Kiran Potula, Ray Xinrui Wang, Hanson Qin, Valerie Youssef, Nayoung Park
MATH 189
9 June, 2019

Final Project

1.1 Introduction

In this project, we use various statistical methods to predict the occupancy of rooms by analyzing various factors. The response variable (the variable we want to predict) is the Occupancy of rooms, with 1 indicating occupied rooms and 0 indicating empty, or unoccupied rooms. The indicator variables, or parameters we will be using to predict the occupancy are: Temperature, Humidity, Light Intensity, Carbon Dioxide (CO2) level, and the Humidity Ratio of the rooms. We will first try to get a general idea for our data by visualizing and standardizing the data. For our analysis we will use the following classifiers, or prediction methods:: LDA, Random Forests, SVM and hierarchical clustering. We will compare the error rates obtained from each prediction model to figure out the best method of analysis for our dataset.

1.2 Data Visualization

The pairwise scatter plot (Figure 1.1**)** compares pairs of variables, with blue points indicating occupied rooms and yellow indicating unoccupied rooms. In this plot, we can see that for the pairwise plots ‘Light’ with ‘Temperature’, ‘Humidity’, ‘CO2’, ‘Humidity Ratio’ show a clear indication of two linear clusters of data points, meaning that Light can be a good indicator to pair with our other classifiers.

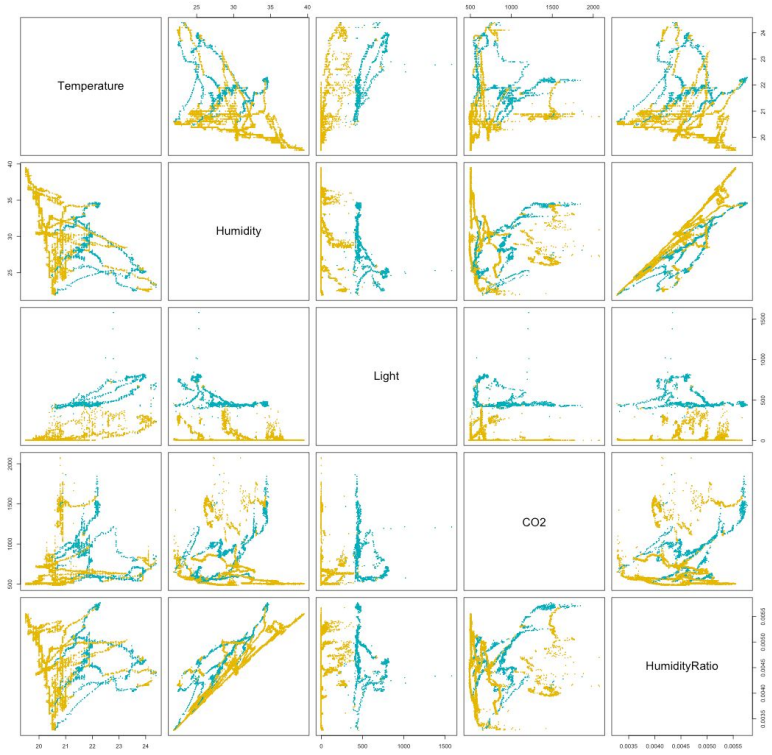

Figure 1.1 Pairwise Scatter Plot of all Indicator Variables

1.3 Standardized Data

To find the relations between each variables and Occupancy throughout the time frame, we constructed some time plots as follows to show their potential correlations. We standardized the data so that the variables are more comparable.

