# **Data Classification on Zoo Dataset**

By

Soumya Yamsani (700693403)

Ashwini Bhoomi Brahmanand(700693129)

Vineesha Paladugu(700691238)

**2. Contribution to the project by each group member:**

Soumya Yamsani:

Performed all the required classification techniques on Zoo Data set. Rechecked the solutions with teammates. Contribution in designing the project report and presentation .

Ashwini Bhoomi Brahmanand:

Performed all the required classification techniques on Zoo Data set. Rechecked the solutions with teammates. Contribution in designing the project report and presentation.

Vineesha Paladugu:

Performed Decision Tree techniques for Holdout, RandomForest and Bagging on Zoo Data Set. Rechecked the solutions with teammates

**3. INTRODUCTION:**

This is the smallest dataset belongs to Richard Forsyth . He donated this dataset to the UCI repository in 1990. This is a simple dataset containing 101 animals from a zoo. There are 18 different variables which describes the animals. Among them most of the attributes are Boolean valued attributes. This dataset does not contain any missing values and do not have uniformly distributed data. The main aim of this dataset is to classify a wide range of animals into 7 classes by the help of attributes that are related to animal characteristics There are 7 Class Types by which the animals are differentiated ,they are: Mammal, Bird, Reptile, Fish, Amphibian, Bug and Invertebrate.

The major purpose for this dataset is based on the variables, we can predict the classification of the animals. It dataset is perfect for the new Datamining Learners.

**Information about Zoo DataSet:**  
Class: Set of animals  
  
1 -- (41) aardvark, antelope, bear, boar, buffalo, calf, cavy, cheetah, deer, dolphin, elephant, fruitbat, giraffe, girl, goat, gorilla, hamster, hare, leopard, lion, lynx, mink, mole, mongoose, opossum, oryx, platypus, polecat, pony, porpoise, puma, pussycat, raccoon, reindeer, seal, sealion, squirrel, vampire, vole, wallaby, wolf  
2 -- (20) chicken, crow, dove, duck, flamingo, gull, hawk, kiwi, lark, ostrich, parakeet, penguin, pheasant, rhea, skimmer, skua, sparrow, swan, vulture, wren  
3 -- (5) pitviper, seasnake, slowworm, tortoise, tuatara  
4 -- (13) bass, carp, catfish, chub, dogfish, haddock, herring, pike, piranha, seahorse, sole, stingray, tuna  
5 -- (4) frog, frog, newt, toad  
6 -- (8) flea, gnat, honeybee, housefly, ladybird, moth, termite, wasp  
7 -- (10) clam, crab, crayfish, lobster, octopus, scorpion, seawasp, slug, starfish, worm.

Attribute Information:

|  |  |
| --- | --- |
| 1. animal name | Unique for each instance |
| 2. hair | Boolean |
| 3.feathers | Boolean |
| 4. eggs | Boolean |
| 5. milk | Boolean |
| 6. airborne | Boolean |
| 7. aquatic | Boolean |
| 8. predator | Boolean |
| 9. toothed | Boolean |
| 10. backbone | Boolean |
| 11. breathes | Boolean |
| 12. venomous | Boolean |
| 13. fins | Boolean |
| 14. legs | Numeric (set of values: {0,2,4,5,6,8}) |
| 15. tail | Boolean |
| 16. domestic | Boolean |
| 17. catsize | Boolean |
| 18. type | Numeric (integer values in range [1,7]) |

**4. DATA PREPROCESSING:**

* Initially when we read the zoo data set we found that there are 2 instances of "frog“ . So we deleted the duplicate before performing any classification techniques on it.
* We dropped the column “animal name” as it didn’t add value.
* We discretized the class label in our dataset which is “type” from ranges[0-7] to Boolean “Yes” and “No”.

**5. HOLDOUT METHOD:**

Holdout is one of the methods to evaluate the classification accuracy technique. In this method initially the zoo dataset is randomly partitioned into two different independent sets. Training set for model construction and Test set for accuracy estimation. And also, the technique of Random sampling is implemented in this method, in which the hold out method is repeated on the same dataset for k number of times, where accuracy is the average of all accuracies which are obtained.

