# OCCUPANCY DETECTION

Engineering Data Analysis – Team 14

## Abstract

This is the final report as part of the deliverable for Engineering Data Analysis project

**Prajwal Gonnade**
**Supreet Nayak**
**ISEN 613**

# 1. Executive Summary

## 1.1. Clear statement of the importance

During our initial project review our statement of project was ***"Classify occupancy in a room based on various experimental factors viz. Temperature, Humidity, CO2 and Light?".*** However, we found that the learnings were limited in the above project statement and thus the updated statement of the project is:

***"Identify the best classification method for occupancy in a room based on various experimental factors viz. Temperature, Humidity, CO2 and Light?"***

## 1.2. Statement of objectives

Occupancy Detection in a building has been one of the prime areas of interest for computer science community to impart efficient energy management and desk allocation systems throughout the organizations. Our primary focus was to detect the occupancy of the building using various factors like temperature, humidity, CO2 and Light. The best method used could then be implemented in the industry to save energy, one of the most focused areas of science in recent times.

## 1.3. What was gap in the literature (i.e., in what way your project is not a mere copy or regurgitation of what somebody else did)

Our analysis through variety of papers and literature review helped us to understand that though there are many methods successful at occupancy detection, none of the research papers had forayed into finding out the best classification methods for the same. Occupancy detection through opportunistic context sources did provide alternative to this method, but the accuracy was not the same as what we could have achieved through classification methods implemented in our project. Thus, our focus was to identify the best method of classification for occupancy detection which would provide us with greater accuracy while not sacrificing on computational speeds.

## 1.4. What methods were used? What challenges were addressed

Our primary areas of interest were to find occupancy detection using the classification methods learnt in class as well as through advanced techniques learnt through our research online. We implemented a wide variety of classification methods like GLM, LDA, QDA, CART, KNN, Random Forest, Artificial Neural Networks(ANN), Multilayer Perceptron and Support Vector Machines(SVM). The major challenge faced during the entire project was to zero down on the right kind of approach for different classification methods we were aiming to implement. With limited time and a broad scope for the project, we also found it challenging to relate the methods to each other which would help us better distinguish the best method for classification.

## 1.5. What are the key results?

The training and testing data obtained from UCI learning for occupancy detection was clean with little to no garbage values. After implementing various classification methods on our data, we found that Artificial Neural Networks, Support Vector Machine and Multilayer perceptron provided us with similar accuracies along with high performance and computation speeds. Accuracy for different classification methods was as follows:

## 1.6. What is the implication of the results in terms of the objectives you had set?

Our research showed that decision trees was the industry wide method which was preferred for occupancy detection case. However, for the kind of data that we could work on and the results that we obtained after applying the classification techniques, any of the methods like ANN, SVM or MLP could be used to get more efficient results while not sacrificing on computational speeds.

# 2. Introduction

## 2.1. Importance of the problem (detail with facts)

Energy conservation has been the prime area of focus for scientific community in the past decade. United States uses 66% fossil fuels in its generation of electric energy and with increasing consumption, this is likely to go up. Our implementation of occupancy detection is based on the idea of energy conservation by accurately predicting whether the room is occupied or not, and then taking the results to further switch all the appliances in the room on or off.

Furthermore, we can also connect the output of our project to other projects like: The Neural Network House: An Environment that Adapts to its Inhabitants: by M. C. Mozer, thus controlling the environment before the room is occupied to obtain our energy management goals. Our main goal is to provide results with the highest accuracy possible while not compromising on computation speeds to provide an efficient Launchpad to techniques which were discussed above.

## 2.2. Objectives, and provide specifics, scope of work (include the updated proposal sheet)

## Problem statement and approach

### *Problem Statement*

The updated research question we will be trying to answer as a part of this project is –

***"Identify the best classification method for occupancy in a room based on various experimental factors viz. Temperature, Humidity, CO2 and Light?"***

### *Updated Approach*

Following will be the probable approach for this project –

**1. Analyze the dataset**

1.1. Summarize and learn more about the dataset

1.2. Find correlations between various variables

1.3. Plot and visualize the dataset

**2. Apply one or more of the Classification methods viz. GLM, LDA, QDA, CART, KNN**

2.1. Decide upon significant predictors

2.2. Use various Classification methods to find the best model and method

**3. Apply advanced algorithms for better prediction**

3.1. Gain more understanding about Artificial Neural Networks, Multilayer Perceptron and Support Vector Machines algorithm

3.2. Find the application and suitability of these classification methods for the current dataset

3.3. If these methods are unsuitable, find another neural network algorithm suitable for the current dataset

## Dataset details (not the actual dataset)

### *Data Set Link*

**We will be using dataset from UCI Machine Learning data repository -**
http://archive.ics.uci.edu/ml/datasets/Occupancy+Detection**+**

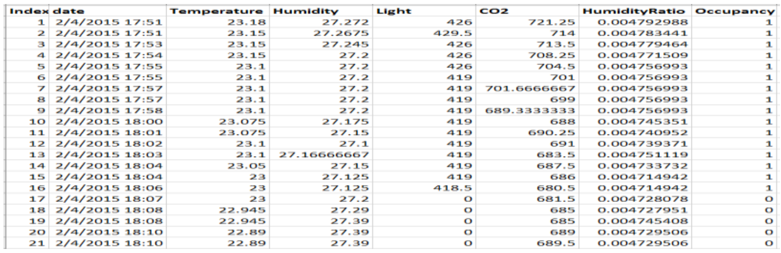
### *Details about the data*

The dataset contains experimental data used for binary classification (room occupancy) from Temperature, Humidity, Light and CO2.

The columns included in this dataset are:

- date time → Format {year-month-day} {hour: minute: second}
- Temperature → Units - Celsius
- Relative Humidity → Units - Percentage
- Light → Units - Lux
- CO2 → Units – Pats per million (ppm)
- Humidity Ratio → Derived quantity from temperature and relative humidity, Units - kg water-vapor/kg-air
- Occupancy → 0 or 1, 0 for not occupied, 1 for occupied status

### *Snapshot of Data*



| Index | date | Temperature | Humidity | Light | CO2 | HumidityRatio | Occupancy |
|---|---|---|---|---|---|---|---|
| 1 | 2/4/2015 17:51 | 23.18 | 27.272 | 426 | 721.25 | 0.004792988 | 1 |
| 2 | 2/4/2015 17:51 | 23.15 | 27.2675 | 429.5 | 714 | 0.004783441 | 1 |
| 3 | 2/4/2015 17:53 | 23.15 | 27.245 | 426 | 713.5 | 0.004779464 | 1 |
| 4 | 2/4/2015 17:54 | 23.15 | 27.2 | 426 | 708.25 | 0.004771509 | 1 |
| 5 | 2/4/2015 17:55 | 23.1 | 27.2 | 426 | 704.5 | 0.004756993 | 1 |
| 6 | 2/4/2015 17:55 | 23.1 | 27.2 | 419 | 701 | 0.004756993 | 1 |
| 7 | 2/4/2015 17:57 | 23.1 | 27.2 | 419 | 701.6666667 | 0.004756993 | 1 |
| 8 | 2/4/2015 17:57 | 23.1 | 27.2 | 419 | 699 | 0.004756993 | 1 |
| 9 | 2/4/2015 17:58 | 23.1 | 27.2 | 419 | 689.3333333 | 0.004756993 | 1 |
| 10 | 2/4/2015 18:00 | 23.075 | 27.175 | 419 | 688 | 0.004745351 | 1 |
| 11 | 2/4/2015 18:01 | 23.075 | 27.15 | 419 | 690.25 | 0.004740952 | 1 |
| 12 | 2/4/2015 18:02 | 23.1 | 27.1 | 419 | 691 | 0.004739371 | 1 |
| 13 | 2/4/2015 18:03 | 23.1 | 27.16666667 | 419 | 683.5 | 0.004751119 | 1 |
| 14 | 2/4/2015 18:04 | 23.05 | 27.15 | 419 | 687.5 | 0.004733732 | 1 |
| 15 | 2/4/2015 18:04 | 23 | 27.125 | 419 | 686 | 0.004714942 | 1 |
| 16 | 2/4/2015 18:06 | 23 | 27.125 | 418.5 | 680.5 | 0.004714942 | 1 |
| 17 | 2/4/2015 18:07 | 23 | 27.2 | 0 | 681.5 | 0.004728078 | 0 |
| 18 | 2/4/2015 18:08 | 22.945 | 27.29 | 0 | 685 | 0.004727951 | 0 |
| 19 | 2/4/2015 18:08 | 22.945 | 27.39 | 0 | 685 | 0.004745408 | 0 |
| 20 | 2/4/2015 18:10 | 22.89 | 27.39 | 0 | 689 | 0.004729506 | 0 |
| 21 | 2/4/2015 18:10 | 22.89 | 27.39 | 0 | 689.5 | 0.004729506 | 0 |

## 3. Literature Review

**3.1. Title:** Artificial Neural Networks: A Tutorial by AK Jain, J Mao, KM Mohiuddin (**Cited by 1607**)
**Reviewed by:** Supreet Nayak
**Complete reference:**
Links: [1] https://scholar.google.com/scholar?cluster=11348675984878115003&hl=en&as_sdt=0,44
[2] http://emotion.psychdept.arizona.edu/Jclub/Artificial%20Neural%20Networks%20-%20A%20Tutorial.pdf

**1) Objective:**

The paper addresses the essence of Artificial Neural Network and explains the mechanics of how ANN work through largely connected networks of neurons to work on a specific problem. The idea is to learn and understand neural network in depth, learning algorithms involved in creating neural networks and its implementation to solve engineering problems. Artificial Neural Networks have great advantages like adaptive learning, self-organization, real time operation and fault tolerance. ANN have applications like pattern recognition, prediction, optimization, associative memory and control which makes it essential to learn and be clear about basics of ANN as much as possible.

The objective of the paper is to help people with little to no understanding of neural networks gain a deeper perspective into how they work and how the concept evolves from proposed working of neurons in biology. Key challenges to be addressed using ANN, derived from working of neurons, were applications like pattern classification, clustering, function approximation, prediction and optimization. The paper describes "the basic biological neuron and the artificial computational model, outline network architectures and learning processes, and present some of the most commonly used ANN models" [2]

**2. Approach:**

The paper goes through the working of ANN by helping the readers understand how neurons are recreated in McCulloch-Pitts model of neuron. The paper then moved into classifying the different types of neural networks with major classification being feed forward and recurrent networks. A major emphasis was given throughout the paper on the importance of learning algorithms. A neural networks major point of advantage lies in its ability to learn efficiently based on the algorithm it uses. Key contributions in the paper were the discussion of four methods for learning algorithms which for the basis of feed forward and feedback networks. Error correction rules, Boltzmann learning, Hebbian rules and Competitive method were discussed in detail helping reader understand how leaning algorithms work.

**3. Results:**

The authors are trying to compare the many computational advantages that human brain offers compared to traditional parallel processing of computers, while trying to implement ANN models based on biological neural networks. The paper delves deep into how learning algorithms are applied through different types of ANN and then moves into its application by providing the example of optical character recognition (OCR). The authors found no solid evidence that ANN has advantages overs the traditional pattern classifiers. Though ANN are superior in terms of speed and memory requirements when compared to nearest neighbor algorithms, human brain still tend outperform ANN in OCR systems when trying to detect cursive handwriting. However, ANN works as efficiently as a human brain when working on isolated characters.

**4. Takeaways:**

Artificial Neural Networks have generated a lot of interest through the computer science community by promising significant advantages in computational power by trying to mimic how human brain computes parallel programs which are approximately 100 steps long for running serial computation. However, with no one approach dominating the other in complex computations like pattern recognition or face recognition, the paper concludes that the selection of a method for classification or clustering should be based on the given applications need.

**3.2. Title:** Occupancy Detection in Commercial Buildings using Opportunistic Context Sources by Sunil Kumar Ghai, Lakshmi V Thanayankizil, Deva P. Seetharam, Dipanjan Chakraborty (**Cited by 34**)
**Reviewed by:** Supreet Nayak
**Complete reference:**
Links: [1] https://scholar.google.com/scholar?cluster=194936897904874240&hl=en&as_sdt=0,44
[2] http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6197536

**1) Objective:**

The main objective of this paper is to use systems like instant messaging clients, Wi-Fi access points and mobile computing devices (called opportunistic context sources) installed in commercial buildings to predict occupancy of the building. Major advantages of occupancy detection in a building includes energy management and desk space allocation. The paper is split into seven sections with primary focus on applying cost saving techniques using machine learning algorithm and evaluating whether the proposed project is feasible by conducting a test run in an office. The paper tries to challenge traditional occupancy detection methods as it involves installation of additional sensors through the office space.

The paper also focuses on working through the challenge while using opportunistic context sources while working through the unreliability of input sources, conflict in the input provided and heterogeneity of spaces. Location detection of an employee to then detect the occupancy was also on top priority to achieve accurate results. Use of regression and classification models, and comparing these models to determine the best approach for the current problem was also discussed at length in the paper.

**2. Approach:**

The researchers used Weka to build linear regression models and classifiers, also using a k-fold cross validation approach with k=10. C.45 algorithm was used in the form of J48 which is the Java application of the same algorithm in Weka tool and over-fitting was avoided by using decision trees and appropriate pruning methodologies. Python was also used for processing and converting the dataset as required for the research. System accuracy and contribution of individual resources were then considered to get the desired results. Data instances which were correctly labeled were used to quantify the performance of the system with all situations being considered for classification and situation with tagged data points sufficient for learning considered in regression. An input vector was generated using only Wi-Fi access points and then other data was added to create a classification model. A confusion matrix was then created to check for accuracy of predicted classes.

**3. Results:**

Pilot study helped the team to achieve accuracy of over 90% with all the assumptions for challenges faced in place. The paper argues that using context based sources in smart buildings and their approach for classification to determine the occupancy of the building, the use of high precision sensors, wireless PIR sensors and requirement of other hardware can be eliminated. The paper further proposes that $CO_2$ based occupancy detection systems are slow to detect changes whereas installation of cameras or acoustic based devices would bear significant expense to the companies. Cameras and acoustic based devices where further rejected by authors due to privacy issues and constraints. Additional pilot project was still conducted by authors to further explore domain specific applications and investigate more into unsupervised learning techniques.

**4. Takeaways:**

The major takeaway from this paper was the alternative approach to room occupancy detection that could be considered while using minimalistic installation procedures and enjoying reduced costs due to the same. Classification and regression techniques when used with right assumptions to ward off potential challenges and opportunistic context sources can provide us with accurate results for occupancy detection. Though major companies throughout the globe can afford installation of precision sensors and other hardware for occupancy detection to impact energy management, medium and small scale companies with infrastructure already in place can use this research methodology to implement energy management within the organization.

**3.3. Title:** The Neural Network House: An Environment that Adapts to its Inhabitants: by M. C. Mozer (**Cited by 753**)
**Reviewed by:** Prajwal Gonnade
**Complete reference:**
Links: [1] https://scholar.google.com/scholar?q=related:ejAf60j7t-sJ:scholar.google.com/&hl=en&as_sdt=0,44
[2] http://www.aaai.org/Papers/Symposia/Spring/1998/SS-98-02/SS98-02-017.pdf

**1) Objective:**

Smart homes have been the talking point of this century since the advent of internet of things and machine learning algorithms. However, costs have always exceeded the benefits such systems would provide which had led this topic to have remained untouched by the computer scientist community. The main objectives of the paper are to provide an alternative approach to the smart home application moving away from the traditional idea of providing your devices greater computational power.

The paper focusses on using machine learning and neural networks to achieve two primary objectives. First, Prediction of inhabitants needs with focus on lighting, temperature and ventilation maintained as per the need of occupant. The system should train itself whenever the inhabitant changes any of the variables which indicate inhabitants being not satisfied with the status of variables. Thus, the authors aim to avoid manual control of the variables. Second, the paper aims to implement energy conservation with the implementation of neural network house. Whenever the system performs changes to the variables, then the option which optimizes and minimizes the use of energy must be given the first preference. For e.g., ambient lighting for the house if chosen by the system should be decided between sunlight and lightbulb based on optimum light from the source and light required by the inhabitant.

**2. Approach:**

The ACHE system, abbreviation for adaptive control of home environment used the optimal control framework against using supervised learning. Supervised learning was rejected as energy costs will not be considered and hence one of the objective would not be met. The authors decided to associate a discomfort cost with the system if the inhabitant's requirements are not met by the ACHE system. The discomfort cost is associated with the environmental state at a time and the decision taken to change the control at that time. This was done with the objective to create a framework or rather a mapping from the state to the decision. The prototype built was then implemented in a three rooms school house from 1905, renovated in 1992 thus forming the ideal candidate as the required hardware could be setup easily. Sensors were installed and observations were recorded for sensory states.

**3. Results:**

The authors tried to find consistency in the response of the inhabitants for ACHE to be working at its full implementation. Intelligent control techniques for complex systems in ever changing environments was the focus of the authors primary area of focus and this being a specific implementation of the same, researchers wanted to look out for patterns that would help in adaptive control of living environments. Discussion regarding irregular daily schedule are also mentioned in the results which though sound deviation from normal behavior for inhabitant is useful for training ACHE systems. The research further involved control of equipment by making simplistic assumptions about the condition of operation and environment around the inhabitant. Long term evaluation revealed systems like intelligent home environments which predict inhabitant behavior are beneficial to energy conservation efforts.

**4. Takeaways:**

The major takeaway for our project through this paper was the ability to control the environment through neural networks and creating a learning algorithm that would adapt to the requirements of the inhabitants. As we ventured into the occupancy data to predict occupation of the room, it is important to consider applications that would further make the system step up the game. An idea that we as a team discussed was based on the data of occupant and an occupant detailed preferences, an adaptive environment could create ambient conditions in the room based upon whether the inhabitant has occupied the room or not. This creates an interesting idea which will work towards both energy conservation as well as making human lives easier through automation.

**3.4. Title:** Artificial Neural Networks (The multilayer perceptron)—a review of applications in the atmospheric sciences: (**Cited by 940**)

**Reviewed by:** Prajwal Gonnade

**Complete reference:**

Links:[1] https://scholar.google.com/scholar?q=multilayer+perceptron+neural+network&hl=en&as_sdt=0,47&as_vis=1
[2] http://www.sciencedirect.com/science/article/pii/S1352231097004470

**1) Objective:**

The objective of the paper was to get help readers understand multilayer perceptron and its increased application in the field of environmental science. The paper dives into explaining the neural networks and its application along with discussing the benefits offered over traditional classification and statistical methods. The author also aim at explaining the applications of multilayer perceptron to atmospheric problems with problems and difficulties associated with artificial neural network. Backpropagation, one of the basic algorithm for training MLP is also explained in the paper while explaining how MLP works based on interconnected neuron structures.

The authors then explain multilayer perceptron in general by going through application of general perceptron, prediction and approximation of function and classification of pattern. Other objectives of paper include explanation of MLP applications in atmospheric sciences by providing examples of prediction in weather such as forecasting Indian monsoon, function approximation for ozone concentration and pattern classification to determine surface pressure over the British Isles. The paper also aimed at explaining the problems and limits of back-propagation learning in multilayer perceptron by giving practical examples.

**2. Approach:**

The paper uses detailed explanation and examples to explain how classification methods, in particular, multilayer perceptron work using the core principle of working of human brains. Explanation regarding how MLP is used for classification by assigning class to output nodes and then providing an accurate classification is also provided in the paper. Since, the paper is aimed at helping a reader understand the implementations of multilayer perceptron and its application, a lot of details regarding how prediction and function approximation is carried out has been provided. Furthermore, a lot of examples provide us with explanations as to how experts in atmospheric studies are using MLP to perform predictions. The approach used by the authors to achieve the objectives of the papers is by providing detailed explanations, equations and examples used in practical applications to help readers with knowing more about MLP.

**3. Results:**

The authors conclude by informing the readers about the benefits MLP when reproducing the relation between any variable and the relationship between the variable is significant. The limitations and problems faced in the MLP can be balanced with the benefits that they provide when compared to traditional statistical techniques. The authors also point out about commercially free packages being available to implement neural networks, which prevent users from the difficult part of parameter selection for the models. However, with training algorithms being limited in number, the MLP packages provide limited utility to people using them. The paper concludes by informing users about the benefit of generating code for MLP on your own thus providing greater flexibility and more adaptations for training dataset.

**4. Takeaways:**

The major takeaway for our project from this paper was the understanding and implementation of MLP while focusing on practical applications that are implemented in atmospheric studies. Our project involved analyzing variables like temperature, humidity and $CO_2$ like the variables used in the paper which drew our interest further. The paper helped us to gain insights into the MLP with its detailed explanation of how the algorithm works and how it can be used to generate accurate predictions as well as gain accurate pattern recognitions. This helped us to implement MLP in our project with greater clarity and confidence.

# 4. Project Approach

## 4.1. Description of the new technique

Following are the advanced techniques we have implemented in this project -

**1. Artificial Neural Network**

Artificial Neural Network is the computational approach which refers to the interconnections between the neural units similar to the biological brain. ANN typically involves an interconnection pattern, a learning process for finding robust weights of interconnections and an activation function which produces final output.

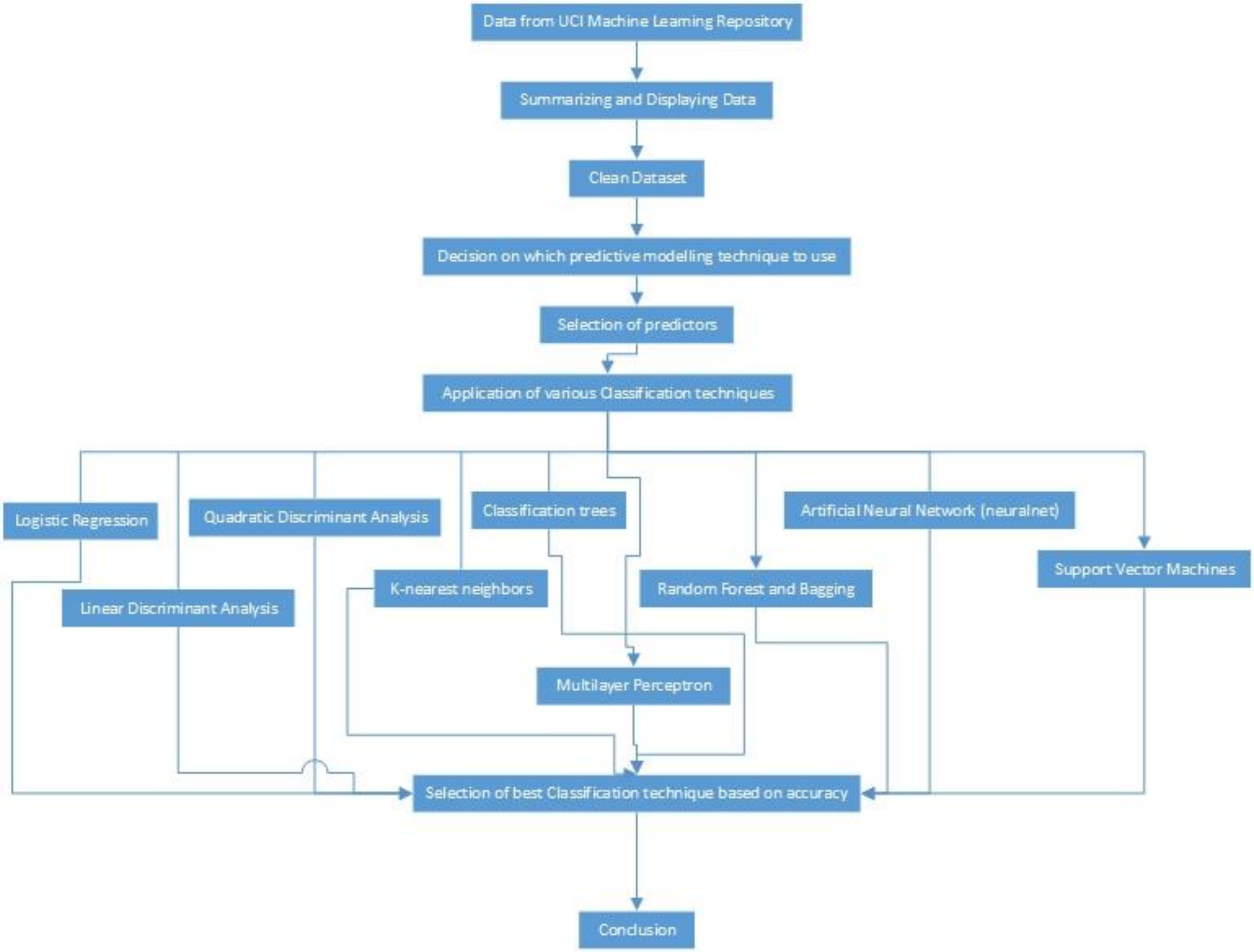
**2. Multilayer Perceptron**

Multilayer Perceptron is a type of ANN which belongs to feedforward category of artificial neural networks. It utilizes a learning technique called backpropagation where the expected result is compared to the amount of error in the output after the connection weights are changed after data processing.

**3. Support Vector Machines**

Support Vector Machines is used to construct a linear classifier i.e. classifier to differentiate data into various classes or groups. It is a supervised learning technique which constructs a hyperplane or a set of hyperplanes which can be used for tasks such as classification and regression.

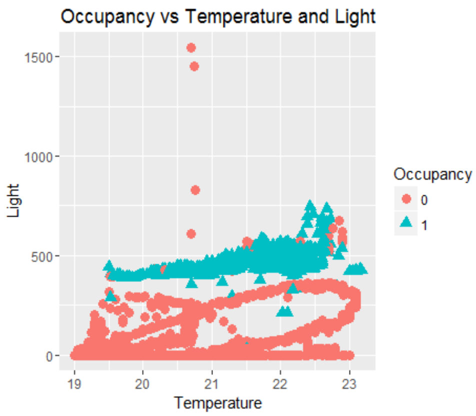
## 4.2. Block Diagram to Project Approach

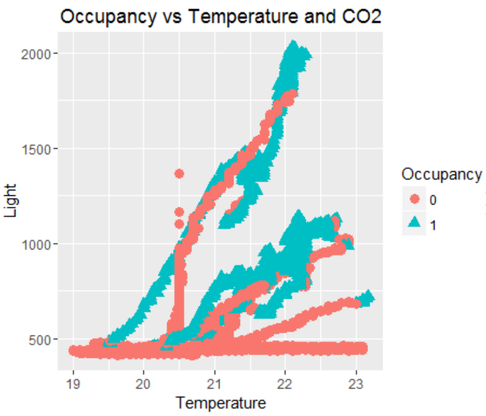

Data from UCI Machine Learning Repository
Summarizing and Displaying Data
Clean Dataset
Decision on which predictive modelling technique to use
Selection of predictors
Application of various Classification techniques
Logistic Regression
Quadratic Discriminant Analysis
Classification trees
Artificial Neural Network (neuralnet)
Support Vector Machines
Linear Discriminant Analysis
K-nearest neighbors
Random Forest and Bagging
Multilayer Perceptron
Selection of best Classification technique based on accuracy
Conclusion

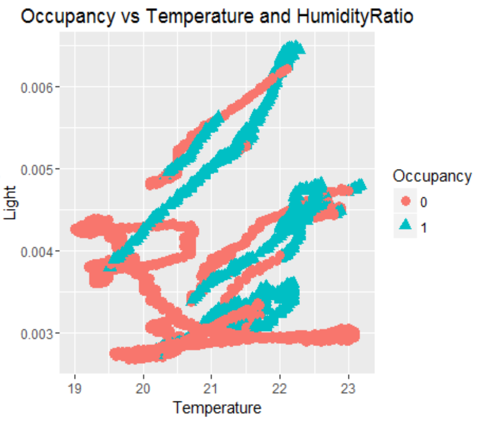
# 5. Implementation details

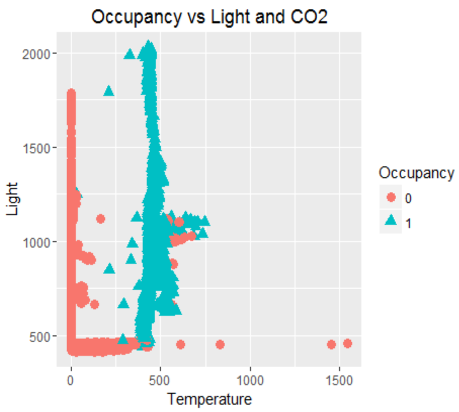
Below are the implementation details of 9 classification techniques we have used. The details include the code to create the model and computing the confusion matrix. To review our entire work, please refer the Appendix and the shared R Markdown files with the Knit Word documents.

