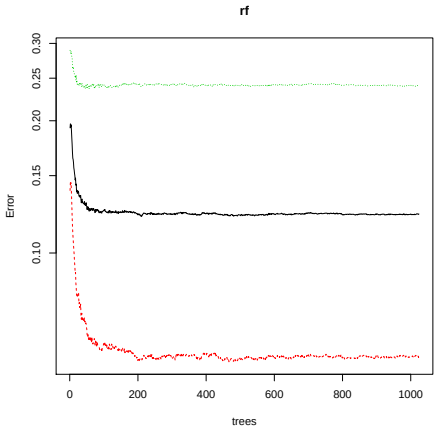


(d) Forest Type Mapping

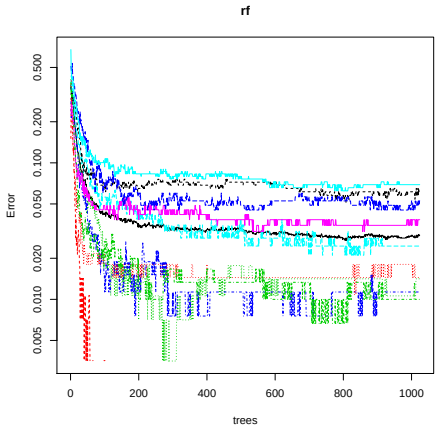
Figure 9: Plots for number of trees and misclassification errors for different datasets (contd. on next page)



(a) Letter Recognition

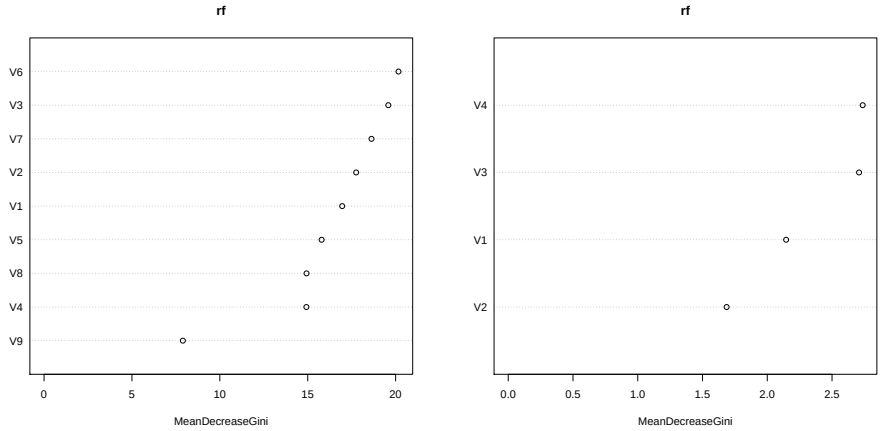


(b) Magic



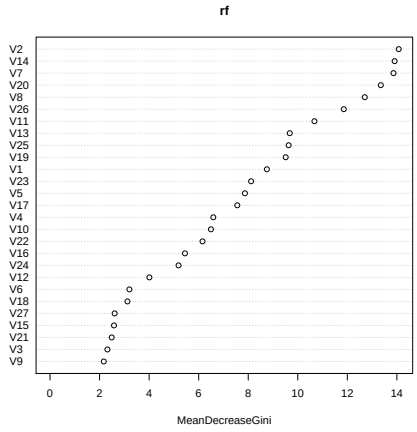
(c) Optical Handwritten

Figure 10: Plots for number of trees and misclassification errors for different datasets