Library used: library(tree)

Hold-Out Implementation:

A screenshot of a social media post

Description automatically generated

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a social media post

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a social media post

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

Test Error rates:

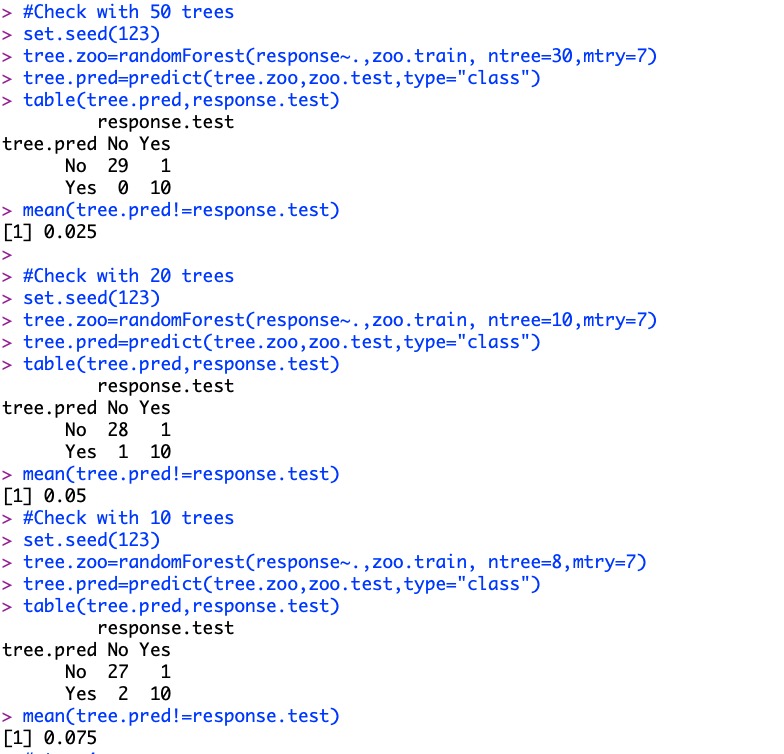
|  |  |  |
| --- | --- | --- |
|  | Tree Size | Test Error Rate |
| **Before random sampling** | - | 0.025 |
| **After Random sampling** | 4 | 0.025 |
| **After Random sampling** | 3 | 0.025 |
| **After Random sampling** | 2 | 0.15 |

**6. BAGGING:**

Even Bagging is also one of the methods to evaluate the classification accuracy technique. It is the bootstrap aggregation technique. In this the zoo dataset is divided into few number of tuples at each iteration on which a bootstrap technique is implemented. From each training set we get a classifier model where models return its class prediction.

Library used: library(randomForest)

Bagging Implementation:



Test Error rates:

|  |  |
| --- | --- |
| parameters | Test Error Rate |
| ntree: 30, mtry:7 | 0.025 |
| ntree: 10, mtry:7 | 0.05 |
| ntree: 8, mtry:7 | 0.075 |

**7. RANDOM FOREST:**

In this technique each classifier is a decision tree classifier. This decision tree is generated only when at each node the attributes are randomly selected to determine the split. And as the part of the classification process each tree will vote and class which is the most popular will be returned.

Library used: library(tree), library(randomForest)

Random Forest Implementation:

A screenshot of a social media post

Description automatically generated

Test Error rates:

|  |  |
| --- | --- |
| parameters | Test Error Rate |
| ntree: 50, mtry:4 | 0.025 |
| ntree: 50, mtry:5 | 0.05 |
| ntree: 50, mtry:3 | 0.025 |

**8. BOOSTING:**

Boosting is a technique in which once we get the classifier model, the weights are updated to allow the subsequent classifier model which has more priority for training tuples that were misclassified by initial classifier model.

Library used: library(tree), library(gbm)

Boosting Implementation:

A screenshot of a social media post

Description automatically generated

Test Error rates:

|  |  |
| --- | --- |
| parameters | Test Error Rate |
| ntree: 40 | 0.075 |
| ntree: 15 | 0.075 |
| ntree: 10 | 0.175 |

**9. NAÏVE BAYES CLASSIFIER**

The e1071 library contains implementations for different classification methods including

Support Vector Machine and Naive Bayes classification.

Library used: e1071

Naïve Bayes Implementation:

A screenshot of a cell phone

Description automatically generated

**A screenshot of a cell phone

Description automatically generated**

Test Error rates:

|  |  |
| --- | --- |
| parameters | Test Error Rate |
| Before Sampling | 0.05 |
| After Sampling | 0.06666667 |

