Room Occupation Prediction Final Project

1.Project Overview

1.1 Literature Review

There is a scientific evidence showing that there will be a lower energy consumption if we appropriately control the HVAC and lighting systems in buildings. In order to save energy, a model that can accurately detect the occupancy of a building is desired. However, due to privacy concerns, we could not use a camera. Alternatively, we will approach this model by using occupancy sensors. They are capable of detecting occupancy from multiple ways. For example, environment sensors are able to detect the presence of humans through the change of temperature, humidity, and density of carbon dioxide. Also, other detectors, such as PIR detector that measures infrared radiation and ultrasonic detector that works like a radar, can detect the occupancy of human based on their unique way of functioning.

Previously, a stochastic high-resolution domestic building occupancy model was built by Ian Richardson, Murray Thompson, and David Infield[1]. Their work shows an elaborate method of creating realistic data of occupancy in UK household from their surveyed time-use data that shows people's behavior in certain time. More importantly, their model is capable of telling the number of occupants in the detecting area in the given time, which helps calculate the share energy use and serves other purposes related to occupancy level.

1.2 Our Goal

We want to build a model that can accurately detect the occupancy of a room. The variables that we are concerned with include date (in the form year-month-day-hour-minute-second), temperature (in Celsius), relative humidity (in percentage), light (in Lux), carbon dioxide (in ppm), humidity ratio (in water-vapor/air), and occupancy (dummy variable). We focus on building the model that demonstrates the accountability of each of these features detected.

1.3 Our Approach

Intuitively, some of these variables may have a more significant impact on the models than others. Based on three statistical models: LDA(Linear Discriminant Analysis), RF(Random Forest) and CART(Classification and Regression Tree), we first find out the relationship

---

[1] Richardson, Ian, Murray Thomson, and David Infield. "A High-resolution Domestic Building Occupancy Model for Energy Demand Simulations." *Energy and Buildings*40, no. 8 (2008): 1560-566. doi:10.1016/j.enbuild.2008.02.006.

between the occupancy and the other variables, in addition to the correlation between every two parameters. Then, we choose the tuning parameters based on our findings, and compare the prediction performance of the validation set on each combination. Finally, we decide on the best model and test its accuracy prediction rate.

2. Data Visualization

2.1 Data sets

There are three data sets collected by the sensors. The first one is treated as a training set, which is used to fit a model. The second one is the validation set, that is used to select tuning parameters. The last one is the test set, which we use to assess the prediction performance. The table below summarizes the total observations of each data set and the overall occupancy rate & non-occupancy rate.

Data.set	Number.of. observations	Non.occupancy.rate	Occupancy.rate
Training	**8141**	**0.788**	**0.212**
Validation	**2665**	**0.635**	**0.365**
Test	**9752**	**0.790**	**0.210**

