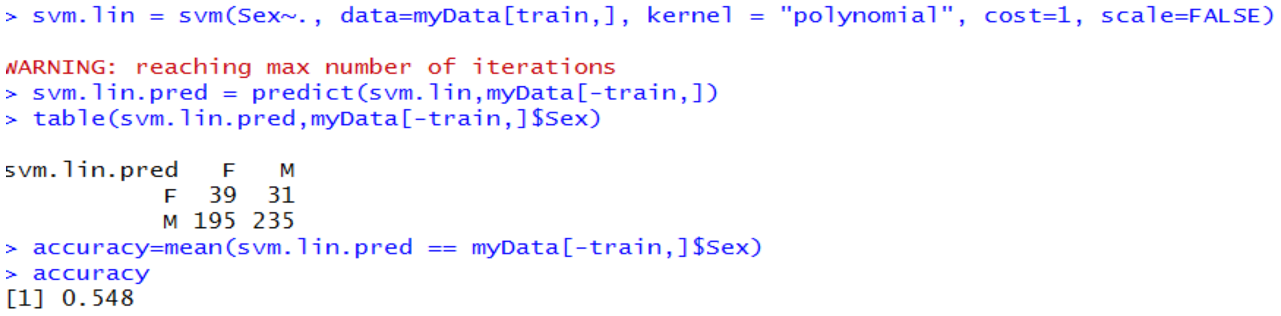
## Part I: (Abalone Dataset)

**Purpose:** *"This assignment helped us study the application of Bagging, Logical Regression, Random Forests and Support Vector Machines on the Abalone Dataset."*

**Dataset(s):** **Abalone Data Set**

**Approach:**

- ***Method used: Support Vector Machines***
- *We unclass the training data and pick 2000 out of 2500 entries for training the model before testing on the remaining 335 entries.*
- *After analyzing the training dataset, we find that it is useful to convert the first column (Sex) from characters to factors and unclassing the data.*
- *We try running logistic regression, random forests, boosting and SVM on the dataset among which we got the best accuracy using SVM with linear kernel with a test accuracy of 54.8%. Below is the confusion matrix on application of SVM.*



```
> svm.lin = svm(Sex~., data=myData[train,], kernel = "polynomial", cost=1, scale=FALSE)

WARNING: reaching max number of iterations
> svm.lin.pred = predict(svm.lin,myData[-train,])
> table(svm.lin.pred,myData[-train,]$Sex)

svm.lin.pred   F   M
           F  39  31
           M 195 235
> accuracy=mean(svm.lin.pred == myData[-train,]$Sex)
> accuracy
[1] 0.548
```

- *Finally, we use this model to predict the labels of Abalone_Test that have 335 test rows and 8 columns that exclude the response variable column (Sex).*
- *Using SVM delivered a comparable accuracy of around 54.8%. However training Neural Networks on the dataset proved difficult and either resulted in a lower accuracy (about 49.5%) or the weights failed to converge. Also, random forest showed impressive results of about 50-53% accuracy.*
- *For predicting the final "Sex" column in the "Abalone_Test" dataset we use the model obtained from SVM and append the predicted values into the csv file.*
- *In summary, SVM, Random forests and Neural Networks had superior results in predicting values for the Sex column in case of this dataset.*

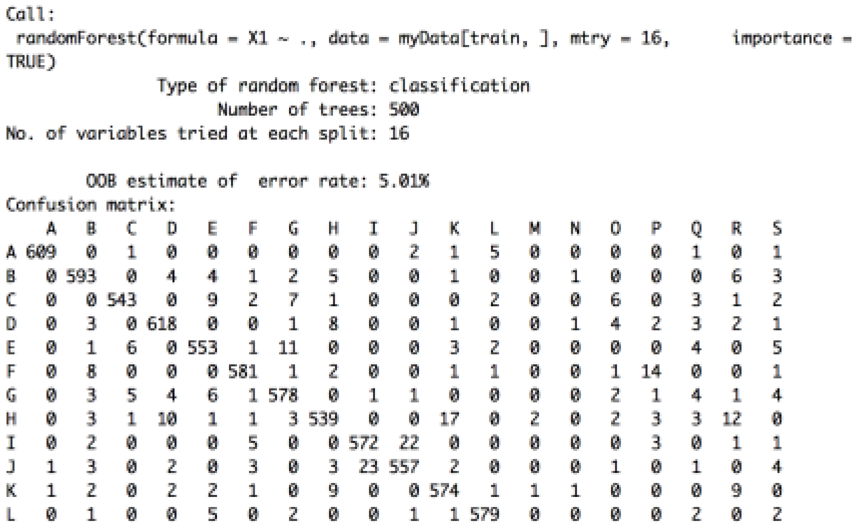
## Part II: (Letters Dataset)

**Purpose:** *"Our goal is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. This assignment helped us study the application of Bagging, Random Forests and Support Vector Machines on the Letter Recognition Dataset."*