**10. SUPPORT VECTOR MACHINE USING LINEAR KERNEL WITH DIFFERENT COSTS**

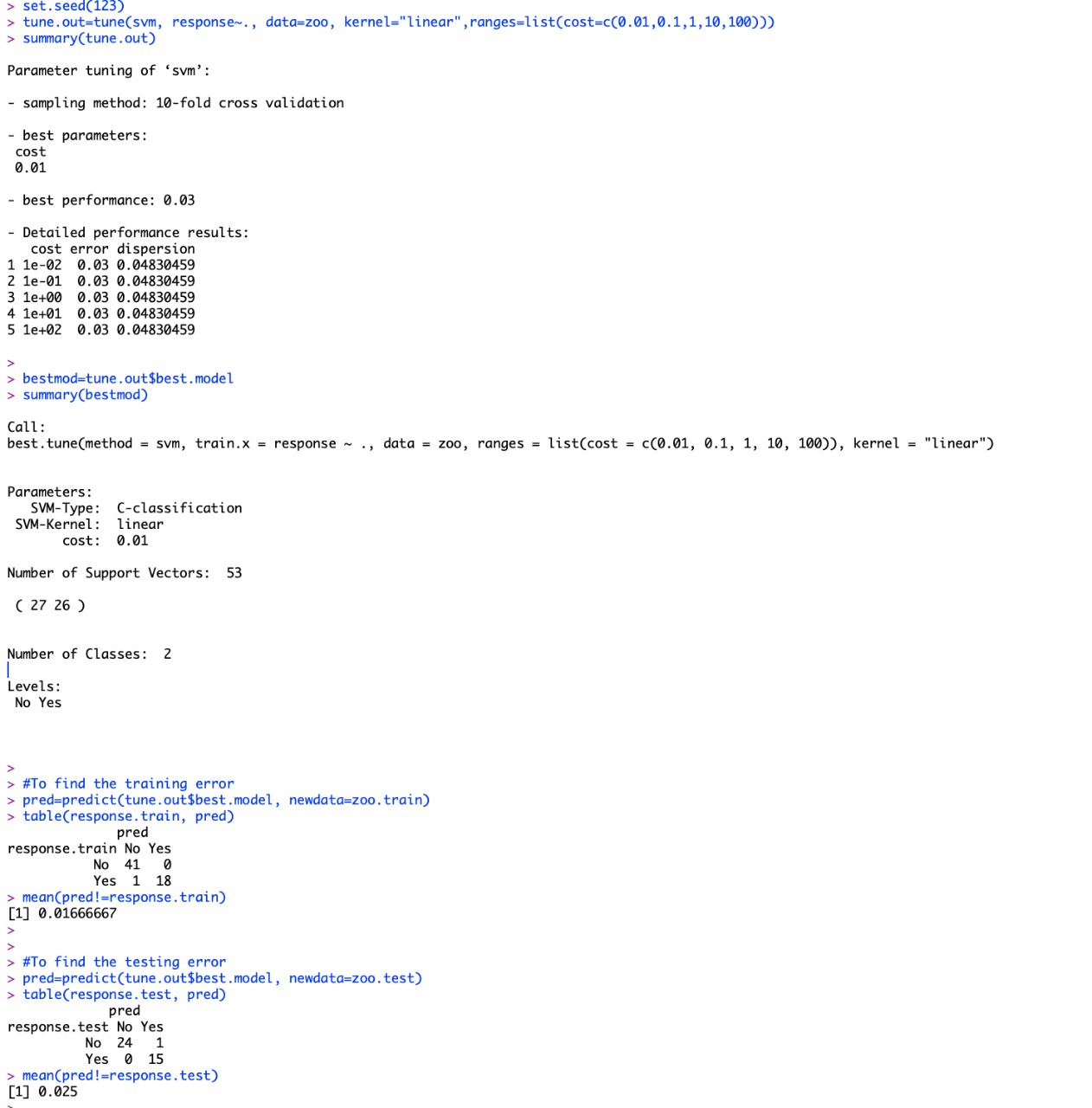
The main objective of the linear Support vector machine is to maximize the margin to feasible extent. The Decision boundary will only depend on the Support vectors. It will not change only when the dataset has same support vectors. . Initially we find the optimal cost for the zoo dataset then we implement this technique using linear kernel and the optimal cost. In the next step we find the train and test error rates.

Library used: library(e1071)

SVM Implementation for Linear Kernel:

A screenshot of a social media post

Description automatically generated



Test Error rates:

|  |  |  |
| --- | --- | --- |
|  | Train Error Rate | Test Error Rate |
| **SVM - Linear Kernel**  **(cost=c(0.01,0.1,1,10,100)**  **Best cost:0.01**  **Train set – 60%**  **Test set – 40%** | 0.01666 | 0.025 |