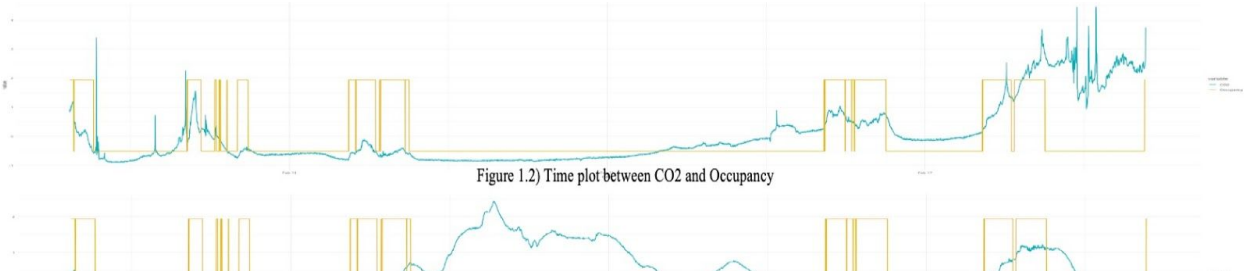

Figure 1.2) Time plot between CO2 and Occupancy

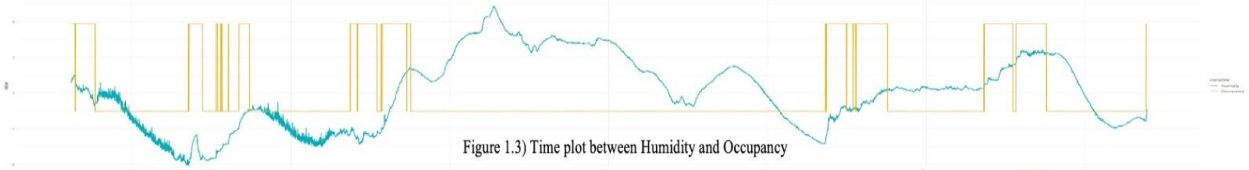

Figure 1.3) Time plot between Humidity and Occupancy

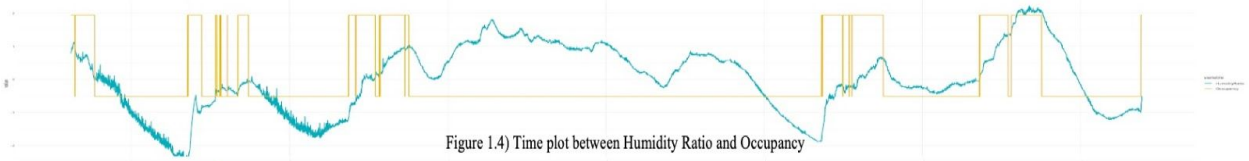

Figure 1.4) Time plot between Humidity Ratio and Occupancy

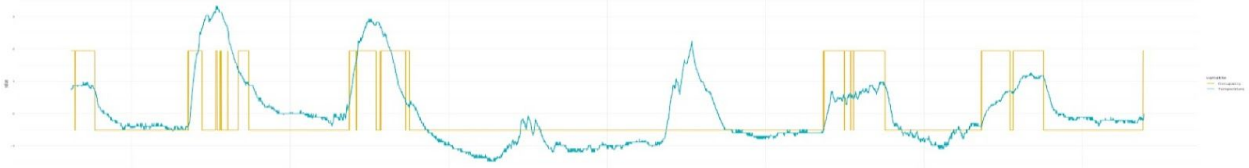

Figure 1.5) Time plot between Temperature and Occupancy

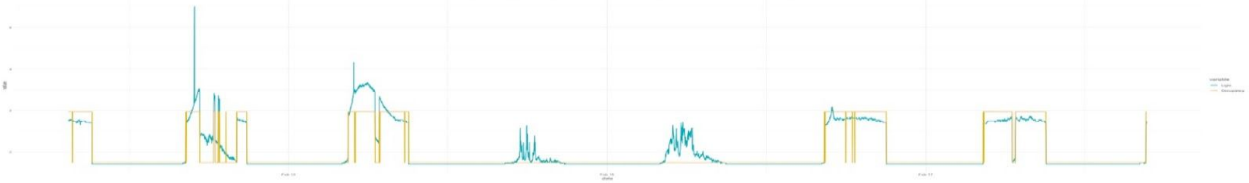

Figure 1.6) Time plot between Light and Occupancy

These time plots (Figures 1.2-6) compare the standardized data of each indicator variable ('CO2', 'Humidity', 'Humidity Ratio', 'Temperature', and 'Light') shown in blue lines and the response variable ('Occupancy') shown in yellow. The graphs show that all indicator variables are somewhat correlated to the response variable, with 'Light', 'Temperature', and 'CO2' having the strongest correlations with 'Occupancy' as a surge in those variables during certain time periods results the 'Occupancy' changing from 1 to 0.

1.4 PCA (Principal Component Analysis)

PCA is another statistical method to help visualize data by breaking it down to its principal components, which are linear combinations of the features that would account for the most variability of the data. Here, we have broken the data down to 5 principal components. In Figure 1.7, we see that the first two principal components represents almost 89% of the total variation, and that the first three represents 95% of the total variation.