**Dataset(s):** Letters Data Set

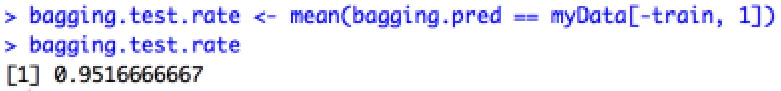
**Approach:**

- ***Method used: Random Forests***
- *There are 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15.*
- *We unclass the training data and pick 16000 out of 19000 entries for training the model before testing on the remaining 3000 entries.*
- *We try running random forests, boosting and SVM on the dataset among which we got the best accuracy using Bagging with a test accuracy of 95.16%. Below is the confusion matrix.*
  *bagging.fit <- randomForest(X1~.,myData[train,],mtry=16,importance=TRUE)*
  *bagging.fit*



```
Call:
 randomForest(formula = X1 ~ ., data = myData[train, ], mtry = 16,      importance =
TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 16

        OOB estimate of  error rate: 5.01%
Confusion matrix:
    A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S
A 609   0   1   0   0   0   0   0   0   2   1   5   0   0   0   0   1   0   1
B   0 593   0   4   4   1   2   5   0   0   1   0   0   1   0   0   0   6   3
C   0   0 543   0   9   2   7   1   0   0   0   2   0   0   6   0   3   1   2
D   0   3   0 618   0   0   1   8   0   0   1   0   0   1   4   2   3   2   1
E   0   1   6   0 553   1  11   0   0   0   3   2   0   0   0   0   4   0   5
F   0   8   0   0   0 581   1   2   0   0   1   1   0   0   1  14   0   0   1
G   0   3   5   4   6   1 578   0   1   1   0   0   0   0   2   1   4   1   4
H   0   3   1  10   1   1   3 539   0   0  17   0   2   0   2   3   3  12   0
I   0   2   0   0   0   5   0   0 572  22   0   0   0   0   0   3   0   1   1
J   1   3   0   2   0   3   0   3  23 557   2   0   0   0   1   0   1   0   4
K   1   2   0   2   2   1   0   9   0   0 574   1   1   1   0   0   0   9   0
L   0   1   0   0   5   0   2   0   0   1   1 579   0   0   0   0   2   0   2
```



```
> bagging.test.rate <- mean(bagging.pred == myData[-train, 1])
> bagging.test.rate
[1] 0.9516666667
```

- *We predict the results of the Test file which contains 1000 entries, based on the model and attach the predicted values.*
- *Using SVM delivered a comparable accuracy of around 94%. However training Neural Networks on the dataset proved difficult and either resulted in a lower accuracy or the weights failed to converge. One way to eliminate this problem was to set a very high stepmax(around 1e6) which allowed more time for the weights to converge.*

- *In summary, Bagging, Random Forests and SVM proved to be superior methods for predicting the letter based on the 16 predictors for this dataset compared to Neural Networks.*

## Part III: Website Phishing

**Purpose:** *“This assignment helped us study the application of Bagging and Neural Networks on the Website Phishing Dataset.”*

**Dataset(s):** **Website Phishing Dataset**

**Approach:**

- ***Method used: Neural Networks***
- *After analyzing the training dataset, we find that it is useful to convert the dataset to factors and unclassing the data.*
- *We try running Random forests on the dataset after selecting 1700 rows for training and testing on the remaining 300 entries.*

  ```
  myData = as.data.frame(unclass(Website_Phishing_Train))
  bagging.fit <- randomForest(factor(Result)~.,myData[train,],mtry=16,importance=TRUE)
  ```

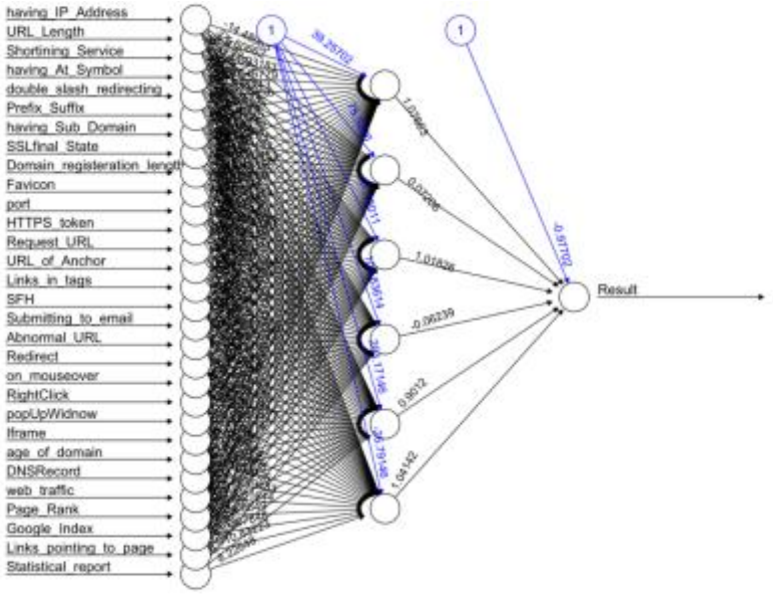
- *We obtain accuracy of 98.33% on the test data of 300 entries.*

  ```
  bagging.pred <- predict(bagging.fit,myData[-train,])
  bagging.test.rate <- mean(bagging.pred == myData[-train, 31])
  ```

  *The accuracy of this model is :*

  ```
  [1] 0.9833333
  ```

- *Finally, we use this model to predict the labels of Website_Phishing_Test.*
- *We also try Neural Networks on the dataset which performs fairly well resulting in around 94% accuracy. Below is a screenshot of the nn.fit model using 6 hidden layers using which we obtain the final converged weights for each predictor.*



- *After training the model, the predicted values need to be rounded to values 0 or 1 based on the following code snippet:*

  ```
  nn.pred$net.result[nn.pred$net.result > .5] = 1
  nn.pred$net.result[nn.pred$net.result <= .5] = 0
  ```

- *For predicting the final "Result" column in the "Website_Phishing_Test" dataset we use the model obtained earlier and append the predicted values into the csv file.*
- *In summary, Random forests, SVM and Neural Networks provided a fairly accurate predicted values for the Result column in case of this dataset.*