University of Cape Town

Department of Statistical Sciences

STA5076Z Supervised Learning

Assignment 2

EVRCHA001 - Charl Everts

29 June 2020

Department of Statistical Sciences

Plagiarism Declaration Form

A copy of this form completed and signed, to be attached to all coursework submissions to the Statistical Sciences Department. Submissions without this form will not be marked.

COURSE CODE: STA5076Z

COURSE NAME: Supervised Learning

STUDENT NAME: Charl Everts

STUDENT NUM: EVRCHA001

Plagiarism Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used a generally accepted citation and referencing style. Each contribution to, and quotation in, this tutorial/report/project from the work(s) of other people has been attributed and has been cited and referenced.
3. This tutorial/report/project is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.
5. I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.

Note that agreement to this statement does not exonerate you from the University's plagiarism rules (http://www.uct.ac.za/uct/policies/plagiarism_students.pdf).

Signature:  Date: 29 June 2020

ABSTRACT

The purpose of this report is to evaluate the occupancy dataset as collected by Candenado and Feldheim (2016). Logistic Regression, Classification Trees, Bagging/Random Forest and Gradient Boosted methods were used to model the binary classification problem relating to the occupancy dataset. The models were trained on an 80/20% train and test split from the original training dataset.

The predictor variable “Date” was not included in the model building to as to ensure the generalizability of the models. It was realised that Light, Temperature and CO2 were the most important predictors throughout the report. The most accurate model in each section was evaluated using the unseen test set and subsequently ranked 1-4 based on classification accuracy. The most accurate model was Model7: a logistic regression model with Lasso penalty using only Light, Temperature and CO2 as predictors – classification accuracy: 96.7355%.

Table of Contents

1. Introduction 6
1.1 The “Occupancy” Dataset 6
1.2 Data Quality Assessment 6
1.3 Descriptive Statistics 7
1.4 The “Date” Variable 11
1.5 Assignment Objectives 11
2. Logistic Regression 12
2.1 Methodology 12
2.2 Results 12
3. Classification Trees 14
3.1 Methodology 14
3.2 Results 14
4. Bagging and Random Forests 16
4.1 Methodology 16
4.2 Results 16
5. Gradient Boosting 19
5.1 Methodology 19
5.2 Results 19
6. Best Model Evaluation on Unseen Test Data 21
6.1 Logistic Regression (Model7): 21
6.2 Classification Trees (Model9): 21
6.3 Bagging and Random Forests (Model11): 21
6.4 Gradient Boosting (Model13): 21
7. Conclusion 22
8. References 44

List of Figures

Figure 1.1: *Generalized pairs plot.*
Figure 1.2: *Frequency distribution of Light, coloured by Occupancy.*
Figure 1.3: *Frequency distribution of Temperature, coloured by Occupancy.*
Figure 1.4: *Correlation cluster plot.*
Figure 1.5: *Correlation heatmap.*
Figure 1.6: *Time-series plots of Temperature, Humidity, HumidityRatio and CO2.*
Figure 1.7: *Time-series plots of Light and Occupancy.*
Figure 3.1: *CV errors vs number of terminal nodes used.*
Figure 3.2: *Classification tree with 8 nodes.*
Figure 3.3: *Classification tree with 4 nodes.*
Figure 4.1: *Bagged model variable importance.*
Figure 4.2: *Model10 variable importance.*
Figure 4.3: *Model comparison based on OOB error.*
Figure 5.1: *Variable importance plot.*
Figure 5.2: *Partial dependence plots for Temperature, Light and CO2.*

List of Tables

Table 1.1: *Correlation coefficients.*
Table 2.1: *Summary coefficients of* model1.
Table 2.2: *Choice of predictor variables.*
Table 2.3: *Models ranked on Classification Accuracy.*
Table 2.4: *Models ranked on AUC.*
Table 2.5: *Model7 confusion matrix.*
Table 2.6: *Model7 coefficients.*
Table 3.1: *Model8 confusion matrix.*
Table 3.2: *Model9 confusion matrix.*
Table 4.1: *Model10&11 confusion matrices.*
Table 5.1: *Model12&13 confusion matrices.*
Table 6.1: *Classification accuracy and AUC for Model7 on unseen data.*
Table 6.2: *Classification accuracy for Model9 on unseen data.*
Table 6.3: *Classification accuracy for Model11 on unseen data.*
Table 6.4: *Classification accuracy for Model13 on unseen data.*

Abbreviations/Symbols

HVAC	heating, ventilation and air conditioning
°C	degrees Centigrade
Lux	SI unit of illuminance
ppm	parts per million
MSE	mean squared error
AUC	area under curve
ROC	receiver operating characteristics
CV	cross validation
OOB	out-of-bag error
B	number of trees
λ	shrinkage parameter
d	interaction depth

Chapter 1

Introduction

1.1 The “Occupancy” Dataset

The purpose of constructing the “Occupancy” dataset is to build models that accurately predict the occupancy of a room based on sensor readings. Modelling the occupancy of a room could in turn provide valuable information to HVAC (heating, ventilation and air conditioning) controls and lighting systems in buildings. This data allows these systems to implement appropriate control measures, increase efficiency and hence save on energy costs.

In their study, Candenado and Feldheim (2016, p.28) used data recorded from light, temperature, humidity and CO2 sensors. An advantage of using sensors instead of a camera to verify the occupancy of a room has substantial privacy-preserving benefits.

The predictor features in the dataset are:

i. Temperature (°C)
ii. Humidity (measured in relative humidity as %)
iii. Light (Lux)
iv. CO_2 (ppm)
v. Humidity Ratio (derived from temperature and relative humidity, in kgwater-vapor/kg-air)
vi. Date (year/month/day with a timestamp every minute)

The response variable is Occupancy, which indicates whether or not a room is occupied or not. Occupancy is a categorical variable with two levels: 0 – indicating an empty room, and 1 – indicating that a room is occupied.

1.2 Data Quality Assessment

The data used in this report is an amalgamation of the observations recorded by Candenado and Feldheim (2016), split into training and test sets with 6751 and 2665 observations respectively.

The test set is treated as completely unseen data and hence not used in any model building. The original training set undergoes an 80/20% split between train/test respectively, upon which models are built and predictive accuracy is measured. The final models are then evaluated using the original test set containing 2665 observations

The data recorded in the training set ranges from 2015/02/05 (Thursday) 06:20 - 2015/02/09 (Monday) 22:50. The data recorded in the testing set ranges from 2015/02/02 (Monday)14:19 - 2015/02/04 (Wednesday) 10:43.

There are no missing or empty values in the datasets. This is aided by the fact that all sensors are automated, eliminating the risk of erroneous data due to manual entries.

1.3 Descriptive Statistics

Figure 1.1 depicts the correlations amongst the predictor variables in the dataset. There exists strong positive correlations between Light and Temperature, HumidityRatio and CO2, Light and CO2, and Temperature and CO2. It is clear that Humidity and HumidityRatio have near perfect correlations, this is however due to the manner in which HumidityRatio is derived. Not much is known about the HumidityRatio value, other than the fact that it is a ratio with Humidity either in the numerator or denominator of the expression, hence the high correlation.

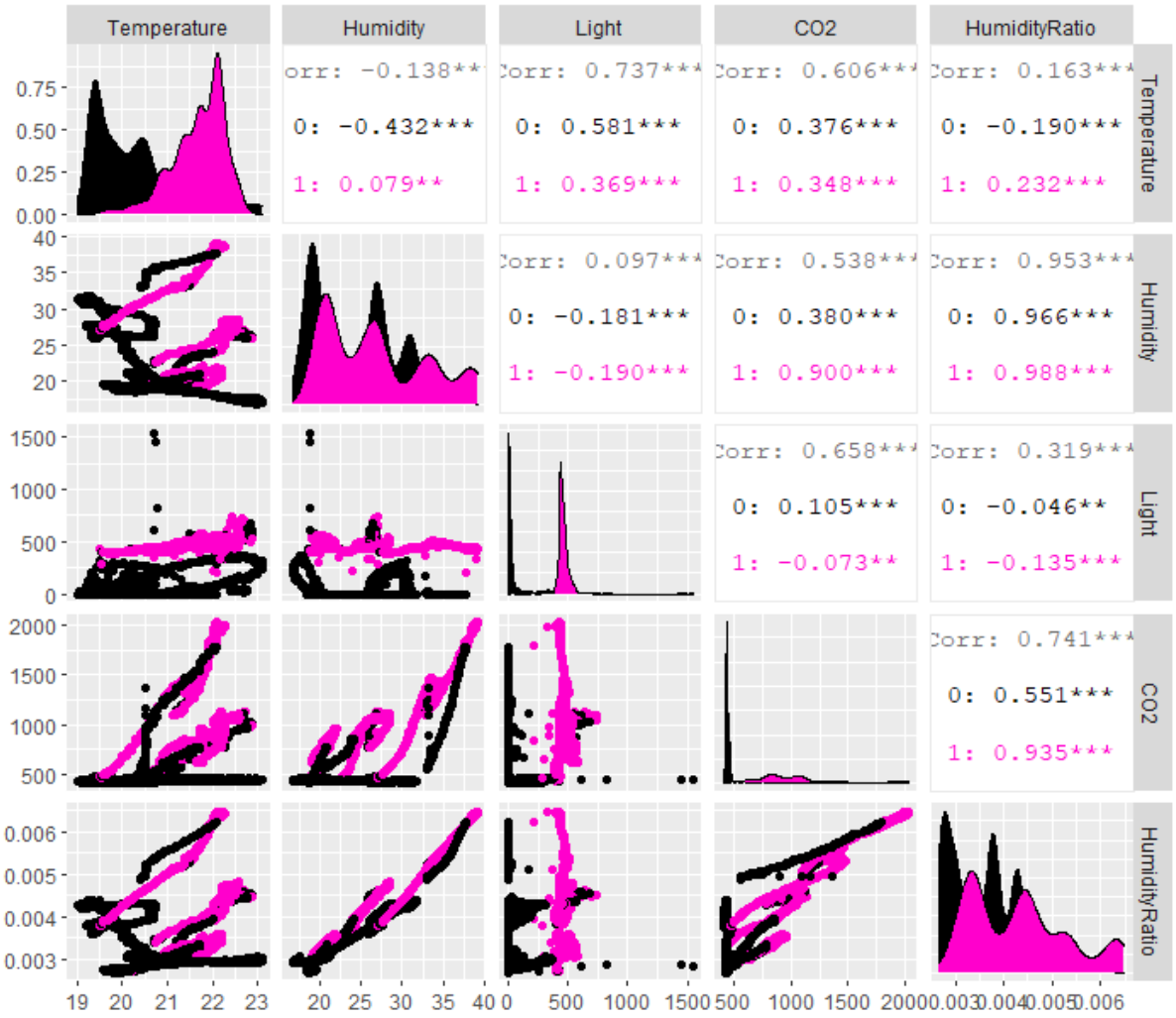

Figure 1.1: *Generalized pairs plot.*

Table 1.1 tabulates the correlation coefficients in a more legible fashion.

	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
Temperature	1	-0.14	0.74	0.61	0.16	0.6
Humidity	-0.14	1	0.1	0.54	0.95	0.19
Light	0.74	0.1	1	0.66	0.32	0.9
CO2	0.61	0.54	0.66	1	0.74	0.71
HumidityRatio	0.16	0.95	0.32	0.74	1	0.38
Occupancy	**0.6**	**0.19**	**0.9**	**0.71**	**0.38**	**1**

Table 1.1: *Correlation coefficients.*

Figures 1.2-3 are alternative ways in which to graphically represent the correlations in Table 1.1, by means of a cluster plot and heatmap.