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5
Standard deviation     1.6542 1.3036 0.59062 0.46300 0.02845
Proportion of Variance 0.5473 0.3399 0.06977 0.04287 0.00016
Cumulative Proportion  0.5473 0.8872 0.95696 0.99984 1.00000
```

Table 1.7

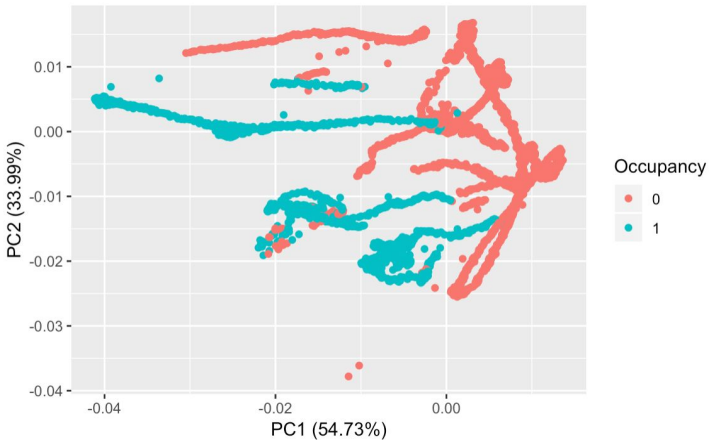

Figure 1.8

Figure 1.8 shows the PCA plot between the first two principal components, with the red dots representing an empty room, and the teal dots representing an occupied room. We see that the data is not cleanly separable when viewed with respect to the first two principal components, but they are lined up and grouped in a peculiar way.

2. LDA (Linear Discriminant Analysis)

For this method, we assumed that all variables follow a normal distribution. We first divided the training set into two subsets as follows: subset Occupied contains all the data (1729 entries) in the training set whose 'Occupancy' value equals 1; subset Unoccupied contains the rest of the data (6414 entries) in the training set where the 'Occupancy' column is 0. We found the priors from the training set according to the relative sample size and used according training data to estimate the coefficients for the linear discriminant functions. With this information, we calculated the linear discriminant functions for each observation in the Test set and classified each result into either 'Occupied' (prediction = 1) or 'Unoccupied' (prediction = 0). In the end, we compared our predictions with the truth in the Test set and our results are the following:

Out of 9752 entries from the Test data set, we correctly predicted 9631 observations using LDA methods. Only 121 observations are misclassified, where the truth is different from our prediction. With an accuracy of **0.9876**, the LDA method does pretty well in predicting the Occupancy of the room.

Aiming to further fine tune our classifier, we chose different combinations of indicator variables. We found out that when dropping 'Humidity Ratio' as an input variable, the test result actually becomes more accurate with only 85 errors and an accuracy of **0.9913**. Performing LDA using other sets of parameters showed no improvement of our classifier. We also found that using the humidity ratio as a predictor shows almost identical results as using the humidity does (both had 85 errors when either one of the two were not used, and 208 or 210 errors when the 'Humidity' or 'Humidity Ratio' column was dropped along with 'Temperature').