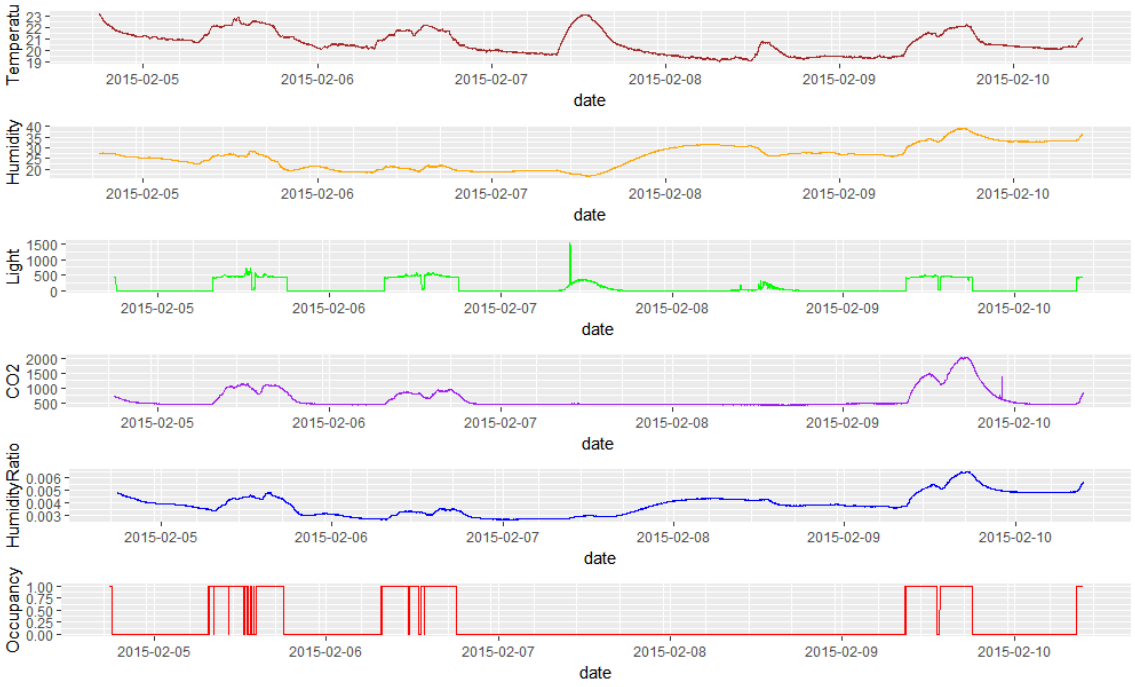
Table 1. Data set Description

2.2 Data Visualization

To have a clearer idea on the data set we are studying, we choose to visual the training set as an example. Fig. 1 shows data between the afternoon of Feb 4, 2015 to morning of Feb 10, 2015. The figure was created with the ggplot2 package and was combined by ggpubr package. We observed that the patterns of those six parameters change consistently with each other. Thus, it is reasonable to use them to build our model. Next, we want to find out which of these parameters do(es) a better prediction on the occupancy.

Fig.1. Time series plot of Temperature, Humidity, Light, CO2, HumidityRatio and Occupancy for 6 days (afternoon of Feb 4, 2015 to morning of Feb 10, 2015)

We first find the relationship between each parameter. To achieve this, we draw a paired plot (Fig. 2.) that shows the relationship for every two parameters while separates the status of occupancy. The dark blue dots represent the occupied status and the light blue dots correspond to the unoccupied status. It is easy to observe that there is a clear separation boundary in the combination pairs of temperature and light, humidity and light, light and CO2, light and humidity ratio, and light and time in seconds. Thus, we are more interested in training on those pairs for

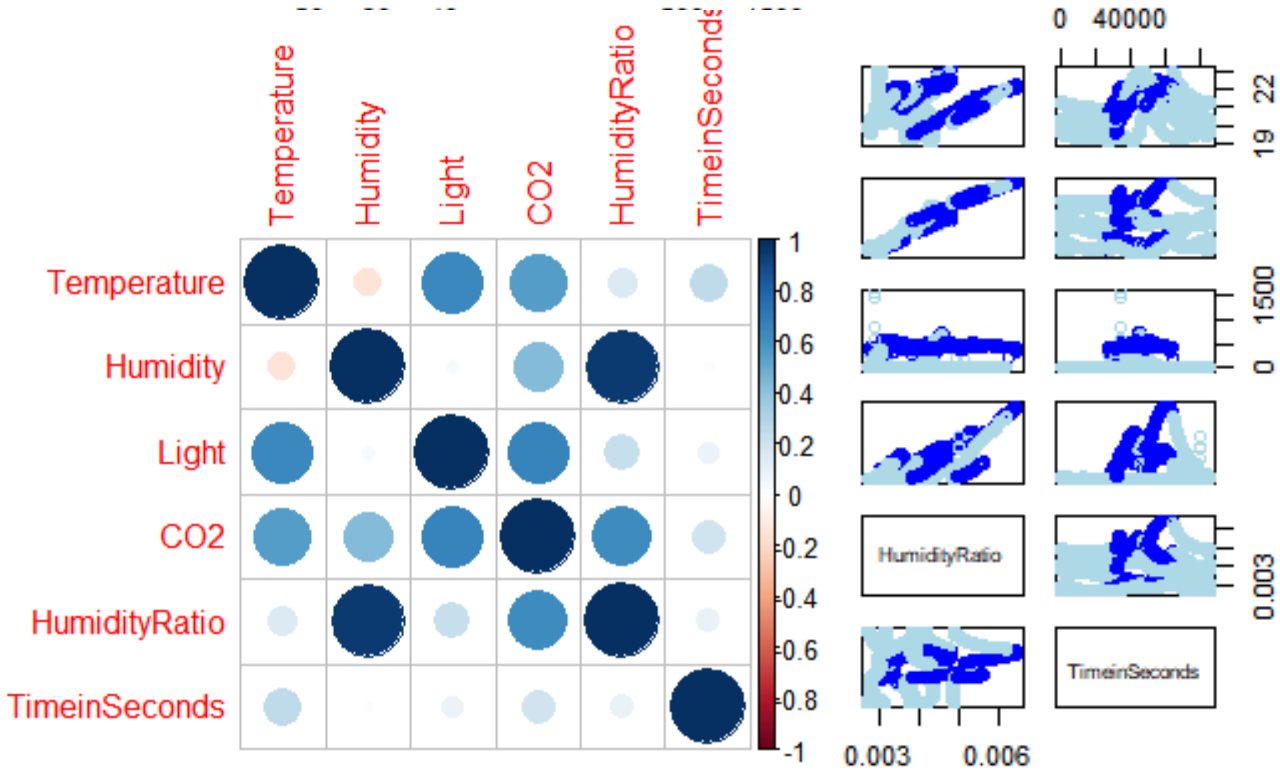


the classification model.