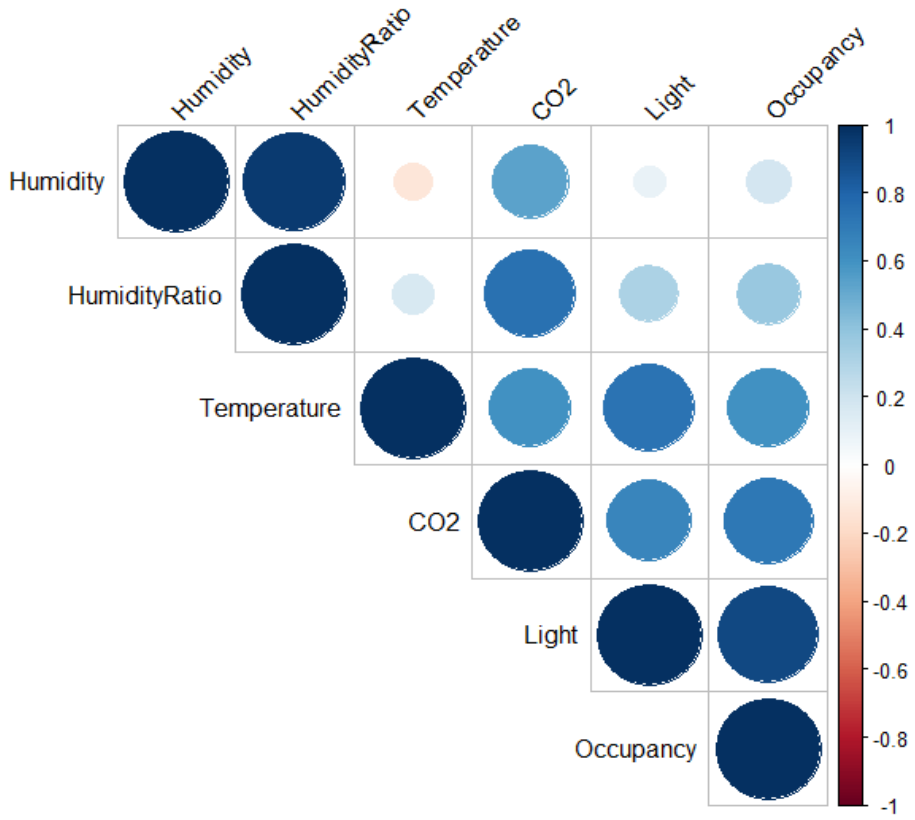

Figure 1.2: *Correlation cluster plot.*

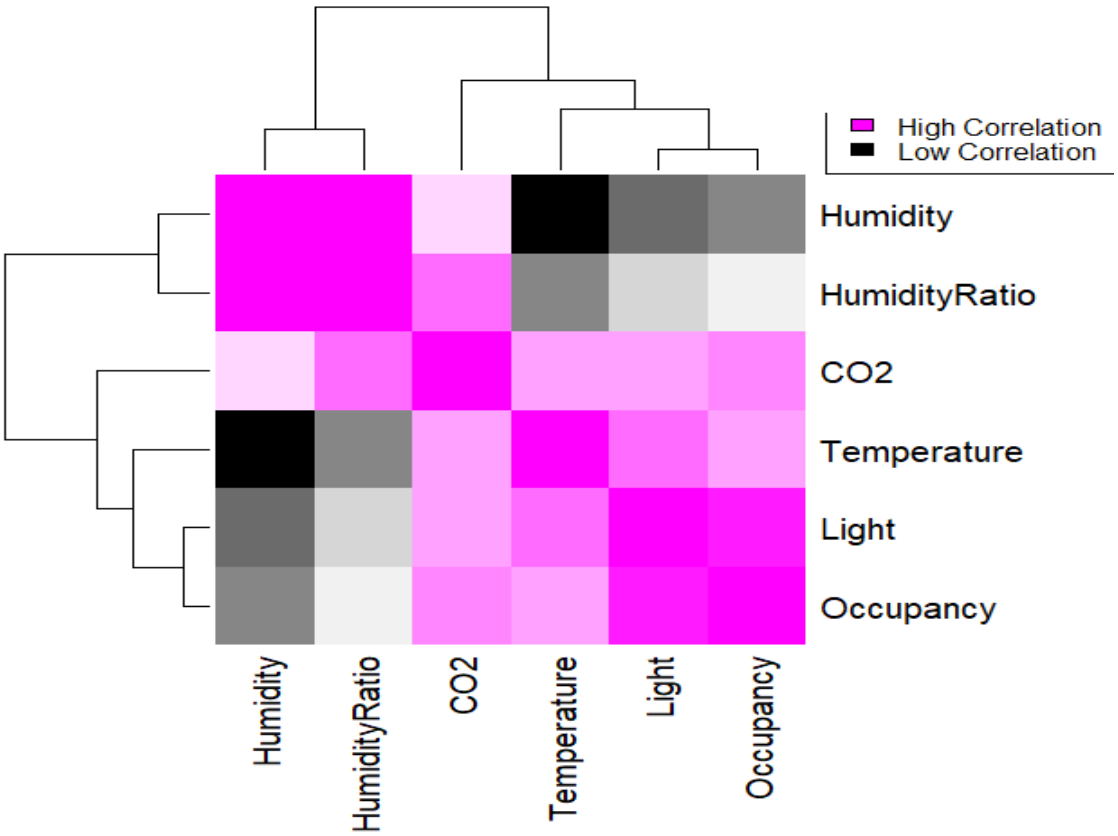

Figure 1.3: *Correlation heatmap.*

Figures 1.4-5 offer an interesting insight into the distribution of Light and Temperature, coloured by the different states of Occupancy. In Figure 1.4 it is almost immediately obvious that for a light reading of below 400 Lux, the probability of the room being occupied is extremely low. The converse is true for values above 400 Lux. This indicates that occupants were quite disciplined with respect to turning the light off upon exiting the room.

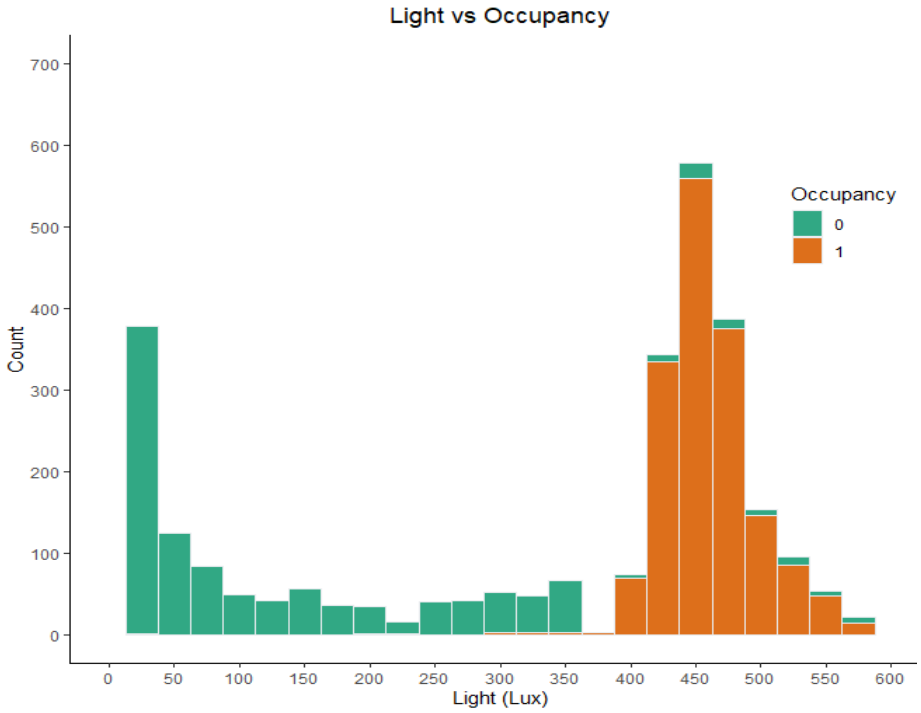

Figure 1.4: *Frequency distribution of Light, coloured by Occupancy.*

Figure 1.5 depicts the distribution of temperatures and the ideal range when the room is occupied. Probability of a room being occupied is high when the office temperature is in the 21.0 – 22.5°C range. According to Candenado and Feldheim (2016, p.28), the room was heated by hot water radiators and turned off in the evenings when occupants left. This explains the high probability of an unoccupied room at low temperatures.

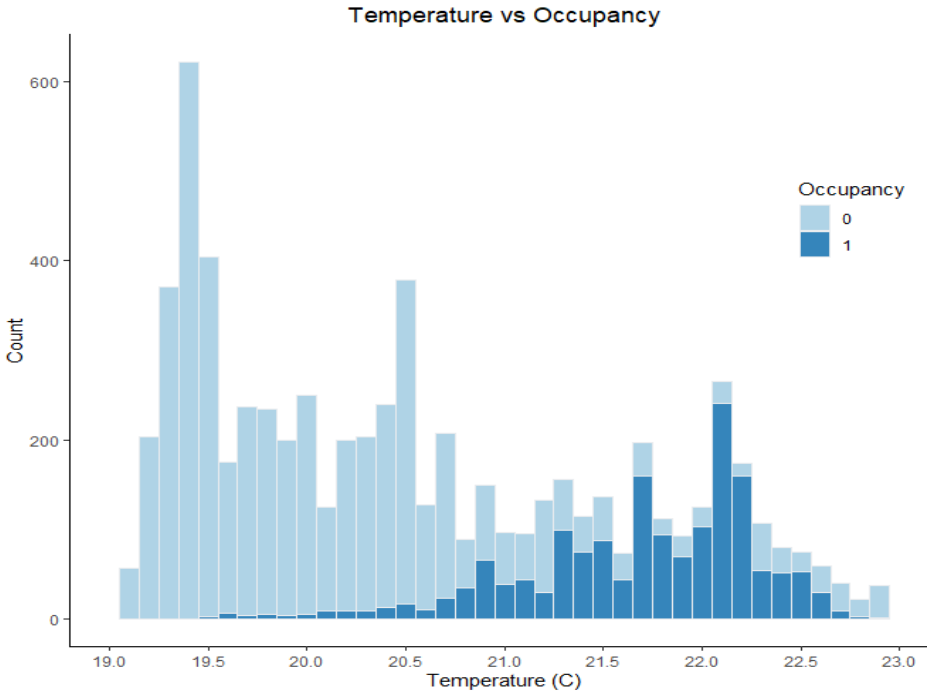

Figure 1.5: *Frequency distribution of Temperature, coloured by Occupancy.*

The time series plots in Figure 1.6 allows one to evaluate the change in predictor variables over a standardized time frame. In this case, the information displayed is for the time period 2015-02-03 (Tuesday) 7:00 to 2015-02-04 (Wednesday) 8:00. Interestingly, the shape of the Humidity curve almost exactly replicates the CO2 curve. Temperature also follows this trend, although the curve is slightly flatter.