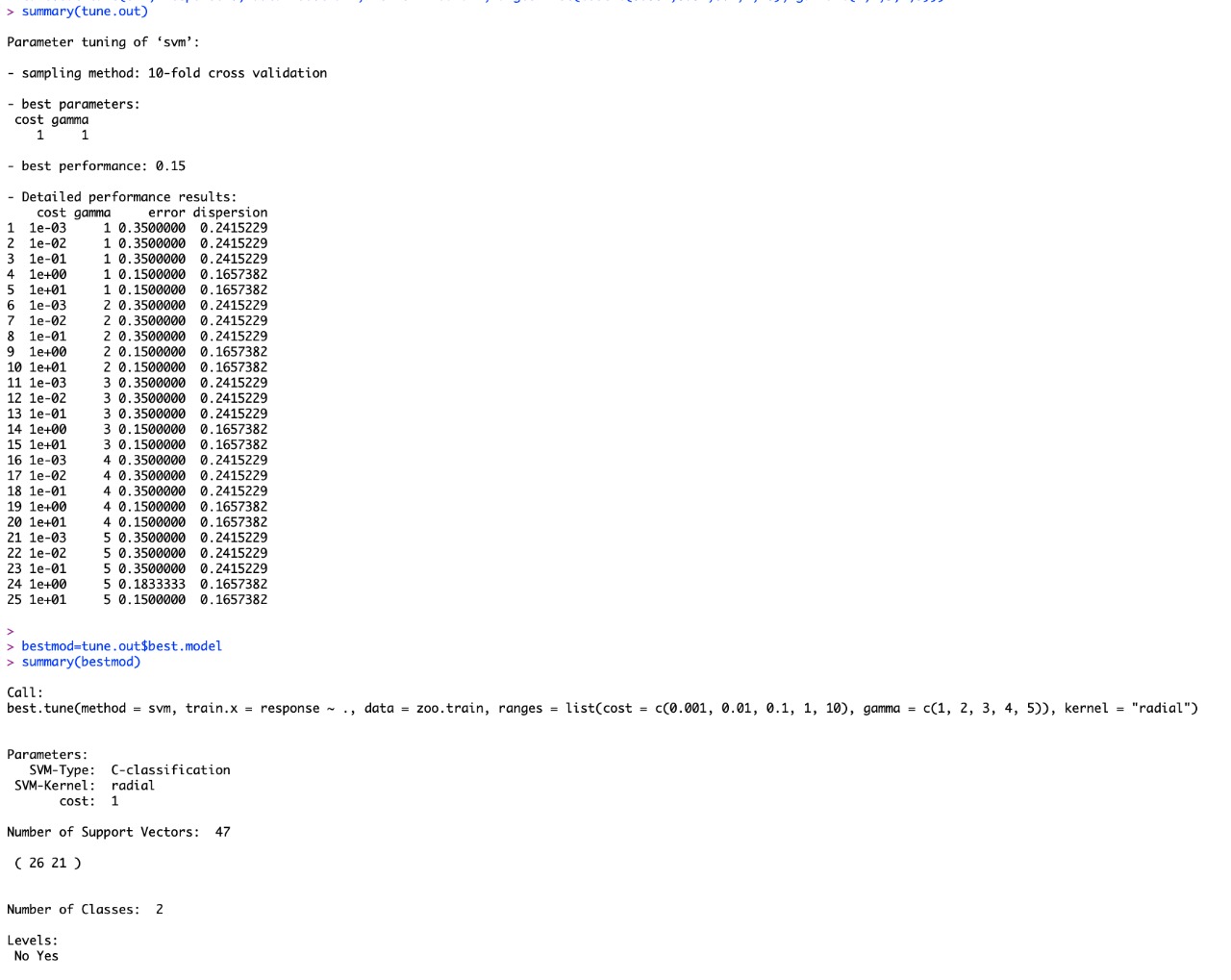
**11. SUPPORT VECTOR MACHINE USING RADIAL KERNEL WITH DIFFERENT COSTS AND GAMMAS**

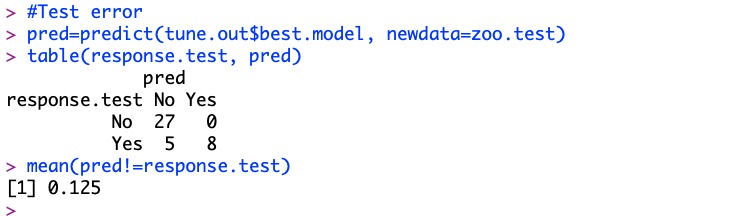
Similarly, if the decision boundary is not linear this scenario comes to existence. The main logic here is to transform data into higher dimensional space. Initially we find the optimal cost for the zoo dataset then we implement this technique using radial kernel, the optimal cost and different gamma values. In the next step we find the train and test error rates.

Library used: library(e1071)

SVM Implementation for Radial Kernel:







Test Error rates:

|  |  |
| --- | --- |
|  | Test Error Rate |
| **SVM - Radial Kernel**  **(cost=c(0.001,0.01,0.1,1,10)**  **(Gamma=** **c(1, 2, 3,4,5))**  **Best cost: 1**  **Best gamma=1**  **Train set – 60%**  **Test set – 40%** | 0.125 |

**12. SUPPORT VECTOR MACHINE USING POLYNOMIAL KERNEL WITH DIFFERENT COSTS AND GAMMAS**

Similarly, if the decision boundary is not linear this scenario comes to existence. The main logic here is to transform data into higher dimensional space. Initially we find the optimal cost for the zoo dataset thenwe implement this technique using polynomial kernel, the optimal cost and different gamma values. In the next step we find the train and test error rates.