It is possible that a combination of more than two parameters can have a better effect on fitting the model. Therefore, we still need to study the correlation between parameters. Fig. 3. is the correlation plot, where the bigger and darker the ball is, the stronger the linear relation will be. Thus, we also pick the combination pairs which have stronger correlation between each other, such as carbon dioxide concentration and humidity.

Fig. 2. Pairs plot. The dark blue data correspond to the occupied status and light blue correspond to the unoccupied status

Fig. 3. Correlation Plot.
The positive correlations are shown in blue and the negative correlations in red.
The sizes of the circles are proportional to the correlation values in Table 2.



	Temperature	Humidity	Light	CO2	HumidityRatio	TimeinSeconds
Temperature	**1.00**	**-0.143**	**0.654**	**0.560**	**0.152**	**0.260**
Humidity	**-0.143**	**1.00**	**0.040**	**0.440**	**0.955**	**0.017**
Light	**0.654**	**0.040**	**1.00**	**0.670**	**0.234**	**0.086**
CO2	**0.560**	**0.440**	**0.670**	**1.00**	**0.627**	**0.209**
HumidityRatio	**0.152**	**0.955**	**0.234**	**0.627**	**1.00**	**0.096**
TimeinSeconds	**0.260**	**0.017**	**0.086**	**0.209**	**0.096**	**1.00**

Table 2. Correlation Matrix of Temperature, Humidity ,Light, CO2, HumidityRatio and TimeinSeconds.

	Temperature	Humidity	Light	CO2	HumidityRatio
Temperature	**NA**	**0.0000000000**	**0.000000e+00**	**0**	**0**
Humidity	**0**	**NA**	**6.396081e-04**	**0**	**0**

Light	**0**	**0.0006396081**	**NA**	**0**	**0**
CO2	**0**	**0.0000000000**	**0.000000e+00**	**NA**	**0**
HumidityRatio	**0**	**0.0000000000**	**0.000000e+00**	**0**	**NA**
TimeinSeconds	**0**	**0.1256260500**	**1.154632e-14**	**0**	**0**

Table 3. Corresponding P-value of the Correlation Matrix.

3. Classifiers Comparison

Here, we use three different statistical models to analyse these data sets where we consider all parameters(Temperature, Humidity, CO2, Light, HumidityRatio, Date(in seconds)).

3.1 LDA

LDA model is subject to solve a classification problem. From the training set, it calculates the prior probabilities of each variable by using Baye's theorem and estimates the parameters of the probability density function. Then, the discriminant function is calculated. Last but not least, we use the model to classify the observations in the validation set and calculate the accuracy rates.

3.2 Classification and Regression Tree

The CART models are trained with **rpart** package. By finding the best cp value that gives less x error for each model, the models are trained more accurately. Fig 4. and Fig 5. are created with **rattle** package, green box represents non-occupied status and blue box represents occupied status. As we can see from both plots, the top node are both Light, which means Light is the most important factor in determining the occupancy status.