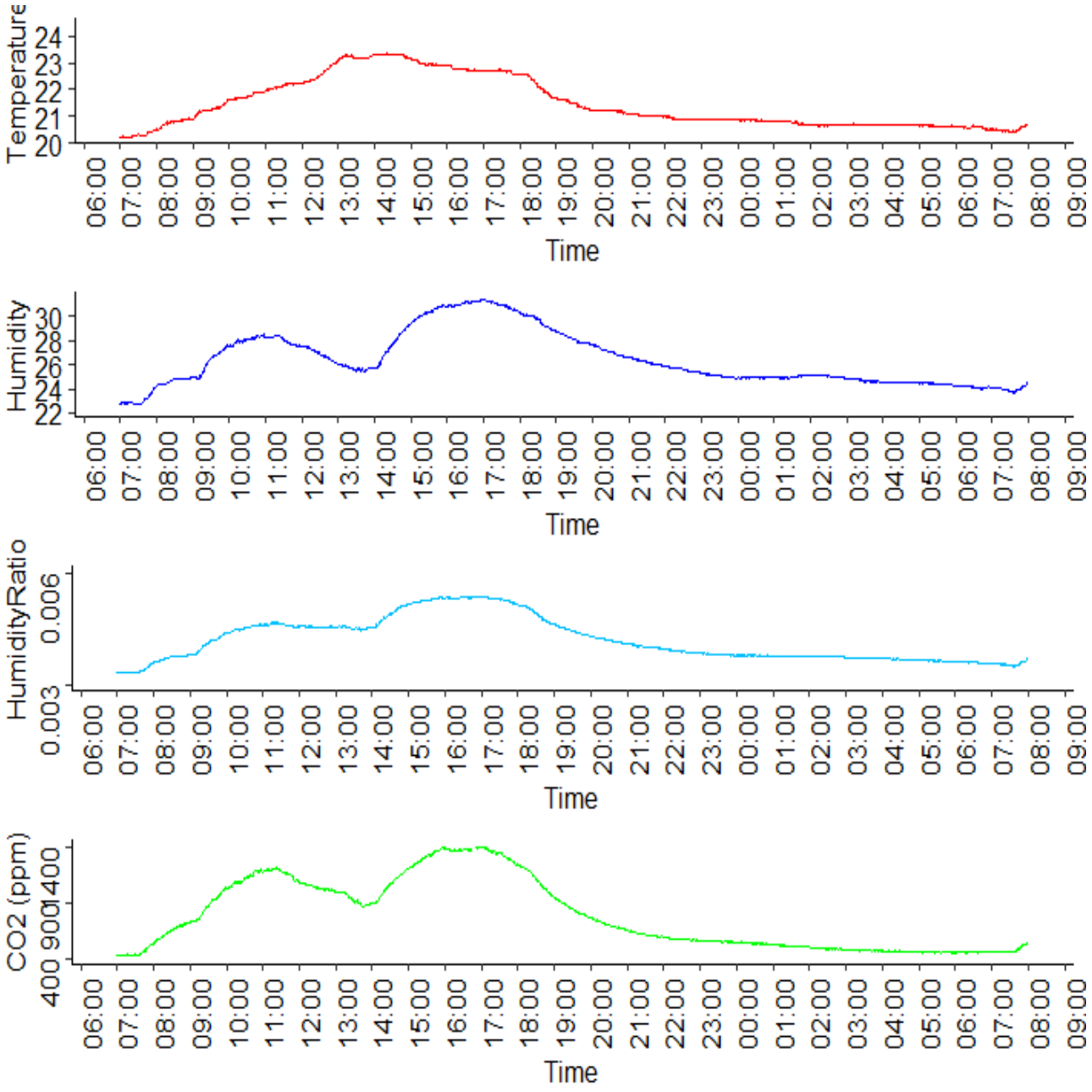

Figure 1.6: *Time-series plots of Temperature, Humidity, HumidityRatio and CO2.*

Figure 1.7 depicts the clear relationship between Light and Occupancy. This relationship may lead one to believe that there is not any natural light that enters the room since the lights are always on when the room is occupied. If sufficient natural light entered the room this would not be necessary. The short periods of the room being unoccupied could relate to situations where individuals briefly leave the room to visit a colleague or during brief toilet breaks. One could further draw the conclusion that the work day ranges from 07:30 – 18:00, with a 30-minute lunchbreak between 13:00 and 14:00.

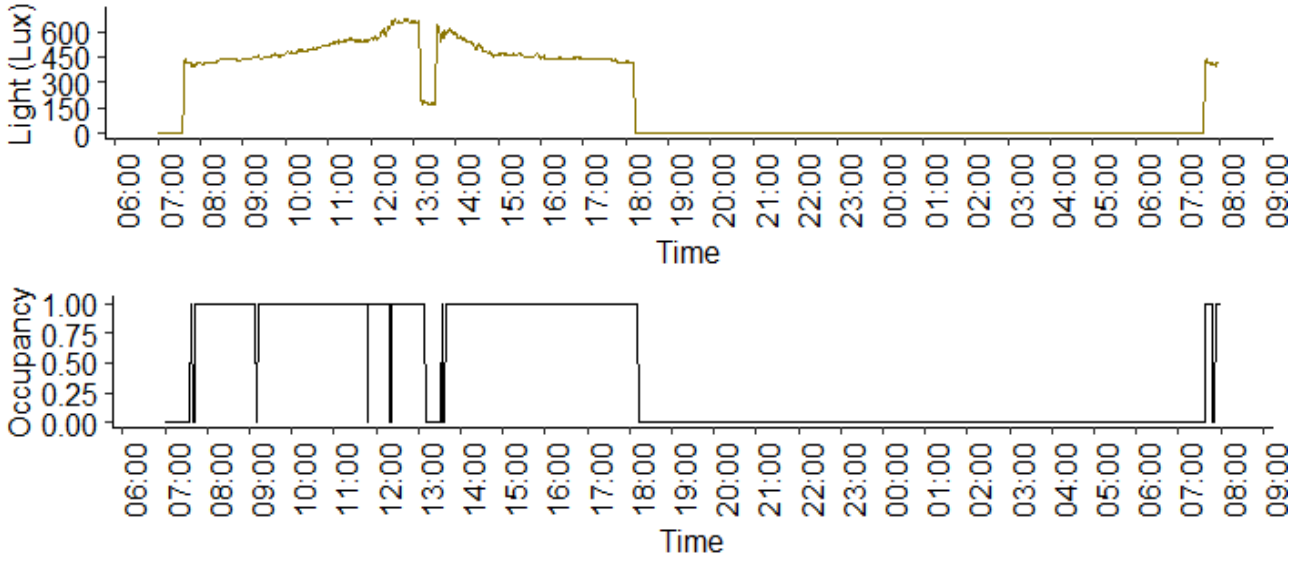

Figure 1.7: *Time-series plots of Occupancy and Light.*

1.4 The “Date” Variable

The date variable was omitted as a predictor variable in this report. This was not without extensive thought. Using the date variable and its timestamp to model a room’s occupancy will only be accurate for models predicting the occupancy of other offices who follow the same office hours. The purpose of the study by Candenado and Feldheim was to predict occupancy in order to make HVAC controls more efficient in general. By modelling the data using the timestamp, the model will be aimed at offices in particular, and not generalised communal spaces where HVAC controls are implemented.

Therefore, omitting the timestamp would allow the model to accurately predict occupancy for all rooms or office spaces that potentially have abnormal office hours, not just offices that implement the conventional 9-5 work day. Examples may include hospital communal areas, student areas in a 24-hour library, or a breakroom in a fire station.

1.5 Assignment Objectives

The objective of this assignment is to analyse the dataset created by Candenado and Feldheim (2016). Using the training and testing datasets provided, the following classification techniques are used to model the occupancy status of an office room:

i. Logistic Regression
ii. Classification Trees
iii. Bagging/Random Forest
iv. Gradient Boosted Trees

Build and analyse these models by commenting on their respective outputs and predictive accuracy.

Chapter 2

Logistic Regression

2.1 Methodology

A binomial logistic regression model is used to model a dichotomous outcome variable, in this case Occupancy, which can take a value of 0 or 1. The log odds of the outcome is modeled as a linear combination of the predictor variables. f(X) defines the logistic map representing the probability of an event which depends on p predictor variables:

$$f(X) = \frac{e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}$$

The logistic function ensures that the predictor variables produce outputs as probabilities between [0,1], hence appropriate for binary classification.

2.2 Results

The first five models were implemented as standard logistic regression models without a lasso penalty. The summary coefficients of Model1 (Table 2.1) indicate that temperature is of no significance to the model since Pr(>|z|) > 0.05.

	Estimate	Std. Error	z value	Pr(>|z|)	Significance
(Intercept)	-4.007e+01	1.972e+01	-2.032	0.0421	*
Temperature	1.468e+00	9.201e-01	1.595	0.1107	
Humidity	2.501e+00	6.190e-01	4.041	5.33e-05	***
Light	2.179e-02	1.126e-03	19.351	< 2e-16	***
CO2	1.347e-02	1.071e-03	12.576	< 2e-16	***
HumidityRatio	-1.822e+04	4.030e+03	-4.522	6.14e-06	***

Table 2.1: *Summary coefficients of Model1.*

The unused predictor variables in the subsequent models (Table 2.2) were excluded based on the significance values in Table 2.1.

Model	Predictors Excluded
Model1	-
Model2	Temperature
Model3	Temperature | HumidityRatio
Model4	Temperature | HumidityRatio | Humidity
Model5	HumidityRatio | Humidity

Table 2.2: *Predictor variables excluded.*

In Model6, all predictor variables were used in a logistic regression with lasso penalty. In Model7, the same procedure was followed, while only the Light, Temperature and CO2 predictors were used. 10-fold cross validation was conducted to find the lambda value corresponding to the lowest MSE. In model6, the min MSE corresponded to a model using all predictor variables except Humidity.

Once the seven models were established, predictions were made using the 20% split test data (`pi_hat_n`). A decision threshold of 0.5 was implemented on the respective predictions. If `pi_hat_n` >= 0.5, a classification of “1” – occupied is assigned, otherwise “0” – unoccupied (`Y_hat_n`).

Misclassification rates are calculated as the proportion of the incorrectly classified observations (predictions) out of the entire test dataset (20% split) observations. Classification accuracy (Table 2.3) is the found by: (1 - Misclassification rate)*100%.

Rank	Model	Classification Accuracy (%)
1	Model7	98.7398
2	Model5	98.6657
3	Model6	98.6657
4	Model1	98.5915
5	Model2	98.5915
6	Model3	98.5174
7	Model4	98.2950

Table 2.3: *Models ranked on Classification Accuracy.*

An alternative measure of predictive accuracy is by making use of ROC curves and their accompanying AUC values for each model (Table 2.4). ROC curves are used to illustrate the sensitivity/specificity trade-off in a binary classification problem.

Rank	Model	AUC*100
1	Model1	99.48065
2	Model2	99.47115
3	Model6	99.42396
5	Model5	99.30763
4	Model7	99.30169
6	Model4	99.18833
7	Model3	99.18536

Table 2.4: *Models ranked on AUC.*

The AUC criteria evaluates the predictive accuracy of a classification algorithm. However, the objective function optimized in most classification algorithms is the error rate and not the AUC value. Thus, algorithms designed to minimize the error rate may not lead to the best possible AUC values (Cortes & Mohri, 2004). Classification accuracy is therefore the defining metric used to rank the predictive accuracy of Models1-7.

Model7 is therefore the best model: a logistic regression with lasso penalty using only Temperature, Light and CO2 as predictors. The confusion matrix (Table 2.5) expresses correct classifications on the diagonal, and misclassifications on off-diagonal entries.