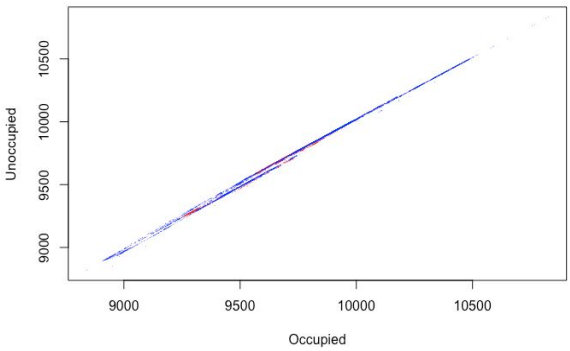

Figure 2.1 Classification plot using LDA on all columns

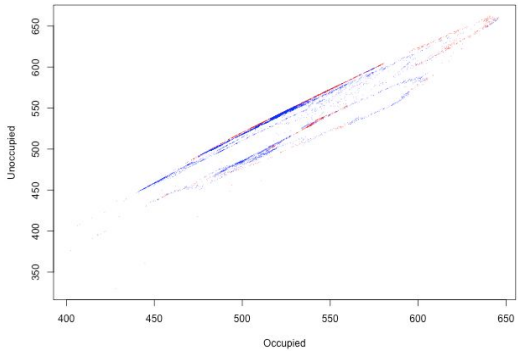

Figure 2.2 Classification plot without Humidity Ratio

Some other combinations of parameters:

1) Param = CO2, Humidity, Light
    a) Errors: 210
    b) Accuracy: (9752-210)/9752 := 0.9785
2) Param = CO2, Humidity Ratio, Light
    a) Misclassified = 208
    b) Accuracy = (9752-208)/9752 = 0.9787
3) Param = CO2, Humidity Ratio, Light, Temperature
    a) Misclassified = 85
    b) Accuracy = (9752-85)/9752 = 0.9913
4) Param = CO2, Light, Temperature
    a) Misclassified = 94
    b) Accuracy = (9752-94)/9752 = 0.9904

3. Random Forests

Random forests provide an improvement over bagged trees by a random small tweak that decorrelates the trees. As in bagging, we build a number of decision trees on bootstrapped training samples. But when building these decision trees, each time a split in a tree is considered, a random sample of *m* predictors is chosen as split candidates instead of the full set of *p* predictors. We generated 100 random forests and plotted their error rates in the histogram provided below.

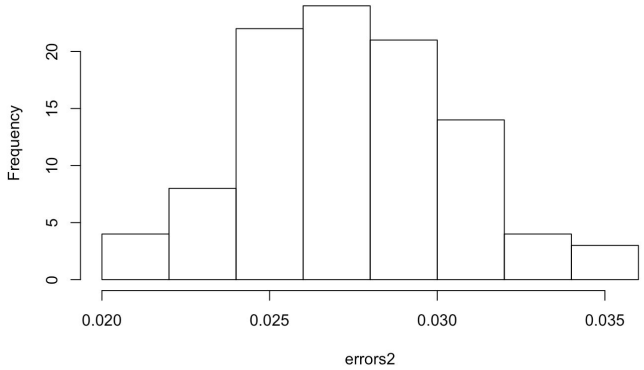

Figure 3.1

From Figure 3.1, we can see that the Test Error Rates approximately follow a normal distribution. Therefore, it would be reasonable to conduct the process several times and take the average of the Test error rates.

When using Random Forest to analyze the data, we tried generating a random forest using every single number of variables randomly sampled as candidates at each split. For every value of *m*, we generated 50 random forests and took the average of the values of the Test Error Rates. Below is a table showing the result of the test error rate corresponding to the 5 different m values.

m value	Test Error Rate
1	0.02902994
2	0.02716981
3	0.0278589
4	0.02933757
5	0.03208573

Table 3.2

From the values in Table 3.2, we can conclude that using the value of $m = 2$ gives the best results because it minimizes the error rate.

Figure 3.3 is an example of a Random Forest generated with the value $m = 2$. The Error rate generated from the Testing data set was equal to .0276, which is within the accepted rate, resulting in an accuracy of **0.9724.**