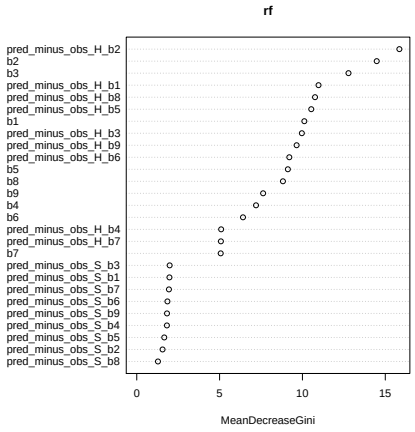


(a) Breast Cancer

(b) Iris

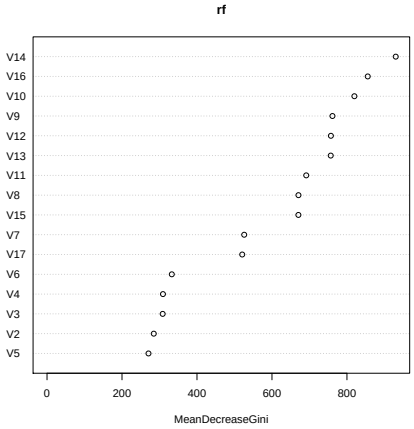


(c) Tic Tac Toe

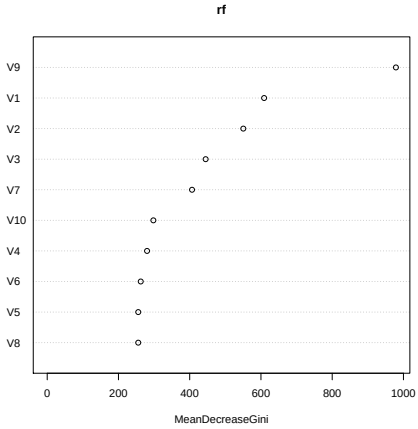


(d) Forest Type Mapping

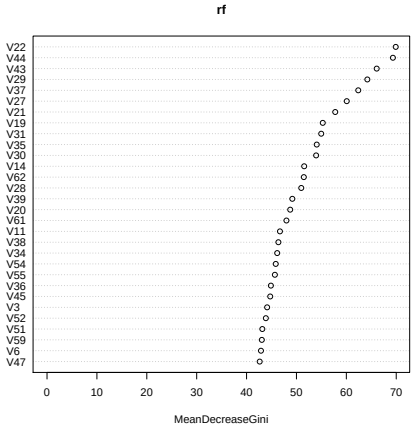
Figure 11: Plots indicating feature importance of different datasets (contd on next page)



(a) Letter Recognition

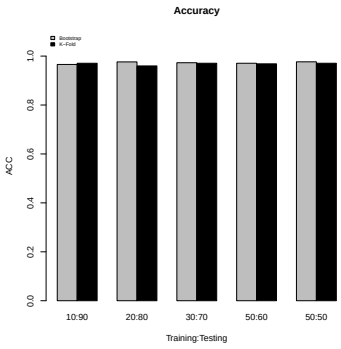


(b) Magic

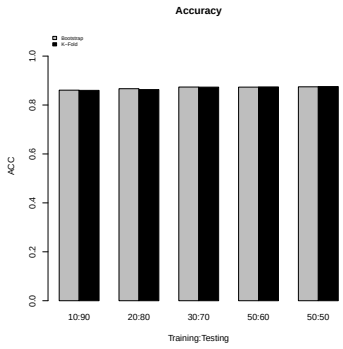


(c) Optical Handwritten

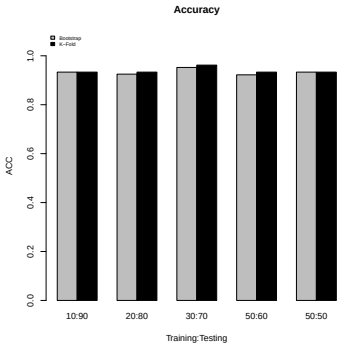
Figure 12: Plots indicating feature importance of different datasets



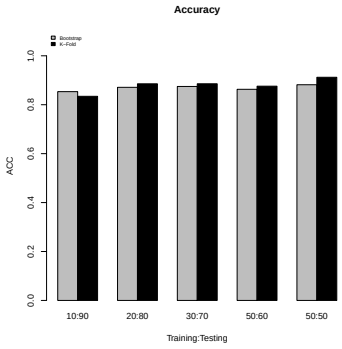
(a) Breast Cancer



(b) Magic



(c) Iris



(d) Forest Type Mapping

Figure 13: Accuracy for all datasets with varying ratio between training and test set

5 Summary of all results

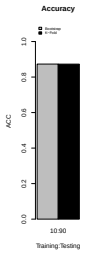
The results are shown in the table 6 to table 12.

References

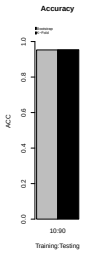
[1] Support Vector Machines. In Wikipedia. Retrieved April 17, 2016, from https://en.wikipedia.org/wiki Support_vector_machine

[2] O. L. Mangasarian and W. H. Wolberg: "Cancer diagnosis via linear programming", SIAM News, Volume 23, Number 5, September 1990, pp 1 & 18.