		True		**Accuracy_ Model7 (%)**
		0	1	
Predicted	0	1002	1	
	1	16	330	**98.7398**

Table 2.5: *Model7 confusion matrix.*

Upon evaluating the variable coefficients in Table 2.5, the following conclusions can be drawn:

1. An increase in temperature decreases the probability that a room is occupied.
2. An increase in light increases the probability that a room is occupied, and the same effect when increasing CO2 levels, albeit to a lesser degree.

Variable	Coefficient
(Intercept)	31.7469 (β_0)
Temperature (X_1)	-2.1372 (β_1)
Light (X_2)	0.0237 (β_2)
CO2 (X_3)	0.0075 (β_3)

Table 2.6: *Model7 coefficients.*

Chapter 3

Classification Trees

3.1 Methodology

A classification tree follows a hierarchical structure which yields a final outcome after traversing the nodes of the tree. As the tree is traversed, the level of impurity/uncertainty decreases. The manner in which the splits are decided depends on the splitting criteria. Two splitting criteria are evaluated, Gini and Deviance.

The Gini index is a measure of node impurity or variability within the nodes. It is therefore desirable to have each leaf as pure as possible, meaning each leaf node would include only the observations of a single response outcome. Each split is therefore chosen such that it produces the greatest reduction in the Gini index.

The Deviance is constructed by viewing a classification tree as a probability model. Growing a classification tree based on the Deviance splitting criteria aims to reduce the deviance at each split, which maximises the likelihood function.

Cross validation is applied to facilitate cost complexity pruning by reducing the number of terminal nodes, whilst minimising the CV error.

3.2 Results

The first model was split using the Gini splitting criteria. This resulted in a low misclassification rate but low interpretability since the tree had 32 nodes. Splitting on Deviance resulted in a slightly worse misclassification rate, but the tree had only 6 terminal nodes and therefore easily interpreted.

The deviance splitting criteria model was improved by utilizing the `mindev` argument which specifies the minimum within-node deviance for the node to be split on. Upon plotting CV error vs Terminal Nodes (Figure 3.1), it is evident that a tree with 8 terminal nodes (Model8) yields the lowest CV error. Figure 3.2 illustrates the resulting tree for Model8.

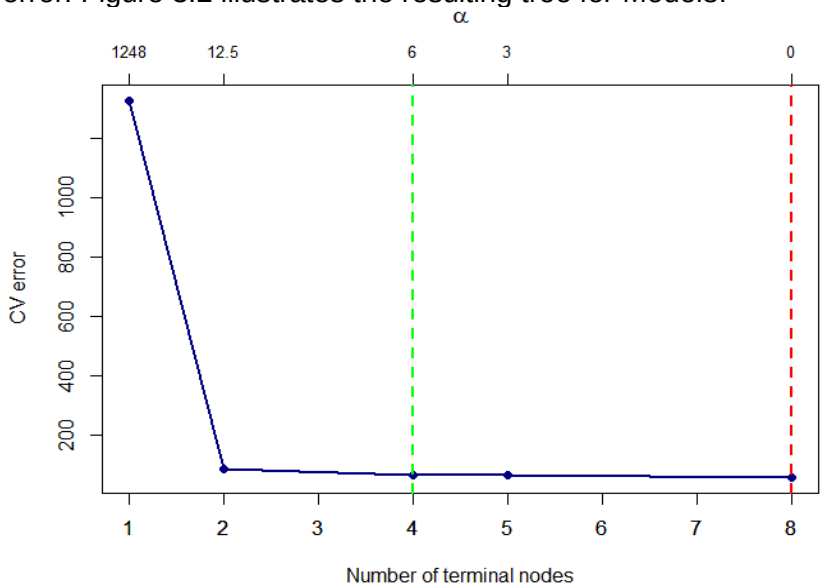

Figure 3.1: *CV errors vs number of terminal nodes used.*

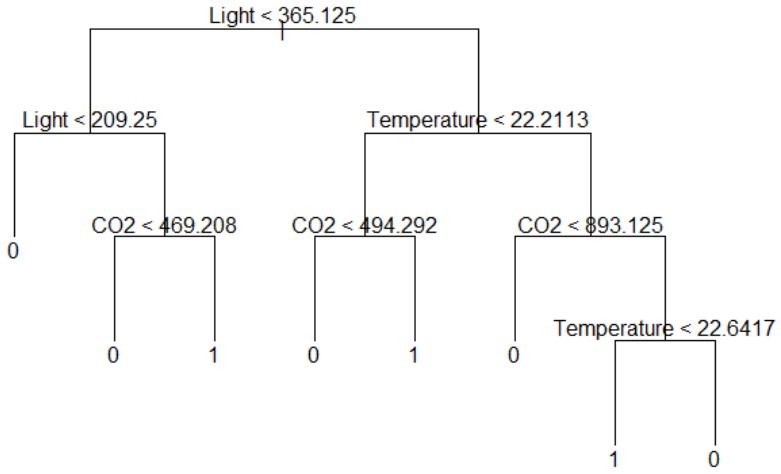

Figure 3.2: *Classification tree with 8 nodes (Model8).*

According to Figure 3.1, pruning the tree even further to 4 terminal nodes, results in an almost negligible increase in CV error. This trade off for model simplification seems justifiable, and therefore Model9 is created (Figure 3.3). The corresponding confusion matrices for Models8&9 are presented in Tables 3.1.

		True		**Accuracy_ Model8 (%)**
		0	1	
Predicted	0	1015	8	
	1	3	323	**99.1846**
		True		**Accuracy_ Model9 (%)**
		0	1	
Predicted	0	1009	1	
	1	9	330	**99.2587**

Table 3.2: *Model8&9 confusion matrix.*

The classification results are intriguing, as the simpler 4 node model marginally out performs the 8 node model on the test set. Model9 is therefore selected as the best classification tree. The initial and hence most important split occurs on the Light variable. It states that if the light reading in a room is less than 365.125 Lux, one can conclude that no one is present in the room. If the light reading is more than this, but Temperature is less than 22.2113°C, then you can conclude that the room is occupied, and so on.

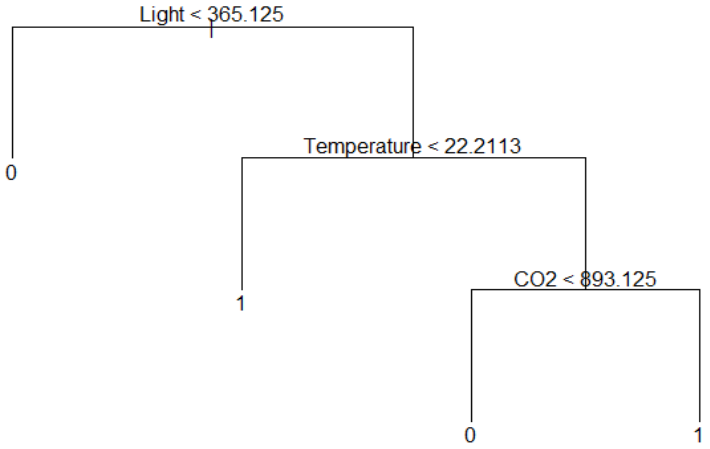

Figure 3.3: *Classification tree with 4 nodes (Model9).*

Chapter 4

Bagging and Random Forests

4.1 Methodology

Bagging (bootstrap aggregating) is an ensemble-based algorithm which aims to make decision trees more robust by reducing the variance of the model. Bagging trains unpruned decision trees on random subsets of the training data, hence reducing the variance and minimizing overfitting. All training data features are considered when splitting a node.

The Random Forest method is a modified version of Bagging. Instead of using all features split on, the Random Forest method forces the tree to use only a subset of available predictors. Subsequently, each decision tree in the Random Forest is different since each was trained on a different subset of data. This result in a more robust model as overfitting is minimized.

In Bagging and Random Forests, it is not required to conduct cross-validation to yield an estimate of the test error. Instead the out-of-bag (OOB) error is estimated internally during the execution of the algorithm. Each tree is constructed using a different sample from the training data, approximately 2/3^{rds}. Each tree can be tested on the samples not used in building that specific tree. The out of sample error is subsequently computed.

4.2 Results

Initially, running a bagged model using all predictors yielded a variable importance plot based on the mean decrease in Gini index as follows:

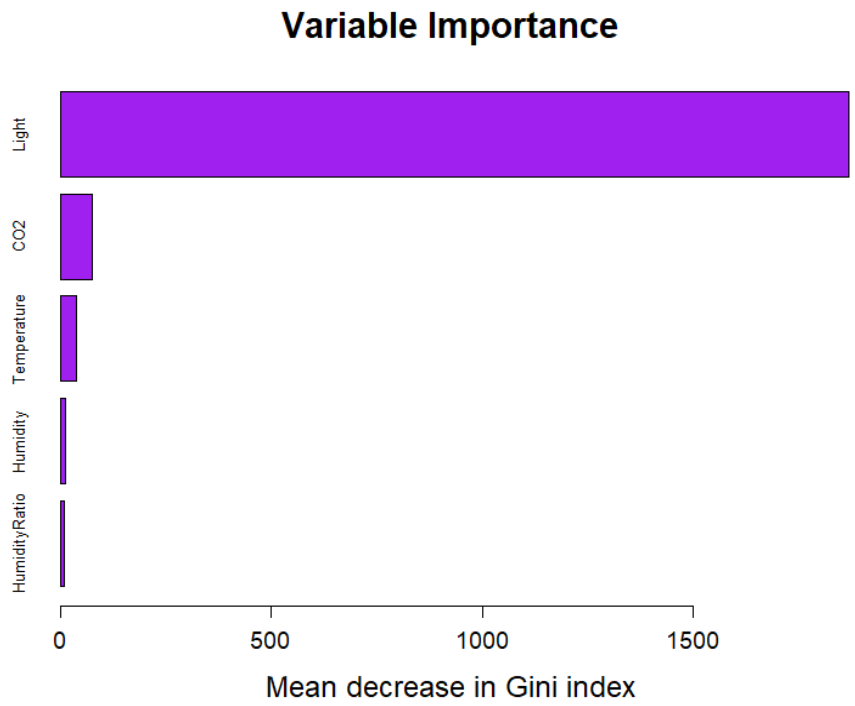

Figure 4.1: *Bagged model variable importance.*

It is evident that Light is by far the most prominent variable, with Humidity and HumidityRatio the least important of the predictors. Model10 was constructed as a 1000 tree bagged model using all predictors except Humidity and HumidityRatio.

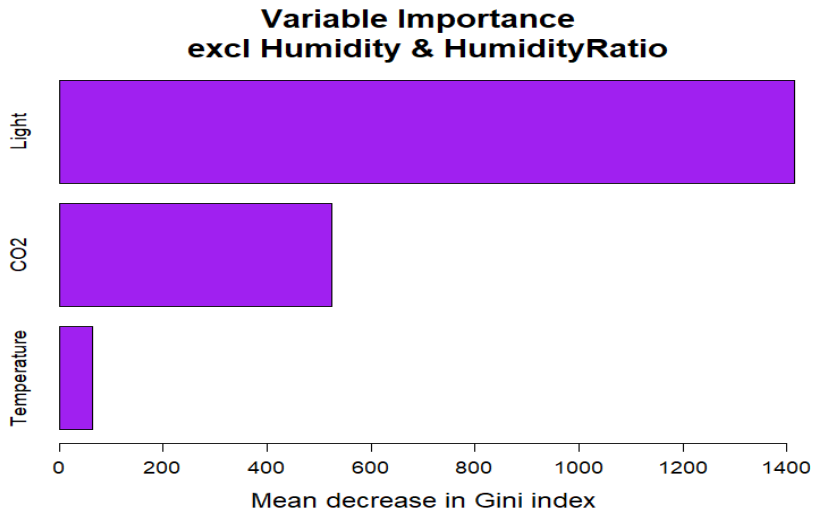

Figure 4.2: *Model10 variable importance.*

Figure 4.3 is a plot of different models and their associated OOB error estimates. Bagged and Random Forest models including and excluding Humidity and HumidityRatio are compared.