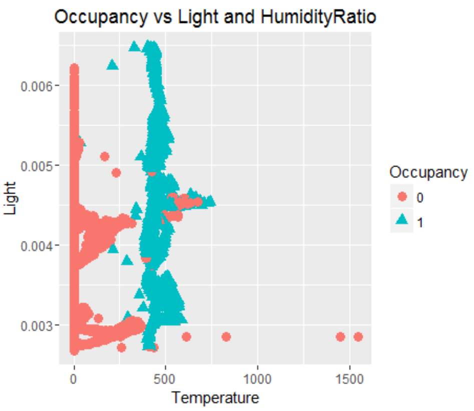
**Data Visualization**

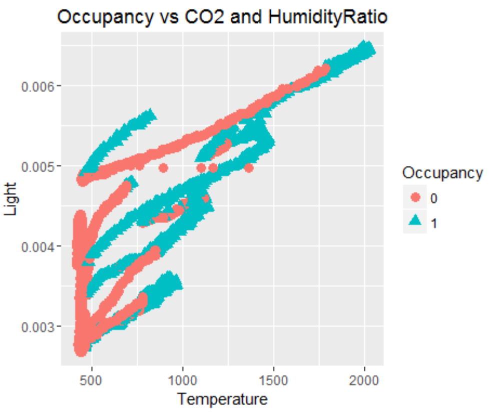












### a. Logistic Regression

Implemented logistic regression technique for predictor selection and computed confusion matrix to measure Misclassification Error Rate and Accuracy

```
glm(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupancy_Train, family = "binomial")
glm_probs_1 = predict(Occupancy_glm, Occupancy_Test1, type = "response")
glm_pred_y_1 = rep(0, length(Occupancy_Test1$Occupancy))
glm_pred_y_1[glm_probs_1 > 0.5] = 1
table(glm_pred_y_1, Occupancy_Test1$Occupancy)
mean(glm_pred_y_1 != Occupancy_Test1$Occupancy)
confusionMatrix(Occupancy_Test1$Occupancy, glm_pred_y_1)
glm_probs_1 = predict(Occupancy_glm, Occupancy_Test2, type = "response")
glm_pred_y_1 = rep(0, length(Occupancy_Test2$Occupancy))
glm_pred_y_1[glm_probs_1 > 0.5] = 1
table(glm_pred_y_1, Occupancy_Test2$Occupancy)
mean(glm_pred_y_1 != Occupancy_Test2$Occupancy)
confusionMatrix(Occupancy_Test2$Occupancy, glm_pred_y_1)
```

### b. Linear Discriminant Analysis

Implemented LDA technique for classification and computed confusion matrix to measure Misclassification Error Rate and Accuracy. Cross validation details can be found in Appendix.

```
Occupancy_lda <- lda(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupancy_Train)
Occupancy_lda
lda_pred = predict(Occupancy_lda, Occupancy_Test1)
names(lda_pred)
table(lda_pred$class, Occupancy_Test1$Occupancy)
mean(lda_pred$class != Occupancy_Test1$Occupancy)
confusionMatrix(Occupancy_Test1$Occupancy,lda_pred$class)
lda_pred = predict(Occupancy_lda, Occupancy_Test2)
names(lda_pred)
table(lda_pred$class, Occupancy_Test2$Occupancy)
mean(lda_pred$class != Occupancy_Test2$Occupancy)
confusionMatrix(Occupancy_Test2$Occupancy,lda_pred$class)
```

### c. Quadratic Discriminant Analysis

Implemented LDA technique for classification and computed confusion matrix to measure Misclassification Error Rate and Accuracy. Cross validation details can be found in Appendix.

```
Occupancy_qda <- qda(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupancy_Train)
Occupancy_qda
qda.pred <- predict(Occupancy_qda, Occupancy_Test1)
table(qda.pred$class, Occupancy_Test1$Occupancy)
mean(qda.pred$class != Occupancy_Test1$Occupancy)
confusionMatrix(Occupancy_Test1$Occupancy,qda.pred$class)
qda.pred <- predict(Occupancy_qda, Occupancy_Test2)
table(qda.pred$class, Occupancy_Test2$Occupancy)
mean(qda.pred$class != Occupancy_Test2$Occupancy)
confusionMatrix(Occupancy_Test2$Occupancy,qda.pred$class)
```

### d. Classification Trees

Implemented Classification tree technique and computed confusion matrix to measure Misclassification Error Rate and Accuracy.

```
#CART
set.seed(2)
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
str(Occupancy_Train)
tree.Occupancy_Train <- tree(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupancy_Train)
plot(tree.Occupancy_Train)
summary(tree.Occupancy_Train)
text(tree.Occupancy_Train, pretty = 0)
tree.pred <- predict(tree.Occupancy_Train, Occupancy_Test1, type = "class")
table(tree.pred, Occupancy_Test1$Occupancy)
mean(tree.pred != Occupancy_Test1$Occupancy)
confusionMatrix(Occupancy_Test1$Occupancy, tree.pred)

#Cross Validation and Pruning
set.seed(3)
cv.Occupancy_Train <- cv.tree(tree.Occupancy_Train, FUN=prune.misclass)
names(cv.Occupancy_Train)
cv.Occupancy_Train
par(mfrow = c(1,2))
plot(cv.Occupancy_Train$size, cv.Occupancy_Train$dev, type = "b")
plot(cv.Occupancy_Train$k, cv.Occupancy_Train$dev, type = "b")
prune.Occupancy_Train = prune.misclass(tree.Occupancy_Train, best = 9)
plot(prune.Occupancy_Train)
text(prune.Occupancy_Train, pretty = 0)
tree.pred = predict(prune.Occupancy_Train, Occupancy_Test1, type = "class")
table(tree.pred, Occupancy_Test1$Occupancy)
mean(tree.pred != Occupancy_Test1$Occupancy)
confusionMatrix(Occupancy_Test1$Occupancy, tree.pred)
```

### e. K-Nearest Neighbors

Implemented K-Nearest Neighbor technique with K=1, 3, 5, 10, 25, 50, 100. Below is sample code. For more details, see Appendix.

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test1$Occupancy <- factor(Occupancy_Test1$Occupancy)
attach(Occupancy_Train)
test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test1))]
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Train))]
```

```
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 3, prob = TRUE)
table(knn.pred, Occupancy_Test1$Occupancy)
mean(knn.pred != Occupancy_Test1$Occupancy)
confusionMatrix(Occupancy_Test1$Occupancy, knn.pred)
```

### f. Random Forest and Bagging

Implemented Random Forest and Bagging technique for classification and computed confusion matrix to measure Misclassification Error Rate and Accuracy.

```
#Random Forest Ntree
set.seed(123)
bag.Occupancy_Train = randomForest(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data =
    Occupancy_Train, mtry = 4, ntree = 15)
bag.Occupancy_Train
yhat.bag = predict(bag.Occupancy_Train, Occupancy_Test1)
plot(yhat.bag, Occupancy_Test1$Occupancy)
abline(0,1)
mean((as.numeric(yhat.bag) - as.numeric(Occupancy_Test1$Occupancy)) ^ 2)

#Random Forest
set.seed(123)
rf.Occupancy_Train = randomForest(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data =
    Occupancy_Train, mtry = 4, importance = TRUE)
rf.Occupancy_Train
yhat.rf = predict(bag.Occupancy_Train, Occupancy_Test1)
plot(yhat.rf, Occupancy_Test1$Occupancy)
abline(0,1)
mean((as.numeric(yhat.rf) - as.numeric(Occupancy_Test1$Occupancy)) ^ 2)
importance(rf.Occupancy_Train)
varImpPlot(rf.Occupancy_Train)

#Random Forest Confusion Matrix
rf.pred <- predict(rf.Occupancy_Train, Occupancy_Test1, type = "class")
table(rf.pred, Occupancy_Test1$Occupancy)
mean(rf.pred != Occupancy_Test1$Occupancy)
confusionMatrix(Occupancy_Test1$Occupancy, rf.pred)
```

### g. Artificial Neural Network (neuralnet)

Implemented ANN technique using neuralnet package with 2 hidden layers for classification and computed confusion matrix to measure Misclassification Error Rate and Accuracy.

```
#Neural Network
library(neuralnet)
nn <- neuralnet(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data=Occupancy_Train,hidden=2,threshold
    = 0.01, linear.output=FALSE)
plot(nn, rep = "best")
pr.nn1 <- compute(nn,Occupancy_Test1[,c(2,4,5,6)])
results1 <- data.frame(actual = Occupancy_Test1$Occupancy, prediction = pr.nn1$net.result)
table(round(pr.nn1$net.result),Occupancy_Test1$Occupancy)
mean(round(pr.nn1$net.result) != Occupancy_Test1$Occupancy)
confusionMatrix(Occupancy_Test1$Occupancy, round(pr.nn1$net.result))
```

### h. 1. Multilayer Perceptron [4]

Implemented multilayer perceptron technique for classification and computed confusion matrix to measure Misclassification Error Rate and Accuracy.

```
Occupancy.monmlp <- monmlp.fit(Occupancy_subset, Occupancy_response, hidden1=3, n.ensemble=15, monotone=1,
    bag=TRUE)
monmlp.test1 <- monmlp.predict(x = data.matrix(Occupancy_Test1[,c(2,4,5,6)]), weights = Occupancy.monmlp)
mean(round(monmlp.test1) != Occupancy_Test1$Occupancy)
confusionMatrix(Occupancy_Test1$Occupancy, round(monmlp.test1))

monmlp.test2 <- monmlp.predict(x = data.matrix(Occupancy_Test2[,c(2,4,5,6)]), weights = Occupancy.monmlp)
mean(round(monmlp.test2) != Occupancy_Test2$Occupancy)
confusionMatrix(Occupancy_Test2$Occupancy, round(monmlp.test2))
```

### 2. Perceptron Classification algorithm[3]

```
Occupancy_subset <- Occupancy_Train[,c(2,4,5,6,7)]
Occupancy_response <- Occupancy_Test2[,7]
feature_plot <- function (Occupancy_subset, Occupancy_response) {
  mtmelt <<- melt(Occupancy_subset, id.vars = Occupancy_response)
  p <- ggplot(mtmelt, aes(x = value, y = mtmelt[, 5])) +
    facet_wrap(~variable, scales = "free") +
    geom_point() +
    labs(y = Occupancy_response)
  p
}
feature_plot(Occupancy_subset, Occupancy_response)
Occupancy_subset[, 6] <- 1
Occupancy_subset[Occupancy_subset[, 5] == 0, 6] <- -1
x <- Occupancy_subset[, c(1,2,3,4)]
y <- Occupancy_subset[, 6]
head(x)
head(y)
perceptron <- function(x, y, eta, niter) {
        # initialize weight vector
        weight <- rep(0, dim(x)[2] + 1)
        errors <- rep(0, niter)
        # loop over number of epochs niter
        for (jj in 1:niter) {
                # loop through training data set
                for (ii in 1:length(y)) {
                        # Predict binary label using Heaviside activation
                        # function
                        z <- sum(weight[2:length(weight)] *
                                        as.numeric(x[ii, ])) + weight[1]
                        if(z < 0) {
                                ypred <- -1
                        } else {
                                ypred <- 1
                        }
                        # Change weight - the formula doesn't do anything
                        # if the predicted value is correct
```

```
                    weightdiff <- eta * (y[ii] - ypred) *
                        c(1, as.numeric(x[ii, ]))
                    weight <- weight + weightdiff

                    # Update error function
                    if ((y[ii] - ypred) != 0.0) {
                        errors[jj] <- errors[jj] + 1
                    }
                }
            }
        # weight to decide between the two species
        print(weight)
        return(errors)
}
err <- perceptron(x, y, 0.1, 5)
plot(0.1:5, err, type="l", lwd=2, col="red", xlab="epoch #", ylab="errors")
title("Errors vs epoch - learning rate eta = 0.1")
```

### i. Support Vector Machines

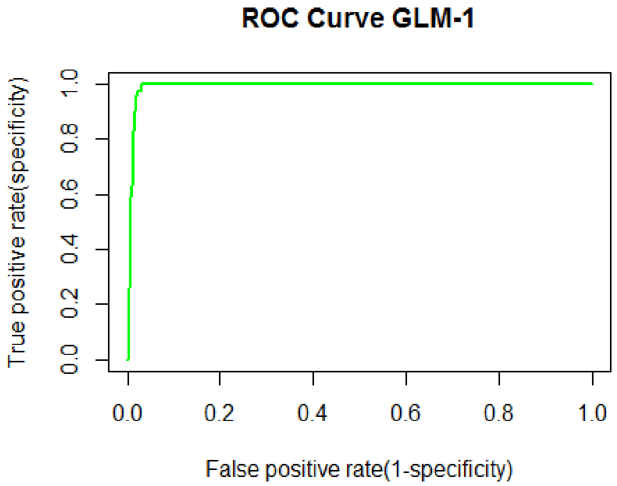
Implemented SVM learning technique for classification and computed confusion matrix to measure Misclassification Error Rate and Accuracy.

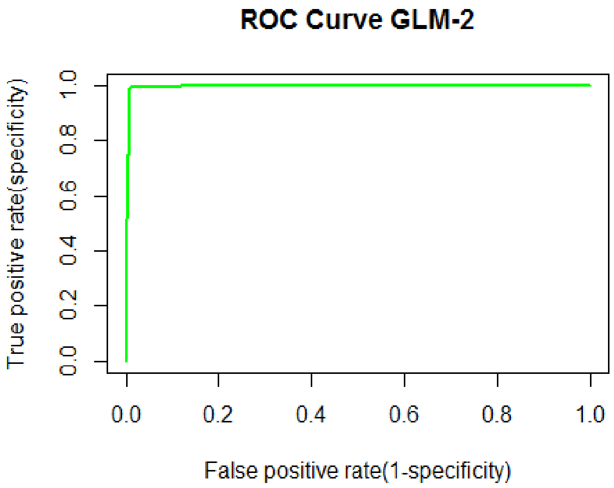
```
Occupancy_subset=data.frame(x=Occupancy_Train[,c(2,4,5,6)], y=as.factor(Occupancy_Train[,7]))
Occupancy.svm=svm(y~., data=Occupancy_subset, kernel="linear",cost=10)
summary(Occupancy.svm)
table(Occupancy.svm$fitted, Occupancy_subset$y)
Occupancy_subset.te=data.frame(x=Occupancy_Test1[,c(2,4,5,6)], y=as.factor(Occupancy_Test1[,7]))
pred.te=predict(Occupancy.svm, newdata=Occupancy_subset.te,decision.values=TRUE)
Occupancy.svm.probs<-attr(pred.te,"decision.values")
table(pred.te, Occupancy_subset.te$y)
Occupancy.svm.confusion <- confusionMatrix(Occupancy_subset.te$y, pred.te)
Occupancy.svm.confusion
Occupancy.svm.accuracy <- mean(pred.te == Occupancy_subset.te$y)
Occupancy.svm.accuracy
Occupancy.svm.prediction<-prediction(Occupancy.svm.probs,Occupancy_subset.te$y)
Occupancy.svm.performance<-performance(Occupancy.svm.prediction,"tpr","fpr")
Occupancy.svm.auc<-performance(Occupancy.svm.prediction,"auc")@y.values[[1]]
plot(Occupancy.svm.performance)
```

## 6. Results

### a) Logistic Regression

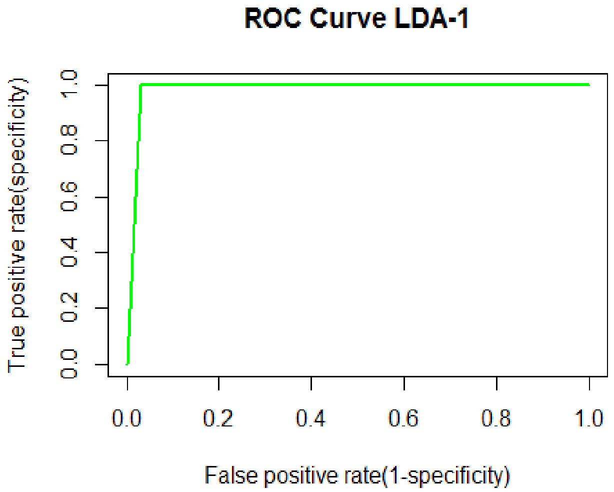
| Measures | Testing Data - 1 | Testing Data - 2 |
|---|---|---|
| **Prediction Table** | ##          Reference<br>## Prediction    0    1<br>##             0 1639   54<br>##             1   12     960 | ##          Reference<br>## Prediction    0    1<br>##             0 7648   55<br>##             1  135    1914 |
| **Mean error** | 0.02476548 | 0.01948318 |
| **Accuracy** | **0.9752** | **0.9805** |
| **Sensitivity** | 0.9927 | 0.9827 |
| **Specificity** | 0.9467 | 0.9721 |
| **Pos Pred Value** | 0.9681 | 0.9929 |
| **Neg Pred Value** | 0.9877 | 0.9341 |

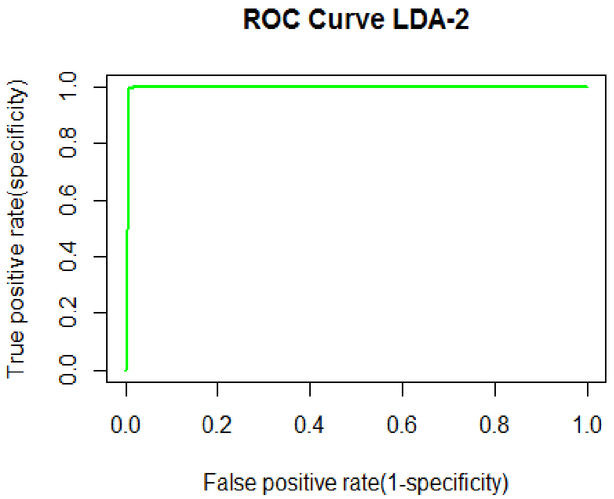




### b) Linear Discriminant Analysis

| Measures | Testing Data - 1 | Testing Data - 2 |
|---|---|---|
| Prediction Table | ## Reference<br>## Prediction 0 1<br>## 0 1638 55<br>## 1 1 971 | ## Reference<br>## Prediction 0 1<br>## 0 7626 77<br>## 1 8 2041 |
| Mean error | 0.02101313 | 0.008716161 |
| Accuracy | **0.979** | **0.9913** |
| Sensitivity | 0.9994 | 0.9990 |
| Specificity | 0.9464 | 0.9636 |
| Pos Pred Value | 0.9675 | 0.9900 |
| Neg Pred Value | 0.9990 | 0.9961 |

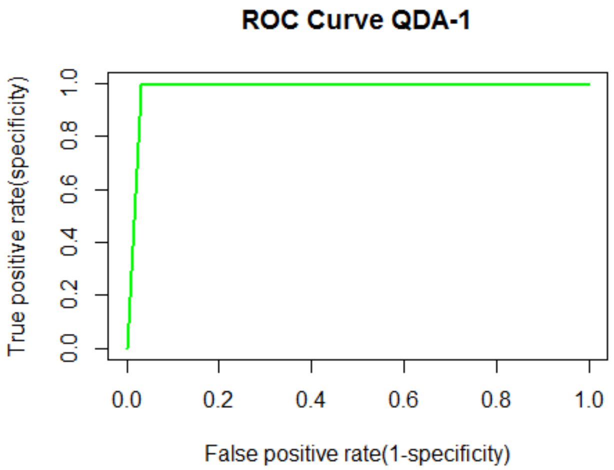


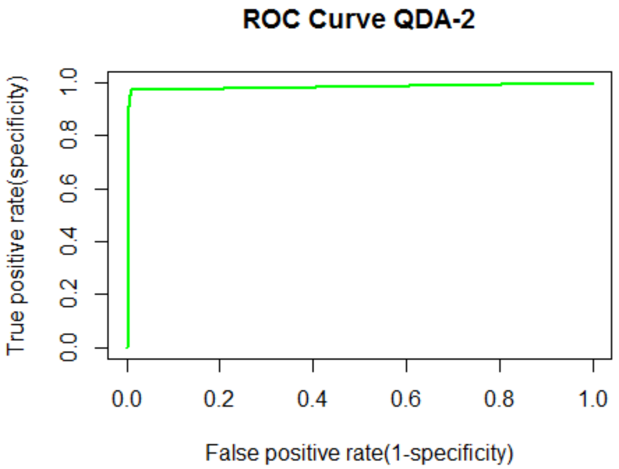


### c) Quadratic Discriminant Analysis

| Measures | Testing Data - 1 | Testing Data - 2 |
|---|---|---|
| Prediction Table | ## Reference<br>## Prediction 0 1<br>## 0 1639 54<br>## 1 6 966 | ## Reference<br>## Prediction 0 1<br>## 0 7645 58<br>## 1 166 1883 |
| Mean error | 0.02251407 | 0.02296965 |
| Accuracy | **0.9775** | **0.977** |

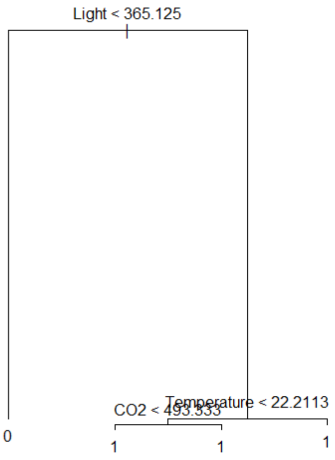
| | | |
|---|---|---|
| Sensitivity | 0.9964 | 0.9787 |
| Specificity | 0.9471 | 0.9701 |
| Pos Pred Value | 0.9681 | 0.9925 |
| Neg Pred Value | 0.9938 | 0.9190 |

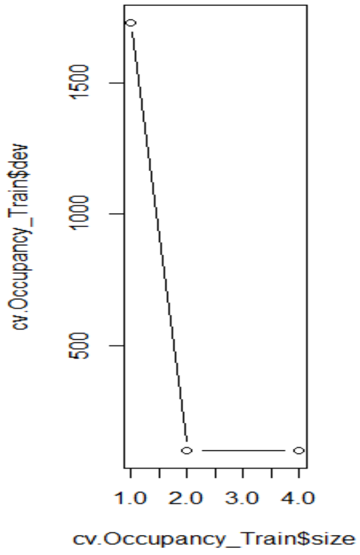


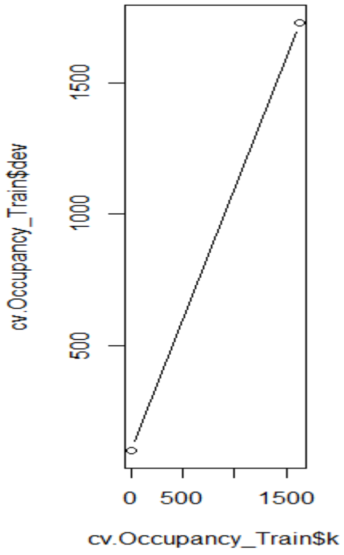


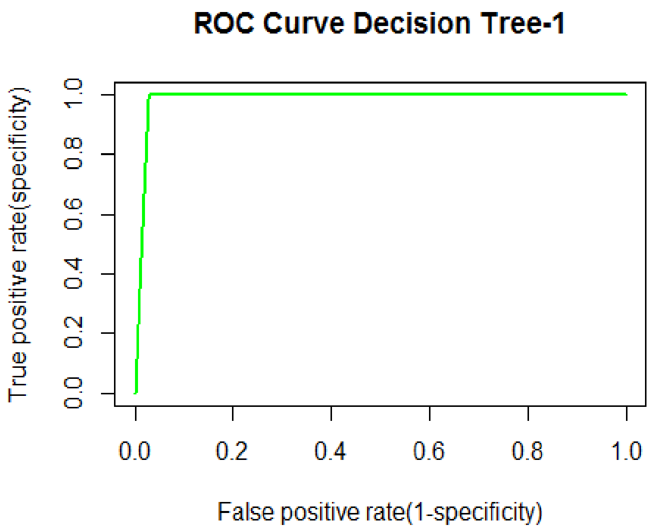
## d) Classification Trees

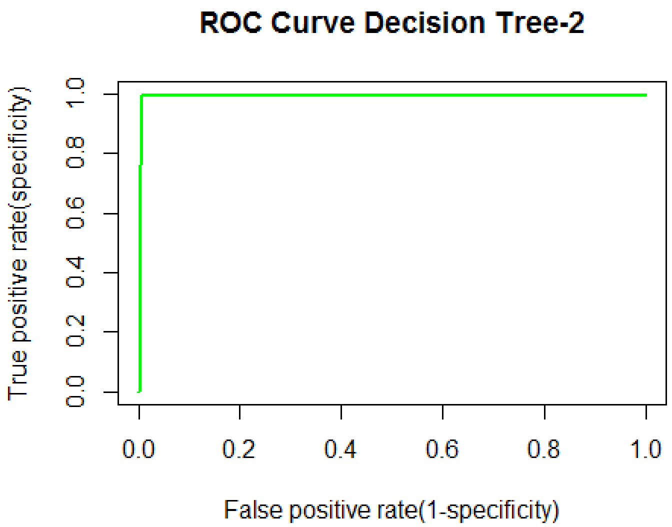
| Measures | Testing Data - 1 | Testing Data - 2 |
|---|---|---|
| Prediction Table | ## Reference<br>## Prediction 0 1<br>## 0 1639 54<br>## 1 3 969 | ## Reference<br>## Prediction 0 1<br>## 0 7648 55<br>## 1 12 2037 |
| Mean error | 0.02138837 | 0.006870386 |
| Accuracy | **0.9786** | **0.9931** |
| Sensitivity | 0.9982 | 0.9984 |
| Specificity | 0.9472 | 0.9737 |
| Pos Pred Value | 0.9681 | 0.9929 |
| Neg Pred Value | 0.9969 | 0.9941 |











### e) K-Nearest Neighbors

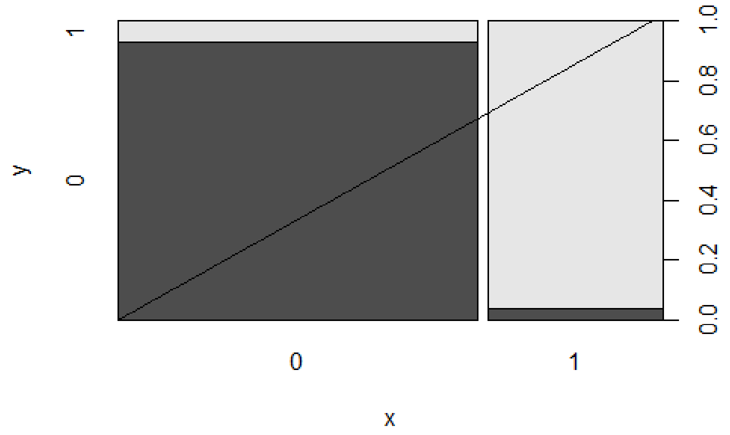
| Measures | K=1 | K=3 | K=5 |
|---|---|---|---|
| Prediction Table | ## Reference<br>## Prediction 0 1<br>## 0 1231 462<br>## 1 525 447 | ## Reference<br>## Prediction 0 1<br>## 0 1231 462<br>## 1 523 449 | ## Reference<br>## Prediction 0 1<br>## 0 1240 453<br>## 1 524 448 |
| Mean error | 0.3703565 | 0.369606 | 0.3666041 |
| Accuracy | **0.6296** | **0.6304** | **0.6334** |
| Sensitivity | 0.7010 | 0.7018 | 0.7029 |
| Specificity | 0.4917 | 0.4929 | 0.4972 |
| Pos Pred Value | 0.7271 | 0.7271 | 0.7324 |
| Neg Pred Value | 0.4599 | 0.4619 | 0.4609 |

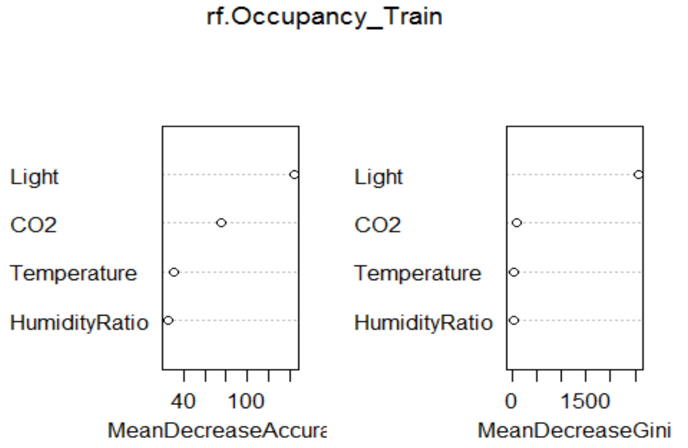
| Measures | K=10 | K=25 | K=50 | K=100 |
|---|---|---|---|---|
| Prediction Table | ## Reference<br>## Prediction 0 1<br>## 0 1249 444<br>## 1 528 444 | ## Reference<br>## Prediction 0 1<br>## 0 1255 438<br>## 1 529 443 | ## Reference<br>## Prediction 0 1<br>## 0 1290 403<br>## 1 551 421 | ## Reference<br>## Prediction 0 1<br>## 0 1320 373<br>## 1 548 424 |
| Mean error | 0.364728 | 0.3628518 | 0.3579737 | 0.345591 |
| Accuracy | **0.6353** | **0.6371** | **0.642** | **0.6544** |
| Sensitivity | 0.7029 | 0.7035 | 0.7007 | 0.7066 |
| Specificity | 0.5000 | 0.5028 | 0.5109 | 0.5320 |
| Pos Pred Value | 0.7377 | 0.7413 | 0.7620 | 0.7797 |
| Neg Pred Value | 0.4568 | 0.4558 | 0.4331 | 0.4362 |

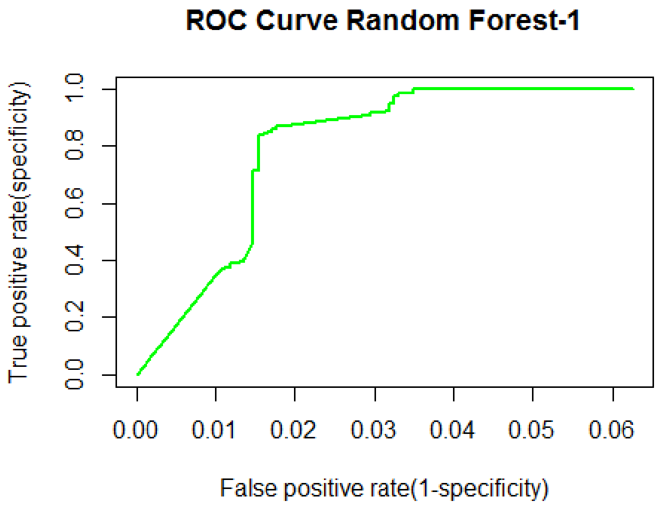
### f) Random Forest and Bagging

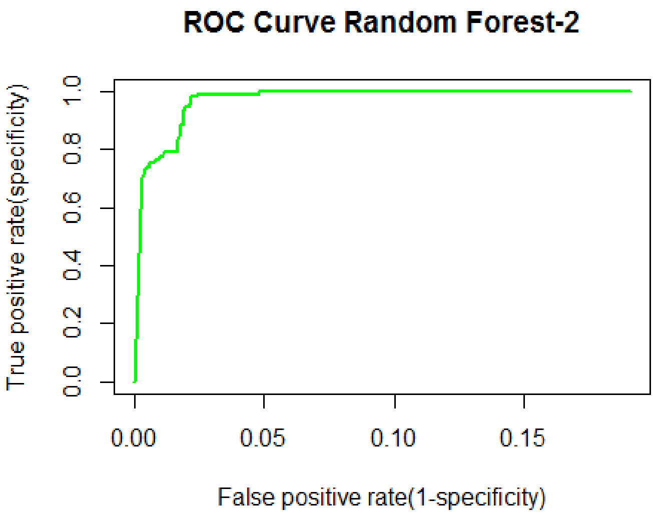
| Measures | Testing Data - 1 | Testing Data - 2 |
|---|---|---|
| Prediction Table | ## Reference<br>## Prediction 0 1<br>## 0 1663 30<br>## 1 130 842 | ## Reference<br>## Prediction 0 1<br>## 0 7554 149<br>## 1 132 1917 |
| Mean error | 0.06003752 | 0.0288146 |
| Accuracy | **0.94** | **0.9712** |

|  |  |  |
|---|---|---|
| Sensitivity | 0.9275 | 0.9828 |
| Specificity | 0.9656 | 0.9279 |
| Pos Pred Value | 0.9823 | 0.9807 |
| Neg Pred Value | 0.8663 | 0.9356 |



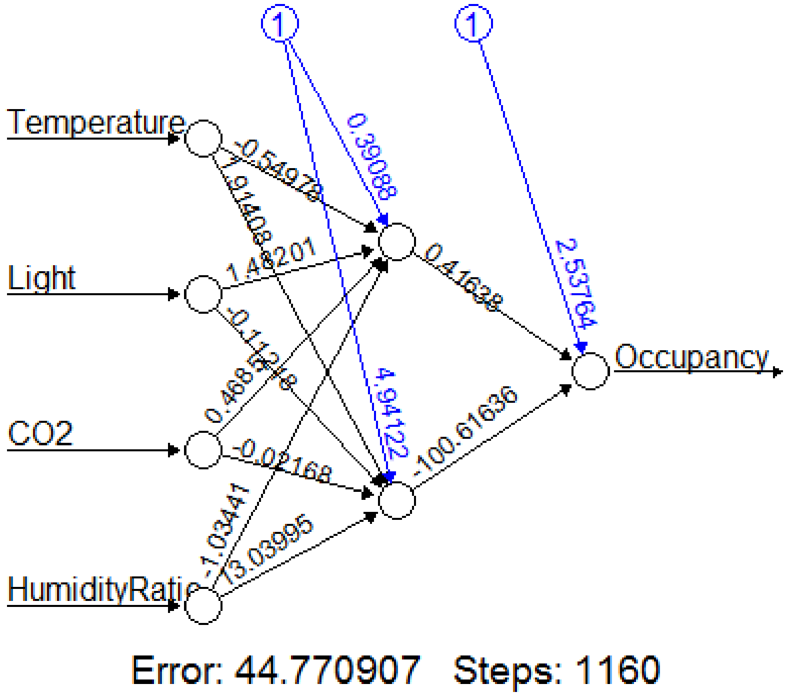






### g) Artificial Neural Network (neuralnet package)

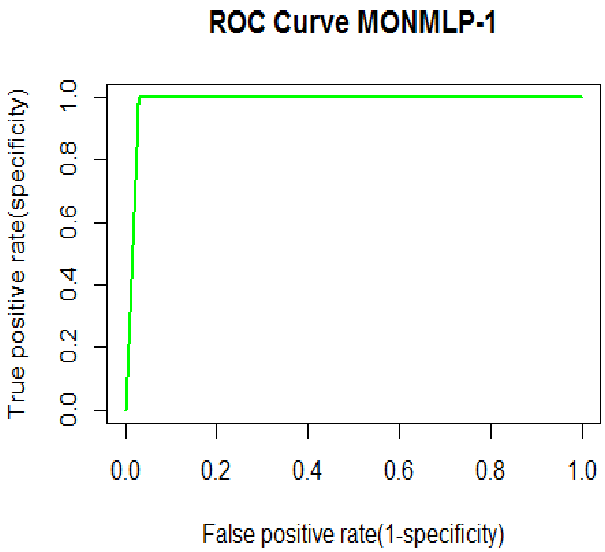
| Measures | Testing Data - 1 | Testing Data - 2 |
|---|---|---|
| Prediction Table | ## Reference<br>## Prediction 0 1<br>## 0 1638 55<br>## 1 2 970 | ## Reference<br>## Prediction 0 1<br>## 0 7624 79<br>## 1 9 2040 |
| Mean error | 0.02138836773 | 0.009023789992 |
| Accuracy | **0.9786116** | **0.9909762** |
| Sensitivity | 0.9987805 | 0.9988209 |
| Specificity | 0.9463415 | 0.9627183 |
| Pos Pred Value | 0.9675133 | 0.9897443 |
| Neg Pred Value | 0.9979424 | 0.9956076 |

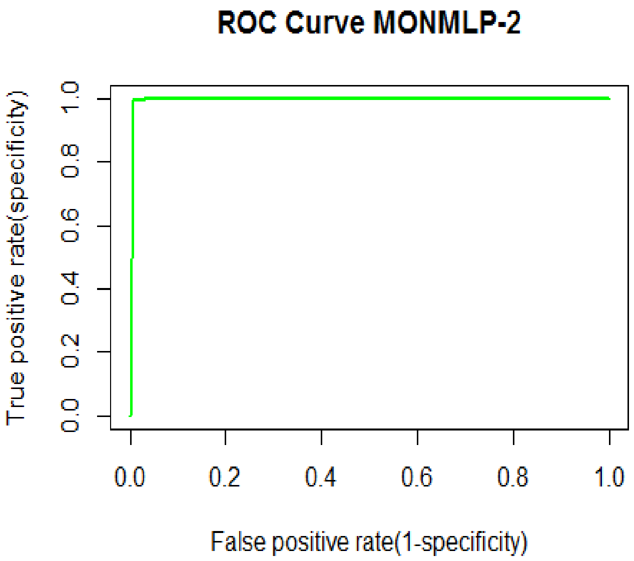


h)

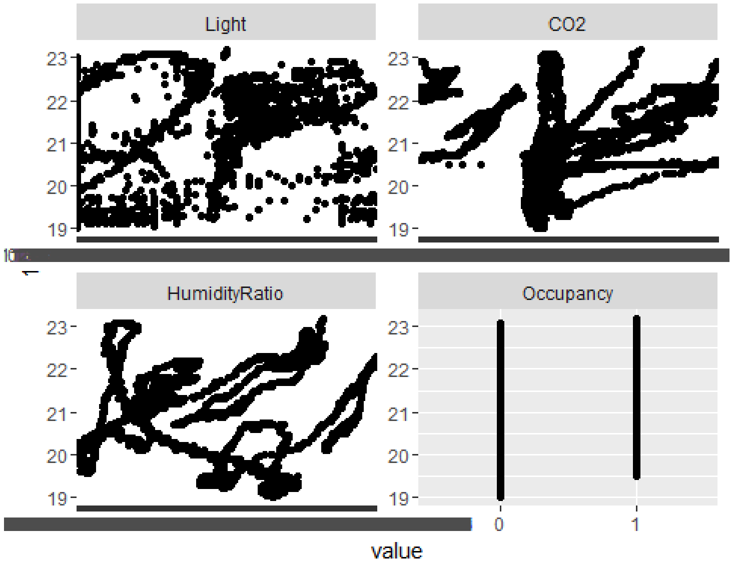
1. **Multilayer Perceptron (monmlp package)**

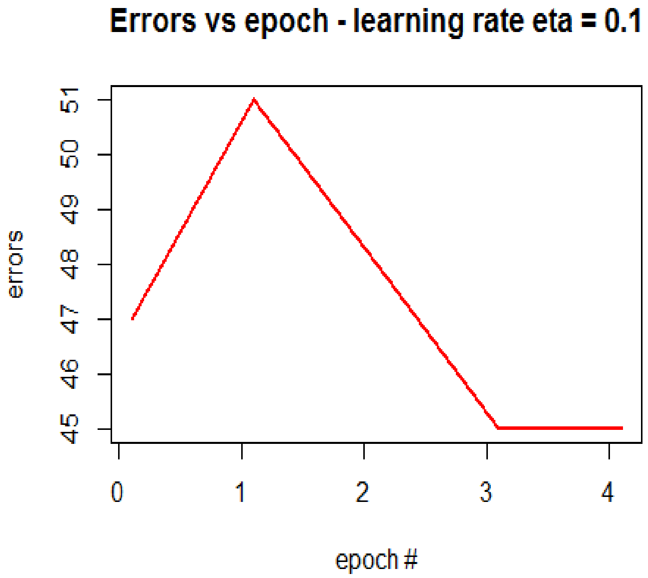
| Measures | Testing Data - 1 | Testing Data - 2 |
|---|---|---|
| Prediction Table | ## Reference<br>## Prediction 0 1<br>## 0 1638 55<br>## 1 2 970 | ## Reference<br>## Prediction 0 1<br>## 0 7609 94<br>## 1 10 2039 |
| Mean error | 0.02138837 | 0.01066448 |
| Accuracy | **0.9786** | **0.9893** |
| Sensitivity | 0.9988 | 0.9987 |
| Specificity | 0.9463 | 0.9559 |
| Pos Pred Value | 0.9675 | 0.9878 |
| Neg Pred Value | 0.9979 | 0.9951 |





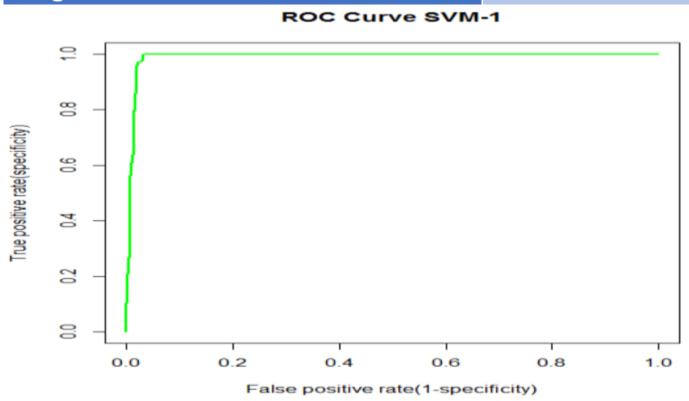
### 2. Perceptron Classification Algorithm

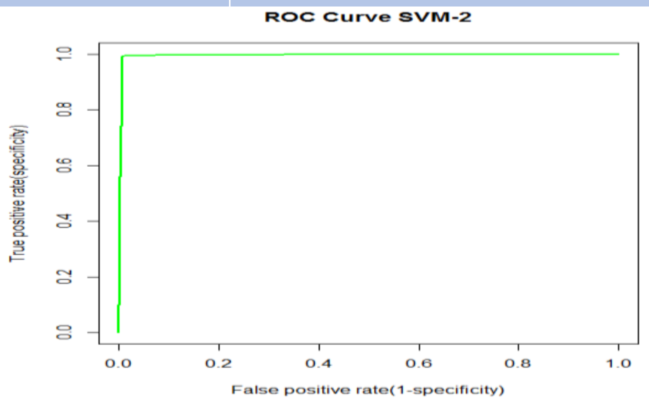




### i) Support Vector Machines

| Measures | Testing Data - 1 | Testing Data - 2 |
|---|---|---|
| Prediction Table | ##          Reference<br>## Prediction    0       1<br>##               0    1639    54<br>##               1      3       969 | ##          Reference<br>## Prediction    0       1<br>##               0    7644    59<br>##               1     14      2035 |
| Mean error | 0.02138837 | 0.007485644 |
| Accuracy | **0.9786116** | **0.9925144** |
| Sensitivity | 0.9982 | 0.9982 |
| Specificity | 0.9472 | 0.9718 |
| Pos Pred Value | 0.9681 | 0.9923 |
| Neg Pred Value | 0.9969 | 0.9932 |





## Comparisons

| | GLM | LDA | QDA | CART | KNN | Random Forest | ANN | MLP | SVM |
|---|---|---|---|---|---|---|---|---|---|
| Accuracy – Testing Data 1 | 0.9752 | 0.979 | 0.9775 | 0.9786 | 0.6544 | 0.94 | 0.9786116 | 0.9786 | 0.9786116 |
| Accuracy – Testing Data 2 | 0.9805 | 0.9913 | 0.977 | 0.9931 | NA | 0.9712 | 0.9909762 | 0.9893 | 0.9925144 |

## 7. Conclusions

The various classification techniques we applied as a part of this project, helped us to gain more understanding and clarity regarding the strengths and weaknesses of each and every classification model. By using R and the various packages like neuralnet, RSNNS, e1071 and monmlp; we were able to implement complex algorithms like artificial neural networks and support vector machines. As a part of our data analysis, we found that removing Humidity from the list of predictors highly improved the Accuracy of the model as it had low correlation with Occupancy.

From the analysis and implementation of the 9 classification techniques, we found that the accuracy was on a higher side for the advanced techniques of ANN, SVM and MLP as can be seen from the above table with minor fluctuations depending upon the testing data. Our observations of ROC graphs helped us to conclude that advanced techniques provide better results for Occupancy Detection.

The high accuracy rate of above 97% in this project forms an excellent baseline to build on for futuristic projects like ACHE i.e. adaptive control of home environment as mentioned in detail in the Literature Review. Furthermore, higher efficiency in energy management could be achieved due to the highly accurate baseline created in this project.

## 8. Individual Roles & Responsibilities

| **Team Member** | **Roles and Responsibilities** |
|---|---|
| Prajwal Gonnade | 1. Plan the project tasks<br>2. Perform Literature review<br>3. Data Analysis<br>4. Develop R code<br>5. Learn and apply neural network algorithm<br>6. Create Final project report |
| Supreet Nayak | 1. Perform Literature review<br>2. Data Visualization<br>3. Develop R code<br>4. Learn and apply neural network algorithm<br>5. Optimize the R code<br>6. Collate all project deliverables |

## 9. References

[2] "Artificial Neural Networks - A Tutorial"http://emotion.psychdept.arizona.edu/Jclub/Artificial%20Neural%20Networks%20-%20A%20Tutorial.pdf

[3] "Perceptron - Binary classification algorithm" https://rpubs.com/FaiHas/197581

[4] "Create and train a multi-layer perceptron (MLP)"http://artax.karlin.mff.cuni.cz/r-help/library/RSNNS/html/mlp.html

[5] "NeuralNetTools - Neural Network" https://beckmw.wordpress.com/tag/neural-network/

[6] "Multilayer perceptron neural network for downscaling rainfall in arid region: A case study of Baluchistan, Pakistan" https://pure.utm.my/en/publications/multilayer-perceptron-neural-network-for-downscaling-rainfall-in-

[7] "The use of multilayer perceptron artificial neural networks for the classification of ethanol samples by commercialization region" http://search.proquest.com/openview/952ffea1f5e8571c8e9e3a89ed826eab/1?pq-origsite=gscholar&cbl=2037655

[8] "Decomposition Techniques for Multilayer Perceptron Training" http://ieeexplore.ieee.org.ezproxy.library.tamu.edu/document/7273903/

[9] "Occupancy Detection in Commercial Buildings using Opportunistic Context Sources" http://www-07.ibm.com/in/research/documents/p469-ghai.pdf

## Appendix

**Appendix – 1 {GLM, LDA, QDA, CART, KNN, Random Forest}**

# Team14_Project_Occupancy_Detection

Prajwal Gonnade, Supreet Nayak

December 5, 2016