Library used: library(e1071)

SVM Implementation for Polynomial Kernel:





Test Error rates:

|  |  |
| --- | --- |
|  | Test Error Rate |
| **SVM - Polynomial Kernel**  **cost=c(0.001,0.01,0.1,1,10),**  **degree=2**  **gamma=c(0.2,0.5,1,2,3)**  **Best gamma=0.2**  **Best Cost Value: 1**  **Train set – 60%**  **Test set – 40%** | 0.025 |

**13. COMPARISION OF MULTIPLE CLASSIFICATION TECHNIQUES**

|  |  |  |
| --- | --- | --- |
| **Name of the Classifier** | **Parameters** | **Testing error** |
| **Hold-out Method**  **Train set – 60%**  **Test set – 40%** | **Tree Size: 4**  **Tree Size: 3**  **Tree Size: 2** | **0.025**  **0.025**  **0.15** |
| **Bagging**  **Train set – 60%**  **Test set – 40%** | **ntree: 30, mtry=7**  **ntree: 10, mtry=7**  **ntree: 8, mtry=7** | **0.025**  **0.05**  **0.075** |
| **Random Forest**  **Train set – 60%**  **Test set – 40%** | **ntree:50, Mtry: 4**  **ntree:50, Mtry: 5**  **ntree:50, Mtry: 3** | **0.025**  **0.05**  **0.025** |
| **Boosting**  **Train set – 60%**  **Test set – 40%** | **n.trees: 40**  **n.trees: 15**  **n.trees: 10** | **0.075**  **0.075**  **0.175** |
| **Naïve Bayes**  **Train set – 70%**  **Test set – 30%** | **-** | **Before sampling:**  **0.05**  **After Sampling:**  **0.066667** |
| **SVM - Linear Kernel**  **(cost=c(0.01,0.1,1,10,100)**  **Train set – 60%**  **Test set – 40%** | **Best cost value: 0.01** | **0.025** |
| **SVM – Radial Linear Kernel**  **Cost=(0.001, 0.01,0.1,1,10), gamma=c(1,2,3,4,5)**  **Best gamma=1**  **Train set – 60%**  **Test set – 40%** | **Best cost value:**  **1** | **0.125** |
| **SVM - Polynomial Kernel**  **cost=c(0.001,0.01,0.1,1,10),**  **degree=2**  **gamma=c(0.2,0.5,1,2,3)**  **Best gamma=0.2**  **Train set – 60%**  **Test set – 40%** | **Best cost value: 1** | **0.025** |

**14. POTENTIAL PERFORMANCE ISSUES AND POSSIBLE FUTURE STUDY**

* For each characteristic, we analyzed how the results vary whenever test mode is changed
* Decision Tree: The model performed better with training set and test set in 60-40%.
  + Bagging had issues with larger tree size and performed better with small tree size of 50 and less
  + Random Forest performed worst with mtry value = 5 and Boosting with tree size =40
* Naïve Bayes performance was weak with 60-40% and improved on 70-30%
* SVM with Linear Kernel have performed best with cost value 0.01.
* SVM with Radial Kernel have performed best with cost value 1 & gamma value 1
* SVM with Polynomial Kernel have performed best with cost value 1 & gamma value 0.2

**15. CONCLUSION**

This project studied the performance of a variety of classification techniques on a zoo dataset, by varying the training size.

By examining different classification techniques on zoo dataset, we observed that Random Forest and SVM classification methods has less Test Error rate when compared to other classification methods.

**16. REFERENCES**

[1]. Richard Forsyth -- Donor: Richard S. Forsyth 8 Grosvenor Avenue Mapperley Park Nottingham NG3 5DX 0602-621676

<https://data.world/uci/zoo/workspace/project-summary?agentid=uci&datasetid=zoo>

[2]. <https://www.kaggle.com/uciml/zoo-animal-classification>

[[3]. https://rdrr.io/cran/VDA/man/zoo.html](file:///C:\Users\mogul\Downloads\%5b3%5d.%20https:\rdrr.io\cran\VDA\man\zoo.html)

[[4]. http://tunedit.org/repo/UCI/zoo.arff](file:///C:\Users\mogul\Downloads\%5b4%5d.%20http:\tunedit.org\repo\UCI\zoo.arff)

[[5]. https://sci2s.ugr.es/keel/dataset.php?cod=69](file:///C:\Users\mogul\Downloads\%5b5%5d.%20https:\sci2s.ugr.es\keel\dataset.php%3fcod=69)

**17. APPENDIX OF R CODES:**

**library(tree)**

**#To load the dataset**

**# Please use header=F**

**zoo=read.csv("C:/Users/group/SEM-3/Data Mining/Project/zoo.data", header=F)**

**#Variable headers**

**names(zoo)=c("animal", "hair", "feathers", "eggs", "milk", "airborne","aquatic", "predator", "toothed", "backbone", "breathes", "venomous",**