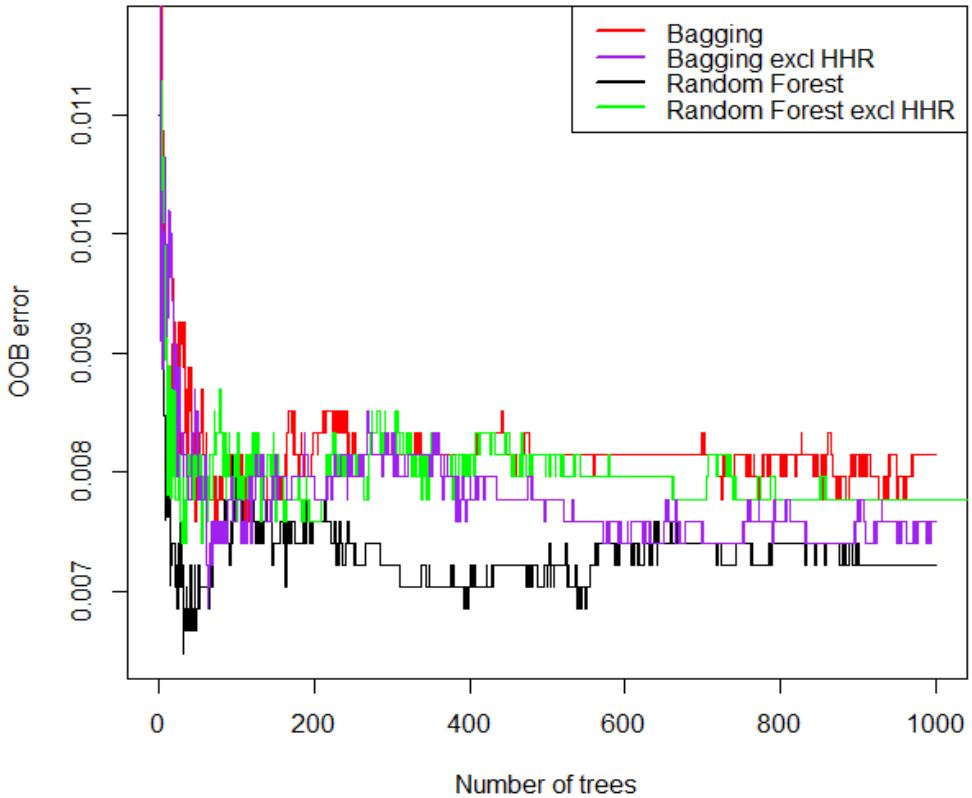

Figure 4.3: *Model comparison based on OOB error.*

The best performing models were the 1000 tree Bagged model excluding Humidity and HumidityRatio (Model10) and the 1000 tree Random Forest using all predictors (Model11) and. Their respective test set classification accuracies are displayed in Table 4.1.

		True		**Accuracy_ Model10 (%)**
		0	1	
Predicted	0	1016	3	
	1	2	328	**99.6294**
		True		**Accuracy_ Model11 (%)**
		0	1	
Predicted	0	1015	1	
	1	3	330	**99.7035**

Table 4.1: *Model10&11 confusion matrices.*

Perhaps unsurprisingly, the Random Forest (Model11) outperformed the Bagged model, albeit by 0.07%.

Chapter 5

Gradient Boosting

5.1 Methodology

Gradient Boosting algorithms build an ensemble of shallow trees sequentially, with each subsequent tree learning from the previous. The process essentially converts weak learners into strong learners. There are three primary tuning parameters which can be set in Gradient Boosting:

1. Number of trees used (B): Gradient Boosting often requires a large number of trees which can lead to overfit. The goal is therefore to find the optimal number of trees such that the loss function is minimized.

2. Shrinkage parameter or learning rate (λ): The shrinkage parameter controls the rate at which the algorithm proceeds down the descent. There exists a trade off between a small learning rate which reduces overfitting, but takes longer to find the optimal fit, and vice versa.

3. Interaction depth (d): The interaction depth controls the number of splits performed on the tree which directly affects the complexity of the model.

5.2 Results

The initial model used was a standard Gradient Boosted model with 10’000 trees, an interaction depth of 2 and λ = 0.005. Built in cross validation found the best iteration at 9914 with all five predictors having non-zero influence.

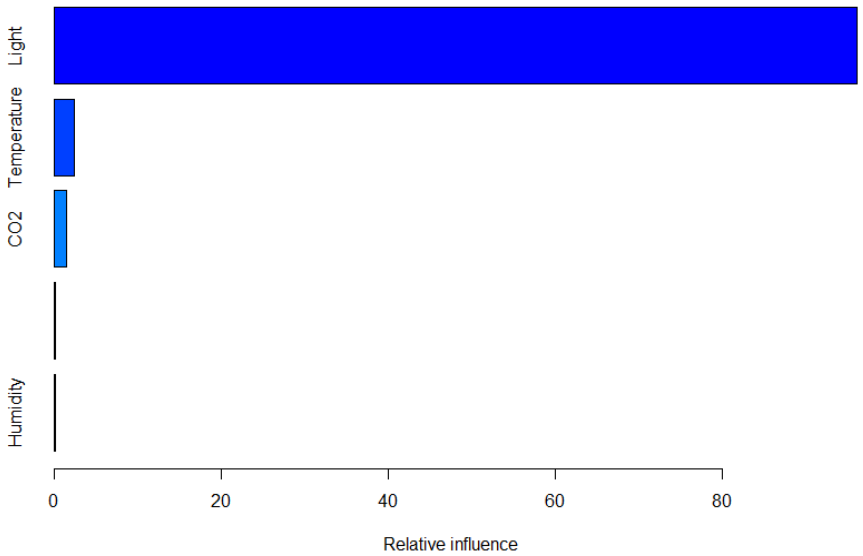

Figure 5.1: *Variable importance plot.*

Again, Light is the most important predictor, with Humidity and HumidityRatio the least important for Gradient Boosting. Figure 5.2 depicts the partial dependence plots of the variables Temperature, Light and CO2. It illustrates that an increase in Temperature from 22°C onwards drastically decreases the probability of a room being unoccupied (0). Additionally, Light and Temperature exhibit similar relationships. An increase in Light and CO2 measurements relate to an increase in the probability that a room is occupied (1).

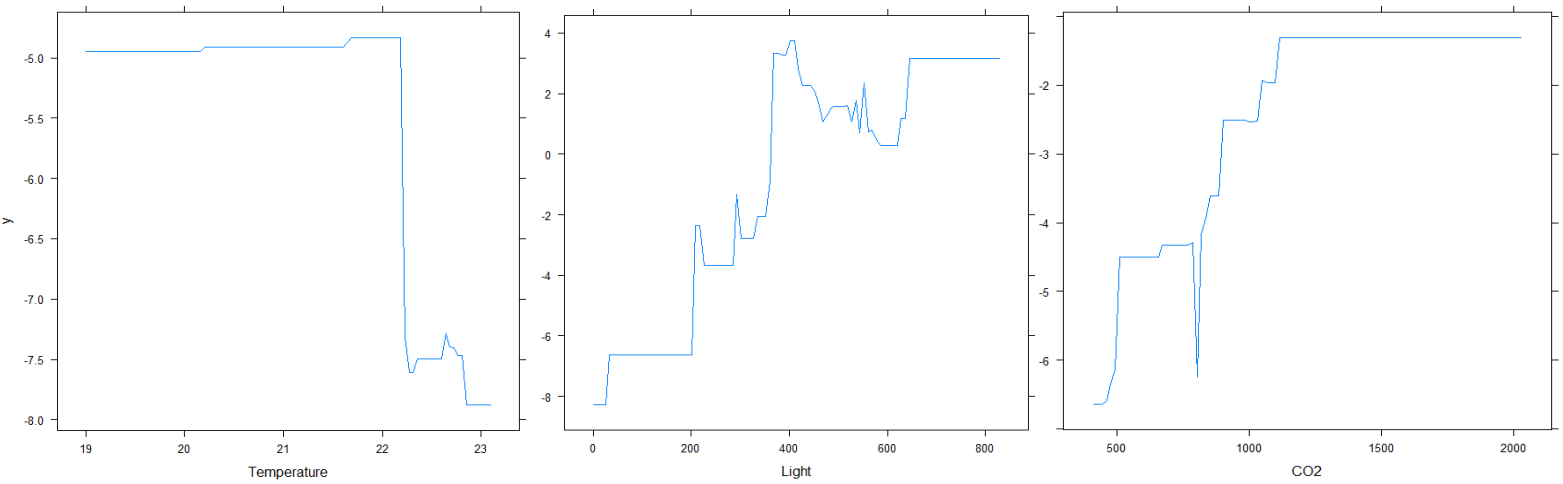

Figure 5.2: *Partial dependence plots for Temperature, Light and CO2.*

Models12&13 were built making use of the `tuneGrid()` parameter which evaluates the accuracy of different models depending on the combination of values for B(1000,1500,2000), d(1,2,6) and λ(0.01, 0.005, 0.001).

Model12 was created using all predictors available, B = 2000, d = 6 and λ = 0.001.
Model13 used all predictors except for Humidity and HumidityRatio, B = 2000, d = 6 and λ = 0.001. The reasoning for omitting these two predictors was based on their low relative influence (0.09 and 0.11 respectively). Their respective confusion matrices based on their classification accuracy on the test set are presented in Table 5.1.

		True		**Accuracy_ Model12 (%)**
		0	1	
Predicted	0	1013	2	
	1	5	329	**99.4811**
		True		**Accuracy_ Model13 (%)**
		0	1	
Predicted	0	1015	2	
	1	3	329	**99.6294**

Table 5.1: *Model12&13 confusion matrices.*

Chapter 6

Best Model Evaluation on Unseen Test Data

As stated previously, the models were built using the 80/20% data split into train and test data sets from the original training.csv. The testing.csv has not been used in any model building and therefore treated as completely unseen. The best four models will now be tested on the unseen data, their classification matrices are presented in Tables 6.1-4.

6.1 Logistic Regression (Model7):

		True		**Classification Accuracy (%)**	**AUC*100**
		0	1		
Predicted	0	1660	54		
	1	33	918	**96.7355**	**99.0749**

Table 6.1: *Classification accuracy and AUC for Model7.*

6.2 Classification Trees (Model9):

		True		**Classification Accuracy (%)**
		0	1	
Predicted	0	1662	91	
	1	31	881	**95.4221**

Table 6.2: *Classification accuracy for Model9.*

6.3 Bagging and Random Forests (Model11):

		True		**Classification Accuracy (%)**
		0	1	
Predicted	0	1642	83	
	1	51	889	**94.9719**

Table 6.3: *Classification accuracy for Model11.*

6.4 Gradient Boosting (Model13):

		True		**Classification Accuracy (%)**
		0	1	
Predicted	0	1644	113	
	1	49	859	**93.9212**

Table 6.4: *Classification accuracy for Model13.*

Chapter 7

Conclusion

In this report, Logistic Regression, Classification Trees, Bagging/Random Forest and Gradient Boosted methods were used to model the binary classification problem relating to the occupancy dataset. Candenado and Feldheim (2016, p.28) recorded measurements of Light, Temperature, Humidity and CO2 levels in an office as well as the relevant states of occupancy: unoccupied assigned a value of 0, whilst an occupied room was assigned a value of 1.