```
library(boot)

## Warning: package 'boot' was built under R version 3.2.5

library(caret)

## Warning: package 'caret' was built under R version 3.2.5

## Warning: package 'ggplot2' was built under R version 3.2.5

library(class)
library(ROCR)

## Warning: package 'ROCR' was built under R version 3.2.5

## Warning: package 'gplots' was built under R version 3.2.5

library(MASS)
library(tree)

## Warning: package 'tree' was built under R version 3.2.5

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.2.5

library(chemometrics)

## Warning: package 'chemometrics' was built under R version 3.2.5

library(reshape2)

## Warning: package 'reshape2' was built under R version 3.2.5
```

## Load CSV

```
Occupancy_Train <- read.csv(file.choose(),header=T)
Occupancy_Test1 <- read.csv(file.choose(),header=T)
Occupancy_Test2 <- read.csv(file.choose(),header=T)
```

## Analyzing Data

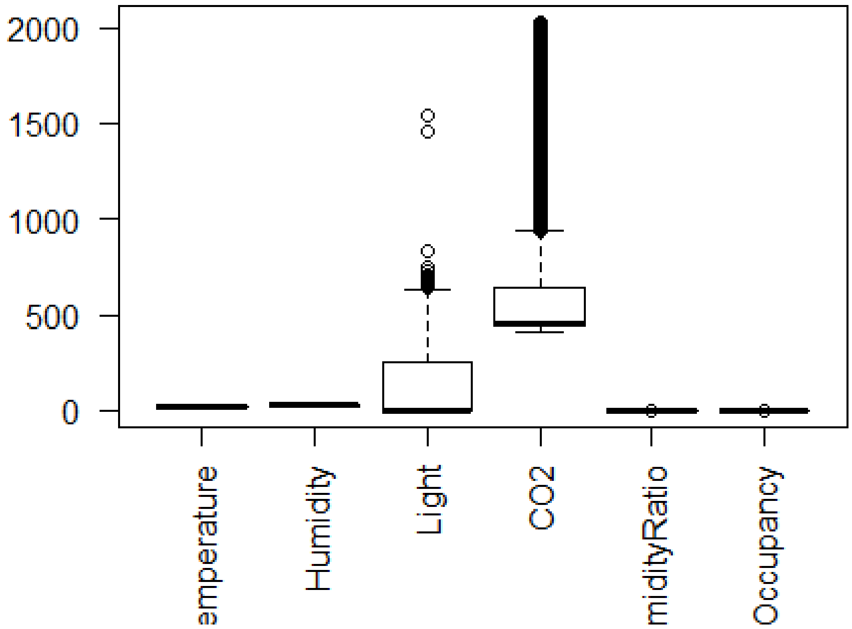
```
names(Occupancy_Train)

## [1] "date"          "Temperature"   "Humidity"      "Light"
## [5] "CO2"           "HumidityRatio" "Occupancy"

str(Occupancy_Train)

## 'data.frame':    8143 obs. of  7 variables:
##  $ date         : Factor w/ 8143 levels "2015-02-04 17:51:00",..: 1 2 3 4 5 6 7 8 9 10
```

```
...
##  $ Temperature  : num  23.2 23.1 23.1 23.1 23.1 ...
##  $ Humidity     : num  27.3 27.3 27.2 27.2 27.2 ...
##  $ Light        : num  426 430 426 426 426 ...
##  $ CO2          : num  721 714 714 708 704 ...
##  $ HumidityRatio: num  0.00479 0.00478 0.00478 0.00477 0.00476 ...
##  $ Occupancy    : int  1 1 1 1 1 1 1 1 1 1 ...
```

```r
summary(Occupancy_Train)
```

```
##                   date         Temperature       Humidity    
##  2015-02-04 17:51:00:   1   Min.   :19.00   Min.   :16.75  
##  2015-02-04 17:51:59:   1   1st Qu.:19.70   1st Qu.:20.20  
##  2015-02-04 17:53:00:   1   Median :20.39   Median :26.22  
##  2015-02-04 17:54:00:   1   Mean   :20.62   Mean   :25.73  
##  2015-02-04 17:55:00:   1   3rd Qu.:21.39   3rd Qu.:30.53  
##  2015-02-04 17:55:59:   1   Max.   :23.18   Max.   :39.12  
##  (Other)            :8137                                  
##      Light             CO2          HumidityRatio       Occupancy     
##  Min.   :   0.0   Min.   : 412.8   Min.   :0.002674   Min.   :0.0000  
##  1st Qu.:   0.0   1st Qu.: 439.0   1st Qu.:0.003078   1st Qu.:0.0000  
##  Median :   0.0   Median : 453.5   Median :0.003801   Median :0.0000  
##  Mean   : 119.5   Mean   : 606.5   Mean   :0.003863   Mean   :0.2123  
##  3rd Qu.: 256.4   3rd Qu.: 638.8   3rd Qu.:0.004352   3rd Qu.:0.0000  
##  Max.   :1546.3   Max.   :2028.5   Max.   :0.006476   Max.   :1.0000  
## 
```

```r
boxplot(Occupancy_Train[,-1],las=2)
```



```r
cor(Occupancy_Train[,-1])
```

```
##                Temperature    Humidity      Light       CO2 HumidityRatio
## Temperature      1.0000000 -0.14175931 0.64994184 0.5598938     0.1517616
## Humidity        -0.1417593  1.00000000 0.03782794 0.4390228     0.9551981
## Light            0.6499418  0.03782794 1.00000000 0.6640221     0.2304202
## CO2              0.5598938  0.43902276 0.66402206 1.0000000     0.6265559
## HumidityRatio    0.1517616  0.95519808 0.23042022 0.6265559     1.0000000
## Occupancy        0.5382197  0.13296424 0.90735211 0.7122352     0.3002816
##                Occupancy
## Temperature    0.5382197
## Humidity       0.1329642
## Light          0.9073521
## CO2            0.7122352
## HumidityRatio 0.3002816
## Occupancy      1.0000000
```

## Logistic Regression

glm(Occupancy ~ Temperature + Humidity + Light + CO2 + HumidityRatio, data = Occupancy_Train, family = "binomial")

```
Occupancy_glm <- glm(Occupancy ~ Temperature + Humidity + Light + CO2 + HumidityRatio, da
ta = Occupancy_Train, family = "binomial")
summary(Occupancy_glm)

## 
## Call:
## glm(formula = Occupancy ~ Temperature + Humidity + Light + CO2 + 
##     HumidityRatio, family = "binomial", data = Occupancy_Train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -6.9573  -0.0654  -0.0384  -0.0186   2.7332  
## 
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)    
## (Intercept)    8.695e+00  1.367e+01   0.636    0.525    
## Temperature   -9.039e-01  6.398e-01  -1.413    0.158    
## Humidity       3.099e-01  4.149e-01   0.747    0.455    
## Light          2.061e-02  7.662e-04  26.892   <2e-16 ***
## CO2            6.430e-03  5.667e-04  11.348   <2e-16 ***
## HumidityRatio -2.259e+03  2.682e+03  -0.842    0.400    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 8420.30  on 8142  degrees of freedom
## Residual deviance:  956.12  on 8137  degrees of freedom
## AIC: 968.12
## 
## Number of Fisher Scoring iterations: 8
```

## Explanation -

## Removing insignificant predictors

```
summary(glm(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupancy_Train
, family = "binomial"))
```

```
##
## Call:
## glm(formula = Occupancy ~ Temperature + Light + CO2 + HumidityRatio,
##     family = "binomial", data = Occupancy_Train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.9702  -0.0650  -0.0399  -0.0191   2.7333
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.873e+01  2.838e+00   6.602 4.07e-11 ***
## Temperature   -1.373e+00  1.428e-01  -9.613  < 2e-16 ***
## Light          2.061e-02  7.656e-04  26.915  < 2e-16 ***
## CO2            6.259e-03  5.131e-04  12.198  < 2e-16 ***
## HumidityRatio -2.590e+02  1.426e+02  -1.817   0.0692 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8420.30  on 8142  degrees of freedom
## Residual deviance:  956.68  on 8138  degrees of freedom
## AIC: 966.68
##
## Number of Fisher Scoring iterations: 8
```

### Final Model -

```
Occupancy_glm <- glm(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupa
ncy_Train, family = "binomial")
```

## Confusion Matrix on Testing Data - 1

```
glm_probs_1 = predict(Occupancy_glm, Occupancy_Test1, type = "response")
glm_pred_y_1 = rep(0, length(Occupancy_Test1$Occupancy))
glm_pred_y_1[glm_probs_1 > 0.5] = 1
table(glm_pred_y_1, Occupancy_Test1$Occupancy)
```

```
##
## glm_pred_y_1    0    1
##            0 1639   12
##            1   54  960
```

```
mean(glm_pred_y_1 != Occupancy_Test1$Occupancy)
```

```
## [1] 0.02476548
```

```
confusionMatrix(Occupancy_Test1$Occupancy, glm_pred_y_1)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 1639   54
##          1   12  960
##                                           
##                Accuracy : 0.9752          
##                  95% CI : (0.9686, 0.9808)
##     No Information Rate : 0.6195          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.947           
##  Mcnemar's Test P-Value : 4.494e-07       
##                                           
##             Sensitivity : 0.9927          
##             Specificity : 0.9467          
##          Pos Pred Value : 0.9681          
##          Neg Pred Value : 0.9877          
##              Prevalence : 0.6195          
##          Detection Rate : 0.6150          
##    Detection Prevalence : 0.6353          
##       Balanced Accuracy : 0.9697          
##                                           
##        'Positive' Class : 0               
## 
```

```
Test_MSE <- sum((glm_pred_y_1 - as.numeric(Occupancy_Test1$Occupancy))^2)/nrow(Occupancy_
Test1)
Test_MSE
```

```
## [1] 0.02476548
```

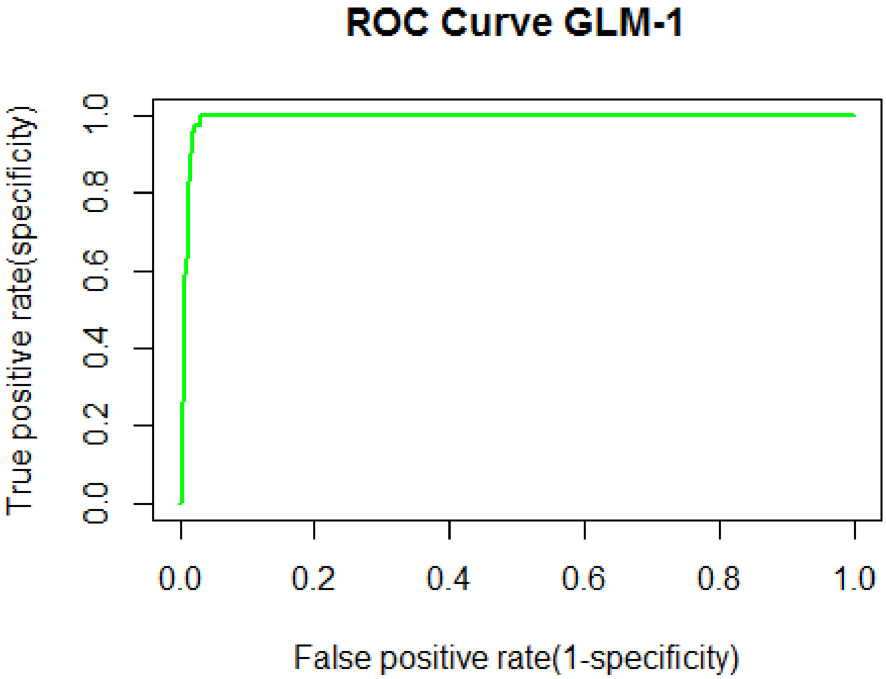
## ROC Analysis GLM - 1

```
roc.curve=function(s,print=FALSE){
Ps=(glm_probs_1>s)*1
FP=sum((Ps==1)*(Occupancy_Test1$Occupancy == 0))/sum(Occupancy_Test1$Occupancy == 0)
TP=sum((Ps==1)*(Occupancy_Test1$Occupancy == 1))/sum(Occupancy_Test1$Occupancy == 1)
if(print==TRUE){
print(table(Observed=Occupancy_Test1$Occupancy,Predicted=Ps))
}
vect=c(FP,TP)
names(vect)=c("FPR","TPR")
return(vect)
}
threshold = 0.5
roc.curve(threshold,print=TRUE)
```

```
##         Predicted
## Observed    0    1
##        0 1639   54
##        1   12  960
```

```
##        FPR        TPR 
## 0.03189604 0.98765432
```

```
ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,],M.ROC[2,], xlab='False positive rate(1-specificity)', ylab='True positive
rate(specificity)',main = 'ROC Curve GLM-1', col="green",lwd=2,type="l")
```



#Confusion Matrix on Testing Data - 2

```
glm_probs_1 = predict(Occupancy_glm, Occupancy_Test2, type = "response")
glm_pred_y_1 = rep(0, length(Occupancy_Test2$Occupancy))
glm_pred_y_1[glm_probs_1 > 0.5] = 1
table(glm_pred_y_1, Occupancy_Test2$Occupancy)

##
## glm_pred_y_1    0    1
##             0 7648  135
##             1   55 1914

mean(glm_pred_y_1 != Occupancy_Test2$Occupancy)