```
 randomForest(formula = Occupancy ~ ., data = Training.data, mtry = 2,      importance = TRUE,
xtest = Test.data[, -6], ytest = Test.data$Occupancy,      keep.forest = TRUE, ntree = 500)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 0.53%
Confusion matrix:
     0    1 class.error
0 6389   25 0.003897724
1   18 1711 0.010410642
                Test set error rate: 2.76%
Confusion matrix:
     0    1 class.error
0 7471  232  0.03011814
1   37 2012  0.01805759
```

Figure 3.3

4. SVM (Support Vector Machines)

If we have data that could be represented by a multi-dimensional space, a support vector machine would attempt to draw a hyperplane that would attempt to separate the data based on another feature, which in this case is the occupancy, the best it can. This way, it is very useful for discriminating between two classes of data, which is precisely what we need. It is different than the LDA in that it allows for a lot more flexibility in the data.

For this method, we start by scaling both the Training and Test data. We train the svm model using a linear kernel on the training set, then test them on the testing set. We train the models on the training set, and run them on the test set. We get a score of **0.9924** for the scaled data, and **0.9923** for the raw data. It seems like scaling the data produces slightly better scores, but the linear svm model works very well on both types of data.

Since from the data analysis, we saw that the first two principal components of the data comprises around 88% of the total variation, we wanted to try running the svm model on those first two principal components. We do this by calculating the first two principal components of the training set, training the svm model on that set of principal components, transforming the data in the test set to be in terms of the first two principal components of the training set using R's predict function, and using the newly trained svm model to make predictions on the transformed test set.Running that newly trained linear svm model, we got a score of **0.9285**, which is considerably worse than that of the scaled data.

Number of Principal Components	1	2	3	4	5
Linear SVM Score	0.8549	0.9285	0.9924	0.9924	0.9916
Cumulative Proportion of Total Variation	0.5473	0.8872	0.9570	0.9998	1.0000

Table 4.1

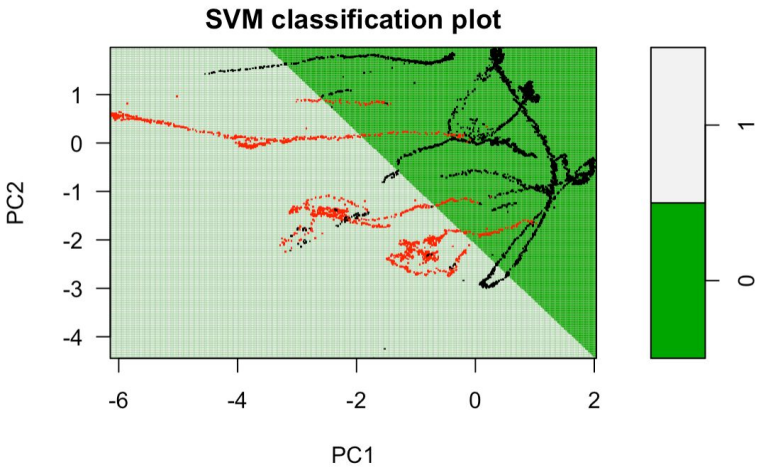

Figure 4.2

We experimented by running our model on different numbers of principal components. Table 4.1 summarizes our results. Figure 4.2 shows plots for linear svm on the first two principal components; the dark green region represents the space that the svm model classifies as unoccupied (0), and the light green as occupied (1). The black points correspond to empty rooms, and the red points represent the occupied rooms.

To improve performance, we could experiment with non-linear kernels, like the gaussian or polynomial kernels. However, since our score with the linear kernel is so high, we doubt that using non-linear methods would further increase our accuracy. Furthermore, the high accuracy of the LDA method further supports the fact that the data is able to be linearly separated by the occupancy very well, and that non-linear methods would likely be intrusive.

5. Hierarchical Clustering

We chose hierarchical clustering as one of our classifiers because we wanted to explore whether or not the dataset had previously undefined cluster groups. This is important because by finding these undefined cluster groups we can pair different, seemingly diverse data together. This can lead to identifying relationships between variables that were previously unseen. To do this we first looked at the data and noticed that there were many more unoccupied values than occupied values. Meaning the clusters that were created in the test would tend to group alongside each other. Of the two clusters created the bigger one would represent the unoccupied people with the other being occupied. We could then use this information to solve for the error rates.