The models were trained on the training data comprised of an 80/20% split into the train and test sets. The single best model out of each section is evaluated in Chapter 6 using the unseen testing data. Model performance is based on the classification accuracy and presented in Tables 6.1-4. The models are ranked as follows:

1. Model7: a logistic regression model with Lasso penalty using only Light, Temperature and CO2 as predictors – classification accuracy: 96.7355%

Variable	Coefficient
(Intercept)	31.7469 (β_0)
Temperature (X_1)	-2.1372 (β_1)
Light (X_2)	0.0237 (β_2)
CO2 (X_3)	0.0075 (β_3)

Table 7.1: *Model7 coefficients.*

2. Model9: a classification tree pruned down to four nodes using cost complexity pruning based on deviance splitting criteria - classification accuracy: 95.4221%

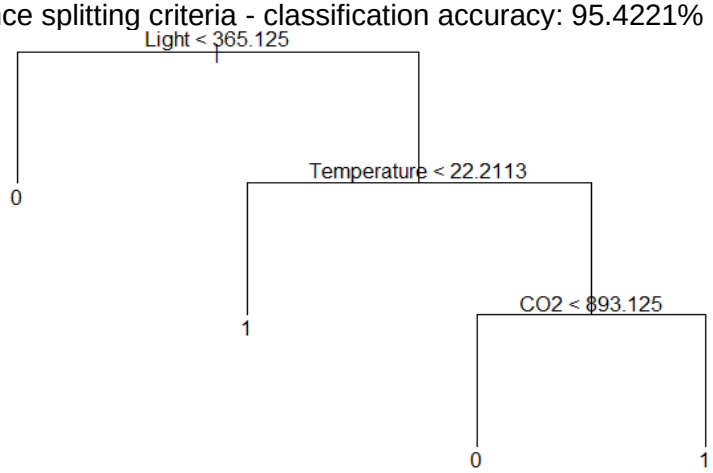

Figure 3.3: *Classification tree with 4 nodes (Model9).*

3. Model11: a random forest made up of 1000 trees using all predictor variables - classification accuracy: 94.9719%

4. Model13: a gradient boosted model using all predictors except for Humidity and HumidityRatio, number of trees = 2000, interaction depth = 6 and shrinkage parameter = 0.001 – classification accuracy: 93.9212%

The most accurate two models in this report require only readings from Temperature, Light and CO2 gauges in order to predict occupancy in a room. The models are simple and easily interpretable, yet robust, and offer insight into the most important variables required to predict occupancy of a room.

A-0: Descriptive Statistics

```
#### LOAD AND SPLIT DATA ####

rm(list=ls())

setwd("C:/Users/user/Desktop/FinTech 2020/Supervised Learning/Assignment 2")


#Read in train and test sets

occtrain <- read.csv("occupancy_training.csv", header = TRUE)

occtrain$date <- NULL

occtrain$Occupancy <- as.factor(occtrain$Occupancy)

head(occtrain)

str(occtrain) #inspect structure


finaltest <- read.csv("occupancy_testing.csv", header = TRUE)

finaltest$date <- NULL

finaltest$Occupancy <- as.factor(finaltest$Occupancy)

head(finaltest)

str(finaltest) #inspect structure


#Split occtrain into test and train sets

library(caret)

set.seed(23)

samp <- createDataPartition(occtrain[,"Occupancy"],1,.8)[[1]]

samp


#80% train split

train = occtrain[samp,]

train$Occupancy <- as.factor(train$Occupancy)

head(train)

str(train)


#20% test split

test = occtrain[-samp,]

test$Occupancy <- as.factor(test$Occupancy)

head(test)

str(test)
```

```
#### 0. DESCRIPTIVE STATISTICS ####

# Since levels of $Occupancy (0&1) have unequal distributions;
# createDataPartition allows you to create a sample that maintains
# the ratio or balance of the factor classes.

#Number of 0 vs 1 entries
#Same proportions maintained
table(occtrain$Occupancy)#0:5092 1:1659
round(5092/1659, 2) #3.07

table(train$Occupancy) #0:4074 1:1328
round(4074/1328, 2) #3.07

table(test$Occupancy) #0:1018 1:331
round(1018/331, 2) #3.08

#Correlation Table
num_occtrain <- read.csv("occupancy_training.csv", header = TRUE)
num_occtrain$date <- NULL
correlation <- cor(num_occtrain)
t <- round(correlation, 2)
t
# write.table(t, file = "correlation.txt", sep = ",",
#           quote = FALSE, row.names = F)

#Correlation cluster plot
library(corrplot)
corrplot(correlation, type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45,)

#Correlation heatmap
col <- colorRampPalette(c("black", "white", "magenta"))(20)
heatmap(x = correlation, col = col, symm = TRUE, margins = c(10, 10))
legend(x = "topright", legend = c("High Correlation", "Low Correlation"),
       fill = colorRampPalette(c("magenta", "black"))(2))
```

```
#GGplot table comparing features (as per Candenado, L.,
#Occupancy-detection-data, (2016), GitHub repository:
#https://github.com/LuisM78/Occupancy-detection-data)

library(ggplot2)
library(grid)
library(gridExtra)
library(scales)
pushViewport(viewport(layout = grid.layout(6, 1)))

datatesting <- read.table("datatest.txt",header=TRUE,sep=",")
datatesting$Occupancy <- datatesting$Occupancy
datatesting$Occupancy  <- as.factor(datatesting$Occupancy)
datatesting$date <- as.POSIXct(datatesting$date,tz="UTC")
str(datatesting) #inspect

myplot1 <- ggplot(datatesting,aes(date)) +
  geom_line(color="Red", aes(y = Temperature)) +
  ylab("Temperature") + xlab("Time") +
  scale_x_datetime(breaks = date_breaks("60 min"), labels = date_format("%H:%M"),
                   limits = as.POSIXct(c("2015-02-03 7:00","2015-02-04 8:00"),
tz="GMT")) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

myplot2 <- ggplot(datatesting, aes(date)) +
  geom_line(color = "Blue", aes(y = Humidity)) +
  ylab("Humidity") + xlab("Time")+
  scale_x_datetime(breaks = date_breaks("60 min"), labels = date_format("%H:%M"),
                   limits = as.POSIXct(c("2015-02-03 7:00","2015-02-04 8:00"),
                                       tz = "GMT")) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

myplot3 <- ggplot(datatesting, aes(date)) +
  geom_line(color = "deepskyblue1", aes(y = HumidityRatio)) +
  ylab("HumidityRatio") + xlab("Time") +
  scale_x_datetime(breaks = date_breaks("60 min"), labels = date_format("%H:%M"),
                   limits = as.POSIXct(c("2015-02-03 7:00","2015-02-04 8:00"),
```

```
                                        tz="GMT")) +
  scale_y_continuous(breaks = seq(0.003, 0.006, by = 0.003),
                     limits = c(0.003, 0.006)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        axis.text.y = element_text(angle = 90, hjust = 1))

myplot4 <- ggplot(datatesting, aes(date)) +
  geom_line(color = "Green", aes(y = CO2))+
  ylab("CO2 (ppm)") + xlab("Time") +
  scale_x_datetime(breaks = date_breaks("60 min"), labels = date_format("%H:%M"),
                   limits = as.POSIXct(c("2015-02-03 7:00","2015-02-04 8:00"),
                                       tz = "GMT")) +
  scale_y_continuous(breaks = seq(400, 1400, by = 500),
                     limits = c(400, 1400)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        axis.text.y = element_text(angle = 90, hjust = 1))

myplot5 <- ggplot(datatesting, aes(date)) +
  geom_line(color = "gold4", aes(y = Light)) +
  ylab("Light (Lux)") + xlab("Time") +
  scale_x_datetime(breaks = date_breaks("60 min"), labels = date_format("%H:%M"),
                   limits = as.POSIXct(c("2015-02-03 7:00","2015-02-04 8:00"),
                                       tz = "GMT")) +
  scale_y_continuous(breaks = seq(0, 700, by = 150),
                     limits = c(0, 700)) +
theme(axis.text.x = element_text(angle = 90, hjust = 1))

myplot6 <- ggplot(datatesting, aes(date)) +
  geom_line(color = "Black", aes(y = as.numeric(Occupancy) - 1)) +
  ylab("Occupancy") + xlab("Time") +
  scale_x_datetime(breaks = date_breaks("60 min"), labels = date_format("%H:%M"),
                   limits = as.POSIXct(c("2015-02-03 7:00","2015-02-04 8:00"),
                                       tz = "GMT"))+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

print(myplot1, vp = viewport(layout.pos.row = 1, layout.pos.col = 1))
print(myplot2, vp = viewport(layout.pos.row = 2, layout.pos.col = 1))
```

```
print(myplot3, vp = viewport(layout.pos.row = 3, layout.pos.col = 1))
print(myplot4, vp = viewport(layout.pos.row = 4, layout.pos.col = 1))
print(myplot5, vp = viewport(layout.pos.row = 5, layout.pos.col = 1))
print(myplot6, vp = viewport(layout.pos.row = 6, layout.pos.col = 1))

myplot1 <- ggplot_gtable(ggplot_build(myplot1))
myplot2 <- ggplot_gtable(ggplot_build(myplot2))
myplot3 <- ggplot_gtable(ggplot_build(myplot3))
myplot4 <- ggplot_gtable(ggplot_build(myplot4))
myplot5 <- ggplot_gtable(ggplot_build(myplot5))
myplot6 <- ggplot_gtable(ggplot_build(myplot6))

maxWidth = unit.pmax(myplot1$widths[2:3], myplot2$widths[2:3],
                     myplot3$widths[2:3], myplot4$widths[2:3],
                     myplot5$widths[2:3],myplot6$widths[2:3])

myplot1$widths[2:3] <- maxWidth
myplot2$widths[2:3] <- maxWidth
myplot3$widths[2:3] <- maxWidth
myplot4$widths[2:3] <- maxWidth
myplot5$widths[2:3] <- maxWidth
myplot6$widths[2:3] <- maxWidth

#Temperature, Humidity, HumidityRatio & CO2
grid.arrange(myplot1, myplot2, myplot3, myplot4, ncol=1)

#Light & Occupancy
grid.arrange(myplot5, myplot6, ncol=1)

# occtrain <- read.csv("occupancy_training.csv", header = TRUE)
# occtrain$date <- NULL
# cols2 <- character(nrow(occtrain))
# cols2[] <- "black"
# cols2[occtrain$Occupancy %in% c("0")] <- "black"
# cols2[occtrain$Occupancy %in% c("1")] <- "gold"
# pairs(occtrain[1:5], col = cols2, cex = 1.1, cex.labels = 1.5)
```

```
# legend("bottomright", fill = occtrain$Occupancy, legend =
c(levels(occtrain$Occupancy)))


#Generalized pairs plot
library(GGally)
occtrain <- read.csv("occupancy_training.csv", header = TRUE)
occtrain$date <- NULL
ggpairs(occtrain, aes(color = as.factor(Occupancy)), columns = 1:5) +
  scale_fill_manual(values=c("black", "#FF00CC")) +
  scale_color_manual(values=c("black", "#FF00CC"))


library(ggplot2)
library(ggpubr)
theme_set(theme_pubr())
library(viridis)
library(dplyr)
library(RColorBrewer)
data <- occtrain %>%
  mutate(Occupancy = as.factor(Occupancy),
         Temperature = as.numeric(Temperature))


#Temperature vs Occupancy
ggplot(data, aes(x = as.numeric(Temperature), fill = Occupancy)) +
  geom_histogram(binwidth=0.1, color="#e9ecef", alpha=0.9) +
  theme_classic() +
  xlab("Temperature") +
  ylab("Count") +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = c(0.9, 0.7)) +
  #scale_color_brewer(palette = "Dark2") +
  scale_fill_brewer(palette = "Paired") + #check all available palettes
  ggtitle("Temperature vs Occupancy") +
  scale_x_continuous(name = "Temperature (C)",
                     breaks = seq(19, 23, by = 0.5),
                     limits=c(19, 23))


#Light vs Occupancy
```

```
ggplot(data, aes(x = as.numeric(Light), fill = Occupancy)) +
  geom_histogram(binwidth = 25, color="#e9ecef", alpha = 0.9) +
  theme_classic() +
  xlab("Light") +
  ylab("Count") +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = c(0.9, 0.7)) +
  scale_fill_brewer(palette = "Dark2") + #check all available palettes
  ggtitle("Light vs Occupancy") +
  scale_x_continuous(name = "Light (Lux)",
                     breaks = seq(0, 600, by = 50),
                     limits=c(0, 600)) +
  scale_y_continuous(breaks = seq(0, 700, by = 100),
                     limits=c(0, 700))


#CO2 vs Occupancy
ggplot(data, aes(x = as.numeric(CO2), fill = Occupancy)) +
  geom_histogram(binwidth = 50, color="#e9ecef", alpha = 0.9) +
  theme_classic() +
  xlab("CO2") +
  ylab("Count") +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = c(0.9, 0.7)) +
  scale_fill_hue(h = c(180, 280), c = 80, l = 50, h.start = 0, direction = 1,
aesthetics = "fill") +
  ggtitle("CO2 vs Occupancy") +
  scale_x_continuous(name = "CO2 (ppm)",
                     breaks = seq(400, 2000, by = 100),
                     limits=c(400, 2000)) +
  scale_y_continuous(breaks = seq(0, 250, by = 50),
                     limits=c(0, 250))


#Humidity vs Occupancy
ggplot(data, aes(x = as.numeric(Humidity), fill = Occupancy)) +
  geom_histogram(binwidth = 1, color="#e9ecef", alpha = 0.9) +
  theme_classic() +
  xlab("Humidity") +
```

```
  ylab("Count") +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = c(0.9, 0.7)) +
  scale_fill_hue(h = c(180, 280), c = 80, l = 50, h.start = 0, direction = 1,
aesthetics = "fill") +
  ggtitle("Humidity vs Occupancy") +
  scale_x_continuous(name = "Humidity (%)",
                     breaks = seq(15, 40, by = 1),
                     limits=c(15, 40)) +
  scale_y_continuous(breaks = seq(0, 1000, by = 100),
                     limits=c(0, 1000))


#HumidityRatio vs Occupancy
ggplot(data, aes(x = as.numeric(HumidityRatio), fill = Occupancy)) +
  geom_histogram(binwidth = 0.1e-3, color="#e9ecef", alpha = 0.9) +
  theme_classic() +
  xlab("HumidityRatio") +
  ylab("Count") +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = c(0.9, 0.7)) +
  scale_fill_hue(h = c(180, 280), c = 80, l = 50, h.start = 0, direction = 1,
aesthetics = "fill") +
  ggtitle("HumidityRatio vs Occupancy") +
  scale_x_continuous(name = "HumidityRatio",
                     breaks = seq(2.5e-3, 6.5e-3, by = 1e-3),
                     limits=c(2.5e-3, 6.5e-3)) +
  scale_y_continuous(breaks = seq(0, 600, by = 100),
                     limits=c(0, 600))
```