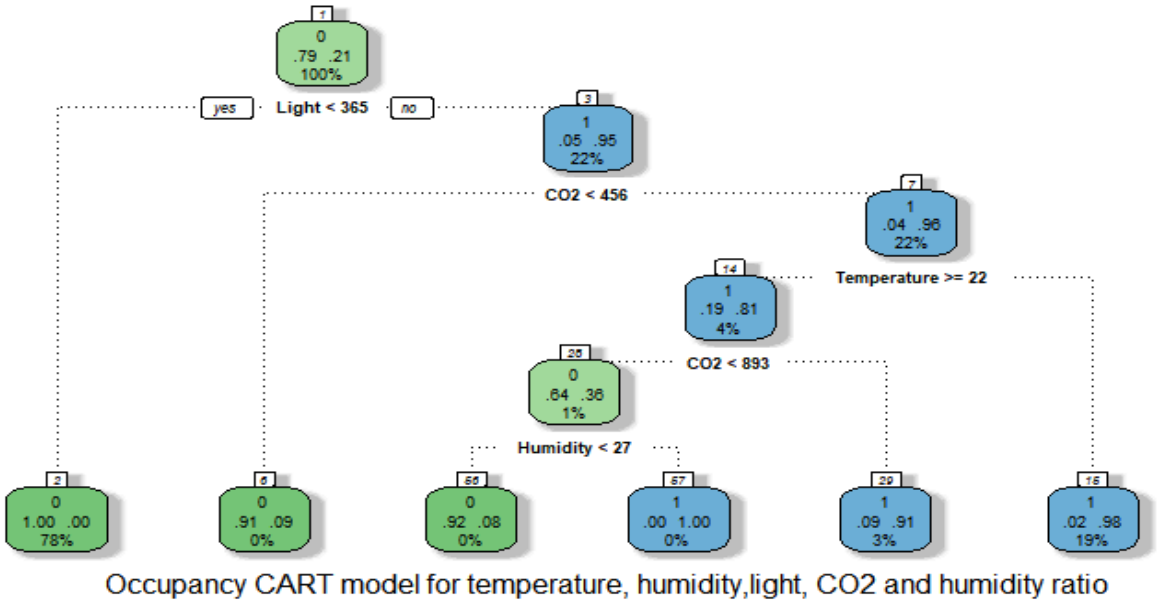


Fig 4. CART model tree plot of parameters including temperature, humidity, light, CO2 and HumidityRatio

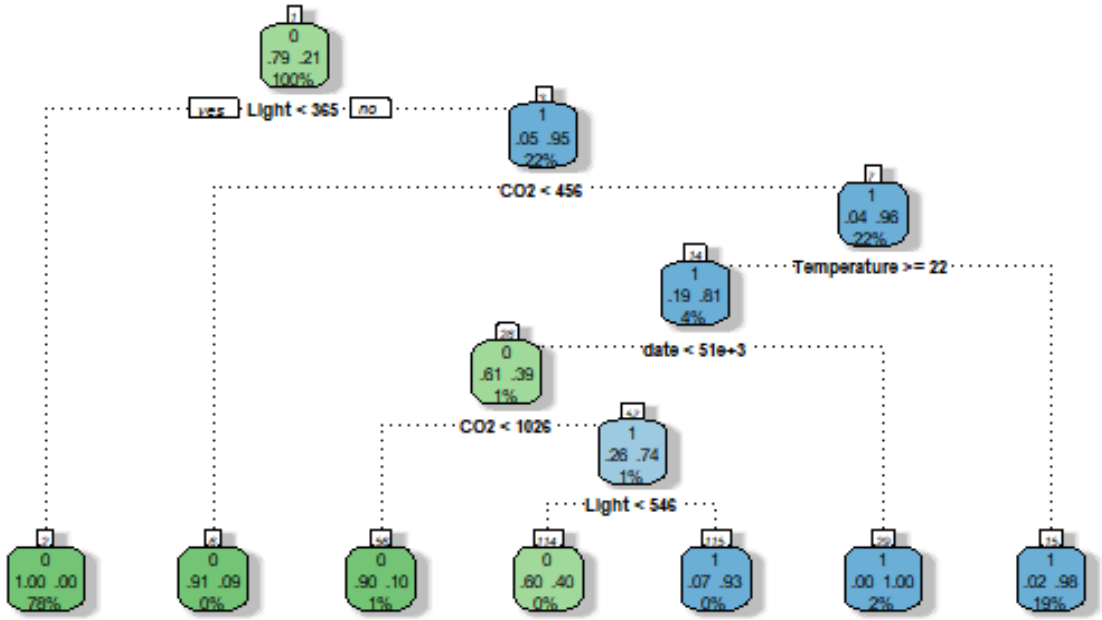


Fig 5. CART model tree plot of parameters including temperature, humidity, light, CO2, date(in terms of time in seconds), and humidity ratio

3.3 Random Forest

Random Forests are ways to provide us classification or regression of the samples through many runs. Each time, the algorithm picks up random sample of predictors and choose

the best decision tree for if. Then it runs for 1000 times and outputs the mostly used classification among those trees. And then use the data that are not used in obtaining the classification to estimate the error of the prediction.

3.4 Informal Comparison

Without grouping the parameters into different combinations, we run the three statistical models for all the parameters respectively. An informal observation from the results is that LDA does the best in prediction.

Model	Accuracy training	Accuracy validation
LDA	0.9839086	0.9789869
CART	0.9942282	0.9452158
RF	0.9992361	0.9664463

Table 8. Accuracy of each model's prediction for all parameters.