**"fins", "legs", "tail", "domestic", "size", "type")**

**#To attach dataset**

**attach(zoo)**

**#To check missing values**

**zoo=na.omit(zoo)**

**#Deleting duplicate data**

**zoo=zoo[-c(26),]**

**rownames(zoo)=NULL**

**#Dropping the column animal name**

**zoo$animal=NULL**

**#Discretization of class label**

**response=ifelse(zoo$type<=3,"No","Yes")**

**zoo=data.frame(zoo,response)**

**zoo=zoo[,-17]**

**#To check row and column dimensions**

**dim(zoo)**

**#------------------------Hold-out Method------------------------------------**

**#To create a decision tree with response as the class label based on all other attributes**

**tree.zoo=tree(response~.,zoo)**

**#Summary of the created tree**

**summary(tree.zoo)**

**#To display the tree structures and node labels**

**plot(tree.zoo)**

**text(tree.zoo,pretty=0)**

**tree.zoo**

**#Testing the model using predict function**

**tree.pred=predict(tree.zoo,type="class")**

**tree.pred**

**#Confusion matrix**

**table(tree.pred,response)**

**#Correct prediction rate**

**mean(tree.pred==response)**

**#Error prediction rate**

**mean(tree.pred!=response)**

**#Random select a sample of 60 observations of the data set as a training set and the rest**

**#of the data set as a test set**

**set.seed(123)**

**train=sample(1:nrow(zoo), 60)**

**zoo.train=zoo[train,]**

**zoo.test=zoo[-train,]**

**response.train=response[train]**

**response.test=response[-train]**

**tree.zoo=tree(response~.,zoo.train)**

**#Train error**

**tree.pred=predict(tree.zoo,zoo.train,type="class")**

**table(tree.pred,response.train)**

**mean(tree.pred!=response.train)**

**#Test error**

**tree.pred=predict(tree.zoo,zoo.test,type="class")**

**table(tree.pred,response.test)**

**mean(tree.pred!=response.test)**

**#Cross validation to understand optimal level of tree complexity**

**set.seed(123)**

**cv.zoo=cv.tree(tree.zoo,FUN=prune.misclass)**

**cv.zoo**

**plot(cv.zoo$size ,cv.zoo$dev ,type="b")**

**#Check the tree with size 4**

**set.seed(123)**

**prune.zoo=prune.misclass(tree.zoo,best=4)**

**plot(prune.zoo)**

**text(prune.zoo,pretty=0)**

**tree.pred=predict(prune.zoo,zoo.test,type="class")**

**table(tree.pred,response.test)**

**mean(tree.pred!=response.test)**

**#Check the tree with size 3**

**set.seed(123)**

**prune.zoo=prune.misclass(tree.zoo,best=3)**

**plot(prune.zoo)**

**text(prune.zoo,pretty=0)**

**tree.pred=predict(prune.zoo,zoo.test,type="class")**

**table(tree.pred,response.test)**

**mean(tree.pred!=response.test)**

**#Check the tree with size 2**

**set.seed(123)**

**prune.zoo=prune.misclass(tree.zoo,best=2)**

**plot(prune.zoo)**

**text(prune.zoo,pretty=0)**

**tree.pred=predict(prune.zoo,zoo.test,type="class")**

**table(tree.pred,response.test)**

**mean(tree.pred!=response.test)**

**#------------------------Bagging Method------------------------------------**

**##Decision tree using bagging**

**#Package includes randomForest() to perform both bagging and random forest**

**library(randomForest)**

**#bagging - special case of a random forest with m = p**

**#ntree indicates the number of trees are generated by bagging**

**#mtry indicates the number of variables are used at each split.**

**#Check with 50 trees**

**set.seed(123)**

**tree.zoo=randomForest(response~.,zoo.train, ntree=30,mtry=7)**

**tree.pred=predict(tree.zoo,zoo.test,type="class")**

**table(tree.pred,response.test)**

**mean(tree.pred!=response.test)**

**#Check with 20 trees**

**set.seed(123)**

**tree.zoo=randomForest(response~.,zoo.train, ntree=10,mtry=7)**

**tree.pred=predict(tree.zoo,zoo.test,type="class")**

**table(tree.pred,response.test)**

**mean(tree.pred!=response.test)**

**#Check with 10 trees**

**set.seed(123)**

**tree.zoo=randomForest(response~.,zoo.train, ntree=8,mtry=7)**

**tree.pred=predict(tree.zoo,zoo.test,type="class")**

**table(tree.pred,response.test)**

**mean(tree.pred!=response.test)**

**#------------------------RandomForest Method------------------------------------**

**#Decision tree using RandomForest**

**#By default, randomForest() uses about sqrt(p) variables when building a random forest of classification trees. sqrt(16)=4**