A-1: Logistic Regression

```
#### 1. LOGISTIC REGRESSION ####

# Logistic Regression without Lasso or Ridge Penalty (Standard LR)
# All predictors
model1 <- glm(Occupancy ~ ., data = train, family = 'binomial')
print(model1)
summary(model1)
round(exp(coef(model1)),3)

# Excluding Temperature
model2 <- glm(Occupancy ~ .-Temperature, data = train, family = 'binomial')
print(model2)
summary(model2)
round(exp(coef(model2)),3) # Odds effect

# Excluding Temperature & HumidityRatio
model3 <- glm(Occupancy ~ .-Temperature-HumidityRatio, data = train, family =
'binomial')
print(model3)
summary(model3)
round(exp(coef(model3)),3) # Odds effect

# Excluding Temperature & HumidityRatio & Humidity
model4 <- glm(Occupancy ~ .-Temperature-HumidityRatio-Humidity, data = train,
family = 'binomial')
print(model4)
summary(model4)
round(exp(coef(model4)),3) # Odds effect

model5 <- glm(Occupancy ~ .-HumidityRatio-Humidity, data = train, family =
'binomial')
print(model5)
summary(model5)
round(exp(coef(model5)),3) # Odds effect

# Logistic Regression with Lasso Penalty
library(glmnet)

X <- train[ , -c(6)]
head(X)
X <- as.matrix(X)
head(X)
Y <- train[ , -c(1:5)]
head(Y)
Y <- as.matrix(Y)
head(Y)

testX <- test[ , -c(6)]
testX <- as.matrix(testX)
head(testX)
testY <- test[ , -c(1:5)]
testY <- as.matrix(testY)
head(testY)

#model 6
modelCV <- glmnet(X, Y, alpha = 1, standardize = TRUE, family = 'binomial')
plot(modelCV, xvar = 'lambda', label=TRUE)

X_dropped <- train[ , -c(2,5,6)]
```

```
X_dropped <- as.matrix(X_dropped)
head(X_dropped)

testX_dropped <- test[ , -c(2,5,6)]
testX_dropped <- as.matrix(testX_dropped)
head(testX_dropped)

#model 7
modelCV_dropped <- glmnet(X_dropped, Y, alpha = 1, standardize = TRUE, family =
'binomial')
plot(modelCV_dropped, xvar = 'lambda', label=TRUE)


#model 6
# 10-fold CV results for Logistic Regression with Lasso Penalty
set.seed(23)
cv <- cv.glmnet(X, Y, alpha = 1, nfolds = 10, type.measure = 'mse',
                standardize = T, family = 'binomial')
plot(cv)
abline(v = log(cv$lambda.min), lwd = 2)
cv$lambda.min
log(cv$lambda.min)
coef(modelCV, s = cv$lambda.min)

#model 7
# 10-fold CV results for Logistic Regression with Lasso Penalty
set.seed(23)
cv_dropped <- cv.glmnet(X_dropped, Y, alpha = 1, nfolds = 10, type.measure = 'mse',
                standardize = T, family = 'binomial')
plot(cv_dropped)
abline(v = log(cv_dropped$lambda.min), lwd = 2)
cv_dropped$lambda.min
log(cv_dropped$lambda.min)
coef(modelCV_dropped, s = cv_dropped$lambda.min)

# Predict under standard LR:
pi_hat_1 <- predict(model1, newdata = test, type = 'response')
head(pi_hat_1)

pi_hat_2 <- predict(model2, newdata = test, type = 'response')
head(pi_hat_2)

pi_hat_3 <- predict(model3, newdata = test, type = 'response')
head(pi_hat_3)

pi_hat_4 <- predict(model4, newdata = test, type = 'response')
head(pi_hat_4)

pi_hat_5 <- predict(model5, newdata = test, type = 'response')
head(pi_hat_5)

# LASSO with lambda chosen by CV:
set.seed(23)
pi_hat_cv <- predict(modelCV, newx = testX,
                     s = cv$lambda.min, type = 'response')

set.seed(23)
pi_hat_cv_dropped <- predict(modelCV_dropped, newx = testX_dropped,
                     s = cv_dropped$lambda.min, type = 'response')

Y_hat_1 <- ifelse(pi_hat_1 >= 0.5, "1", "0")
Y_hat_2 <- ifelse(pi_hat_2 >= 0.5, "1", "0")
Y_hat_3 <- ifelse(pi_hat_3 >= 0.5, "1", "0")
```

```
Y_hat_4 <- ifelse(pi_hat_4 >= 0.5, "1", "0")
Y_hat_5 <- ifelse(pi_hat_5 >= 0.5, "1", "0")
Y_hat_cv <- ifelse(pi_hat_cv >= 0.5, "1", "0")
Y_hat_cv_dropped <- ifelse(pi_hat_cv_dropped >= 0.5, "1", "0")

# Misclassification error rate:
N <- length(testY)
Error_1 <- sum((Y_hat_1!=testY)/N)*100 #model1
round((Accuracy_1 <- 100 - Error_1), 4)
Error_2 <- sum((Y_hat_2!=testY)/N)*100 #model2
round((Accuracy_2 <- 100 - Error_2), 4)
Error_3 <- sum((Y_hat_3!=testY)/N)*100  #model3
round((Accuracy_3 <- 100 - Error_3), 4)
Error_4 <- sum((Y_hat_4!=testY)/N)*100  #model4
round((Accuracy_4 <- 100 - Error_4), 4)
Error_5 <- sum((Y_hat_5!=testY)/N)*100  #model5
round((Accuracy_5 <- 100 - Error_5), 4)
Error_cv <- sum((Y_hat_cv!=testY)/N)*100  #model6
round((Accuracy_cv <- 100 - Error_cv), 4)
Error_cv_dropped <- sum((Y_hat_cv_dropped!=testY)/N)*100  #model7
round((Accuracy_cv_dropped <- 100 - Error_cv_dropped), 4)

# Create confusion matrices
(conf_mod1 <- cbind(table(Y_hat_1, testY, dnn = c("predict", "true")), Accuracy_1))
(conf_mod2 <- cbind(table(Y_hat_2, testY, dnn = c("predict", "true")), Accuracy_2))
(conf_mod3 <- cbind(table(Y_hat_3, testY, dnn = c("predict", "true")), Accuracy_3))
(conf_mod4 <- cbind(table(Y_hat_4, testY, dnn = c("predict", "true")), Accuracy_4))
(conf_mod5 <- cbind(table(Y_hat_5, testY, dnn = c("predict", "true")), Accuracy_5))
(conf_modcv <- cbind(table(Y_hat_cv, testY, dnn = c("predict", "true")),
Accuracy_cv))
(conf_modcv_dropped <- cbind(table(Y_hat_cv_dropped,
                                   testY, dnn = c("predict", "true")),
                             Accuracy_cv_dropped))

# ROC curves
library(ROCR)

# Model1:
pred_1 <- prediction(pi_hat_1, testY)
perf_1  <- performance(pred_1, 'tpr', 'fpr')
plot(perf_1, colorize = FALSE, col = 'black')
lines(c(0,1), c(0,1), col = 'gray', lty = 4)
performance(pred_1, measure = 'auc')@y.values[[1]]*100 #area under curve

# Model2:
pred_2 <- prediction(pi_hat_2, testY)
perf_2  <- performance(pred_2, 'tpr', 'fpr')
plot(perf_2, colorize = FALSE, col = 'black')
lines(c(0,1), c(0,1), col = 'gray', lty = 4)
performance(pred_2, measure = 'auc')@y.values[[1]]*100 #area under curve

# Model3:
pred_3 <- prediction(pi_hat_3, testY)
perf_3  <- performance(pred_3, 'tpr', 'fpr')
plot(perf_3, colorize = FALSE, col = 'black')
lines(c(0,1), c(0,1), col = 'gray', lty = 4)
performance(pred_3, measure = 'auc')@y.values[[1]]*100 #area under curve

# Model4:
pred_4 <- prediction(pi_hat_4, testY)
perf_4  <- performance(pred_4, 'tpr', 'fpr')
plot(perf_4, colorize = FALSE, col = 'black')
lines(c(0,1), c(0,1), col = 'gray', lty = 4)
```

```
performance(pred_4, measure = 'auc')@y.values[[1]]*100 #area under curve

# Model5:
pred_5 <- prediction(pi_hat_5, testY)
perf_5  <- performance(pred_5, 'tpr', 'fpr')
plot(perf_5, colorize = FALSE, col = 'black')
lines(c(0,1), c(0,1), col = 'gray', lty = 4)
performance(pred_5, measure = 'auc')@y.values[[1]]*100 #area under curve

# Model6 (with Lasso)
pred_cv <- prediction(pi_hat_cv, testY)
perf_cv  <- performance(pred_cv, 'tpr', 'fpr')
plot(perf_cv, colorize = FALSE, col = 'black')
lines(c(0,1), c(0,1), col = 'gray', lty = 4)
performance(pred_cv, measure = 'auc')@y.values[[1]]*100

# Model7 (With Lasso)
pred_cv_dropped <- prediction(pi_hat_cv_dropped, testY)
perf_cv_dropped  <- performance(pred_cv_dropped, 'tpr', 'fpr')
plot(perf_cv_dropped, colorize = FALSE, col = 'black')
lines(c(0,1), c(0,1), col = 'gray', lty = 4)
performance(pred_cv_dropped, measure = 'auc')@y.values[[1]]*100
```