4. Tuning Parameters

Now, in order to achieve a higher prediction rate, we fit models and do predictions by selecting tuning parameters from the training set instead of using all of them. Same choices of tuning parameters are conducted by three different models respectively. The way we choose has been indicated by the data visualization above. After conducting the following ten groups that use different combinations of tuning parameters, we select the model that has the highest accuracy validation rate and use it to test the test set.

Model	Parameters	Accuracy training	Accuracy validation	Accuracy testing
LDA	Light	0.9597101	0.9786116	
CART	Light	0.9878	0.9786	
RF	Light	0.9983624	0.9683305	
LDA	Light, Humidity	0.964255	0.9786116	
CART	Light, Humidity	0.9925	0.9568	

RF	Light, Humidity	0.9929578	0.9463607	
LDA	Light, CO2	0.9696597	0.9786116	
CART	Light, CO2	0.9889	0.9782	
RF	Light, CO2	0.9978805	0.9405976	
LDA	Light, HR	0.963518	0.9786116	
CART	Light, HR	0.9878	0.9786	
RF	Light, HR	0.9980918	0.9679693	
LDA	Light, Temp	0.9625353	0.9789869	0.9843109
CART	Light, Temp	0.9916	0.8837	
RF	Light, Temp	0.997632	0.9564639	
LDA	CO2, Humidity	0.9243336	0.8754221	
CART	CO2, Humidity	0.9832	0.8071	
RF	CO2, Humidity	0.9967849	0.8256744	
LDA	Time, Temp, CO2	0.9253163	0.8765478	
CART	Time, Temp, CO2	0.9953	0.9482	
RF	Time, Temp, CO2	0.9991205	0.9509418	
LDA	Time, Light, Humidity	0.964255	0.9786116	
CART	Time, Light, Humidity	0.9952	0.9674	
RF	Time, Light, Humidity	0.9992193	0.9662563	
LDA	Temp, CO2, Humidity	0.9264218	0.8705441	
CART	Temp, CO2, Humidity	0.9862	0.7456	
RF	Temp, CO2, Humidity	0.9977617	0.7973692	
LDA	CO2, Time, Temp, Humidity	0.9286328	0.8840525	
CART	CO2, Time, Temp, Humidity	0.9925	0.8904	
RF	CO2, Tiime, Temp, Humidity	0.9990727	0.954176	

Table 9. Model performance

5. Conclusion

5.1 Decision

We observe that the LDA model provides a consistent accuracy prediction for the training and validation sets when using the pairs of light and humidity, light and CO2, light and temperature, light and humidity ratio and light alone.

The best model is LDA with tuning parameters light and temperature since it has the highest accuracy on the prediction of the validation set. Moreover, this model achieves an accuracy on prediction of above 98% when we use it to test on the test set.

5.2 Discussion

We are satisfied with the LDA model we choose since it shows a consistent outstanding prediction performance on both the validation set and the test set. However, it is worth pointing out that if we look at the prediction performance on the training set alone, LDA does not do a great job compared to the other two statistical models. It is probably because of the fact that LDA does not consider the response when doing classification. But our goal is to build a model that can perform prediction at a higher rate of accuracy given any new observations. Thus, it does not influence our decision on choosing the LDA model. Moreover, its property on classification even favors its prediction. It has been discussed above that there is a clear separation of the occupancy in the pairs, light and any other parameters. Therefore, our hyper-parameters are more likely to be one from these pairs.

Code:

Data Visualization

```
Test <- read.csv("Test.txt")
Validation<-read.csv("Training.txt")
Training<-read.csv("Validation.txt")

#Data visualization of Validation dataset
library(ggplot2)
Validation$date<-as.POSIXct(Validation$date)

ggplot(Validation,aes(x=date,y=Temperature)) +
  geom_line(color="brown")+
  scale_x_datetime(breaks ="7 hours",
                   minor_breaks = "1 hour")

ggplot(Validation,aes(x=date,y=Humidity)) +
  geom_line(color="orange")+
  scale_x_datetime(breaks = "7 hours",
                   minor_breaks ="1 hour")

ggplot(Validation,aes(x=date,y=Light)) +
  geom_line(color="green")+
  scale_x_datetime(breaks = "7 hours",
                   minor_breaks ="1 hour")

ggplot(Validation,aes(x=date,y=CO2)) +
  geom_line(color = "purple")+
  scale_x_datetime(breaks = "7 hours",
                   minor_breaks ="1 hour")

ggplot(Validation,aes(x=date,y=HumidityRatio)) +
  geom_line(color = "blue")+
  scale_x_datetime(breaks ="7 hours",
                   minor_breaks = "1 hour")

ggplot(Validation,aes(x=date,y=Occupancy)) +
  geom_line(color = "red")+
  scale_x_datetime(breaks ="7 hours",
                   minor_breaks = "1 hour")

#Pairs plot(blue represents occupancy, lightblue represents non-occupancy)
#convert time for each day into seconds (values stored in TimeinSeconds)
TimeinSeconds<-c(1:length(Validation$date))
for(i in 1:length(Validation$date)){
  TimeinSeconds[i]=(as.numeric(strsplit(format(Validation$date,"%H:%M:%S"),":")[[i]]
[1])*3600+
          as.numeric(strsplit(format(Validation$date,"%H:%M:%S"),":")[[i]][2])*60+
           as.numeric(strsplit(format(Validation$date,"%H:%M:%S"),":")[[i]][3]))}

data<-as.matrix(cbind(Validation[,2:6],TimeinSeconds))
color=numeric(nrow(data))
```

```
for(i in 1:nrow(data))
if(Validation$Occupancy[i]=="1"){
  color[i]="blue"
}else{color[i]="lightblue"}
pairs(data,col=color)

#Correlation matrix and its corresponding p-value.
library("Hmisc")
rcorr(data)
rcorr(data)$P
library("corrplot")
corrplot(rcorr(data)$r,method = "circle")

#Table 5. Dataset Description
NumObs<-c(nrow(Training),nrow(Validation),nrow(Test))
count1<-0
count2<-0
count3<-0
for(i in 1:nrow(Training))
{ if(Training$Occupancy[i]=="0")
  count1=count1+1}
for(i in 1:nrow(Validation))
{ if(Validation$Occupancy[i]=="0")
  count2=count2+1}
for(i in 1:nrow(Test))
{ if(Test$Occupancy[i]=="0")
  count3=count3+1}
ratio1=count1/nrow(Training)
ratio2=count2/nrow(Validation)
ratio3=count3/nrow(Test)
distribution_nonoccupancy<-round(c(ratio1,ratio2,ratio3),3)
distribution_occupancy<-round(c(1-ratio1,1-ratio2,1-ratio3),3)
data.frame("Data set"=c("Training","Validation","Test"),
           "Number of observations"=NumObs,
           "Non-occupancy rate"=distribution_nonoccupancy,
           "Occupancy rate"=distribution_occupancy)


LDA
#setup
#setup data sets
training <- read.csv("Training.txt",header = T)
validation <- read.csv("Validation.txt",header = T)
test <- read.csv("Test.txt",header = T)

#delete outlier
training<-training[training$Light<900,]

#set variables
sub_training<-training[,2:7]
sub_validation<-validation[,2:7]
sub_test<-test[,2:7]

#Training
```

```
n_train=nrow(sub_training)
n_test=nrow(sub_training)


############## Choose Prior ##############
n_1=sum(sub_training$Occupancy=="1")
n_0=sum(sub_training$Occupancy=="0")


############## Choose Prior ##############
#Prior=relative sample size in train data
p_1=n_1/n_train
p_0=n_0/n_train


############## Calculate sample mean vectors ##############
train_1=sub_training[sub_training$Occupancy=="1",]
train_0=sub_training[sub_training$Occupancy=="0",]

Mean_1=colMeans(train_1[,1:5])
Mean_0=colMeans(train_0[,1:5])

############## Calculate pooled variance-covariance matrix ##############
#Sample variance-covariance matrix for each species
S_1=cov(train_1[,1:5])
S_0=cov(train_0[,1:5])

#Complete fomula
S_pooled= ((n_1-1)*S_1+(n_0-1)*S_0)/(n_1+n_0+-2)

S_inv=solve(S_pooled)

#Simple way
#S_pooled=(S_setosa+S_versicolor+S_virginica)/3

############## Calculate alpha_i ##############

alpha_1= -0.5* t(Mean_1) %*% S_inv %*% Mean_1
alpha_0= -0.5* t(Mean_0) %*% S_inv %*% Mean_0

############## Calculate beta_i ##############

beta_1=S_inv %*% Mean_1
beta_0=S_inv %*% Mean_0

############## Classification ##############
prediction=c()
d_1_vec=c()
d_0_vec=c()

label=c("1", "0")

for(i in 1:nrow(sub_training)){
  #Read an observation in test data