## [1] 0.01948318

confusionMatrix(Occupancy_Test2$Occupancy, glm_pred_y_1)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7648   55
##          1  135 1914
##
##                Accuracy : 0.9805
##                  95% CI : (0.9776, 0.9832)
##     No Information Rate : 0.7981
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.9404
##  Mcnemar's Test P-Value : 9.969e-09
##
##             Sensitivity : 0.9827
##             Specificity : 0.9721
##          Pos Pred Value : 0.9929
##          Neg Pred Value : 0.9341
##              Prevalence : 0.7981
##          Detection Rate : 0.7842
##    Detection Prevalence : 0.7899
##       Balanced Accuracy : 0.9774
##
##        'Positive' Class : 0
##
```

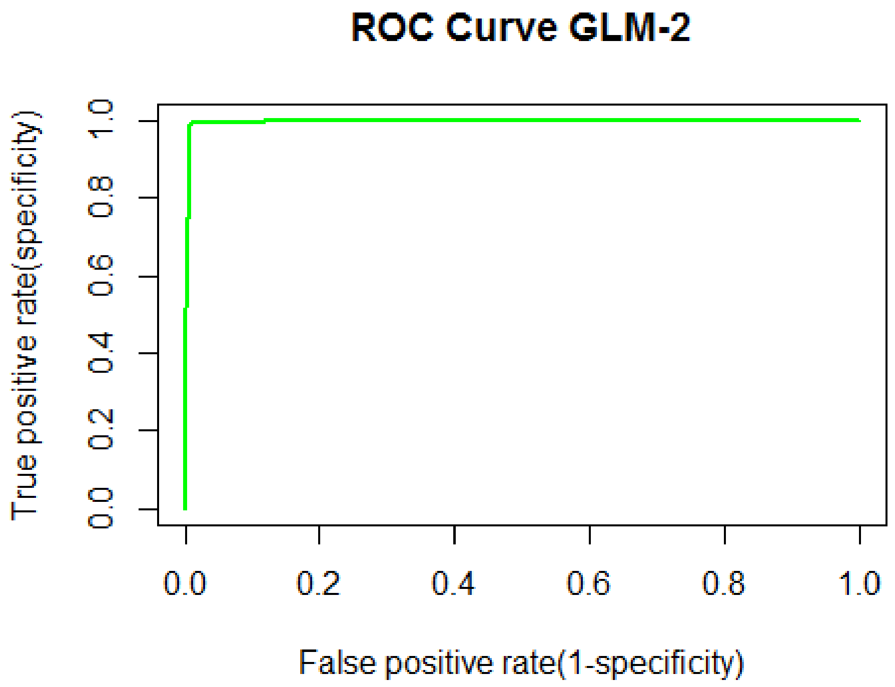
## ROC Analysis GLM - 2

```
roc.curve=function(s,print=FALSE){
Ps=(glm_probs_1>s)*1
FP=sum((Ps==1)*(Occupancy_Test2$Occupancy == 0))/sum(Occupancy_Test2$Occupancy == 0)
TP=sum((Ps==1)*(Occupancy_Test2$Occupancy == 1))/sum(Occupancy_Test2$Occupancy == 1)
if(print==TRUE){
print(table(Observed=Occupancy_Test2$Occupancy,Predicted=Ps))
}
vect=c(FP,TP)
names(vect)=c("FPR","TPR")
return(vect)
}
threshold = 0.5
roc.curve(threshold,print=TRUE)
```

```
##         Predicted
## Observed    0    1
##        0 7648   55
##        1  135 1914
```

```
##         FPR         TPR
## 0.007140075 0.934114202
```

```
ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,],M.ROC[2,], xlab='False positive rate(1-specificity)', ylab='True positive
rate(specificity)',main = 'ROC Curve GLM-2', col="green",lwd=2,type="l")
```



#LDA - Testing Data 1

```
Occupancy_lda <- lda(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupa
ncy_Train)
Occupancy_lda
```

```
## Call:
## lda(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupancy_Train)
##
## Prior probabilities of groups:
##         0         1
## 0.7876704 0.2123296
##
## Group means:
##   Temperature     Light       CO2 HumidityRatio
## 0    20.33493  27.77644  490.3203   0.003729632
## 1    21.67319 459.85435 1037.7048   0.004355428
##
## Coefficients of linear discriminants:
##                         LD1
## Temperature   -4.092751e-01
## Light          1.227500e-02
## CO2            2.326517e-03
## HumidityRatio -1.060925e+02
```

```
lda_pred = predict(Occupancy_lda, Occupancy_Test1)
names(lda_pred)
```

```
## [1] "class"     "posterior" "x"
```

```
table(lda_pred$class, Occupancy_Test1$Occupancy)
```

```
##
##        0    1
##   0 1638    1
##   1   55  971
```

```
mean(lda_pred$class != Occupancy_Test1$Occupancy)
```

```
## [1] 0.02101313
```

```
confusionMatrix(Occupancy_Test1$Occupancy,lda_pred$class)
```

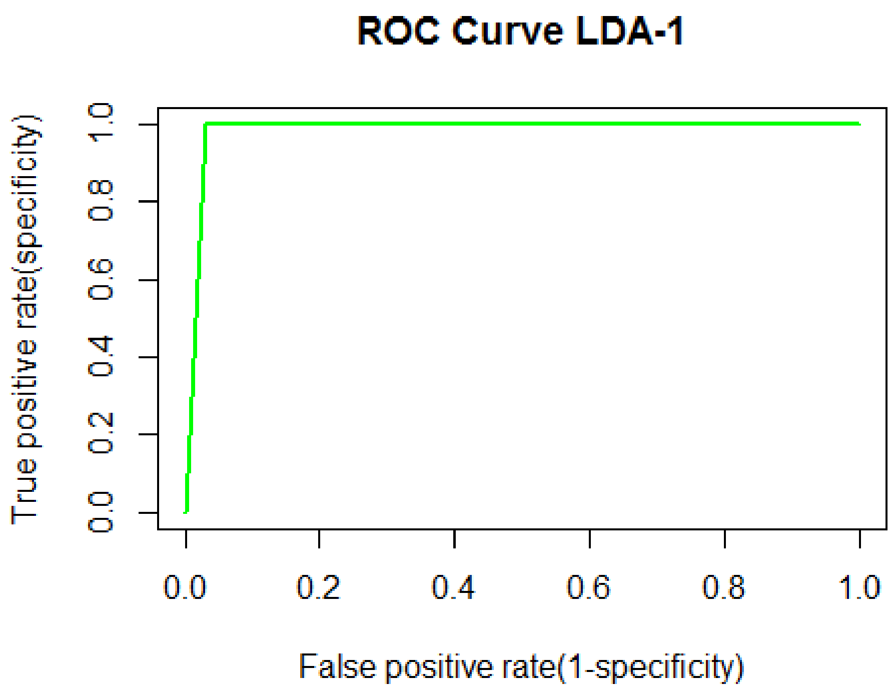
```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 1638   55
##          1    1  971
## 
##                Accuracy : 0.979
##                  95% CI : (0.9728, 0.9841)
##     No Information Rate : 0.615
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9552
##  Mcnemar's Test P-Value : 1.417e-12
## 
##             Sensitivity : 0.9994
##             Specificity : 0.9464
##          Pos Pred Value : 0.9675
##          Neg Pred Value : 0.9990
##              Prevalence : 0.6150
##          Detection Rate : 0.6146
##    Detection Prevalence : 0.6353
##       Balanced Accuracy : 0.9729
## 
##        'Positive' Class : 0
## 
```

## ROC Analysis LDA - Testing Data 1

```
S = lda_pred$posterior[,2]
roc.curve=function(s,print=FALSE){
Ps=(S>s)*1
FP=sum((Ps==1)*(Occupancy_Test1$Occupancy == 0))/sum(Occupancy_Test1$Occupancy == 0)
TP=sum((Ps==1)*(Occupancy_Test1$Occupancy == 1))/sum(Occupancy_Test1$Occupancy == 1)
if(print==TRUE){
print(table(Observed=Occupancy_Test1$Occupancy,Predicted=Ps))
}
vect=c(FP,TP)
names(vect)=c("FPR","TPR")
return(vect)
}
threshold = 0.5
roc.curve(threshold,print=TRUE)
```

```
##         Predicted
## Observed    0    1
##        0 1638   55
##        1    1  971
##         FPR        TPR
## 0.03248671 0.99897119
```

```
ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,],M.ROC[2,],xlab='False positive rate(1-specificity)', ylab='True positive r
ate(specificity)',main = 'ROC Curve LDA-1', col="green",lwd=2,type="l")
```



#LDA - Testing Data 2

```
Occupancy_lda <- lda(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupa
ncy_Train)
Occupancy_lda
```

```
## Call:
## lda(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupancy_Train)
## 
## Prior probabilities of groups:
##         0         1 
## 0.7876704 0.2123296 
## 
## Group means:
##   Temperature     Light       CO2 HumidityRatio
## 0    20.33493  27.77644  490.3203   0.003729632
## 1    21.67319 459.85435 1037.7048   0.004355428
## 
## Coefficients of linear discriminants:
##                         LD1
## Temperature   -4.092751e-01
## Light          1.227500e-02
## CO2            2.326517e-03
## HumidityRatio -1.060925e+02
```

```
lda_pred = predict(Occupancy_lda, Occupancy_Test2)
names(lda_pred)
```

```
## [1] "class"     "posterior" "x"
```

```
table(lda_pred$class, Occupancy_Test2$Occupancy)
```

```
## 
##        0    1
##   0 7626    8
##   1   77 2041
```

```
mean(lda_pred$class != Occupancy_Test2$Occupancy)
```

```
## [1] 0.008716161
```

```
confusionMatrix(Occupancy_Test2$Occupancy,lda_pred$class)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 7626   77
##          1    8 2041
## 
##                Accuracy : 0.9913
##                  95% CI : (0.9892, 0.993)
##     No Information Rate : 0.7828
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9741
##  Mcnemar's Test P-Value : 1.636e-13
## 
##             Sensitivity : 0.9990
##             Specificity : 0.9636
##          Pos Pred Value : 0.9900
##          Neg Pred Value : 0.9961
##              Prevalence : 0.7828
##          Detection Rate : 0.7820
##    Detection Prevalence : 0.7899
##       Balanced Accuracy : 0.9813
## 
##        'Positive' Class : 0
## 
```

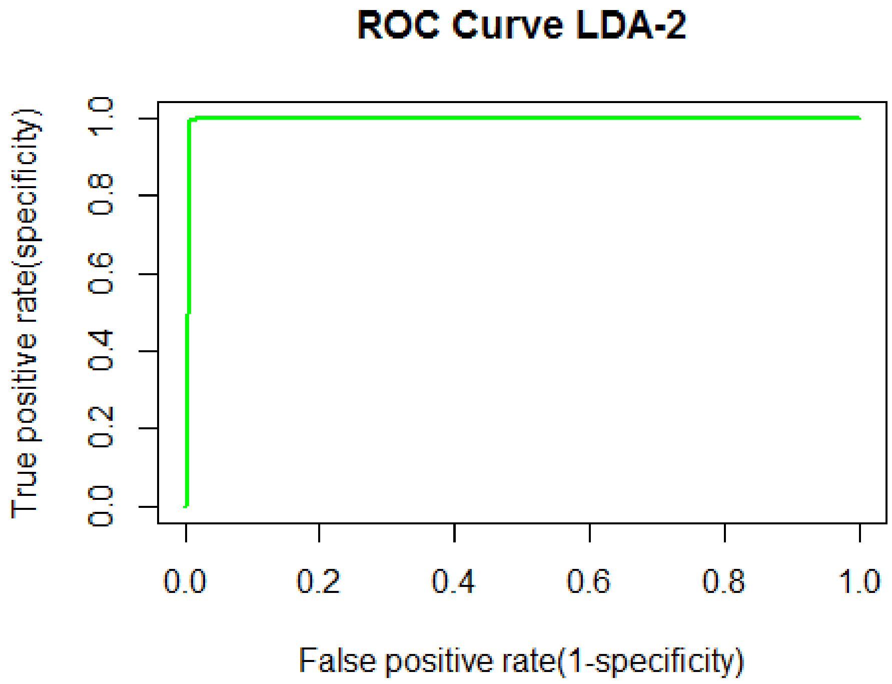
## ROC Analysis LDA - Testing Data 2

```
S = lda_pred$posterior[,2]
roc.curve=function(s,print=FALSE){
Ps=(S>s)*1
FP=sum((Ps==1)*(Occupancy_Test2$Occupancy == 0))/sum(Occupancy_Test2$Occupancy == 0)
TP=sum((Ps==1)*(Occupancy_Test2$Occupancy == 1))/sum(Occupancy_Test2$Occupancy == 1)
if(print==TRUE){
print(table(Observed=Occupancy_Test2$Occupancy,Predicted=Ps))
}
vect=c(FP,TP)
names(vect)=c("FPR","TPR")
return(vect)
}
threshold = 0.5
roc.curve(threshold,print=TRUE)
```

```
##          Predicted
## Observed    0    1
##         0 7626   77
##         1    8 2041
```

```
##         FPR         TPR
## 0.009996105 0.996095656
```

```
ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,],M.ROC[2,],xlab='False positive rate(1-specificity)', ylab='True positive r
ate(specificity)',main = 'ROC Curve LDA-2', col="green",lwd=2,type="l")
```



## Cross Validation LDA

```
inds=sample(1:nrow(Occupancy_Train),0.9*nrow(Occupancy_Train))
df.train=Occupancy_Train[inds,]
df.test=Occupancy_Train[-inds,]
train.model = lda(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupancy
_Train)
preds=predict(train.model, df.test)
```

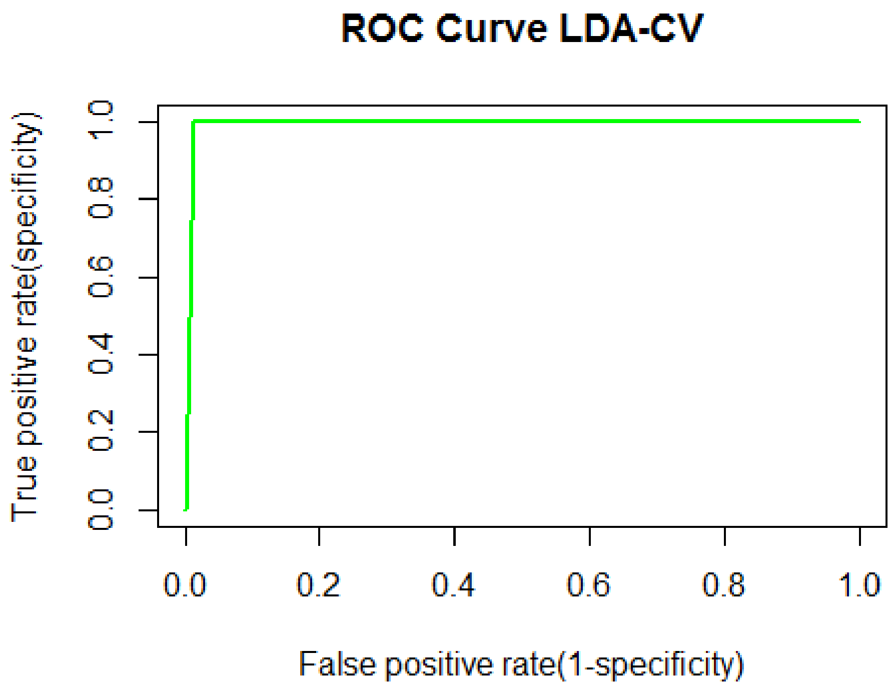
## ROC Analysis LDA - CV

```
S = preds$posterior[,2]
roc.curve=function(s,print=FALSE){
Ps=(S>s)*1
FP=sum((Ps==1)*(df.test$Occupancy == 0))/sum(df.test$Occupancy == 0)
TP=sum((Ps==1)*(df.test$Occupancy == 1))/sum(df.test$Occupancy == 1)
if(print==TRUE){
print(table(Observed=df.test$Occupancy,Predicted=Ps))
}
vect=c(FP,TP)
names(vect)=c("FPR","TPR")
return(vect)
}
threshold = 0.5
roc.curve(threshold,print=TRUE)
```

```
##          Predicted
## Observed   0   1
##         0 629   9
##         1   0 177
```

```
##        FPR        TPR
## 0.01410658 1.00000000
```

```
ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,],M.ROC[2,],xlab='False positive rate(1-specificity)', ylab='True positive r
ate(specificity)',main = 'ROC Curve LDA-CV', col="green",lwd=2,type="l")
```



## QDA - Testing Data 1

```
Occupancy_qda <- qda(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupa
ncy_Train)
Occupancy_qda
```

```
## Call:
## qda(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupancy_Train)
##
## Prior probabilities of groups:
##         0         1
## 0.7876704 0.2123296
##
## Group means:
##   Temperature     Light       CO2 HumidityRatio
## 0    20.33493  27.77644  490.3203   0.003729632
## 1    21.67319 459.85435 1037.7048   0.004355428
```

```
qda.pred <- predict(Occupancy_qda, Occupancy_Test1)
table(qda.pred$class, Occupancy_Test1$Occupancy)
```

```
##
##        0    1
##   0 1639    6
##   1   54  966
```

```
mean(qda.pred$class != Occupancy_Test1$Occupancy)
```

```
## [1] 0.02251407
```

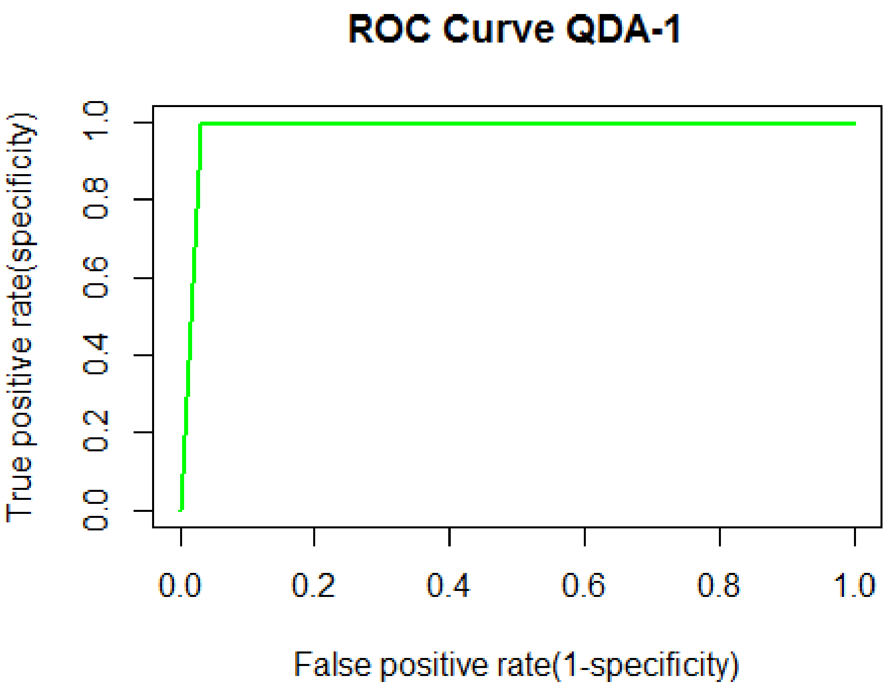
```
confusionMatrix(Occupancy_Test1$Occupancy,qda.pred$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1639   54
##          1    6  966
##
##                Accuracy : 0.9775
##                  95% CI : (0.9711, 0.9828)
##     No Information Rate : 0.6173
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9519
##  Mcnemar's Test P-Value : 1.298e-09
##
##             Sensitivity : 0.9964
##             Specificity : 0.9471
##          Pos Pred Value : 0.9681
##          Neg Pred Value : 0.9938
##              Prevalence : 0.6173
##          Detection Rate : 0.6150
##    Detection Prevalence : 0.6353
##       Balanced Accuracy : 0.9717
##
##        'Positive' Class : 0
##
```

## ROC Analysis QDA - Testing Data 1

```
S = qda.pred$posterior[,2]
roc.curve=function(s,print=FALSE){
```

```
Ps=(S > s)*1
FP=sum((Ps==1)*(Occupancy_Test1$Occupancy == 0))/sum(Occupancy_Test1$Occupancy == 0)
TP=sum((Ps==1)*(Occupancy_Test1$Occupancy == 1))/sum(Occupancy_Test1$Occupancy == 1)
if(print==TRUE){
print(table(Observed=Occupancy_Test1$Occupancy,Predicted=Ps))
}
vect=c(FP,TP)
names(vect)=c("FPR","TPR")
return(vect)
}
threshold = 0.5
roc.curve(threshold,print=TRUE)

##          Predicted
## Observed    0    1
##        0 1639   54
##        1    6  966

##        FPR        TPR
## 0.03189604 0.99382716

ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,],M.ROC[2,],xlab='False positive rate(1-specificity)', ylab='True positive r
ate(specificity)',main = 'ROC Curve QDA-1', col="green",lwd=2,type="l")
```



#QDA - Testing Data 2

```
Occupancy_qda <- qda(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupa
ncy_Train)
Occupancy_qda

## Call:
## qda(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupancy_Train)
```

```
##
## Prior probabilities of groups:
##         0         1
## 0.7876704 0.2123296
##
## Group means:
##   Temperature     Light       CO2 HumidityRatio
## 0    20.33493  27.77644  490.3203   0.003729632
## 1    21.67319 459.85435 1037.7048   0.004355428
```

```
qda.pred <- predict(Occupancy_qda, Occupancy_Test2)
table(qda.pred$class, Occupancy_Test2$Occupancy)
```

```
##
##        0    1
##   0 7645  166
##   1   58 1883
```

```
mean(qda.pred$class != Occupancy_Test2$Occupancy)
```

```
## [1] 0.02296965
```

```
confusionMatrix(Occupancy_Test2$Occupancy,qda.pred$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7645   58
##          1  166 1883
##
##                Accuracy : 0.977
##                  95% CI : (0.9739, 0.9799)
##     No Information Rate : 0.801
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9294
##  Mcnemar's Test P-Value : 8.726e-13
##
##             Sensitivity : 0.9787
##             Specificity : 0.9701
##          Pos Pred Value : 0.9925
##          Neg Pred Value : 0.9190
##              Prevalence : 0.8010
##          Detection Rate : 0.7839
##    Detection Prevalence : 0.7899
##       Balanced Accuracy : 0.9744
##
##        'Positive' Class : 0
##
```

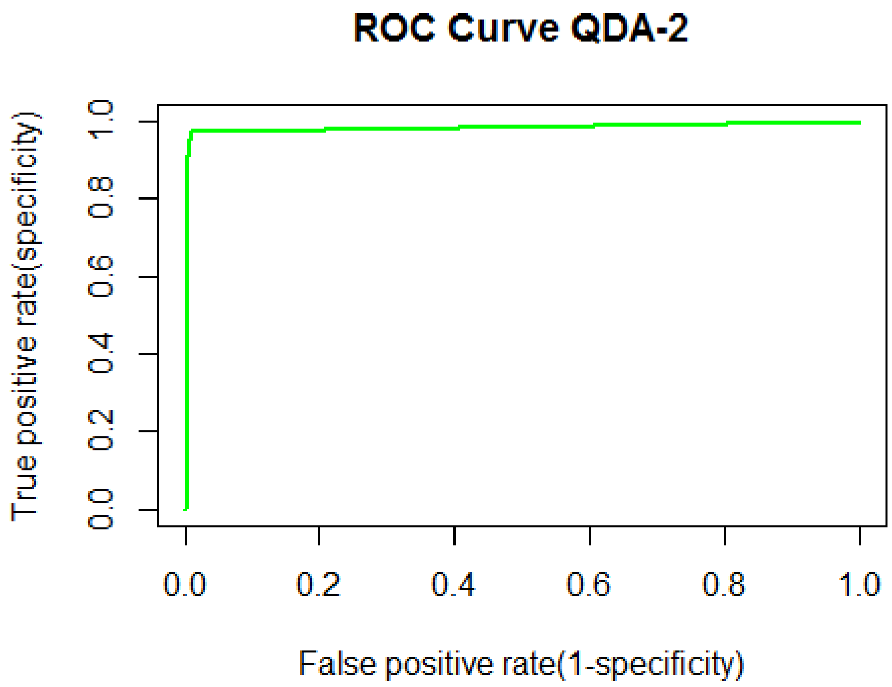
## ROC Analysis QDA - Testing Data 2

```
S = qda.pred$posterior[,2]
roc.curve=function(s,print=FALSE){
Ps=(S > s)*1
FP=sum((Ps==1)*(Occupancy_Test2$Occupancy == 0))/sum(Occupancy_Test2$Occupancy == 0)
TP=sum((Ps==1)*(Occupancy_Test2$Occupancy == 1))/sum(Occupancy_Test2$Occupancy == 1)
if(print==TRUE){
print(table(Observed=Occupancy_Test2$Occupancy,Predicted=Ps))
}
vect=c(FP,TP)
names(vect)=c("FPR","TPR")
return(vect)
}
threshold = 0.5
roc.curve(threshold,print=TRUE)
```

```
##           Predicted
## Observed    0    1
##         0 7645   58
##         1  166 1883
```

```
##         FPR         TPR
## 0.007529534 0.918984871
```

```
ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,],M.ROC[2,],xlab='False positive rate(1-specificity)', ylab='True positive r
ate(specificity)',main = 'ROC Curve QDA-2', col="green",lwd=2,type="l")
```

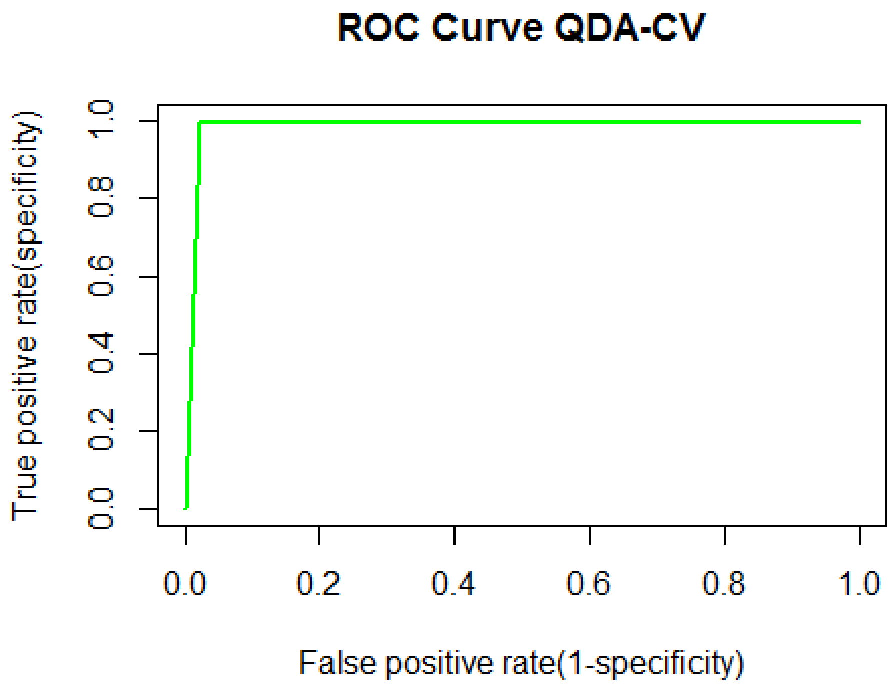


#Cross Validation QDA - 10 Fold

```
inds=sample(1:nrow(Occupancy_Train),0.9*nrow(Occupancy_Train))
df.train=Occupancy_Train[inds,]
df.test=Occupancy_Train[-inds,]
train.model = qda(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data = Occupancy
_Train)
preds=predict(train.model, df.test)
```

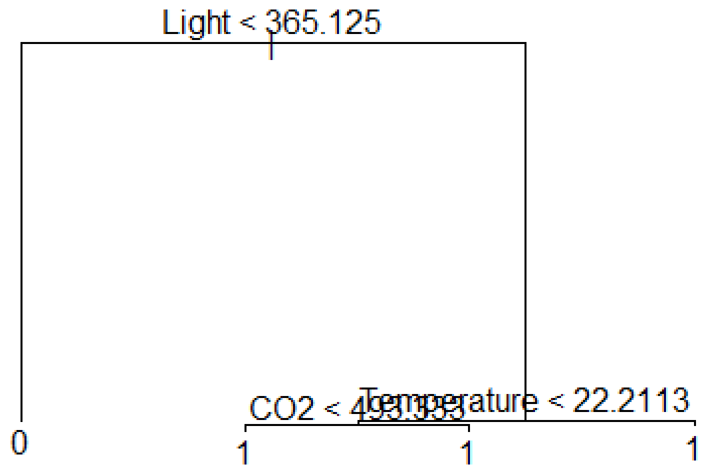
## ROC Analysis QDA - CV

```
S = preds$posterior[,2]
roc.curve=function(s,print=FALSE){
Ps=(S>s)*1
FP=sum((Ps==1)*(df.test$Occupancy == 0))/sum(df.test$Occupancy == 0)
TP=sum((Ps==1)*(df.test$Occupancy == 1))/sum(df.test$Occupancy == 1)
if(print==TRUE){
print(table(Observed=df.test$Occupancy,Predicted=Ps))
}
vect=c(FP,TP)
names(vect)=c("FPR","TPR")
return(vect)
}
threshold = 0.5
roc.curve(threshold,print=TRUE)

##          Predicted
## Observed   0   1
##        0 641  14
##        1   2 158

##        FPR        TPR
## 0.02137405 0.98750000

ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,],M.ROC[2,],xlab='False positive rate(1-specificity)', ylab='True positive rate(specificity)',main = 'ROC Curve QDA-CV', col="green",lwd=2,type="l")
```



## CART - Testing Data 1

```
set.seed(2)
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
str(Occupancy_Train)

## 'data.frame':    8143 obs. of  7 variables:
##  $ date         : Factor w/ 8143 levels "2015-02-04 17:51:00",..: 1 2 3 4 5 6 7 8 9 10
...
##  $ Temperature  : num  23.2 23.1 23.1 23.1 23.1 ...
##  $ Humidity     : num  27.3 27.3 27.2 27.2 27.2 ...
##  $ Light        : num  426 430 426 426 426 ...
##  $ CO2          : num  721 714 714 708 704 ...
##  $ HumidityRatio: num  0.00479 0.00478 0.00478 0.00477 0.00476 ...
##  $ Occupancy    : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...