For this method we had to pick two different parameters: distance and linkage method. For distance we chose euclidean because it was the most direct path between points. We considered doing taxicab distance but using an indirect path felt unnecessary with this particular dataset. Secondly for the linkage method we chose complete. The reasoning behind this decision was because complete linkage works well with data that is very clustered, which in this case it is.

We used the Training data for this analysis and ran three tests. For the first test we ran it without the Humidity Ratio variable in our dataset. We did this because we figured that the inclusion of this variable as well as Humidity would be unnecessary as the formula for Humidity Ratio included Humidity.

Figure 5.1

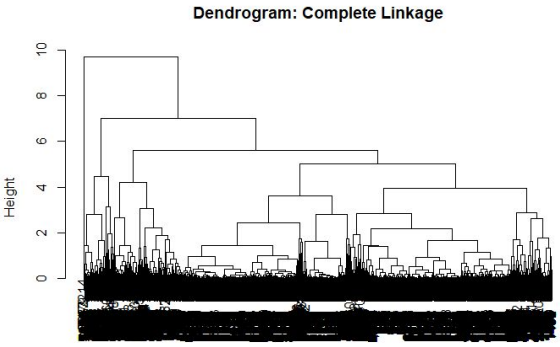

Figure 5.2

Figure 5.1 (cluster plot) and 5.2 (dendrogram) are for the test without the 'Humidity Ratio'. Clearly based on the cluster plot, the red is larger than the blue. As mentioned previously, we took this to mean that red = unoccupied and blue = occupied. Notice the green dots, these we believe were outliers/errors or points that were just caused by using the complete linkage method. With this test though we received an error term (including the green as error) of 0.1862, resulting in an accuracy of **0.8138**. This shows that the cluster test was not effective in predicting the data. As a result we decided to try a test that included the 'Humidity Ratio' to see if the results would change. The below diagrams are represented by this new test.

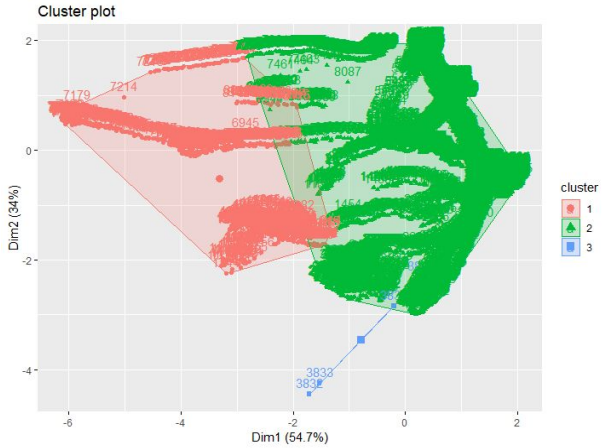

Figure 5.3

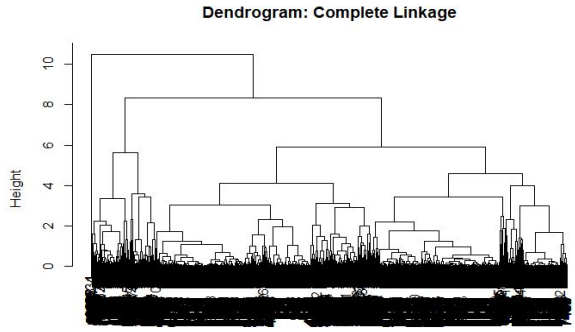

Figure 5.4

As you can see from this new cluster plot, there seems to be a lot more red (occupied) then in the previous test. Again we also got outliers/errors (blue). Our new error rate was 0.1070, and the accuracy **0.8930**. This value is significantly better than the previous test. Which means that the humidity ratio is a very important predictor. However while the error rate did decrease significantly, it still is very high.

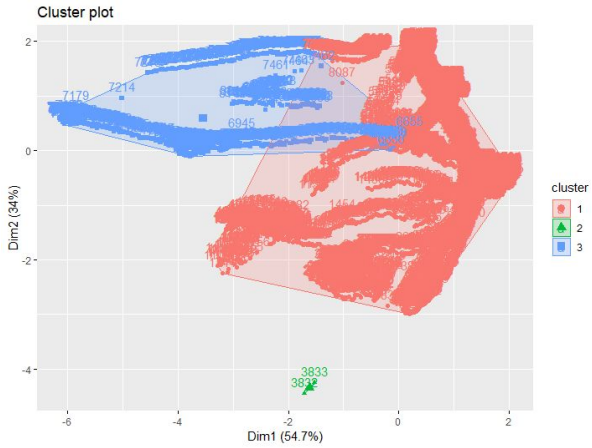

Figure 5.5

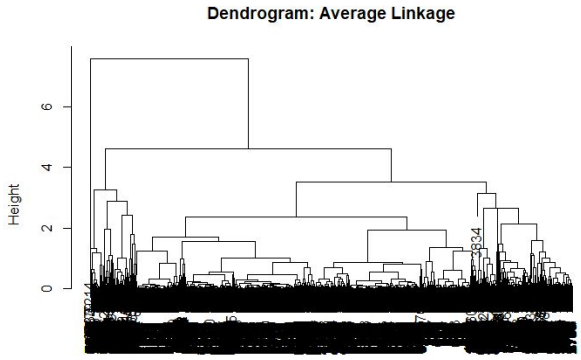

Figure 5.6

For our last test we decided to change the Linkage method to see if using the complete method was justified (results shown in figures 5.5 and 5.6. Upon first inspection, the results do not seem too different from the previous tests. In this case the red represents unoccupied, blue is occupied, and green is some sort of error. However the error rate changed to 0.1630, or an accuracy of **0.8370**. Based on this result we can say that changing the linkage method from complete to average was not beneficial and was not a better predictor.

It seems that although we were able to optimize the tests by switching up the various parameters, the error rate still weren't good enough. In all three of the tests the error rates were above 0.10. Comparing these results with those from the previous tests, we can say that hierarchical clustering was not the best method of analysis.

6. Conclusion

Comparing the accuracy scores of the various statistical techniques we implemented, we see that we got the highest score, which is 0.9924, by scaling the data and using a linear SVM model to classify it. We got similar high scores with running the linear SVM model on the first three principal components of the training data (0.9924), and using an LDA model on the CO2, Humidity, Light, and Temperature columns (0.9913). We believe that our result accuracy achieves satisfactory level and further improvement is unnecessary.

In all of those models, we chose not to use the date column because we did not think that would correlate to the occupancy of the room as much as the other features, although adding it back in might grant us better results. We believe that the reason the SVM model outperforms the other techniques is that we use it to classify two groups, which the method excels at. Furthermore, it is a very flexible classifier that assumes very little about the distribution of the data.

7. References

[1] Candanedo Ibarra, Luis & Feldheim, Veronique. (2015). Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. Energy and Buildings. 112. 10.1016/j.enbuild.2015.11.071.

CODE

Code for pairwise scatter plot:

```
png(file = "pairwise_scatter.png", width = 1080, height = 1080)
cols <- character(nrow(Test))
cols[] <- "black"

cols[Test$Occupancy == 1] <- "#00AFBB"
cols[Test$Occupancy == 0] <- "#E7B800"
pairs(iris,col=cols)

pairs(~Temperature+Humidity+Light+CO2+HumidityRatio,
      data=Test, pch = 19, cex = 0.1,
      col = cols,
      main="Simple Scatterplot Matrix")
dev.off()
```

Code for time plots:

```
Test_scaled$date <- as.POSIXct(Test_scaled$date, format="%Y-%m-%d %H:%M:%S")

df <- Test_scaled %>%
  select(date, Temperature, Occupancy) %>% # change temperature to other column names
  gather(key = "variable", value = "value", -date)
png(file = "timeplot_temperature.png", width = 2400, height = 600)
cols <- c("Temperature" = "#00AFBB", "Occupancy" = "#E7B800")
ggplot(df, aes(x = date, y = value)) +
  geom_line(aes(color = variable), size = 0.8) +
  scale_color_manual(values = cols) +
  theme_minimal()
dev.off()
```

Code for PCA:

```
Train = Training[,c(2,3,4,5,6,7)]
Test.data = Test[,c(2,3,4,5,6,7)]
Train$Occupancy <- as.factor(Train$Occupancy)
Test.data$Occupancy = factor(Test.data$Occupancy)
Train.scaled = scale(Train[,-6])
Test.scaled = scale(Test.data[,-6])
pca_result = prcomp(Train[,-6], center = TRUE,scale = TRUE)
summary(pca_result)
pca_reduce = prcomp(Train[,-6], center = TRUE,scale = TRUE,rank = 2)
#plotting the PCA plot of the first two PCs
library(ggfortify)
autoplot(pca_reduce,data = Train, cex = 0.01, colour = 'Occupancy',legendLabs =
```

```
c('Empty','Occupied'),title = 'PCA of the scaled training data')
```

Code for LDA(+plots):

```
Occupied <- Training[Training$Occupancy == '1',]
Unoccupied <- Training[Training$Occupancy == '0',]
n_occupied = nrow(Occupied)
n_unoccupied = nrow(Unoccupied)
p_occupied = n_occupied/(n_occupied+n_unoccupied)
p_unoccupied = n_unoccupied/(n_occupied+n_unoccupied)
Mean_Occupied=colMeans(Occupied[,2:6]) # change columns for different parameters
Mean_Unoccupied=colMeans(Unoccupied[,2:6])
S_occupied=cov(Occupied[,2:6])
S_unoccupied=cov(Unoccupied[,2:6])
S_pooled=((n_occupied-1)*S_occupied+(n_unoccupied-1)*S_unoccupied)/(n_occupied+n_unoccupied-2)
S_inv=solve(S_pooled)
alpha_occupied= -0.5*t(Mean_Occupied) %*% S_inv %*% Mean_Occupied+log(p_occupied)
alpha_unoccupied= -0.5*t(Mean_Unoccupied) %*% S_inv %*% Mean_Unoccupied+log(p_unoccupied)
beta_occupied=S_inv %*% Mean_Occupied
beta_unoccupied=S_inv %*% Mean_Unoccupied
prediction=c()
d_occupied_vec=c()
d_unoccupied_vec=c()
label=c("1", "0")
for(i in 1:nrow(Test)){
x=t(Test[i,2:6])
d_occupied=alpha_occupied+ t(beta_occupied) %*% x
d_unoccupied=alpha_unoccupied+ t(beta_unoccupied) %*% x
d_vec=c(d_occupied, d_unoccupied)
prediction=append(prediction, label[which.max( d_vec )])
d_occupied_vec=append(d_occupied_vec, d_occupied)
d_unoccupied_vec=append(d_unoccupied_vec, d_unoccupied)
}
Test$prediction=prediction
sum(Test$Occupancy != Test$prediction)
occ_predict <- Test[Test$prediction == '1',]
unocc_predict <- Test[Test$prediction == '0',]

col_vec=c(rep("red", nrow(occ_predict)), rep("blue", nrow(unocc_predict)))
pch_vec=c(rep(15, nrow(occ_predict)), rep(17, nrow(unocc_predict)))
png(file = "Scatter.png", width=640, height=480)
plot(x = d_occupied_vec, y= d_unoccupied_vec, xlab = "Occupied", ylab = "Unoccupied", col= col_vec,
pch = pch_vec, cex = 0.1)
dev.off()
```

Code for randomForest:

generate 100 RFs with m value = 2, plot error in a histogram and take the average of the error rates.

*** note:

```
errors2 = numeric (50)
for (i in 1:100){
  RF =randomForest(Occupancy~.,data=Training.data, mtry=2,
                             importance =TRUE,xtest = Test.data[,-6], ytest= Test.data$Occupancy,
keep.forest=TRUE,ntree=500)
  errors2[i]= RF[["test"]][["err.rate"]][500,1]
}
hist(errors2)
mean(errors2)

RF # get details
plot(RF) # get plot
```

Code for SVM:

```
install.packages('e1071')
library(e1071)
scaled_lin_classifier = svm(Train$Occupancy ~ .,data = Train.scaled,
              type = 'C-classification',kernel = 'linear')
scaled_lin_pred_test <-predict(scaled_lin_classifier,Test.scaled)
scaled_lin_score = mean(scaled_lin_pred_test==Test.data$Occupancy)
raw_lin_classifier = svm(Train$Occupancy ~ .,data = Train,
                         type = 'C-classification',kernel = 'linear')
raw_lin_pred_test <-predict(raw_lin_classifier,Test.data)
raw_lin_score = mean(raw_lin_pred_test==Test.data$Occupancy)
#SVM using PCs
test_pca = predict(pca_reduce,newdata = Test.data)
pca_lin_classifier = svm(Train$Occupancy ~ .,data = pca_reduce$x,
                       type = 'C-classification',kernel = 'linear')
pca_lin_pred_test <-predict(pca_lin_classifier,test_pca)
pca_lin_score = mean(pca_lin_pred_test==Test.data$Occupancy)

#plotting the pca svm
pca_reduce = prcomp(Train[,-6], center = TRUE,scale = TRUE,rank = 2)
pca_classifier = svm(Train$Occupancy ~ .,data = pca_reduce$x,type = 'C-classification',kernel = 'linear')
plot(pca_classifier,data.frame(pca_reduce$x),PC2 ~ PC1,fill = TRUE,grid = 300,svSymbol = '.',
dataSymbol = '.',color.palette = terrain.colors)
```

Clustering -- Distance: Euclidean; Linkage: Complete; w/o Humidity Ratio:

```
library(cluster)
library(factoextra)
library(dendextend)
```

```
Training$date=NULL
Training$Occupancy=NULL
scale(Training)->Training
dist=dist(Training, method="euclidean")
hc.complete = hclust (dist, method ="complete")
sub_grp = cutree(hc.complete,k= 3)
plot(hc.complete , main=" Dendrogram: Complete Linkage ", xlab="", sub ="")
fviz_cluster(list(data = Training, cluster = sub_grp))
data.frame(sub_grp)->sub_grp
Training$prediction = sub_grp
Training$prediction[Training$prediction == "1"] <- "0"
Training$prediction[Training$prediction == "3"] <- "1"
nrow(Training[Training$prediction == '1',])
[1] 527
nrow(Training[Training$prediction == '0',])
[1] 7614
nrow(Training[Training$prediction == '2',])
[1] 2
sum(Training$Occupancy != Training$prediction)
[1] 1516
1516/8143
[1] 0.1861722 = ERROR
```

Clustering -- Distance: Euclidean; Linkage: Complete; w Humidity Ratio

```
library(cluster)
library(factoextra)
library(dendextend)
Training$date=NULL
Training$Occupancy=NULL
scale(Training)->Training
dist=dist(Training, method="euclidean")
hc.complete = hclust (dist, method ="complete")
sub_grp = cutree(hc.complete,k= 3)
plot(hc.complete , main=" Dendrogram: Complete Linkage ", xlab="", sub ="")
fviz_cluster(list(data = Training, cluster = sub_grp))
data.frame(sub_grp)->sub_grp
Training$prediction = sub_grp
Training$prediction[Training$prediction == "2"] <- "0"
nrow(Training[Training$prediction == '1',])
[1] 1098
nrow(Training[Training$prediction == '0',])
[1] 7041
nrow(Training[Training$prediction == '3',])
[1] 4
sum(Training$Occupancy != Training$predict)
[1] 871
871/8143
```

[1] 0.106963 = ERROR

Clustering -- Distance: Euclidean; Linkage: Average

```
library(cluster)
library(factoextra)
library(dendextend)
Training$date=NULL
Training$Occupancy=NULL
scale(Training)->Training
dist=dist(Training, method="euclidean")
hc.complete = hclust (dist, method ="average")
sub_grp = cutree(hc.average,k= 3)
plot(hc.complete , main=" Dendrogram: Average Linkage ", xlab="", sub ="")
fviz_cluster(list(data = Training, cluster = sub_grp))
data.frame(sub_grp)->sub_grp
Training$prediction = sub_grp
Training$prediction[Training$prediction == "1"] <- "0"
Training$prediction[Training$prediction == "3"] <- "1"
nrow(Training[Training$prediction == '1',])
[1] 772
nrow(Training[Training$prediction == '0',])
[1] 7369
nrow(Training[Training$prediction == '2',])
[1] 2
sum(Training$Occupancy != Training$prediction)
[1] 1327
1327/8143
[1] 0.1629621 = ERROR RATE
```