```

```
  x=t(sub_training[i,1:5])

  #Calculate linear discriminant functions for each species
  d_1=alpha_1+ t(beta_1) %*% x
  d_0=alpha_0+ t(beta_0) %*% x


  #Classify the observation to the species with highest function value
  d_vec=c(d_1, d_0)
  prediction=append(prediction, label[which.max( d_vec )])

  d_1_vec=append(d_1_vec, d_1)
  d_0_vec=append(d_0_vec, d_0)

}

#Combine the predicted results to the test dataset.
sub_training$prediction=prediction
sum(sub_training$Occupancy!=sub_training$prediction)

correct_rate= 1-sum(sub_training$Occupancy!=sub_training$prediction)/
nrow(sub_training)
correct_rate


#Validation
n_train=nrow(sub_training)
n_test=nrow(sub_validation)


############### Choose Prior ###############
n_1=sum(sub_training$Occupancy=="1")
n_0=sum(sub_training$Occupancy=="0")


############### Choose Prior ###############
#Prior=relative sample size in train data
p_1=n_1/n_train
p_0=n_0/n_train


############### Calculate sample mean vectors ###############
train_1=sub_training[sub_training$Occupancy=="1",]
train_0=sub_training[sub_training$Occupancy=="0",]

Mean_1=colMeans(train_1[,1:5])
Mean_0=colMeans(train_0[,1:5])


############### Calculate pooled variance-covariance matrix ###############
#Sample variance-covariance matrix for each species
S_1=cov(train_1[,1:5])
S_0=cov(train_0[,1:5])
```

```
#Complete fomula
S_pooled= ((n_1-1)*S_1+(n_0-1)*S_0)/(n_1+n_0+-2)

S_inv=solve(S_pooled)

#Simple way
#S_pooled=(S_setosa+S_versicolor+S_virginica)/3

############### Calculate alpha_i ###############

alpha_1= -0.5* t(Mean_1) %*% S_inv %*% Mean_1
alpha_0= -0.5* t(Mean_0) %*% S_inv %*% Mean_0

############### Calculate beta_i ###############

beta_1=S_inv %*% Mean_1
beta_0=S_inv %*% Mean_0

############### Classification ###############
prediction=c()
d_1_vec=c()
d_0_vec=c()

label=c("1", "0")

for(i in 1:nrow(sub_validation)){
  #Read an observation in test data
  x=t(sub_validation[i,1:5])

  #Calculate linear discriminant functions for each species
  d_1=alpha_1+ t(beta_1) %*% x
  d_0=alpha_0+ t(beta_0) %*% x


  #Classify the observation to the species with highest function value
  d_vec=c(d_1, d_0)
  prediction=append(prediction, label[which.max( d_vec )])

  d_1_vec=append(d_1_vec, d_1)
  d_0_vec=append(d_0_vec, d_0)

}

#Combine the predicted results to the test dataset.
sub_validation$prediction=prediction
sum(sub_validation$Occupancy!=sub_validation$prediction)

correct_rate= 1-sum(sub_validation$Occupancy!=sub_validation$prediction)/
nrow(sub_validation)
correct_rate

#Test
n_train=nrow(sub_training)
n_test=nrow(sub_test)
```

```
############## Choose Prior ##############
n_1=sum(sub_training$Occupancy=="1")
n_0=sum(sub_training$Occupancy=="0")

############## Choose Prior ##############
#Prior=relative sample size in train data
p_1=n_1/n_train
p_0=n_0/n_train

############## Calculate sample mean vectors ##############
train_1=sub_training[sub_training$Occupancy=="1",]
train_0=sub_training[sub_training$Occupancy=="0",]

Mean_1=colMeans(train_1[,1:5])
Mean_0=colMeans(train_0[,1:5])

############## Calculate pooled variance-covariance matrix ##############
#Sample variance-covariance matrix for each species
S_1=cov(train_1[,1:5])
S_0=cov(train_0[,1:5])


#Complete fomula
S_pooled= ((n_1-1)*S_1+(n_0-1)*S_0)/(n_1+n_0+-2)

S_inv=solve(S_pooled)

#Simple way
#S_pooled=(S_setosa+S_versicolor+S_virginica)/3

############## Calculate alpha_i ##############

alpha_1= -0.5* t(Mean_1) %*% S_inv %*% Mean_1
alpha_0= -0.5* t(Mean_0) %*% S_inv %*% Mean_0

############## Calculate beta_i ##############

beta_1=S_inv %*% Mean_1
beta_0=S_inv %*% Mean_0

############## Classification ##############
prediction=c()
d_1_vec=c()
d_0_vec=c()

label=c("1", "0")

for(i in 1:nrow(sub_test)){
  #Read an observation in test data
  x=t(sub_test[i,1:5])

  #Calculate linear discriminant functions for each species
  d_1=alpha_1+ t(beta_1) %*% x
```

```
  d_0=alpha_0+ t(beta_0) %*% x


  #Classify the observation to the species with highest function value
  d_vec=c(d_1, d_0)
  prediction=append(prediction, label[which.max( d_vec )])

  d_1_vec=append(d_1_vec, d_1)
  d_0_vec=append(d_0_vec, d_0)

}

#Combine the predicted results to the test dataset.
sub_test$prediction=prediction
sum(sub_test$Occupancy!=sub_test$prediction)

correct_rate= 1-sum(sub_test$Occupancy!=sub_test$prediction)/nrow(sub_test)
correct_rate
```