tree.Occupancy_Train <- tree(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data
= Occupancy_Train)
plot(tree.Occupancy_Train)
summary(tree.Occupancy_Train)

##
## Classification tree:
## tree(formula = Occupancy ~ Temperature + Light + CO2 + HumidityRatio,
##     data = Occupancy_Train)
## Variables actually used in tree construction:
## [1] "Light"       "Temperature" "CO2"
## Number of terminal nodes:  4
## Residual mean deviance:  0.07946 = 646.8 / 8139
## Misclassification error rate: 0.01216 = 99 / 8143

text(tree.Occupancy_Train, pretty = 0)
```

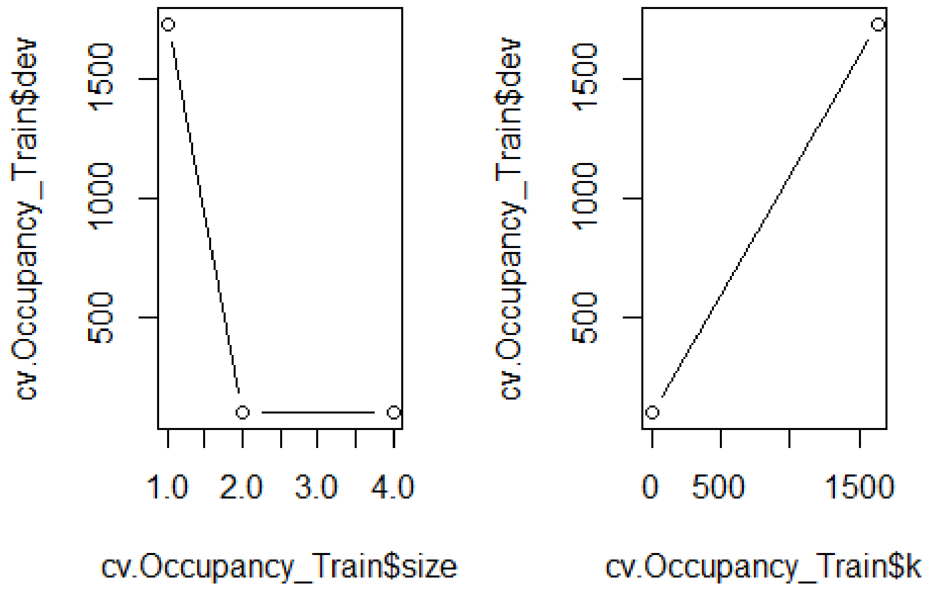


```
tree.pred <- predict(tree.Occupancy_Train, Occupancy_Test1, type = "class")
table(tree.pred, Occupancy_Test1$Occupancy)

##
## tree.pred    0    1
##         0 1639    3
##         1   54  969

mean(tree.pred != Occupancy_Test1$Occupancy)

## [1] 0.02138837

confusionMatrix(Occupancy_Test1$Occupancy, tree.pred)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1639   54
##          1    3  969
##
##                Accuracy : 0.9786
##                  95% CI : (0.9724, 0.9838)
##     No Information Rate : 0.6161
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9544
##  Mcnemar's Test P-Value : 3.528e-11
##
##             Sensitivity : 0.9982
##             Specificity : 0.9472
##          Pos Pred Value : 0.9681
##          Neg Pred Value : 0.9969
##              Prevalence : 0.6161
##          Detection Rate : 0.6150
##    Detection Prevalence : 0.6353
##       Balanced Accuracy : 0.9727
##
##        'Positive' Class : 0
##
```

## Cross Validation and Pruning - Testing Data 1

```
set.seed(3)
cv.Occupancy_Train <- cv.tree(tree.Occupancy_Train, FUN=prune.misclass)
names(cv.Occupancy_Train)

## [1] "size"   "dev"    "k"      "method"

cv.Occupancy_Train

## $size
## [1] 4 2 1
##
## $dev
## [1]  101  101 1729
##
## $k
## [1] -Inf    0 1630
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

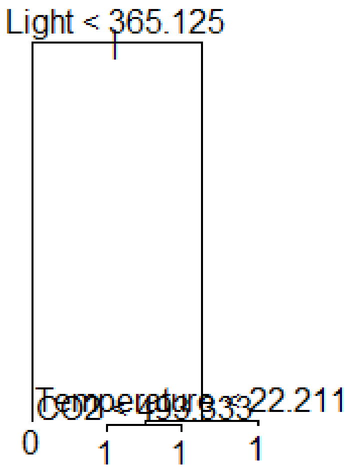
```
par(mfrow = c(1,2))
plot(cv.Occupancy_Train$size, cv.Occupancy_Train$dev, type = "b")
plot(cv.Occupancy_Train$k, cv.Occupancy_Train$dev, type = "b")
```



```
prune.Occupancy_Train = prune.misclass(tree.Occupancy_Train, best = 9)

## Warning in prune.tree(tree = tree.Occupancy_Train, best = 9, method =
## "misclass"): best is bigger than tree size

#prune.mt2Data_Train = prune.misclass(tree.mt2Data_Train)
plot(prune.Occupancy_Train)
text(prune.Occupancy_Train, pretty = 0)
tree.pred = predict(prune.Occupancy_Train, Occupancy_Test1, type = "class")
#tree.pred = predict(prune.mt2Data_Train, mt2Data_Test)
table(tree.pred, Occupancy_Test1$Occupancy)

##
## tree.pred    0    1
##         0 1639    3
##         1   54  969

mean(tree.pred != Occupancy_Test1$Occupancy)

## [1] 0.02138837
```

```
confusionMatrix(Occupancy_Test1$Occupancy, tree.pred)
```

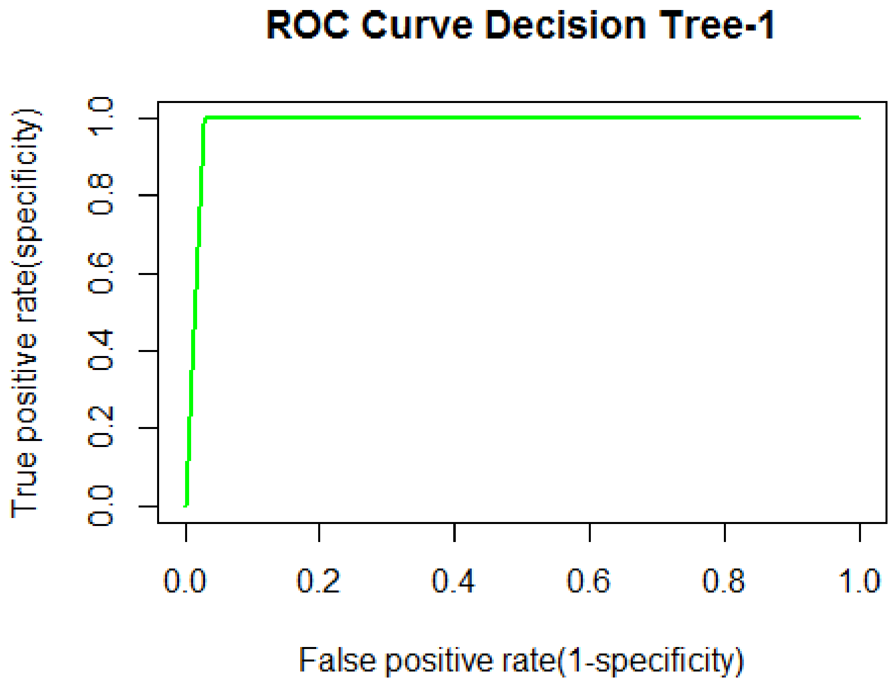
```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 1639   54
##          1    3  969
##                                           
##                Accuracy : 0.9786          
##                  95% CI : (0.9724, 0.9838)
##     No Information Rate : 0.6161          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9544          
##  Mcnemar's Test P-Value : 3.528e-11       
##                                           
##             Sensitivity : 0.9982          
##             Specificity : 0.9472          
##          Pos Pred Value : 0.9681          
##          Neg Pred Value : 0.9969          
##              Prevalence : 0.6161          
##          Detection Rate : 0.6150          
##    Detection Prevalence : 0.6353          
##       Balanced Accuracy : 0.9727          
##                                           
##        'Positive' Class : 0               
## 
```



#ROC Analysis Decision Trees - Testing Data 1

```
tree.pred = predict(prune.Occupancy_Train, Occupancy_Test1, type = "vector")
#tree.prob <- attr(tree.pred, "vector")
```

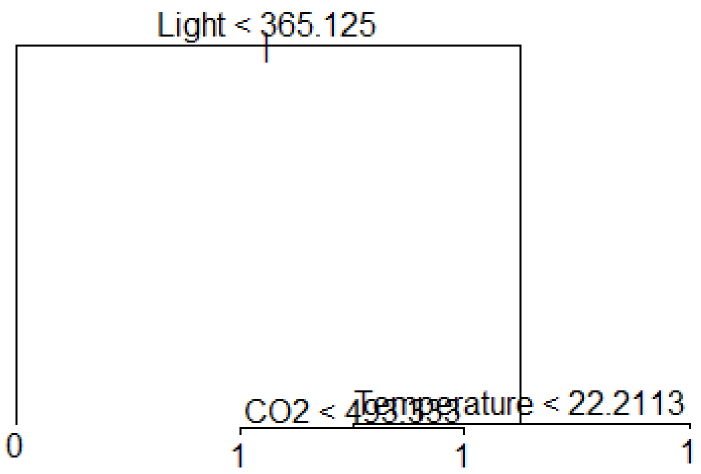
```
roc.curve=function(s,print=FALSE){
Ps=(tree.pred[,2]>s)*1
FP=sum((Ps==1)*(Occupancy_Test1$Occupancy == 0))/sum(Occupancy_Test1$Occupancy == 0)
TP=sum((Ps==1)*(Occupancy_Test1$Occupancy == 1))/sum(Occupancy_Test1$Occupancy == 1)
if(print==TRUE){
print(table(Observed=Occupancy_Test1$Occupancy,Predicted=Ps))
}
vect=c(FP,TP)
names(vect)=c("FPR","TPR")
return(vect)
}
threshold = 0.5
roc.curve(threshold,print=TRUE)

##          Predicted
## Observed    0    1
##         0 1639   54
##         1    3  969

##         FPR         TPR
## 0.03189604 0.99691358

ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,],M.ROC[2,],xlab='False positive rate(1-specificity)', ylab='True positive r
ate(specificity)', main = 'ROC Curve Decision Tree-1', col="green",lwd=2,type="l")
```



## CART - Testing Data 2

```
set.seed(2)
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
str(Occupancy_Train)
```

```
## 'data.frame':    8143 obs. of  7 variables:
##  $ date         : Factor w/ 8143 levels "2015-02-04 17:51:00",..: 1 2 3 4 5 6 7 8 9 10
...
##  $ Temperature  : num  23.2 23.1 23.1 23.1 23.1 ...
##  $ Humidity     : num  27.3 27.3 27.2 27.2 27.2 ...
##  $ Light        : num  426 430 426 426 426 ...
##  $ CO2          : num  721 714 714 708 704 ...
##  $ HumidityRatio: num  0.00479 0.00478 0.00478 0.00477 0.00476 ...
##  $ Occupancy    : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...

tree.Occupancy_Train <- tree(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data
= Occupancy_Train)
plot(tree.Occupancy_Train)
summary(tree.Occupancy_Train)

##
## Classification tree:
## tree(formula = Occupancy ~ Temperature + Light + CO2 + HumidityRatio,
##     data = Occupancy_Train)
## Variables actually used in tree construction:
## [1] "Light"       "Temperature" "CO2"
## Number of terminal nodes:  4
## Residual mean deviance:  0.07946 = 646.8 / 8139
## Misclassification error rate: 0.01216 = 99 / 8143

text(tree.Occupancy_Train, pretty = 0)
```



```
tree.pred <- predict(tree.Occupancy_Train, Occupancy_Test2, type = "class")
table(tree.pred, Occupancy_Test2$Occupancy)

##
## tree.pred    0    1
##         0 7648   12
##         1   55 2037
```

```
mean(tree.pred != Occupancy_Test2$Occupancy)
```

```
## [1] 0.006870386
```

```
confusionMatrix(Occupancy_Test2$Occupancy, tree.pred)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 7648   55
##          1   12 2037
## 
##                Accuracy : 0.9931
##                  95% CI : (0.9913, 0.9947)
##     No Information Rate : 0.7855
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9795
##  Mcnemar's Test P-Value : 2.88e-07
## 
##             Sensitivity : 0.9984
##             Specificity : 0.9737
##          Pos Pred Value : 0.9929
##          Neg Pred Value : 0.9941
##              Prevalence : 0.7855
##          Detection Rate : 0.7842
##    Detection Prevalence : 0.7899
##       Balanced Accuracy : 0.9861
## 
##        'Positive' Class : 0
## 
```

## Cross Validation and Pruning - Testing Data 2

```
set.seed(3)
cv.Occupancy_Train <- cv.tree(tree.Occupancy_Train, FUN=prune.misclass)
names(cv.Occupancy_Train)
```

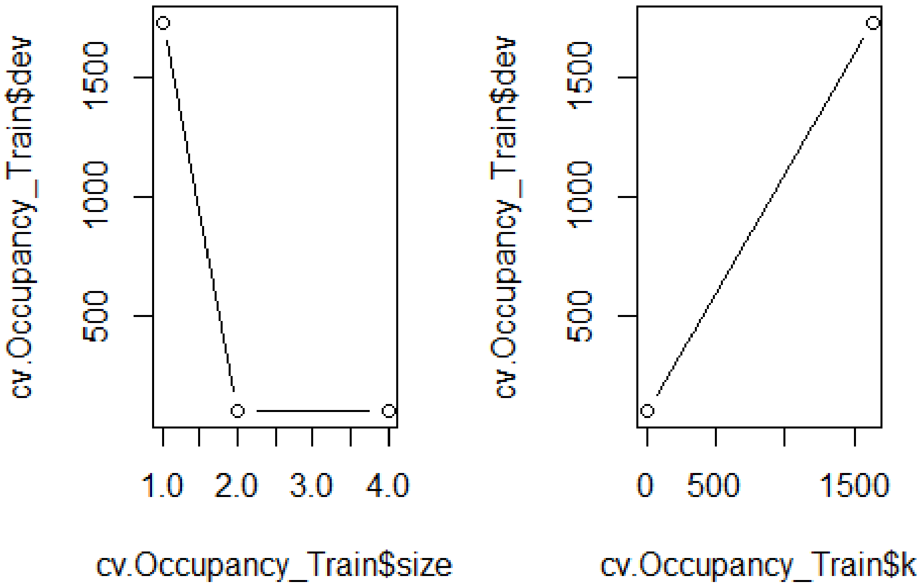
```
## [1] "size"   "dev"    "k"      "method"
```

```
cv.Occupancy_Train
```

```
## $size
## [1] 4 2 1
## 
## $dev
## [1]  101  101 1729
## 
## $k
## [1] -Inf    0 1630
## 
## $method
## [1] "misclass"
```

```
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
par(mfrow = c(1,2))
plot(cv.Occupancy_Train$size, cv.Occupancy_Train$dev, type = "b")
plot(cv.Occupancy_Train$k, cv.Occupancy_Train$dev, type = "b")
```



```
prune.Occupancy_Train = prune.misclass(tree.Occupancy_Train, best = 9)
```

```
## Warning in prune.tree(tree = tree.Occupancy_Train, best = 9, method =
## "misclass"): best is bigger than tree size
```

```
#prune.mt2Data_Train = prune.misclass(tree.mt2Data_Train)
plot(prune.Occupancy_Train)
text(prune.Occupancy_Train, pretty = 0)
tree.pred = predict(prune.Occupancy_Train, Occupancy_Test2, type = "class")
#tree.pred = predict(prune.mt2Data_Train, mt2Data_Test)
table(tree.pred, Occupancy_Test2$Occupancy)
```

```
##
## tree.pred    0    1
##         0 7648   12
##         1   55 2037
```

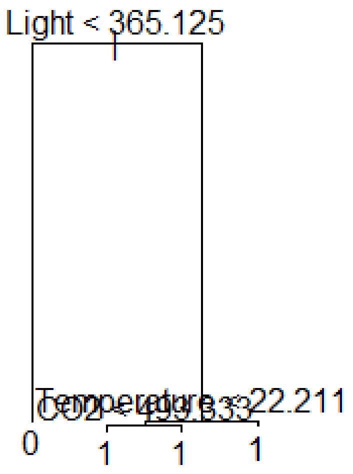
```
mean(tree.pred != Occupancy_Test2$Occupancy)
```

```
## [1] 0.006870386
```

```
confusionMatrix(Occupancy_Test2$Occupancy, tree.pred)
```

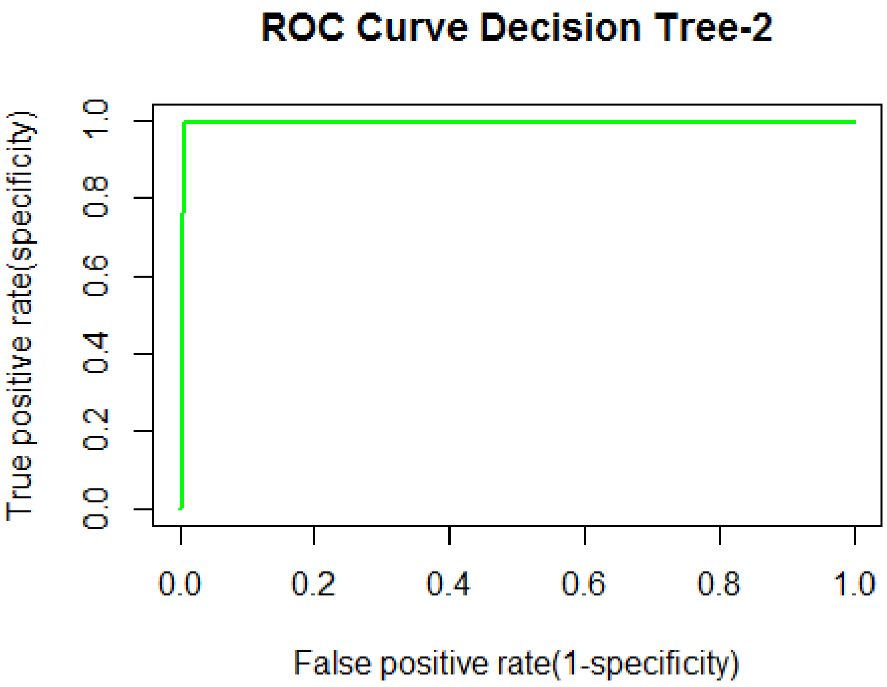
```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    0    1
##          0 7648   55
##          1   12 2037
##                                           
##                Accuracy : 0.9931          
##                  95% CI : (0.9913, 0.9947)
##     No Information Rate : 0.7855          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9795          
##  Mcnemar's Test P-Value : 2.88e-07        
##                                           
##             Sensitivity : 0.9984          
##             Specificity : 0.9737          
##          Pos Pred Value : 0.9929          
##          Neg Pred Value : 0.9941          
##              Prevalence : 0.7855          
##          Detection Rate : 0.7842          
##    Detection Prevalence : 0.7899          
##       Balanced Accuracy : 0.9861          
##                                           
##        'Positive' Class : 0               
##
```



#ROC Analysis Decision Trees - Testing Data 2

```
tree.pred = predict(prune.Occupancy_Train, Occupancy_Test2, type = "vector")
#tree.prob <- attr(tree.pred, "vector")
roc.curve=function(s,print=FALSE){
Ps=(tree.pred[,2]>s)*1
FP=sum((Ps==1)*(Occupancy_Test2$Occupancy == 0))/sum(Occupancy_Test2$Occupancy == 0)
TP=sum((Ps==1)*(Occupancy_Test2$Occupancy == 1))/sum(Occupancy_Test2$Occupancy == 1)
if(print==TRUE){
print(table(Observed=Occupancy_Test2$Occupancy,Predicted=Ps))
}
vect=c(FP,TP)
names(vect)=c("FPR","TPR")
return(vect)
}
threshold = 0.5
roc.curve(threshold,print=TRUE)

##          Predicted
## Observed    0    1
##        0 7648   55
##        1   12 2037

##          FPR         TPR
## 0.007140075 0.994143485

ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,],M.ROC[2,],xlab='False positive rate(1-specificity)', ylab='True positive r
ate(specificity)', main = 'ROC Curve Decision Tree-2', col="green",lwd=2,type="l")
```



###KNN Testing Data 1 #KNN k = 1

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test1$Occupancy <- factor(Occupancy_Test1$Occupancy)
attach(Occupancy_Train)

## The following object is masked from package:datasets:
##
##     CO2
```

```
test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test
1))]
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Tra
in))]
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 1,
prob = TRUE)
table(knn.pred, Occupancy_Test1$Occupancy)

##
## knn.pred    0    1
##        0 1231  525
##        1  462  447

mean(knn.pred != Occupancy_Test1$Occupancy)

## [1] 0.3703565

confusionMatrix(Occupancy_Test1$Occupancy, knn.pred)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1231  462
##          1  525  447
##
##                Accuracy : 0.6296
##                  95% CI : (0.611, 0.648)
##     No Information Rate : 0.6589
##     P-Value [Acc > NIR] : 0.99929
##
##                   Kappa : 0.1896
##  Mcnemar's Test P-Value : 0.04844
##
##             Sensitivity : 0.7010
##             Specificity : 0.4917
##          Pos Pred Value : 0.7271
##          Neg Pred Value : 0.4599
##              Prevalence : 0.6589
##          Detection Rate : 0.4619
##    Detection Prevalence : 0.6353
##       Balanced Accuracy : 0.5964
##
##        'Positive' Class : 0
##
```

## KNN k = 3

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test1$Occupancy <- factor(Occupancy_Test1$Occupancy)
attach(Occupancy_Train)

## The following objects are masked from Occupancy_Train (pos = 3):
##
```

```
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following object is masked from package:datasets:
##
##     CO2
```

```r
test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test
1))]
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Tra
in))]
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 3,
prob = TRUE)
table(knn.pred, Occupancy_Test1$Occupancy)
```

```
##
## knn.pred    0    1
##        0 1231  523
##        1  462  449
```

```r
mean(knn.pred != Occupancy_Test1$Occupancy)
```

```
## [1] 0.369606
```

```r
confusionMatrix(Occupancy_Test1$Occupancy, knn.pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1231  462
##          1  523  449
##
##                Accuracy : 0.6304
##                  95% CI : (0.6117, 0.6488)
##     No Information Rate : 0.6582
##     P-Value [Acc > NIR] : 0.99876
##
##                   Kappa : 0.1916
##  Mcnemar's Test P-Value : 0.05591
##
##             Sensitivity : 0.7018
##             Specificity : 0.4929
##          Pos Pred Value : 0.7271
##          Neg Pred Value : 0.4619
##              Prevalence : 0.6582
##          Detection Rate : 0.4619
##    Detection Prevalence : 0.6353
##       Balanced Accuracy : 0.5973
##
##        'Positive' Class : 0
##
```

### KNN k = 5

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test1$Occupancy <- factor(Occupancy_Test1$Occupancy)
attach(Occupancy_Train)

## The following objects are masked from Occupancy_Train (pos = 3):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 4):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following object is masked from package:datasets:
##
##     CO2

test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test
1))]
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Tra
in))]
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 5,
prob = TRUE)
table(knn.pred, Occupancy_Test1$Occupancy)

##
## knn.pred    0    1
##        0 1240  524
##        1  453  448

mean(knn.pred != Occupancy_Test1$Occupancy)

## [1] 0.3666041

confusionMatrix(Occupancy_Test1$Occupancy, knn.pred)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1240  453
##          1  524  448
##
##                Accuracy : 0.6334
##                  95% CI : (0.6148, 0.6517)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 0.99908
##
##                   Kappa : 0.1964
##  Mcnemar's Test P-Value : 0.02512
##
##             Sensitivity : 0.7029
##             Specificity : 0.4972
##          Pos Pred Value : 0.7324
##          Neg Pred Value : 0.4609
##              Prevalence : 0.6619
##          Detection Rate : 0.4653
##    Detection Prevalence : 0.6353
##       Balanced Accuracy : 0.6001
##
##        'Positive' Class : 0
##
```

## KNN k = 10

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test1$Occupancy <- factor(Occupancy_Test1$Occupancy)
attach(Occupancy_Train)

## The following objects are masked from Occupancy_Train (pos = 3):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 4):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 5):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following object is masked from package:datasets:
##
##     CO2

test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test
1))]
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Tra
in))]
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 10
, prob = TRUE)
table(knn.pred, Occupancy_Test1$Occupancy)

##
## knn.pred    0    1
##        0 1249  528
##        1  444  444

mean(knn.pred != Occupancy_Test1$Occupancy)

## [1] 0.364728

confusionMatrix(Occupancy_Test1$Occupancy, knn.pred)

## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    0    1
##          0 1249  444
##          1  528  444
##                                           
##                Accuracy : 0.6353          
##                  95% CI : (0.6167, 0.6536)
##     No Information Rate : 0.6668          
##     P-Value [Acc > NIR] : 0.999719        
##                                           
##                   Kappa : 0.1982          
##  Mcnemar's Test P-Value : 0.007763        
##                                           
##             Sensitivity : 0.7029          
##             Specificity : 0.5000          
##          Pos Pred Value : 0.7377          
##          Neg Pred Value : 0.4568          
##              Prevalence : 0.6668          
##          Detection Rate : 0.4687          
##    Detection Prevalence : 0.6353          
##       Balanced Accuracy : 0.6014          
##                                           
##        'Positive' Class : 0               
## 
```

## KNN k = 25

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test1$Occupancy <- factor(Occupancy_Test1$Occupancy)
attach(Occupancy_Train)

## The following objects are masked from Occupancy_Train (pos = 3):
## 
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 4):
## 
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 5):
## 
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 6):
## 
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following object is masked from package:datasets:
## 
##     CO2

test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test
1))]
```

```
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Tra
in))]
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 25
, prob = TRUE)
table(knn.pred, Occupancy_Test1$Occupancy)
```

```
## 
## knn.pred    0    1
##        0 1255  529
##        1  438  443
```

```
mean(knn.pred != Occupancy_Test1$Occupancy)
```

```
## [1] 0.3628518
```

```
confusionMatrix(Occupancy_Test1$Occupancy, knn.pred)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 1255  438
##          1  529  443
##                                           
##                Accuracy : 0.6371          
##                  95% CI : (0.6186, 0.6554)
##     No Information Rate : 0.6694          
##     P-Value [Acc > NIR] : 0.999797        
##                                           
##                   Kappa : 0.2011          
##  Mcnemar's Test P-Value : 0.003801        
##                                           
##             Sensitivity : 0.7035          
##             Specificity : 0.5028          
##          Pos Pred Value : 0.7413          
##          Neg Pred Value : 0.4558          
##              Prevalence : 0.6694          
##          Detection Rate : 0.4709          
##    Detection Prevalence : 0.6353          
##       Balanced Accuracy : 0.6032          
##                                           
##        'Positive' Class : 0               
## 
```

## KNN k = 50

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test1$Occupancy <- factor(Occupancy_Test1$Occupancy)
attach(Occupancy_Train)
```

```
## The following objects are masked from Occupancy_Train (pos = 3):
## 
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
## The following objects are masked from Occupancy_Train (pos = 4):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 5):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 6):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 7):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following object is masked from package:datasets:
##
##     CO2
```

```
test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test
1))]
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Tra
in))]
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 50
, prob = TRUE)
table(knn.pred, Occupancy_Test1$Occupancy)
```

```
##
## knn.pred    0    1
##        0 1290  551
##        1  403  421
```

```
mean(knn.pred != Occupancy_Test1$Occupancy)
```

```
## [1] 0.3579737
```

```
confusionMatrix(Occupancy_Test1$Occupancy, knn.pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1290  403
##          1  551  421
##
##                Accuracy : 0.642
##                  95% CI : (0.6235, 0.6603)
##     No Information Rate : 0.6908
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2016
##  Mcnemar's Test P-Value : 1.943e-06
##
##             Sensitivity : 0.7007
##             Specificity : 0.5109
##          Pos Pred Value : 0.7620
##          Neg Pred Value : 0.4331
##              Prevalence : 0.6908
##          Detection Rate : 0.4841
##    Detection Prevalence : 0.6353
##       Balanced Accuracy : 0.6058
##
##        'Positive' Class : 0
##
```

## KNN k = 100

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test1$Occupancy <- factor(Occupancy_Test1$Occupancy)
attach(Occupancy_Train)
```

```
## The following objects are masked from Occupancy_Train (pos = 3):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following objects are masked from Occupancy_Train (pos = 4):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following objects are masked from Occupancy_Train (pos = 5):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following objects are masked from Occupancy_Train (pos = 6):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following objects are masked from Occupancy_Train (pos = 7):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following objects are masked from Occupancy_Train (pos = 8):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following object is masked from package:datasets:
##
##     CO2
```

```
test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test
1))]
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Tra
in))]
```

```
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 10
0, prob = TRUE)
table(knn.pred, Occupancy_Test1$Occupancy)

##
## knn.pred    0    1
##        0 1320  548
##        1  373  424

mean(knn.pred != Occupancy_Test1$Occupancy)