**#mtry=4**

**set.seed(123)**

**tree.zoo=randomForest(response~.,zoo.train, ntree=50, mtry=4)**

**tree.pred=predict(tree.zoo,zoo.test,type="class")**

**table(tree.pred,response.test)**

**mean(tree.pred!=response.test)**

**#mtry=5**

**set.seed(123)**

**tree.zoo=randomForest(response~.,zoo.train, ntree=50, mtry=5)**

**tree.pred=predict(tree.zoo,zoo.test,type="class")**

**table(tree.pred,response.test)**

**mean(tree.pred!=response.test)**

**#mtry=3**

**set.seed(123)**

**tree.zoo=randomForest(response~.,zoo.train, ntree=50, mtry=3)**

**tree.pred=predict(tree.zoo,zoo.test,type="class")**

**table(tree.pred,response.test)**

**mean(tree.pred!=response.test)**

**#------------------------Boosting Method------------------------------------**

**##Decision tree using boosting**

**library(tree)**

**library(gbm)**

**zoo=read.csv("/Users/ashwinibhoomi/Desktop/SEM-3/Data Mining/Project/zoo.data",header=F)**

**names(zoo)=c("animal", "hair", "feathers", "eggs", "milk", "airborne","aquatic", "predator", "toothed", "backbone", "breathes", "venomous",**

**"fins", "legs", "tail", "domestic", "size", "type")**

**attach(zoo)**

**zoo=na.omit(zoo)**

**#Deleting duplicate data**

**zoo=zoo[-c(26),]**

**rownames(zoo)=NULL**

**#Dropping the column animal name**

**zoo$animal=NULL**

**#Discretization of class label**

**class.label=ifelse(zoo$type<=3,"No","Yes")**

**zoo=data.frame(zoo,class.label)**

**zoo=zoo[,-17]**

**#For binary classification, the response variable should be 0 or 1 if using Bernoulli distribution.**

**zoo$class.label=ifelse(zoo$class.label=="Yes",1,0)**

**zoo$class.label**

**set.seed(123)**

**train=sample(1:nrow(zoo),60)**

**zoo.train=zoo[train,]**

**zoo.test=zoo[-train,]**

**class.label.test=class.label[-train]**

**#Check with 40 trees, n.trees =no. of trees**

**set.seed(123)**

**tree.zoo=gbm(class.label~., zoo.train, distribution="bernoulli",n.trees=40)**

**tree.pred.prob=predict(tree.zoo, zoo.test, n.trees=40, type="response")**

**tree.pred=ifelse(tree.pred.prob>0.5, "Yes", "No")**

**table(class.label.test, tree.pred)**

**mean(tree.pred!=class.label.test)**

**#Check with 15 trees**

**set.seed(123)**

**tree.zoo=gbm(class.label~., zoo.train, distribution="bernoulli",n.trees=15)**

**tree.pred.prob=predict(tree.zoo, zoo.test, n.trees=15, type="response")**

**tree.pred=ifelse(tree.pred.prob>0.5, "Yes", "No")**

**table(class.label.test, tree.pred)**

**mean(tree.pred!=class.label.test)**

**#Check with 10 trees**

**set.seed(123)**

**tree.zoo=gbm(class.label~., zoo.train, distribution="bernoulli",n.trees=10)**

**tree.pred.prob=predict(tree.zoo, zoo.test, n.trees=10, type="response")**

**tree.pred=ifelse(tree.pred.prob>0.5, "Yes", "No")**

**table(class.label.test, tree.pred)**

**mean(tree.pred!=class.label.test)**

**#------------------------Naïve Bayes Method------------------------------------**

**##Naïve Bayes classifier**

**#The e1071 library contains implementations for Naive Bayes classification and Support Vector Machine.**

**library(e1071)**

**zoo=read.csv("/Users/ashwinibhoomi/Desktop/SEM-3/Data Mining/Project/zoo.data",header=F)**

**names(zoo)=c("animal", "hair", "feathers", "eggs", "milk", "airborne","aquatic", "predator", "toothed", "backbone", "breathes", "venomous",**