RF

```
Test <- read.csv("Test.txt")
Training <- read.csv('Training.txt', header=TRUE, stringsAsFactors=FALSE, sep=',')
Validation<-read.csv("Validation.txt")
Validation$date<-as.POSIXct(Validation$date)
Training$date<-as.POSIXct(Training$date)
Test$date<-as.POSIXct(Test$date)

TimeinSecondsTraining<-c(1:length(Training$date))
for(i in 1:length(Training$date)){
  TimeinSecondsTraining[i]=(as.numeric(strsplit(format(Training$date,"%H:%M:%S"),":")
[[i]][1])*3600+
                as.numeric(strsplit(format(Training$date,"%H:%M:%S"),":")[[i]][2])*60+
                as.numeric(strsplit(format(Training$date,"%H:%M:%S"),":")[[i]][3]))}
TimeinSeconds <- TimeinSecondsTraining
AdjustedTraining <- as.matrix(cbind(Training,TimeinSeconds))
write.csv(AdjustedTraining,'AdjustedTraining.csv')

TimeinSecondsTest<-c(1:length(Test$date))
for(i in 1:length(Test$date)){
  TimeinSecondsTest[i]=(as.numeric(strsplit(format(Test$date,"%H:%M:%S"),":")[[i]]
[1])*3600+
                as.numeric(strsplit(format(Test$date,"%H:%M:%S"),":")[[i]][2])*60+
                as.numeric(strsplit(format(Test$date,"%H:%M:%S"),":")[[i]][3]))}
TimeinSeconds <- TimeinSecondsTest
AdjustedTest<-as.matrix(cbind(Test,TimeinSeconds))
write.csv(AdjustedTest,'AdjustedTest.csv')

TimeinSecondsValidation<-c(1:length(Validation$date))
for(i in 1:length(Validation$date)){
  TimeinSecondsValidation[i]=(as.numeric(strsplit(format(Validation$date,"%H:%M:
%S"),":")[[i]][1])*3600+
                as.numeric(strsplit(format(Validation$date,"%H:%M:%S"),":")[[i]][2])*60+
                as.numeric(strsplit(format(Validation$date,"%H:%M:%S"),":")[[i]][3]))}
TimeinSeconds <- TimeinSecondsValidation
```

```
AdjustedValidation<-as.matrix(cbind(Validation,TimeinSeconds))
write.csv(AdjustedValidation,'AdjustedValidation.csv')

ATest <- read.csv('AdjustedTest.csv')
ATraining <- read.csv('AdjustedTraining.csv')
AValidation <- read.csv('AdjustedValidation.csv')

RF.train = randomForest(Occupancy~.,data=ATraining[,3:9],
                        mtry=2, importance=TRUE, ntree=1000)

# Validation
yhat.RF_V = predict(RF.train, newdata=AValidation)
train.test_V = (Validation[,'Occupancy'])
MSE.RF_V = mean((yhat.RF_V - train.test_V)^2)
print(1-MSE.RF_V)

#Training
yhat.RF_TR = predict(RF.train, newdata=ATraining)
train.test_TR = (ATraining[,'Occupancy'])
MSE.RF_TR = mean((yhat.RF_TR - train.test_TR)^2)
print(1-MSE.RF_TR)

#Test
yhat.RF_T = predict(RF.train, newdata=ATest)
train.test_T = (ATest[,'Occupancy'])
MSE.RF_T = mean((yhat.RF_T - train.test_T)^2)
print(1-MSE.RF_T)

#CART (rattle package)
install.packages("rattle")
library(rattle)
install.packages("rpart")
library(rpart)
#Validation$Occupancy<-factor(Validation$Occupancy, labels = c("no", "yes"))
Training$Occupancy<-as.factor(Training$Occupancy)
tree<-rpart(Occupancy~
Temperature+Humidity+Light+CO2+HumidityRatio,data=Training,method="class"
          , control = rpart.control(cp=0.005))
fancyRpartPlot(tree,sub = "Occupancy CART model for temperature, humidity, light, CO2
and humidity ratio")
```