## [1] 0.345591

confusionMatrix(Occupancy_Test1$Occupancy, knn.pred)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1320  373
##          1  548  424
##
##                Accuracy : 0.6544
##                  95% CI : (0.636, 0.6725)
##     No Information Rate : 0.7009
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2245
##  Mcnemar's Test P-Value : 9.838e-09
##
##             Sensitivity : 0.7066
##             Specificity : 0.5320
##          Pos Pred Value : 0.7797
##          Neg Pred Value : 0.4362
##              Prevalence : 0.7009
##          Detection Rate : 0.4953
##    Detection Prevalence : 0.6353
##       Balanced Accuracy : 0.6193
##
##        'Positive' Class : 0
##
```

### KNN Testing Data 2

## KNN k = 1

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test2$Occupancy <- factor(Occupancy_Test2$Occupancy)
attach(Occupancy_Train)

## The following objects are masked from Occupancy_Train (pos = 3):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
## The following objects are masked from Occupancy_Train (pos = 4):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following objects are masked from Occupancy_Train (pos = 5):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following objects are masked from Occupancy_Train (pos = 6):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following objects are masked from Occupancy_Train (pos = 7):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following objects are masked from Occupancy_Train (pos = 8):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following objects are masked from Occupancy_Train (pos = 9):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following object is masked from package:datasets:
##
##     CO2
```

```
test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test2))]
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Train))]
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 1,
prob = TRUE)
table(knn.pred, Occupancy_Test2$Occupancy)
```

```
##
## knn.pred    0    1
##        0 6310 1209
##        1 1393  840
```

```
mean(knn.pred != Occupancy_Test2$Occupancy)
```

```
## [1] 0.2668171
```

```
confusionMatrix(Occupancy_Test2$Occupancy, knn.pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6310 1393
##          1 1209  840
##                                           
##                Accuracy : 0.7332          
##                  95% CI : (0.7243, 0.7419)
##     No Information Rate : 0.771           
##     P-Value [Acc > NIR] : 1.0000000       
##                                           
##                   Kappa : 0.2218          
##  Mcnemar's Test P-Value : 0.0003338       
##                                           
##             Sensitivity : 0.8392          
##             Specificity : 0.3762          
##          Pos Pred Value : 0.8192          
##          Neg Pred Value : 0.4100          
##              Prevalence : 0.7710          
##          Detection Rate : 0.6470          
##    Detection Prevalence : 0.7899          
##       Balanced Accuracy : 0.6077          
##                                           
##        'Positive' Class : 0               
## 
```

## KNN k = 3

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test2$Occupancy <- factor(Occupancy_Test2$Occupancy)
attach(Occupancy_Train)

## The following objects are masked from Occupancy_Train (pos = 3):
## 
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 4):
## 
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 5):
## 
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 6):
## 
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 7):
## 
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 8):
## 
```

```
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 9):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 10):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following object is masked from package:datasets:
##
##     CO2
```

```
test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test
2))]
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Tra
in))]
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 3,
prob = TRUE)
table(knn.pred, Occupancy_Test2$Occupancy)
```

```
##
## knn.pred    0    1
##        0 6314 1210
##        1 1389  839
```

```
mean(knn.pred != Occupancy_Test2$Occupancy)
```

```
## [1] 0.2665094
```

```
confusionMatrix(Occupancy_Test2$Occupancy, knn.pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6314 1389
##          1 1210  839
##
##                Accuracy : 0.7335
##                  95% CI : (0.7246, 0.7422)
##     No Information Rate : 0.7715
##     P-Value [Acc > NIR] : 1.0000000
##
##                   Kappa : 0.222
##  Mcnemar's Test P-Value : 0.0004802
##
##             Sensitivity : 0.8392
##             Specificity : 0.3766
##          Pos Pred Value : 0.8197
##          Neg Pred Value : 0.4095
##               Prevalence : 0.7715
##           Detection Rate : 0.6475
##     Detection Prevalence : 0.7899
##        Balanced Accuracy : 0.6079
##
##         'Positive' Class : 0
##
```

## KNN k = 5

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test2$Occupancy <- factor(Occupancy_Test2$Occupancy)
attach(Occupancy_Train)
```

```
## The following objects are masked from Occupancy_Train (pos = 3):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 4):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 5):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 6):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 7):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 8):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 9):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 10):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 11):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
## The following object is masked from package:datasets:
##
##     CO2
```

```
test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test
2))]
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Tra
in))]
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 5,
prob = TRUE)
table(knn.pred, Occupancy_Test2$Occupancy)
```

```
##
## knn.pred    0    1
##        0 6966 1217
##        1  737  832
```

```
mean(knn.pred != Occupancy_Test2$Occupancy)
```

```
## [1] 0.2003692
```

```
confusionMatrix(Occupancy_Test2$Occupancy, knn.pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6966  737
##          1 1217  832
##
##                Accuracy : 0.7996
##                  95% CI : (0.7915, 0.8075)
##     No Information Rate : 0.8391
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3396
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8513
##             Specificity : 0.5303
##          Pos Pred Value : 0.9043
##          Neg Pred Value : 0.4061
##              Prevalence : 0.8391
##          Detection Rate : 0.7143
##    Detection Prevalence : 0.7899
##       Balanced Accuracy : 0.6908
##
##        'Positive' Class : 0
##
```

## KNN k = 10

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test2$Occupancy <- factor(Occupancy_Test2$Occupancy)
attach(Occupancy_Train)
```

```
## The following objects are masked from Occupancy_Train (pos = 3):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 4):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 5):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 6):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 7):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 8):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 9):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 10):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 11):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 12):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following object is masked from package:datasets:
##
##     CO2
```

```
test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test
2))]
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Tra
in))]
```

```
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 10
, prob = TRUE)
table(knn.pred, Occupancy_Test2$Occupancy)
```

```
##
## knn.pred    0    1
##        0 6988 1219
##        1  715  830
```

```
mean(knn.pred != Occupancy_Test2$Occupancy)
```

```
## [1] 0.1983183
```

```
confusionMatrix(Occupancy_Test2$Occupancy, knn.pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6988  715
##          1 1219  830
##
##                Accuracy : 0.8017
##                  95% CI : (0.7936, 0.8096)
##     No Information Rate : 0.8416
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3432
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8515
##             Specificity : 0.5372
##          Pos Pred Value : 0.9072
##          Neg Pred Value : 0.4051
##              Prevalence : 0.8416
##          Detection Rate : 0.7166
##    Detection Prevalence : 0.7899
##       Balanced Accuracy : 0.6943
##
##        'Positive' Class : 0
##
```

## KNN k = 25

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test2$Occupancy <- factor(Occupancy_Test2$Occupancy)
attach(Occupancy_Train)
```

```
## The following objects are masked from Occupancy_Train (pos = 3):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following objects are masked from Occupancy_Train (pos = 4):
##
```

```
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 5):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 6):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 7):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 8):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 9):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 10):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 11):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 12):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 13):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following object is masked from package:datasets:
##
##     CO2
```

```
test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test2))]
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Train))]
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
```

```
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 25
, prob = TRUE)
table(knn.pred, Occupancy_Test2$Occupancy)
```

```
##
## knn.pred    0    1
##        0 6984 1247
##        1  719  802
```

```
mean(knn.pred != Occupancy_Test2$Occupancy)
```

```
## [1] 0.2015997
```

```
confusionMatrix(Occupancy_Test2$Occupancy, knn.pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6984  719
##          1 1247  802
##
##                Accuracy : 0.7984
##                  95% CI : (0.7903, 0.8063)
##     No Information Rate : 0.844
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3292
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8485
##             Specificity : 0.5273
##          Pos Pred Value : 0.9067
##          Neg Pred Value : 0.3914
##              Prevalence : 0.8440
##          Detection Rate : 0.7162
##    Detection Prevalence : 0.7899
##       Balanced Accuracy : 0.6879
##
##        'Positive' Class : 0
##
```

## KNN k = 50

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test2$Occupancy <- factor(Occupancy_Test2$Occupancy)
attach(Occupancy_Train)
```

```
## The following objects are masked from Occupancy_Train (pos = 3):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
```

```
## The following objects are masked from Occupancy_Train (pos = 4):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
## The following objects are masked from Occupancy_Train (pos = 5):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 6):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 7):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 8):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 9):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 10):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 11):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 12):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 13):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 14):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following object is masked from package:datasets:
##
##     CO2

test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test
2))]
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Tra
in))]
```

```
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 50
, prob = TRUE)
table(knn.pred, Occupancy_Test2$Occupancy)

## 
## knn.pred    0    1
##        0 7042 1277
##        1  661  772

mean(knn.pred != Occupancy_Test2$Occupancy)

## [1] 0.1987285

confusionMatrix(Occupancy_Test2$Occupancy, knn.pred)

## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 7042  661
##          1 1277  772
##                                           
##                Accuracy : 0.8013          
##                  95% CI : (0.7932, 0.8092)
##     No Information Rate : 0.8531          
##     P-Value [Acc > NIR] : 1               
##                                           
##                   Kappa : 0.327           
##  Mcnemar's Test P-Value : <2e-16          
##                                           
##             Sensitivity : 0.8465          
##             Specificity : 0.5387          
##          Pos Pred Value : 0.9142          
##          Neg Pred Value : 0.3768          
##              Prevalence : 0.8531          
##          Detection Rate : 0.7221          
##    Detection Prevalence : 0.7899          
##       Balanced Accuracy : 0.6926          
##                                           
##        'Positive' Class : 0               
## 
```

## KNN k = 100

```
Occupancy_Train$Occupancy <- factor(Occupancy_Train$Occupancy)
Occupancy_Test2$Occupancy <- factor(Occupancy_Test2$Occupancy)
attach(Occupancy_Train)

## The following objects are masked from Occupancy_Train (pos = 3):
## 
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 4):
## 
```

```
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 5):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 6):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 7):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 8):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 9):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 10):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 11):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 12):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 13):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 14):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature

## The following objects are masked from Occupancy_Train (pos = 15):
##
##     CO2, date, Humidity, HumidityRatio, Light, Occupancy,
##     Temperature
## The following object is masked from package:datasets:
##
##     CO2
```

```
test.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Test
2))]
train.X = cbind(Temperature, Light, CO2, HumidityRatio)[as.numeric(rownames(Occupancy_Tra
in))]
Occupancy_Train.Occupancy = Occupancy[1:nrow(Occupancy_Train)]
set.seed(1)
knn.pred = knn(data.frame(train.X), data.frame(test.X), Occupancy_Train.Occupancy, k = 10
0, prob = TRUE)
table(knn.pred, Occupancy_Test2$Occupancy)
```

```
##
## knn.pred    0    1
##        0 7076 1272
##        1  627  777
```

```
mean(knn.pred != Occupancy_Test2$Occupancy)
```

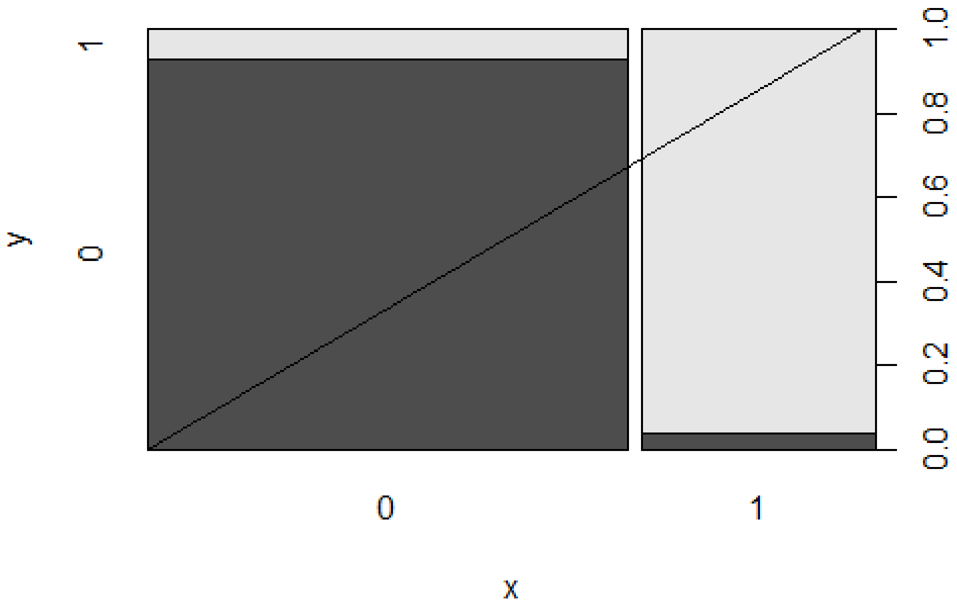
```
## [1] 0.1947293
```

```
confusionMatrix(Occupancy_Test2$Occupancy, knn.pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7076  627
##          1 1272  777
##
##                Accuracy : 0.8053
##                  95% CI : (0.7973, 0.8131)
##     No Information Rate : 0.856
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3367
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8476
##             Specificity : 0.5534
##          Pos Pred Value : 0.9186
##          Neg Pred Value : 0.3792
##              Prevalence : 0.8560
##          Detection Rate : 0.7256
##    Detection Prevalence : 0.7899
##       Balanced Accuracy : 0.7005
##
##        'Positive' Class : 0
##
```

## Random Forest Bag - Testing Data 1

```
set.seed(123)
bag.Occupancy_Train = randomForest(Occupancy ~ Temperature + Light + CO2 + HumidityRatio,
data = Occupancy_Train, mtry = 4, importance = TRUE)
bag.Occupancy_Train

##
## Call:
##  randomForest(formula = Occupancy ~ Temperature + Light + CO2 +      HumidityRatio, da
ta = Occupancy_Train, mtry = 4, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 0.6%
## Confusion matrix:
##      0    1 class.error
## 0 6390   24 0.003741815
## 1   25 1704 0.014459225

yhat.bag = predict(bag.Occupancy_Train, Occupancy_Test1)
plot(yhat.bag, Occupancy_Test1$Occupancy)
abline(0,1)
```



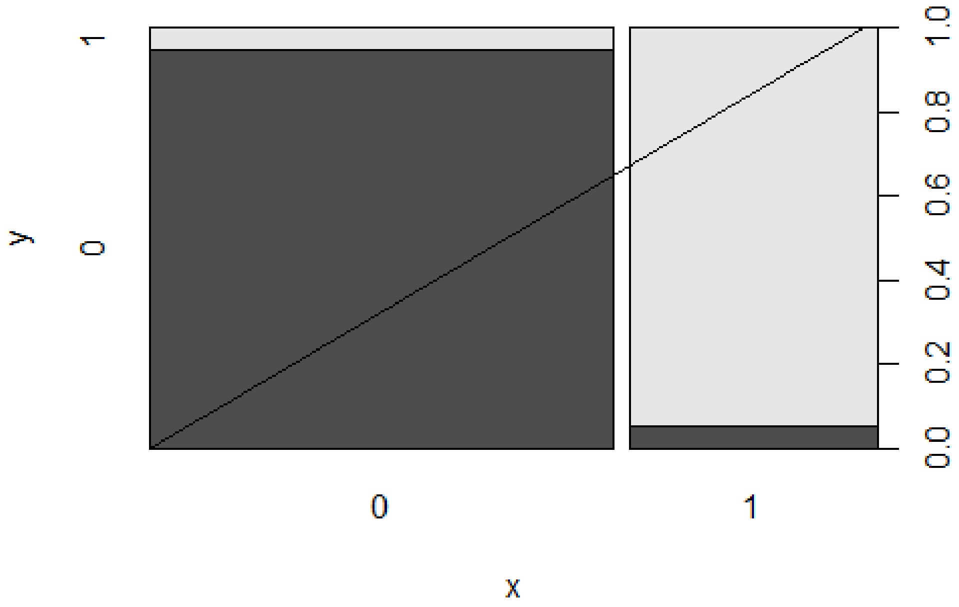
```
mean((as.numeric(yhat.bag) - as.numeric(Occupancy_Test1$Occupancy)) ^ 2)

## [1] 0.06003752
```

## Random Forest Ntree

```
set.seed(123)
bag.Occupancy_Train = randomForest(Occupancy ~ Temperature + Light + CO2 + HumidityRatio,
data = Occupancy_Train, mtry = 4, ntree = 15)
bag.Occupancy_Train
```

```
##
## Call:
##  randomForest(formula = Occupancy ~ Temperature + Light + CO2 +      HumidityRatio, da
ta = Occupancy_Train, mtry = 4, ntree = 15)
##                Type of random forest: classification
##                      Number of trees: 15
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 0.71%
## Confusion matrix:
##      0    1 class.error
## 0 6385   25 0.003900156
## 1   33 1693 0.019119351
```

```
yhat.bag = predict(bag.Occupancy_Train, Occupancy_Test1)
plot(yhat.bag, Occupancy_Test1$Occupancy)
abline(0,1)
```

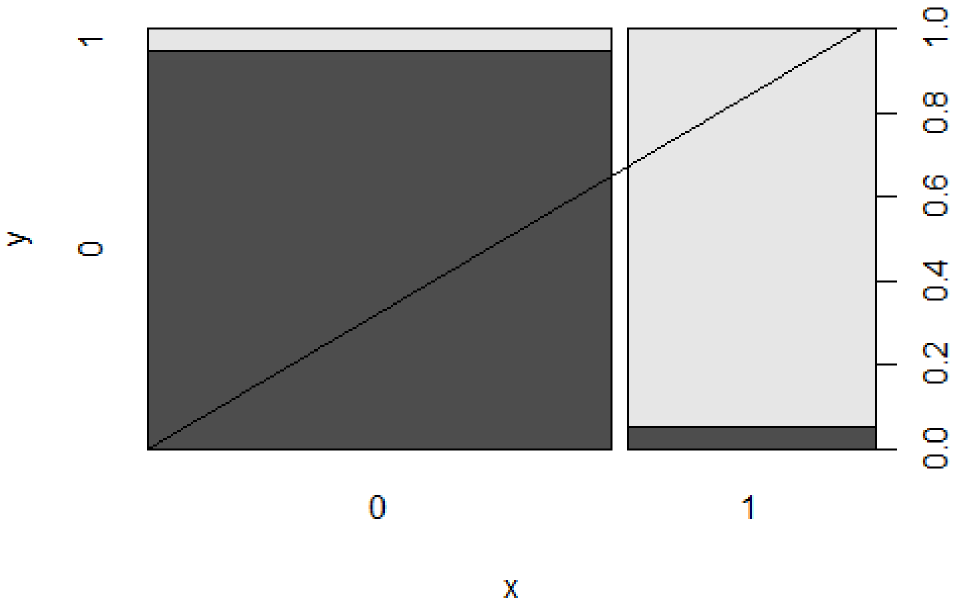


```
mean((as.numeric(yhat.bag) - as.numeric(Occupancy_Test1$Occupancy)) ^ 2)

## [1] 0.05140713
```

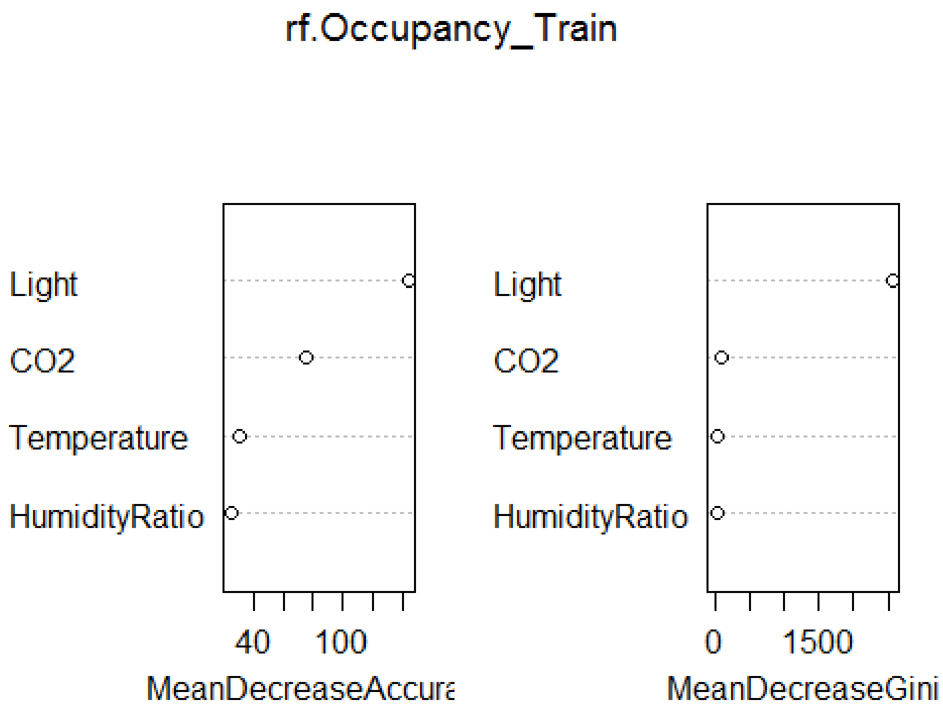
## Random Forest

```
set.seed(123)
rf.Occupancy_Train = randomForest(Occupancy ~ Temperature + Light + CO2 + HumidityRatio,
data = Occupancy_Train, mtry = 4, importance = TRUE)
rf.Occupancy_Train

##
## Call:
##  randomForest(formula = Occupancy ~ Temperature + Light + CO2 +      HumidityRatio, da
ta = Occupancy_Train, mtry = 4, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 0.6%
## Confusion matrix:
##      0    1 class.error
## 0 6390   24 0.003741815
## 1   25 1704 0.014459225
```

```
yhat.rf = predict(bag.Occupancy_Train, Occupancy_Test1)
plot(yhat.rf, Occupancy_Test1$Occupancy)
abline(0,1)
```



```
mean((as.numeric(yhat.rf) - as.numeric(Occupancy_Test1$Occupancy)) ^ 2)

## [1] 0.05140713

importance(rf.Occupancy_Train)

##                       0         1 MeanDecreaseAccuracy MeanDecreaseGini
## Temperature   37.41279  25.49205             31.54602         47.34943
## Light         46.78300 454.59543            143.40751       2566.50569
## CO2           26.80236  66.48899             74.95026         84.60173
## HumidityRatio 26.44377   7.57482             25.56969         25.09048

varImpPlot(rf.Occupancy_Train)
```



#Random Forst Confusion Matrix

```
rf.pred <- predict(rf.Occupancy_Train, Occupancy_Test1, type = "class")
table(rf.pred, Occupancy_Test1$Occupancy)

##
## rf.pred    0    1
##       0 1663  130
##       1   30  842

mean(rf.pred != Occupancy_Test1$Occupancy)

## [1] 0.06003752

confusionMatrix(Occupancy_Test1$Occupancy, rf.pred)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1663   30
##          1  130  842
##
##                Accuracy : 0.94
##                  95% CI : (0.9303, 0.9487)
##     No Information Rate : 0.6728
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8675
##  Mcnemar's Test P-Value : 5.011e-15
##
##             Sensitivity : 0.9275
##             Specificity : 0.9656
##          Pos Pred Value : 0.9823
##          Neg Pred Value : 0.8663
##              Prevalence : 0.6728
##          Detection Rate : 0.6240
##    Detection Prevalence : 0.6353
##       Balanced Accuracy : 0.9465
## 
##        'Positive' Class : 0
## 
```

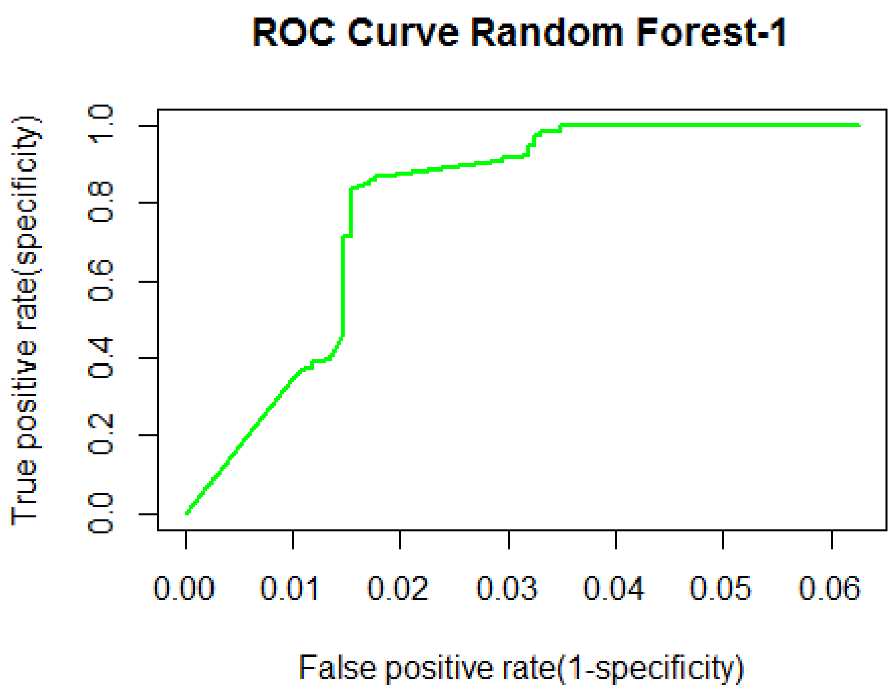
## ROC Analysis Random Forest

```
rf.pred = predict(rf.Occupancy_Train, Occupancy_Test1, type = "prob")
#tree.prob <- attr(tree.pred, "vector")
roc.curve=function(s,print=FALSE){
Ps=(rf.pred[,2]>s)*1
FP=sum((Ps==1)*(Occupancy_Test1$Occupancy == 0))/sum(Occupancy_Test1$Occupancy == 0)
TP=sum((Ps==1)*(Occupancy_Test1$Occupancy == 1))/sum(Occupancy_Test1$Occupancy == 1)
if(print==TRUE){
print(table(Observed=Occupancy_Test1$Occupancy,Predicted=Ps))
}
vect=c(FP,TP)
names(vect)=c("FPR","TPR")
return(vect)
}
threshold = 0.5
roc.curve(threshold,print=TRUE)
```

```
##         Predicted
## Observed    0    1
##        0 1663   30
##        1  130  842
```

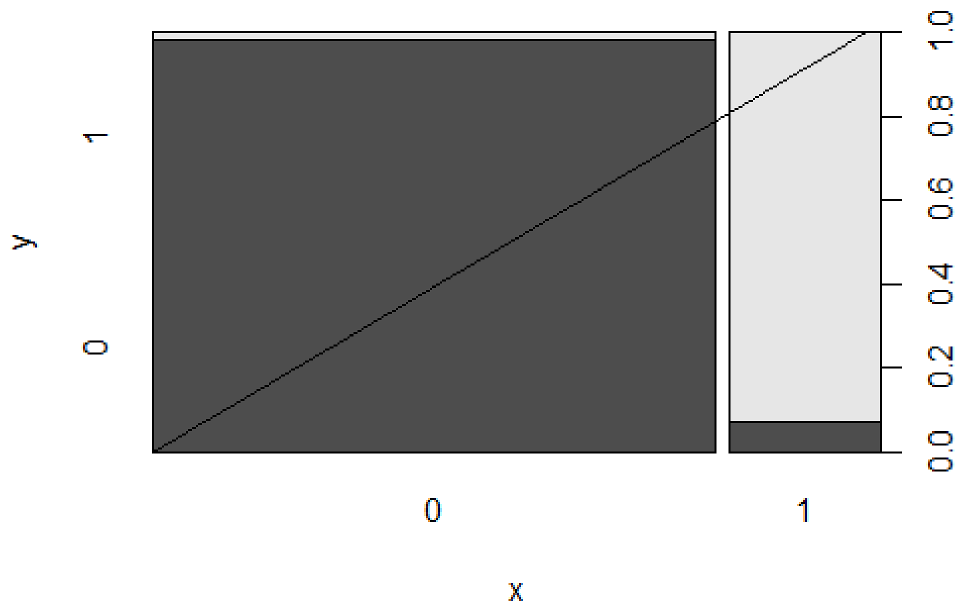
```
##        FPR        TPR 
## 0.01772002 0.86625514
```

```
ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,],M.ROC[2,],xlab='False positive rate(1-specificity)', ylab='True positive r
ate(specificity)', main = 'ROC Curve Random Forest-1', col="green",lwd=2,type="l")
```



#Random Forest Bag - Testing Data 2

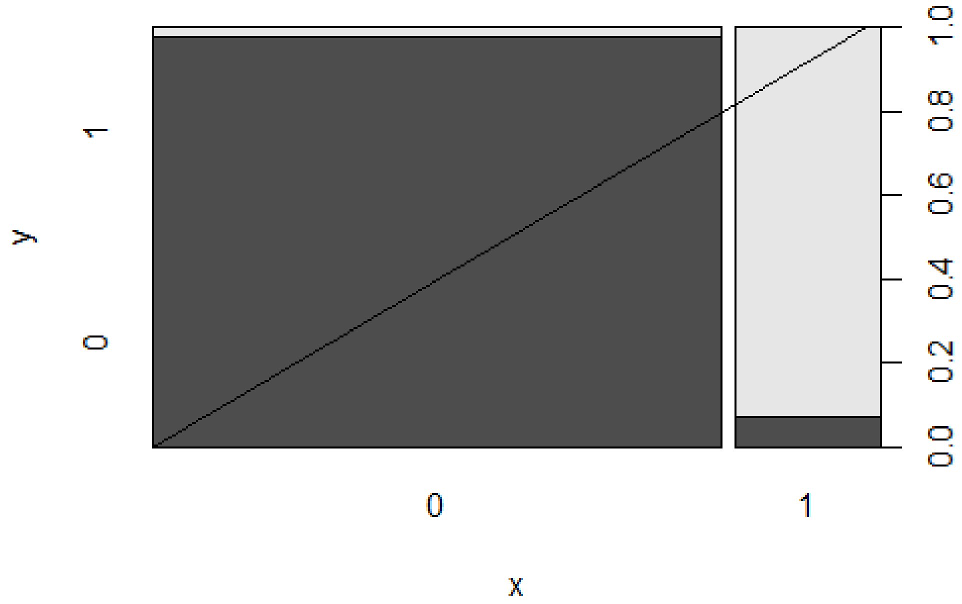
```
set.seed(123)
bag.Occupancy_Train = randomForest(Occupancy ~ Temperature + Light + CO2 + HumidityRatio,
data = Occupancy_Train, mtry = 4, importance = TRUE)
bag.Occupancy_Train