**"fins", "legs", "tail", "domestic", "size", "type")**

**zoo=zoo[-c(26),]**

**rownames(zoo)=NULL**

**zoo$animal=NULL**

**response=ifelse(zoo$type<=3,"No","Yes")**

**zoo=data.frame(zoo,response)**

**zoo=zoo[,-17]**

**dim(zoo)**

**attach(zoo)**

**#Fitting the Naive Bayes model**

**Naive\_Bayes\_Model=naiveBayes(response~., zoo)**

**#Understanding the model summary**

**Naive\_Bayes\_Model**

**#Predicting dataset**

**NB\_Predictions=predict(Naive\_Bayes\_Model,zoo)**

**#Confusion matrix for accuracy**

**table(NB\_Predictions,response)**

**mean(NB\_Predictions!=response)**

**#Train and test set**

**set.seed(123)**

**train=sample(1:nrow(zoo),70)**

**trainSet=zoo[train,]**

**testSet=zoo[-train,]**

**test.label=response[-train]**

**NB\_2=naiveBayes(response~.,trainSet)**

**NB\_Predictions\_2=predict(NB\_2,testSet)**

**table(NB\_Predictions\_2,test.label)**

**mean(NB\_Predictions\_2!=test.label)**

**#------------------------SVM-Linear Method------------------------------------**

**##Support vector machine using liner kernel with different costs**

**library(e1071)**

**zoo=read.csv("/Users/ashwinibhoomi/Desktop/SEM-3/Data Mining/Project/zoo.data",header=F)**

**names(zoo)=c("animal", "hair", "feathers", "eggs", "milk", "airborne","aquatic", "predator", "toothed", "backbone", "breathes", "venomous",**

**"fins", "legs", "tail", "domestic", "size", "type")**

**zoo=zoo[-c(26),]**

**rownames(zoo)=NULL**

**zoo$animal=NULL**

**response=ifelse(zoo$type<=3,"No","Yes")**

**zoo=data.frame(zoo,response)**

**zoo=zoo[,-17]**

**train=sample(1:nrow(zoo), 60)**

**zoo.train=zoo[train,]**

**zoo.test=zoo[-train,]**

**response.train=response[train]**

**response.test=response[-train]**

**#Fitting the model**

**svmfit=svm(response~.,data=zoo.train,kernel="linear",cost=0.01)**

**summary(svmfit)**

**#Training error rate**

**svm1.pred=predict(svmfit,newdata=zoo.train)**

**table(svm1.pred,response.train)**

**mean(svm1.pred!=response.train)**

**#Testing error rate**

**svm1.pred=predict(svmfit,newdata=zoo.test)**

**table(svm1.pred,response.test)**

**mean(svm1.pred!=response.test)**

**#Using tune() for cross validation**

**set.seed(123)**

**tune.out=tune(svm, response~., data=zoo, kernel="linear",ranges=list(cost=c(0.01,0.1,1,10,100)))**

**summary(tune.out)**

**bestmod=tune.out$best.model**

**summary(bestmod)**

**#To find the training error**

**pred=predict(tune.out$best.model, newdata=zoo.train)**

**table(response.train, pred)**

**mean(pred!=response.train)**

**#To find the testing error**

**pred=predict(tune.out$best.model, newdata=zoo.test)**

**table(response.test, pred)**

**mean(pred!=response.test)**

**#------------------------SVM-Radial Method--------------------------------**

**#Support Vector Machine with radial kernel and default gamma before tune-out**

**set.seed(123)**

**svmfit=svm(response~.,data=zoo.train,kernel="radial",gamma=1,cost=0.01)**

**summary(svmfit)**

**svm1.pred=predict(svmfit,newdata=zoo.train)**

**table(svm1.pred,response.train)**

**mean(svm1.pred!=response.train)**

**svm2.pred = predict(svmfit,newdata=zoo.test)**

**table(svm2.pred,response.test)**

**mean(svm2.pred!=response.test)**

**##different cost and gammas**

**tune.out=tune(svm, response~., data=zoo.train, kernel="radial",ranges=list(cost=c(0.001,0.01,0.1,1,10), gamma=c(1,2,3,4,5)))**

**summary(tune.out)**

**bestmod=tune.out$best.model**

**summary(bestmod)**

**#Test error**

**pred=predict(tune.out$best.model, newdata=zoo.test)**

**table(response.test, pred)**

**mean(pred!=response.test)**

**#------------------------SVM-Polynomial Method------------------------------------**

**#Support Vector Machine with polynomial kernel and default gamma before tune-out**

**set.seed(123)**

**svmfit=svm(response~.,data=zoo.train,kernel="polynomial",degree=2,cost=0.01)**

**summary(svmfit)**

**svm1.pred=predict(svmfit,newdata=zoo.train)**

**table(svm1.pred,response.train)**

**mean(svm1.pred!=response.train)**

**svm2.pred = predict(svmfit,newdata=zoo.test)**

**table(svm2.pred,response.test)**

**mean(svm2.pred!=response.test)**

**##different cost and gammas**

**tune.out=tune(svm, response~., data=zoo.train, kernel="polynomial",degree=2,ranges=list(cost=c(0.001,0.01,0.1,1,10), gamma=c(0.2,0.5,1,2,3)))**

**summary(tune.out)**

**#Test error**

**pred=predict(tune.out$best.model, newdata=zoo.test)**

**table(response.test, pred)**

**mean(pred!=response.test)**