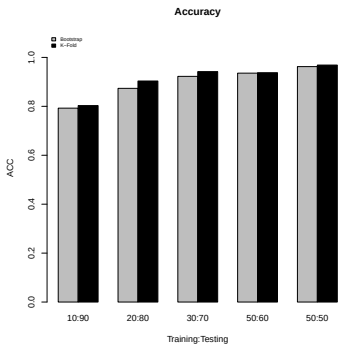
[3] Johnson, B., Tateishi, R., Xie, Z., 2012. Using geographically-weighted variables for image classification. Remote Sensing Letters, 3 (6), 491-499.



(a) Letter



(b) Optical Handwritten



(c) Tic Tac Toe

Figure 14: Accuracy for all datasets with varying ratio between training and test set

	10:90	20:80	30:70	40:60	50:50
Random Forest	97.07	95.97	97.07	96.82	97.07
SVM	98.33	97.63	98.33	98.29	98.54
Deep Learning	92.17	93.71	92.65	93.57	94.44

Table 6: Accuracy for Breast Cancer Dataset

	10:90	20:80	30:70	40:60	50:50
Random Forest	83.43	88.54	88.55	87.57	91.22
SVM	100	99.03	98.71	96.65	94.64
Deep Learning	77.07	84.01	84.47	85.03	86.26

Table 7: Accuracy for Forest Type Mapping Dataset

	10:90	20:80	30:70	40:60	50:50
Random Forest	95.27	-	-	-	
SVM		86.17	-	-	
Deep Learning	76.33	84.05	87.72	90.59	92.24

Table 8: Accuracy for Optical Handwritten Dataset

	10:90	20:80	30:70	40:60	50:50
Random Forest	93.33	92.5	95.23	92.22	93.33
SVM	-	-	-	-	-
Deep Learning	92.31	92.5	95.32	94.44	96

Table 9: Accuracy for Iris Dataset

	10:90	20:80	30:70	40:60	50:50
Random Forest	80.3	90.35	94.18	93.73	96.86
SVM	-	-	-	-	-
Deep Learning	65.81	73.01	84.64	89.73	90.6

Table 10: Accuracy for Tic Tac Toe Dataset

	10:90	20:80	30:70	40:60	50:50
Random Forest	85.99	86.3	87.31	87.39	87.53
SVM	-	-	-	-	-
Deep Learning	81.2	82.9	83.62	83.41	85.75

Table 11: Accuracy for Magic Dataset

	10:90	20:80	30:70	40:60	50:50
Random Forest	87.31	-	-	-	-
SVM	-	-	-	-	-
Deep	79.41	85.04	87.76	88.23	90.58

Table 12: Accuracy for Letter Dataset

[4] UCI Machine Learning repository. Retrieved April 17, 2016, from http://archive.ics.uci.edu/ml/datasets.html

[5] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[6] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

[7] Matlab Svm implementation. Retrieved April 17, 2016, from https://sites.google.com/site/kittipat/libsvm_matlab

[8] Breast Cancer Dataset. Retrieved April 17, 2016, from http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29

[9] Wolberg, W.H., Mangasarian, O.L. (1990). Multisurface method of pattern separation for medical diagnosis applied to breast cytology. In Proceedings of the National Academy of Sciences, 87, 9193–9196.

[10] Zhang, J. (1992). Selecting typical instances in instance-based learning. In Proceedings of the Ninth International Machine Learning Conference (pp. 470–479). Aberdeen, Scotland: Morgan Kaufmann.

[11] Forest Mapping Dataset. Retrieved April 17, 2016, from http://archive.ics.uci.edu/ml/datasets/Forest+type+mapping

[12] Johnson, B., Tateishi, R., Xie, Z., 2012. Using geographically-weighted variables for image classification. Remote Sensing Letters, 3 (6), 491-499.

[13] Handwriting Recognition Dataset. Retrieved April 17, 2016, from http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

[14] C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their Applications to Handwritten Digit Recognition, MSc Thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University.

[15] E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika. [Web Link]

[16] Statlog Heart Dataset. Retrieved April 17, 2016, from http://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29

[17] Caruana, Rich, and Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning algorithms." Proceedings of the 23rd international conference on Machine learning. ACM, 2006.

[18] Random Forests in R. Retrieved April 17, 2016, from https://www.youtube.com/watch?v=IJgR7n-VqSo

[19] Package randomForest. Retrieved April 17, 2016, from https://cran.r-project.org/web/packages/randomForest/index.htm

[20] Predictive Modeling with R and the caret Package. Retrieved April 17, 2016, from https://www.r-project.org/nosvn/conferences/useR-2013/Tutorials/kuhn/user_caret_2up.pdf