##
## Call:
##  randomForest(formula = Occupancy ~ Temperature + Light + CO2 +      HumidityRatio, da
ta = Occupancy_Train, mtry = 4, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 0.6%
## Confusion matrix:
##      0    1 class.error
## 0 6390   24 0.003741815
## 1   25 1704 0.014459225

yhat.bag = predict(bag.Occupancy_Train, Occupancy_Test2)
plot(yhat.bag, Occupancy_Test2$Occupancy)
abline(0,1)
```



```
mean((as.numeric(yhat.bag) - as.numeric(Occupancy_Test2$Occupancy)) ^ 2)

## [1] 0.0288146
```

## Random Forest Ntree

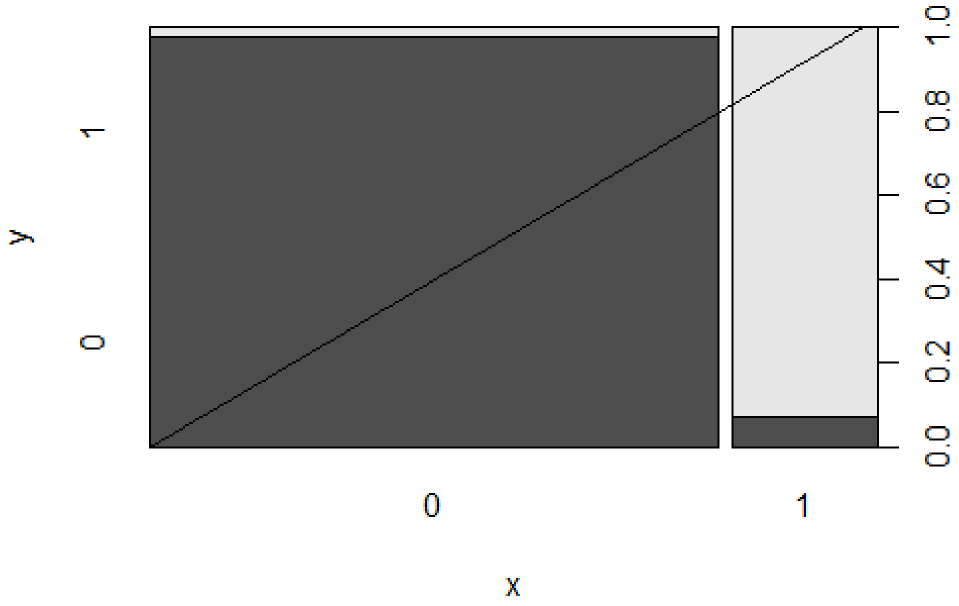
```
set.seed(123)
bag.Occupancy_Train = randomForest(Occupancy ~ Temperature + Light + CO2 + HumidityRatio,
data = Occupancy_Train, mtry = 4, ntree = 15)
bag.Occupancy_Train

##
## Call:
##  randomForest(formula = Occupancy ~ Temperature + Light + CO2 +      HumidityRatio, da
ta = Occupancy_Train, mtry = 4, ntree = 15)
##                Type of random forest: classification
##                      Number of trees: 15
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 0.71%
## Confusion matrix:
##      0    1 class.error
## 0 6385   25 0.003900156
## 1   33 1693 0.019119351

yhat.bag = predict(bag.Occupancy_Train, Occupancy_Test2)
plot(yhat.bag, Occupancy_Test2$Occupancy)
abline(0,1)
```



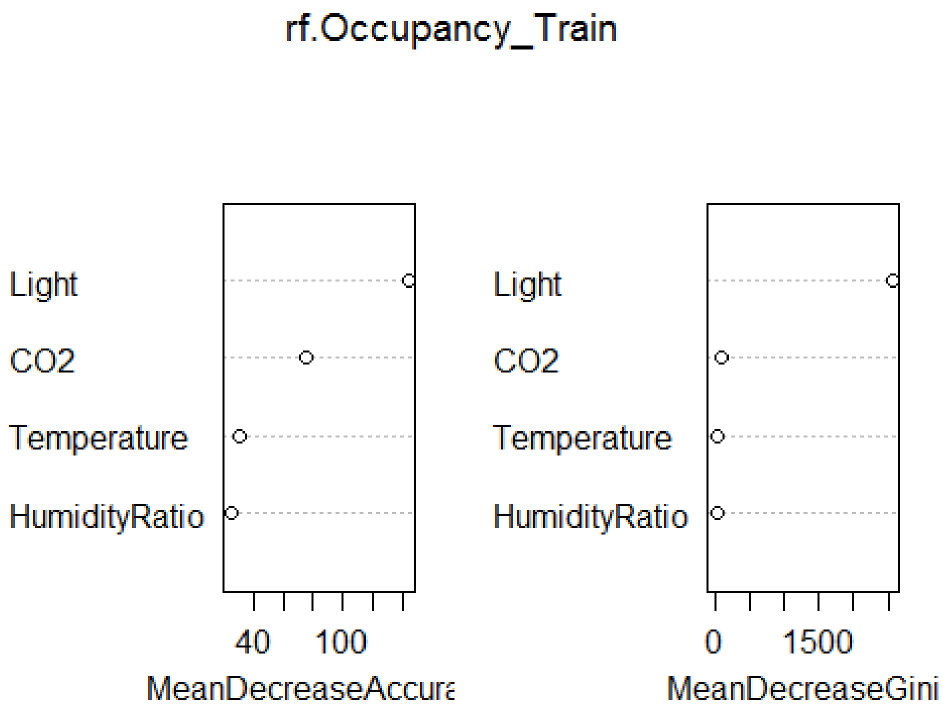
```
mean((as.numeric(yhat.bag) - as.numeric(Occupancy_Test2$Occupancy)) ^ 2)

## [1] 0.03373667
```

## Random Forest

```
set.seed(123)
rf.Occupancy_Train = randomForest(Occupancy ~ Temperature + Light + CO2 + HumidityRatio,
data = Occupancy_Train, mtry = 4, importance = TRUE)
rf.Occupancy_Train

##
## Call:
##  randomForest(formula = Occupancy ~ Temperature + Light + CO2 +      HumidityRatio, da
ta = Occupancy_Train, mtry = 4, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 0.6%
## Confusion matrix:
##      0    1 class.error
## 0 6390   24 0.003741815
## 1   25 1704 0.014459225

yhat.rf = predict(bag.Occupancy_Train, Occupancy_Test2)
plot(yhat.rf, Occupancy_Test2$Occupancy)
abline(0,1)
```



```
mean((as.numeric(yhat.rf) - as.numeric(Occupancy_Test2$Occupancy)) ^ 2)

## [1] 0.03373667

importance(rf.Occupancy_Train)

##                      0         1 MeanDecreaseAccuracy MeanDecreaseGini
## Temperature   37.41279  25.49205             31.54602         47.34943
## Light         46.78300 454.59543            143.40751       2566.50569
## CO2           26.80236  66.48899             74.95026         84.60173
## HumidityRatio 26.44377   7.57482             25.56969         25.09048

varImpPlot(rf.Occupancy_Train)
```



#Random Forst Confusion Matrix

```
rf.pred <- predict(rf.Occupancy_Train, Occupancy_Test2, type = "class")
table(rf.pred, Occupancy_Test2$Occupancy)
```

```
##
## rf.pred    0    1
##       0 7554  132
##       1  149 1917
```

```
mean(rf.pred != Occupancy_Test2$Occupancy)
```

```
## [1] 0.0288146
```

```
confusionMatrix(Occupancy_Test2$Occupancy, rf.pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7554  149
##          1  132 1917
##
##                Accuracy : 0.9712
##                  95% CI : (0.9677, 0.9744)
##     No Information Rate : 0.7881
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9135
##  Mcnemar's Test P-Value : 0.3398
##
##             Sensitivity : 0.9828
##             Specificity : 0.9279
##          Pos Pred Value : 0.9807
##          Neg Pred Value : 0.9356
##              Prevalence : 0.7881
##          Detection Rate : 0.7746
##    Detection Prevalence : 0.7899
##       Balanced Accuracy : 0.9554
##
##        'Positive' Class : 0
##
```

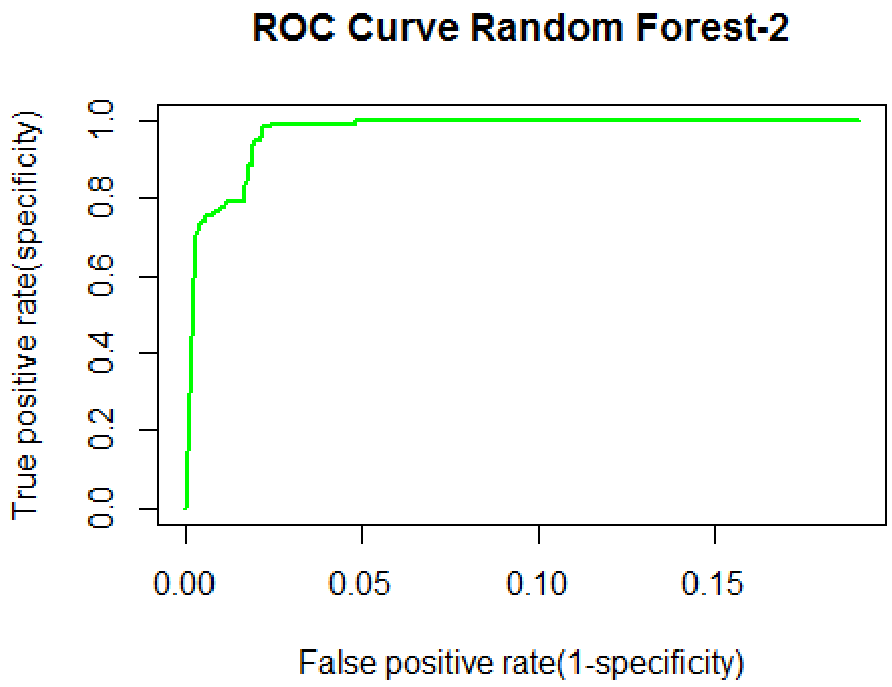
## ROC Analysis Random Forest

```
rf.pred = predict(rf.Occupancy_Train, Occupancy_Test2, type = "prob")
#tree.prob <- attr(tree.pred, "vector")
roc.curve=function(s,print=FALSE){
Ps=(rf.pred[,2]>s)*1
FP=sum((Ps==1)*(Occupancy_Test2$Occupancy == 0))/sum(Occupancy_Test2$Occupancy == 0)
TP=sum((Ps==1)*(Occupancy_Test2$Occupancy == 1))/sum(Occupancy_Test2$Occupancy == 1)
if(print==TRUE){
print(table(Observed=Occupancy_Test2$Occupancy,Predicted=Ps))
}
vect=c(FP,TP)
names(vect)=c("FPR","TPR")
return(vect)
}
threshold = 0.5
roc.curve(threshold,print=TRUE)
```

```
##         Predicted
## Observed    0    1
##        0 7554  149
##        1  132 1917
```

```
##        FPR        TPR
## 0.01934311 0.93557833
```

```
ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,],M.ROC[2,],xlab='False positive rate(1-specificity)', ylab='True positive r
ate(specificity)', main = 'ROC Curve Random Forest-2', col="green",lwd=2,type="l")
```



**Appendix – 2 {Artificial Neural Network}**

# Team14_Project_Neural_Network

Prajwal Gonnade; Supreet Nayak

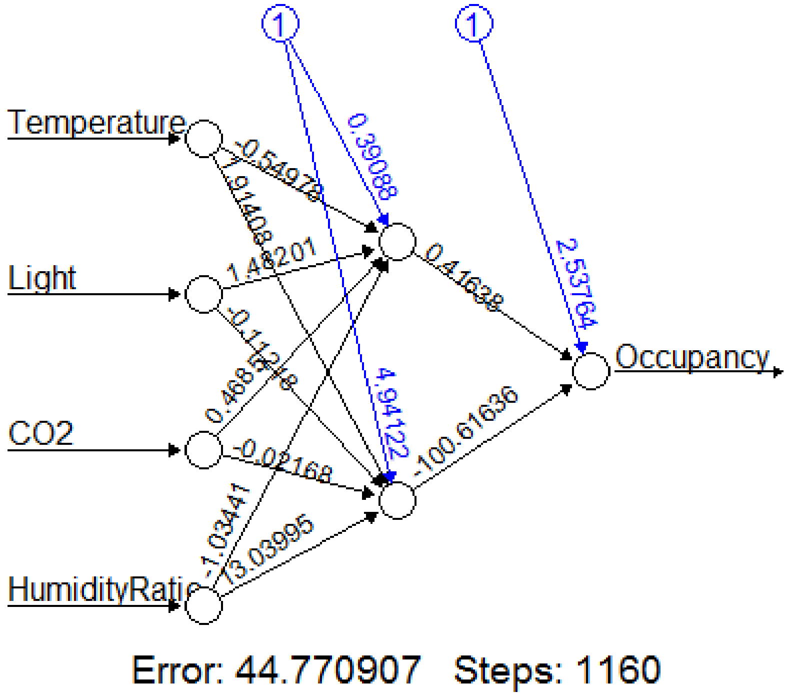
December 9, 2016

```
library(boot)

## Warning: package 'boot' was built under R version 3.2.5

library(caret)

## Warning: package 'caret' was built under R version 3.2.5

## Warning: package 'ggplot2' was built under R version 3.2.5

library(class)
library(ROCR)

## Warning: package 'ROCR' was built under R version 3.2.5

## Warning: package 'gplots' was built under R version 3.2.5

library(MASS)
library(tree)

## Warning: package 'tree' was built under R version 3.2.5

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.2.5
```

```
library(chemometrics)

## Warning: package 'chemometrics' was built under R version 3.2.5

library(reshape2)

## Warning: package 'reshape2' was built under R version 3.2.5
```

## Neural Network

```
Occupancy_Train <- read.csv(file.choose(),header=T)
Occupancy_Test1 <- read.csv(file.choose(),header=T)
Occupancy_Test2 <- read.csv(file.choose(),header=T)

library(neuralnet)

## Warning: package 'neuralnet' was built under R version 3.2.5

##
## Attaching package: 'neuralnet'

## The following object is masked from 'package:ROCR':
##
##     prediction

nn <- neuralnet(Occupancy ~ Temperature + Light + CO2 + HumidityRatio, data=Occupancy_Tra
in,hidden=2,threshold = 0.01, linear.output=FALSE)
plot(nn, rep = "best")
```



```
pr.nn1 <- compute(nn,Occupancy_Test1[,c(2,4,5,6)])

results1 <- data.frame(actual = Occupancy_Test1$Occupancy, prediction = pr.nn1$net.result
)
```

```
table(round(pr.nn1$net.result),Occupancy_Test1$Occupancy)

##
##        0    1
##   0 1638    2
##   1   55  970

mean(round(pr.nn1$net.result) != Occupancy_Test1$Occupancy)

## [1] 0.02138836773

confusionMatrix(Occupancy_Test1$Occupancy, round(pr.nn1$net.result))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1638   55
##          1    2  970
##
##                Accuracy : 0.9786116
##                  95% CI : (0.9723768, 0.9837614)
##     No Information Rate : 0.6153846
##     P-Value [Acc > NIR] : < 0.00000000000000022204
##
##                   Kappa : 0.9543747
##  Mcnemar's Test P-Value : 0.000000000005675414
##
##             Sensitivity : 0.9987805
##             Specificity : 0.9463415
##          Pos Pred Value : 0.9675133
##          Neg Pred Value : 0.9979424
##              Prevalence : 0.6153846
##          Detection Rate : 0.6146341
##    Detection Prevalence : 0.6352720
##       Balanced Accuracy : 0.9725610
##
##        'Positive' Class : 0
##

pr.nn2 <- compute(nn,Occupancy_Test2[,c(2,4,5,6)])

results2 <- data.frame(actual = Occupancy_Test2$Occupancy, prediction = pr.nn2$net.result
)

table(round(pr.nn2$net.result),Occupancy_Test2$Occupancy)

##
##        0    1
##   0 7624    9
##   1   79 2040

mean(round(pr.nn2$net.result) != Occupancy_Test2$Occupancy)

## [1] 0.009023789992

confusionMatrix(Occupancy_Test2$Occupancy, round(pr.nn2$net.result))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 7624   79
##          1    9 2040
##                                                
##                Accuracy : 0.9909762            
##                  95% CI : (0.9888941, 0.9927565)
##     No Information Rate : 0.7827112            
##     P-Value [Acc > NIR] : < 0.00000000000000022204
##                                                
##                   Kappa : 0.9731507            
##  Mcnemar's Test P-Value : 0.000000000000190321 
##                                                
##             Sensitivity : 0.9988209            
##             Specificity : 0.9627183            
##          Pos Pred Value : 0.9897443            
##          Neg Pred Value : 0.9956076            
##              Prevalence : 0.7827112            
##          Detection Rate : 0.7817884            
##    Detection Prevalence : 0.7898893            
##       Balanced Accuracy : 0.9807696            
##                                                
##        'Positive' Class : 0                    
## 
```

**Appendix – 3 {Support Vector Machines}**

# Team14_Project_SVM

Prajwal Gonnade; Supreet Nayak

December 9, 2016

```
library(boot)

## Warning: package 'boot' was built under R version 3.2.5

library(caret)

## Warning: package 'caret' was built under R version 3.2.5

## Warning: package 'ggplot2' was built under R version 3.2.5

library(class)
library(ROCR)

## Warning: package 'ROCR' was built under R version 3.2.5

## Warning: package 'gplots' was built under R version 3.2.5

library(e1071)

## Warning: package 'e1071' was built under R version 3.2.5
```

## Load CSV

```
Occupancy_Train <- read.csv(file.choose(),header=T)
Occupancy_Test1 <- read.csv(file.choose(),header=T)
Occupancy_Test2 <- read.csv(file.choose(),header=T)

Occupancy_subset=data.frame(x=Occupancy_Train[,c(2,4,5,6)], y=as.factor(Occupancy_Train[,
7]))
Occupancy.svm=svm(y~., data=Occupancy_subset, kernel="linear",cost=10)
summary(Occupancy.svm)
```

```
##
## Call:
## svm(formula = y ~ ., data = Occupancy_subset, kernel = "linear",
##     cost = 10)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.25
##
## Number of Support Vectors:  325
##
##  ( 162 163 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
table(Occupancy.svm$fitted, Occupancy_subset$y)
```

```
##
##        0    1
##   0 6321   21
##   1   93 1708
```

```
Occupancy_subset.te=data.frame(x=Occupancy_Test1[,c(2,4,5,6)], y=as.factor(Occupancy_Test
1[,7]))
pred.te=predict(Occupancy.svm, newdata=Occupancy_subset.te,decision.values=TRUE)
Occupancy.svm.probs<-attr(pred.te,"decision.values")
table(pred.te, Occupancy_subset.te$y)
```

```
##
## pred.te    0    1
##       0 1639    3
##       1   54  969
```

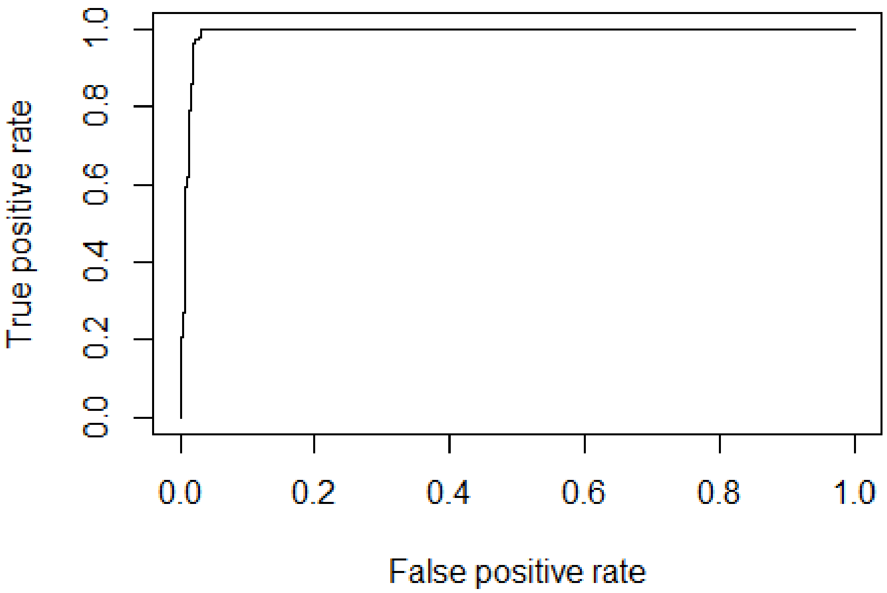
```
Occupancy.svm.confusion <- confusionMatrix(Occupancy_subset.te$y, pred.te)
Occupancy.svm.confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 1639   54
##           1    3  969
##                                           
##                Accuracy : 0.9786          
##                  95% CI : (0.9724, 0.9838)
##     No Information Rate : 0.6161          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9544          
##  Mcnemar's Test P-Value : 3.528e-11       
##                                           
##             Sensitivity : 0.9982          
##             Specificity : 0.9472          
##          Pos Pred Value : 0.9681          
##          Neg Pred Value : 0.9969          
##              Prevalence : 0.6161          
##          Detection Rate : 0.6150          
##    Detection Prevalence : 0.6353          
##       Balanced Accuracy : 0.9727          
##                                           
##        'Positive' Class : 0               
## 
```

```
Occupancy.svm.accuracy <- mean(pred.te == Occupancy_subset.te$y)
Occupancy.svm.accuracy
```

```
## [1] 0.9786116
```

```
Occupancy.svm.prediction<-prediction(Occupancy.svm.probs,Occupancy_subset.te$y)
Occupancy.svm.performance<-performance(Occupancy.svm.prediction,"tpr","fpr")
Occupancy.svm.auc<-performance(Occupancy.svm.prediction,"auc")@y.values[[1]]
plot(Occupancy.svm.performance)
```



```
Occupancy_subset.te=data.frame(x=Occupancy_Test2[,c(2,4,5,6)], y=as.factor(Occupancy_Test
2[,7]))
pred.te=predict(Occupancy.svm, newdata=Occupancy_subset.te,decision.values=TRUE)
Occupancy.svm.probs<-attr(pred.te,"decision.values")
table(pred.te, Occupancy_subset.te$y)
```

```
##
## pred.te    0    1
##       0 7644   14
##       1   59 2035
```

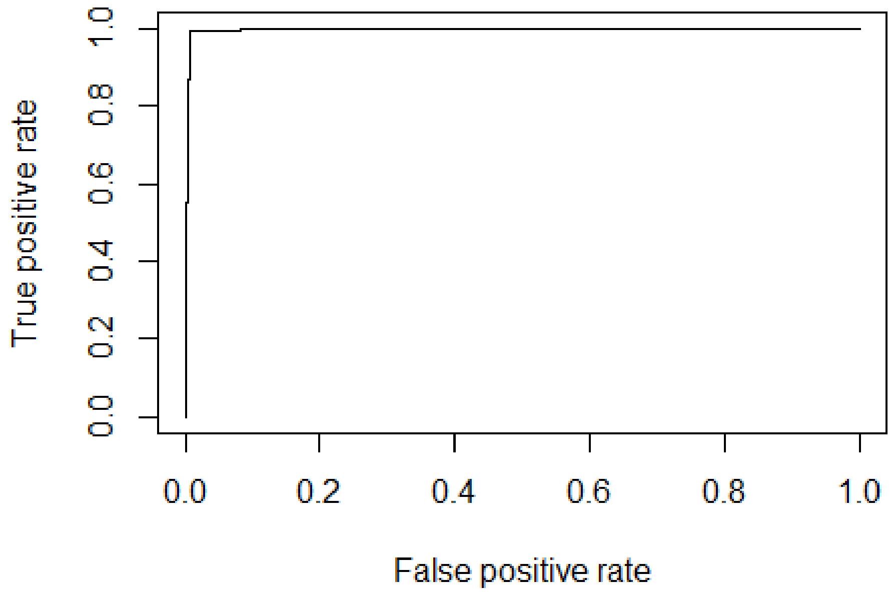
```
Occupancy.svm.confusion <- confusionMatrix(Occupancy_subset.te$y, pred.te)
Occupancy.svm.confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7644   59
##          1   14 2035
##
##                Accuracy : 0.9925
##                  95% CI : (0.9906, 0.9941)
##     No Information Rate : 0.7853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9776
##  Mcnemar's Test P-Value : 2.607e-07
##
##             Sensitivity : 0.9982
##             Specificity : 0.9718
##          Pos Pred Value : 0.9923
##          Neg Pred Value : 0.9932
##              Prevalence : 0.7853
##          Detection Rate : 0.7838
##    Detection Prevalence : 0.7899
##       Balanced Accuracy : 0.9850
##
##        'Positive' Class : 0
##
```

```
Occupancy.svm.accuracy <- mean(pred.te == Occupancy_subset.te$y)
Occupancy.svm.accuracy
```

```
## [1] 0.9925144
```

```
Occupancy.svm.prediction<-prediction(Occupancy.svm.probs,Occupancy_subset.te$y)
Occupancy.svm.performance<-performance(Occupancy.svm.prediction,"tpr","fpr")
Occupancy.svm.auc<-performance(Occupancy.svm.prediction,"auc")@y.values[[1]]
plot(Occupancy.svm.performance)
```



**Appendix – 4 {Perceptron Algorithm}**

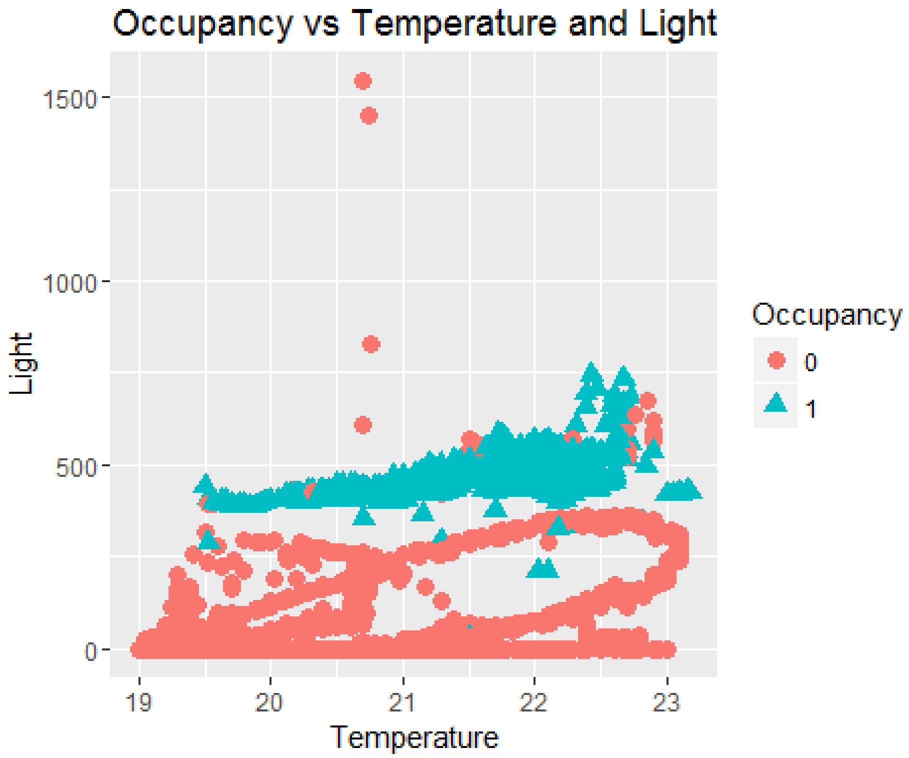
# Team14_Project_Perceptron

Prajwal Gonnade; Supreet Nayak

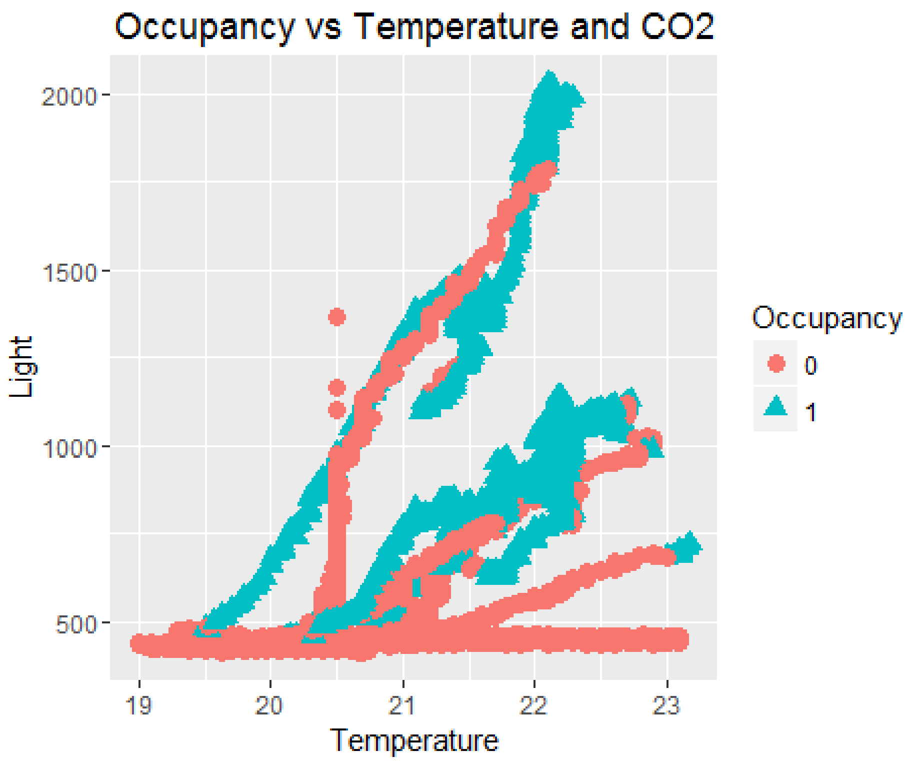
December 9, 2016

```
library(boot)

## Warning: package 'boot' was built under R version 3.2.5

library(caret)

## Warning: package 'caret' was built under R version 3.2.5

## Warning: package 'ggplot2' was built under R version 3.2.5

library(class)
library(ROCR)

## Warning: package 'ROCR' was built under R version 3.2.5

## Warning: package 'gplots' was built under R version 3.2.5

library(MASS)
library(tree)

## Warning: package 'tree' was built under R version 3.2.5

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.2.5
```

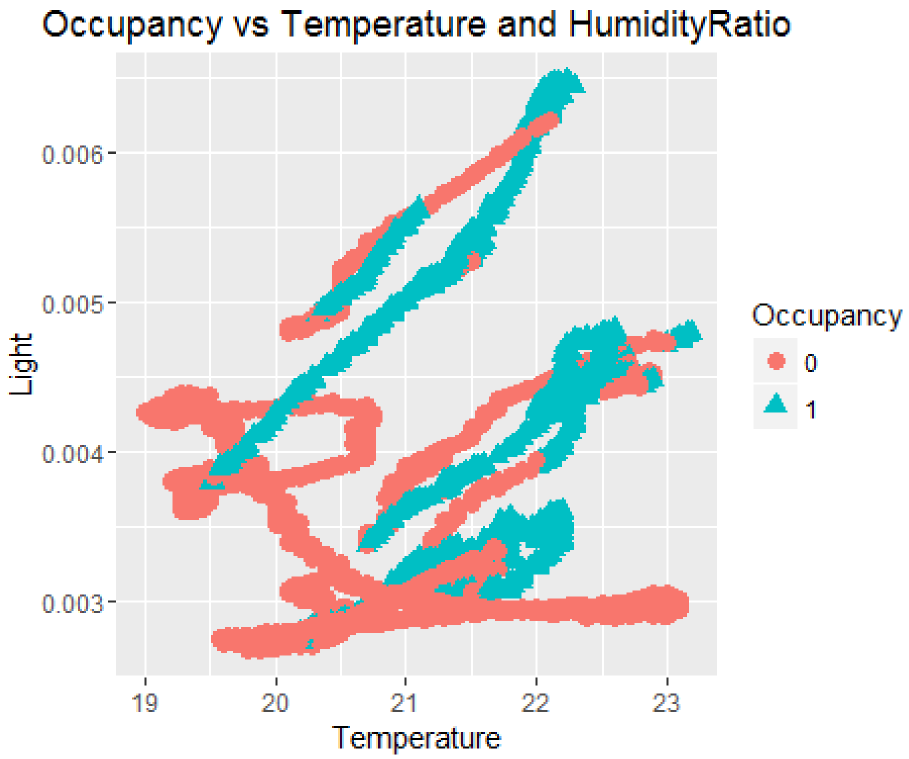
```
library(chemometrics)

## Warning: package 'chemometrics' was built under R version 3.2.5

library(reshape2)

## Warning: package 'reshape2' was built under R version 3.2.5

Occupancy_Train <- read.csv(file.choose(),header=T)
Occupancy_Test1 <- read.csv(file.choose(),header=T)
Occupancy_Test2 <- read.csv(file.choose(),header=T)

Occupancy_Train$Occupancy <- as.factor(Occupancy_Train$Occupancy)
ggplot(Occupancy_Train, aes(x = Temperature, y = Light)) +
     geom_point(aes(colour=Occupancy, shape=Occupancy), size = 3) +
     xlab("Temperature") +
     ylab("Light") +
     ggtitle("Occupancy vs Temperature and Light")
```



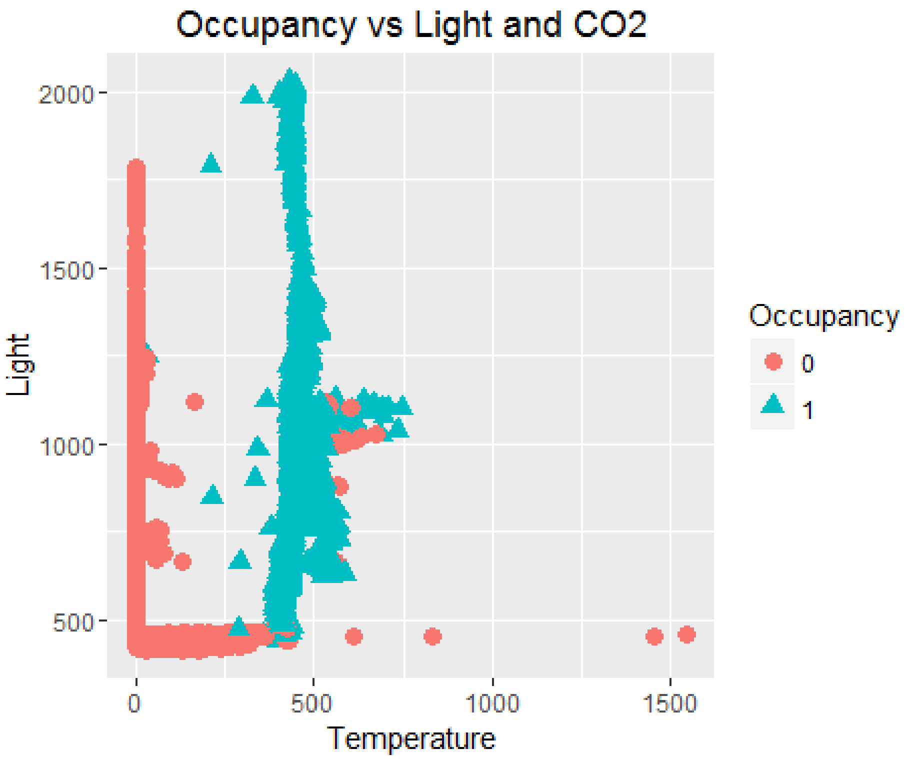
```
ggplot(Occupancy_Train, aes(x = Temperature, y = CO2)) +
     geom_point(aes(colour=Occupancy, shape=Occupancy), size = 3) +
     xlab("Temperature") +
     ylab("Light") +
     ggtitle("Occupancy vs Temperature and CO2")
```



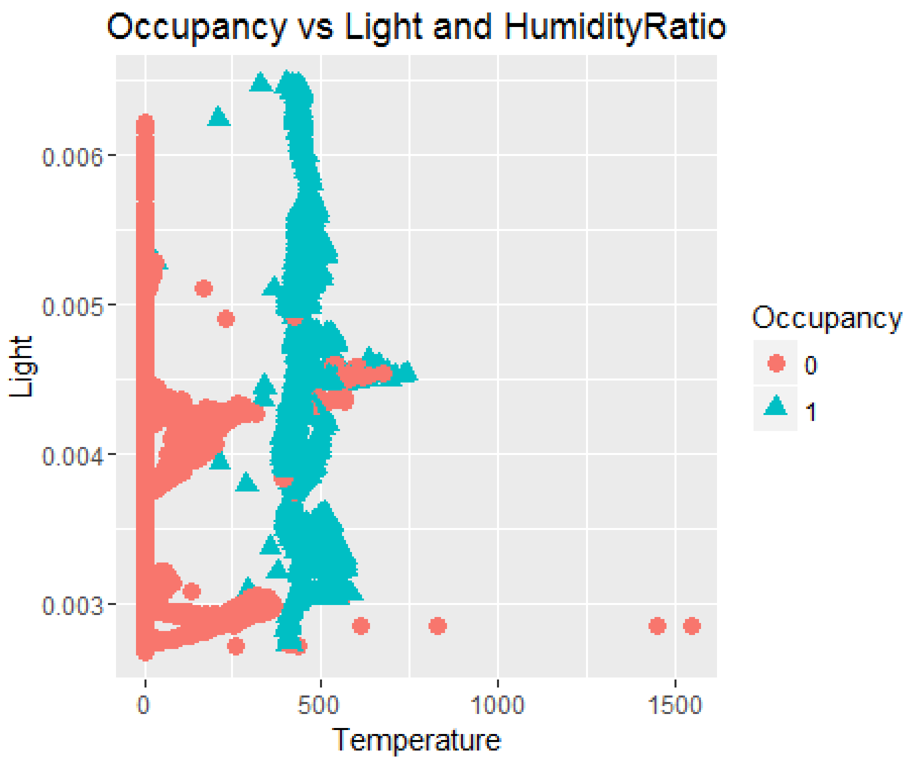
```
ggplot(Occupancy_Train, aes(x = Temperature, y = HumidityRatio)) +
     geom_point(aes(colour=Occupancy, shape=Occupancy), size = 3) +
     xlab("Temperature") +
     ylab("Light") +
     ggtitle("Occupancy vs Temperature and HumidityRatio")
```



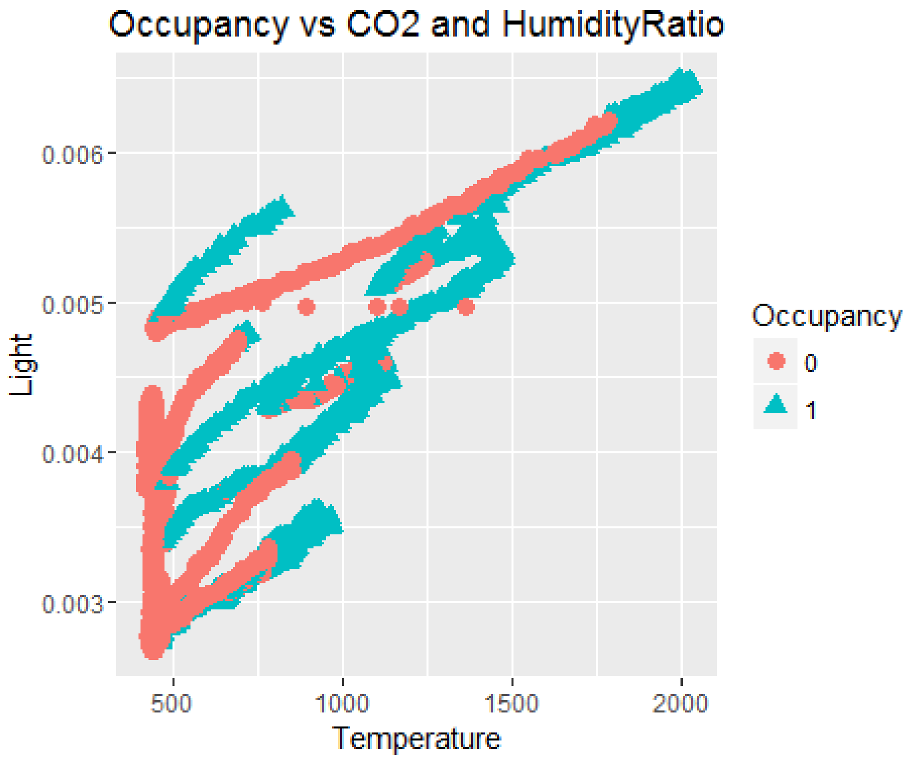
```
ggplot(Occupancy_Train, aes(x = Light, y = CO2)) +
     geom_point(aes(colour=Occupancy, shape=Occupancy), size = 3) +
     xlab("Temperature") +
     ylab("Light") +
     ggtitle("Occupancy vs Light and CO2")
```



```
ggplot(Occupancy_Train, aes(x = Light, y = HumidityRatio)) +
     geom_point(aes(colour=Occupancy, shape=Occupancy), size = 3) +
     xlab("Temperature") +
     ylab("Light") +
     ggtitle("Occupancy vs Light and HumidityRatio")
```



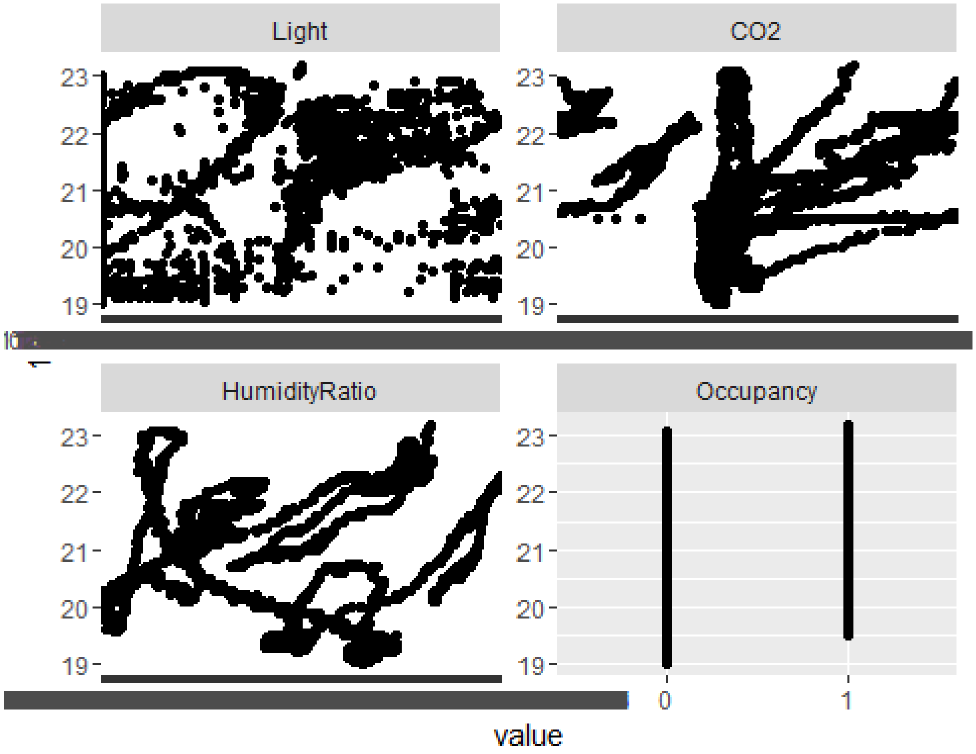
```r
ggplot(Occupancy_Train, aes(x = CO2, y = HumidityRatio)) +
     geom_point(aes(colour=Occupancy, shape=Occupancy), size = 3) +
     xlab("Temperature") +
     ylab("Light") +
     ggtitle("Occupancy vs CO2 and HumidityRatio")
```



```r
#Occupancy_Train$Occupancy <- as.integer(Occupancy_Train$Occupancy)
```

## Perceptron

```r
Occupancy_subset <- Occupancy_Train[,c(2,4,5,6,7)]
Occupancy_response <- Occupancy_Test2[,7]
feature_plot <- function (Occupancy_subset, Occupancy_response) {
  mtmelt <<- melt(Occupancy_subset, id.vars = Occupancy_response)
  p <- ggplot(mtmelt, aes(x = value, y = mtmelt[, 5])) +
    facet_wrap(~variable, scales = "free") +
    geom_point() +
    labs(y = Occupancy_response)
  p
}

feature_plot(Occupancy_subset, Occupancy_response)

## Warning: attributes are not identical across measure variables; they will
## be dropped
```



```r
Occupancy_subset[, 6] <- 1
Occupancy_subset[Occupancy_subset[, 5] == 0, 6] <- -1
x <- Occupancy_subset[, c(1,2,3,4)]
y <- Occupancy_subset[, 6]
head(x)

##   Temperature Light    CO2 HumidityRatio
## 1       23.18 426.0 721.25   0.004792988
## 2       23.15 429.5 714.00   0.004783441
## 3       23.15 426.0 713.50   0.004779464
## 4       23.15 426.0 708.25   0.004771509
## 5       23.10 426.0 704.50   0.004756993
## 6       23.10 419.0 701.00   0.004756993
```

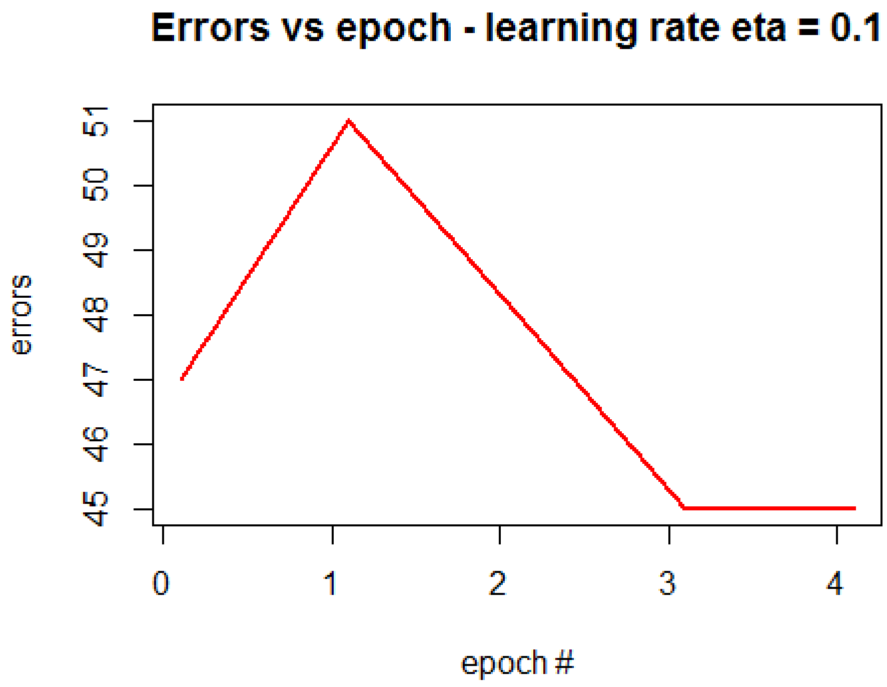
```
head(y)
```

```
## [1] 1 1 1 1 1 1
```

```
perceptron <- function(x, y, eta, niter) {

        # initialize weight vector
        weight <- rep(0, dim(x)[2] + 1)
        errors <- rep(0, niter)

        # loop over number of epochs niter
        for (jj in 1:niter) {

                # loop through training data set
                for (ii in 1:length(y)) {

                        # Predict binary label using Heaviside activation
                        # function
                        z <- sum(weight[2:length(weight)] *
                                 as.numeric(x[ii, ])) + weight[1]
                        if(z < 0) {
                                ypred <- -1
                        } else {
                                ypred <- 1
                        }

                        # Change weight - the formula doesn't do anything
                        # if the predicted value is correct
                        weightdiff <- eta * (y[ii] - ypred) *
                                c(1, as.numeric(x[ii, ]))
                        weight <- weight + weightdiff

                        # Update error function
                        if ((y[ii] - ypred) != 0.0) {
                                errors[jj] <- errors[jj] + 1
                        }

                }
        }

        # weight to decide between the two species
        print(weight)
        return(errors)
}

err <- perceptron(x, y, 0.1, 5)
```

```
## [1] 4.000000e-01 4.387833e+00 2.891000e+02 4.254667e+01 5.286345e-03
```

```
plot(0.1:5, err, type="l", lwd=2, col="red", xlab="epoch #", ylab="errors")
title("Errors vs epoch - learning rate eta = 0.1")
```



**Appendix – 5 {Perceptron Algorithm}**

# Team14_Project_Perceptron

Prajwal Gonnade; Supreet Nayak

December 9, 2016

```
library(boot)

## Warning: package 'boot' was built under R version 3.2.5

library(caret)

## Warning: package 'caret' was built under R version 3.2.5

## Warning: package 'ggplot2' was built under R version 3.2.5

library(class)
library(ROCR)

## Warning: package 'ROCR' was built under R version 3.2.5

## Warning: package 'gplots' was built under R version 3.2.5

library(MASS)
library(monmlp)

## Warning: package 'monmlp' was built under R version 3.2.5

Occupancy_Train <- read.csv(file.choose(),header=T)
Occupancy_Test1 <- read.csv(file.choose(),header=T)
Occupancy_Test2 <- read.csv(file.choose(),header=T)
Occupancy_subset <- Occupancy_Train[,c(2,4,5,6)]
Occupancy_subset <- data.matrix(Occupancy_subset)
```

```
Occupancy_response <- Occupancy_Train[,7]
Occupancy_response <- data.matrix(Occupancy_response)
```

## Perceptron

```
# Fit the model and compute the predictions
Occupancy.monmlp <- monmlp.fit(Occupancy_subset, Occupancy_response, hidden1=3, n.ensemble=15, monotone=1, bag=TRUE)
```

```
## ** Ensemble 1
## ** Bagging on
## 1 0.05689021
## ** 0.05689021
##
## ** Ensemble 2
## ** Bagging on
## 1 0.06621595
## ** 0.06621595
##
## ** Ensemble 3
## ** Bagging on
## 1 0.0762087
## ** 0.0762087
##
## ** Ensemble 4
## ** Bagging on
## 1 0.0714974
## ** 0.0714974
##
## ** Ensemble 5
## ** Bagging on
## 1 0.07462134
## ** 0.07462134
##
## ** Ensemble 6
## ** Bagging on
## 1 0.07064936
## ** 0.07064936
##
## ** Ensemble 7
## ** Bagging on
## 1 0.05917637
## ** 0.05917637
##
## ** Ensemble 8
## ** Bagging on
## 1 0.0565633
## ** 0.0565633
##
## ** Ensemble 9
## ** Bagging on
## 1 0.05639147
## ** 0.05639147
##
## ** Ensemble 10
## ** Bagging on
## 1 0.0667447
## ** 0.0667447
##
## ** Ensemble 11
## ** Bagging on
## 1 0.05977946
## ** 0.05977946
##
## ** Ensemble 12
## ** Bagging on
## 1 0.06211387
## ** 0.06211387
##
## ** Ensemble 13
## ** Bagging on
## 1 0.06509851
## ** 0.06509851
##
## ** Ensemble 14
## ** Bagging on
## 1 0.07589035
## ** 0.07589035
##
## ** Ensemble 15
## ** Bagging on
## 1 0.05706769
## ** 0.05706769
```

```
monmlp.test1 <- monmlp.predict(x = data.matrix(Occupancy_Test1[,c(2,4,5,6)]), weights = O
ccupancy.monmlp)
mean(round(monmlp.test1) != Occupancy_Test1$Occupancy)
```

```
## [1] 0.02138837
```

```
confusionMatrix(Occupancy_Test1$Occupancy, round(monmlp.test1))
```

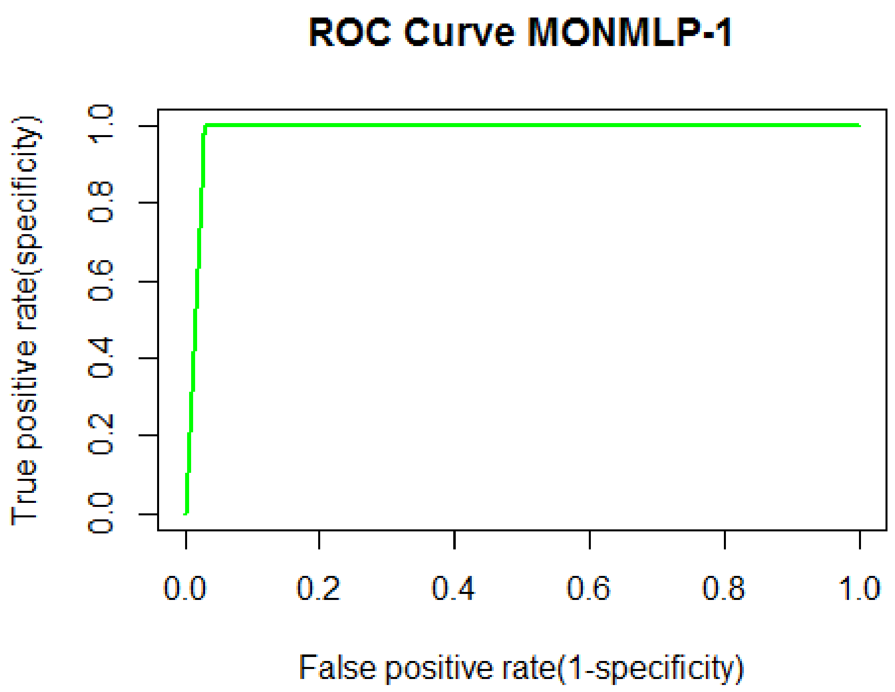
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1638   55
##          1    2  970
##
##                Accuracy : 0.9786
##                  95% CI : (0.9724, 0.9838)
##     No Information Rate : 0.6154
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9544
##  Mcnemar's Test P-Value : 5.675e-12
##
##             Sensitivity : 0.9988
##             Specificity : 0.9463
##          Pos Pred Value : 0.9675
##          Neg Pred Value : 0.9979
##              Prevalence : 0.6154
##          Detection Rate : 0.6146
##    Detection Prevalence : 0.6353
##       Balanced Accuracy : 0.9726
##
##        'Positive' Class : 0
##
```

```
# Compute the AUC
roc.curve=function(s,print=FALSE){
Ps=(monmlp.test1>s)*1
FP=sum((Ps==1)*(Occupancy_Test1$Occupancy == 0))/sum(Occupancy_Test1$Occupancy == 0)
TP=sum((Ps==1)*(Occupancy_Test1$Occupancy == 1))/sum(Occupancy_Test1$Occupancy == 1)
if(print==TRUE){
print(table(Observed=Occupancy_Test1$Occupancy,Predicted=Ps))
}
vect=c(FP,TP)
names(vect)=c("FPR","TPR")
return(vect)
}
threshold = 0.5
roc.curve(threshold,print=TRUE)
```

```
##          Predicted
## Observed    0    1
##        0 1638   55
##        1    2  970
```

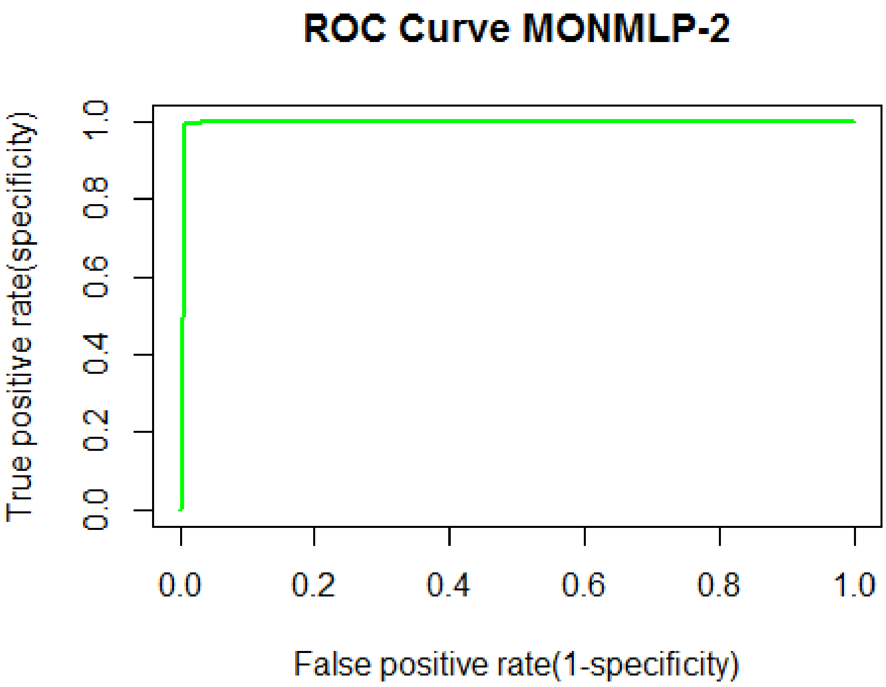
```
##        FPR        TPR
## 0.03248671 0.99794239
```

```
ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,],M.ROC[2,], xlab='False positive rate(1-specificity)', ylab='True positive
rate(specificity)',main = 'ROC Curve MONMLP-1', col="green",lwd=2,type="l")
```



```
monmlp.test2 <- monmlp.predict(x = data.matrix(Occupancy_Test2[,c(2,4,5,6)]), weights = O
ccupancy.monmlp)
mean(round(monmlp.test2) != Occupancy_Test2$Occupancy)

## [1] 0.01066448

confusionMatrix(Occupancy_Test2$Occupancy, round(monmlp.test2))

## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0    1
##          0 7609   94
##          1   10 2039
##
##                Accuracy : 0.9893
##                  95% CI : (0.9871, 0.9913)
##     No Information Rate : 0.7813
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9683
##  Mcnemar's Test P-Value : 3.992e-16
##
##             Sensitivity : 0.9987
##             Specificity : 0.9559
##          Pos Pred Value : 0.9878
##          Neg Pred Value : 0.9951
##              Prevalence : 0.7813
##          Detection Rate : 0.7803
##    Detection Prevalence : 0.7899
##       Balanced Accuracy : 0.9773
##
```

```
##         'Positive' Class : 0
##

# Compute the AUC
roc.curve=function(s,print=FALSE){
Ps=(monmlp.test2>s)*1
FP=sum((Ps==1)*(Occupancy_Test2$Occupancy == 0))/sum(Occupancy_Test2$Occupancy == 0)
TP=sum((Ps==1)*(Occupancy_Test2$Occupancy == 1))/sum(Occupancy_Test2$Occupancy == 1)
if(print==TRUE){
print(table(Observed=Occupancy_Test2$Occupancy,Predicted=Ps))
}
vect=c(FP,TP)
names(vect)=c("FPR","TPR")
return(vect)
}
threshold = 0.5
roc.curve(threshold,print=TRUE)

##          Predicted
## Observed    0    1
##        0 7609   94
##        1   10 2039

##        FPR        TPR
## 0.01220304 0.99511957

ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,],M.ROC[2,], xlab='False positive rate(1-specificity)', ylab='True positive rate(specificity)',main = 'ROC Curve MONMLP-2', col="green",lwd=2,type="l")
```


ROC Curve MONMLP-2
True positive rate(specificity)
False positive rate(1-specificity)