A-2: Classification Trees

```
#### 2. CLASSIFICATION TREE ####

library(tree)
# Using gini index reduction...
tree_occtrain_g <- tree(Occupancy ~ ., data = train, split = 'gini')
# and using deviance reduction as splitting criterion
tree_occtrain_d <- tree(Occupancy ~ ., data = train, split = 'deviance')

# First using Gini index
summary(tree_occtrain_g)
tree_occtrain_g
plot(tree_occtrain_g, type = c("uniform"))
text(tree_occtrain_g, cex = 0.9) #BIG, messy tree!
# Gini doesn't take into account the number of observations in each resulting node
# "Good" gini splits early on lead to very little reduction in the deviance

# Checking deviance
summary(tree_occtrain_d)
tree_occtrain_d
plot(tree_occtrain_d, type = c("uniform"))
text(tree_occtrain_d, cex = 0.9, pretty = 1) #Much more sensible

# Grow the tree deeper
tree_occtrain <- tree(Occupancy ~ ., data = train, split = 'deviance',
                      control = tree.control(nrow(train),
                                             mindev = 0.005))
summary(tree_occtrain) #Checks out
tree_occtrain
plot(tree_occtrain, type = c("uniform"))
text(tree_occtrain, cex = 0.9, pretty = 1) #Much more sensible

# Cost complexity pruning
set.seed(23)
cv_occtrain <- cv.tree(tree_occtrain, FUN = prune.misclass) #use classification
error rate for pruning

# Aesthetics
plot(cv_occtrain$size, cv_occtrain$dev, type='o', pch = 16, col = 'navy', lwd = 2,
     xlab='Number of terminal nodes', ylab='CV error')

cv_occtrain$k[1] <- 0 #by default is neg inf., change to 0
alpha <- round(cv_occtrain$k,1)
axis(3, at = cv_occtrain$size, lab = alpha, cex.axis  =0.8)
mtext(expression(alpha), 3, line = 2.5, cex = 1.2)
axis(side = 1, at = 1:max(cv_occtrain$size))
T <- cv_occtrain$size[which.min(cv_occtrain$dev)] #The minimum CV Error (8 nodes)
abline(v = T, lty = 2, lwd = 2, col = 'red')
abline(v = 4, lty = 2, lwd = 2, col = 'green')

#Model8
# Prune the tree with 8 nodes
pr_tree_8 <- prune.misclass(tree_occtrain, best = T) #8 nodes
plot(pr_tree_8, type = c("uniform"))
text(pr_tree_8, pretty = 0)

# Predict on test set
yhat_8 <- predict(pr_tree_8, test, type = 'class') #type argument = classification!
(c_mat <- table(yhat_8, test$Occupancy)) #predicted vs actaul YES and NO
sum(diag(c_mat))/nrow(test)*100 #classification accuracy %
```

```
#REPEAT

#Model9
# Prune the tree with 4 nodes
pr_tree_4 <- prune.misclass(tree_occtrain, best = 4) #4 nodes
plot(pr_tree_4, type = c("uniform"))
text(pr_tree_4, pretty = 0)

# Predict on test set
yhat_4 <- predict(pr_tree_4, test, type = 'class') #type argument nb for
classification!
(c_mat <- table(yhat_4, test$Occupancy)) #predicted vs actaul YES and NO
sum(diag(c_mat))/nrow(test)*100 #classification accuracy %
```

A-3: Bagging and Random Forests

```
#### 3. BAGGED TREES & RANDOMFOREST ####
library(randomForest)

#With Light
set.seed(23)
bag_occtrain <- randomForest(Occupancy ~ ., data = train,
                mtry = ncol(train) - 1, #for bagging, use all predictors(# features
to use at each split)
                ntree = 1000, #number of trees
                importance = TRUE, #keep track of reduction in loss function
                do.trace = 100)  #print out regular progress
bag_occtrain

## Choose number of trees:
head(bag_occtrain$err.rate)
plot(bag_occtrain$err.rate[, 'OOB'], type = 's', xlab = 'Number of trees', ylab =
'OOB error')

## Variable importance plot
varImpPlot(bag_occtrain, type = 2) #type = 2: Reduction in gini index
set.seed(23)
bag_varimp <- randomForest::importance(bag_occtrain, type = 2)
bag_varimp <- bag_varimp[order(bag_varimp, decreasing = FALSE),]
barplot(bag_varimp, horiz = T, col = 'purple', las = 1,
        las = 0,
        xlab = 'Mean decrease in Gini index', cex.lab = 1.5, cex.axis = 1.2,
        main = 'Variable Importance', cex.main = 1.8, cex.names = 0.8)

#Model10
#Without Humidity and HumidityRatio
set.seed(23)
bag_occtrain_exHHR <- randomForest(Occupancy ~ .-Humidity-HumidityRatio, data =
train,
                            mtry = ncol(train) - 4, #features to use at each split
                            ntree = 1000, #number of trees
                            importance = TRUE, #keep track of reduction in loss
function
                            do.trace = 100)  #print out regular progress
bag_occtrain_exHHR

## Choose number of trees:
head(bag_occtrain_exHHR$err.rate)
plot(bag_occtrain_exHHR$err.rate[, 'OOB'], type = 's', xlab = 'Number of trees',
ylab = 'OOB error')

## Variable importance plot
varImpPlot(bag_occtrain_exHHR, type = 2) #type = 2: Reduction in gini index
set.seed(23)
bag_varimp_exHHR <- randomForest::importance(bag_occtrain_exHHR, type = 2)
bag_varimp_exHHR <- bag_varimp_exHHR[order(bag_varimp_exHHR, decreasing = FALSE),]
barplot(bag_varimp_exHHR, horiz = T, col = 'purple', las = 1,
        las = 0,
        xlab = 'Mean decrease in Gini index', cex.lab = 1.5, cex.axis = 1.2,
        main = 'Variable Importance \n excl Humidity & HumidityRatio', cex.main =
1.8, cex.names = 1.2)

# Model11
# Fit a Random Forest (1000 trees)
# Using all predictors
```

```
set.seed(23)
rf_occtrain_1 <- randomForest(Occupancy ~ ., data = train,
                              ntree = 1000,
                              importance = TRUE,
                              do.trace = 100)
# For classification tree, default mtry = floor(sqrt(ncol(x)))
rf_occtrain_1

## Fit a Random Forest (1500 trees)
set.seed(23)
rf_occtrain_2 <- randomForest(Occupancy ~ ., data = train,
                              ntree = 1500,
                              importance = TRUE,
                              do.trace = 100)
# For classification tree, default mtry = floor(sqrt(ncol(x)))
rf_occtrain_2

## Fit a Random Forest (2000 trees)
set.seed(23)
rf_occtrain_3 <- randomForest(Occupancy ~ ., data = train,
                              ntree = 2000,
                              importance = TRUE,
                              do.trace = 100)
# For classification tree, default mtry = floor(sqrt(ncol(x)))
rf_occtrain_3

## Fit a Random Forest (2000 trees) excluding Humidity and HumidityRatio
set.seed(23)
rf_occtrain_4 <- randomForest(Occupancy ~ .-Humidity-HumidityRatio, data = train,
                              ntree = 2000,
                              importance = TRUE,
                              do.trace = 100)
# For classification tree, default mtry = floor(sqrt(ncol(x)))
rf_occtrain_4

## Variable importance plot
varImpPlot(rf_occtrain_1, type = 2)

# compare OOB errors:
par(mfrow=c(1,1))
plot(rf_occtrain_1$err.rate[, 'OOB'], type = 's', xlab = 'Number of trees', ylab =
'OOB error')
lines(bag_occtrain$err.rate[, 'OOB'], col = 'red', type = 's')
lines(rf_occtrain_4$err.rate[, 'OOB'], col = 'green', type = 's')
lines(bag_occtrain_exHHR$err.rate[, 'OOB'], col = 'purple', type = 's')
legend('topright', legend = c('Bagging', 'Bagging excl HHR',
                              'Random Forest', 'Random Forest excl HHR'),
       col = c('red', 'purple', 'black', 'green'), lwd = 2)

## Use both models for prediction
bag_pred <- predict(bag_occtrain_exHHR, newdata = test) #Model10
rf_pred <- predict(rf_occtrain_1, newdata = test) #Model11

ytest <- test$Occupancy
table(bag_pred, ytest, dnn = c('Predicted', 'True')) #Model10
table(rf_pred, ytest, dnn = c('Predicted', 'True')) #Model11

#Misclassification rates
(bag_err <- mean(bag_pred != ytest)) #Model10
(rf_err <- mean(rf_pred != ytest)) #Model11

#Classification Accuracy
```

```
round((Accuracy_bag <- 100 - bag_err*100), 4) #Model10

round((Accuracy_rf <- 100 - rf_err*100), 4) #Model11

## Check answer using built in caret confusion matrix:
confusionMatrix(bag_pred, ytest) #Model10
confusionMatrix(rf_pred, ytest) #Model11
```

A-4: Gradient Boosting

```
#### 4. GRADIENT BOOSTED TREES ####

library(gbm)

# Requires 0-1 for binary classification (without caret)
train_bin <- train
train_bin$Occupancy <- as.numeric(train_bin$Occupancy) - 1

system.time(
  gbm_occtrain <- gbm(Occupancy ~ ., data = train_bin,
                      distribution = 'bernoulli', #Use bernoulli for binary
classification
                      n.trees = 10000, #B
                      interaction.depth = 2, #d
                      shrinkage = 0.005, #lambda = learning rate
                      bag.fraction = 1, #default = 0.5 for extra randomisation.
                      cv.folds = 10, #built-in CV
                      n.cores = 3, #which can be parallelised
                      verbose = F)
)
gbm_occtrain

#CV Errors per tree
best_B <- gbm.perf(gbm_occtrain, method = 'cv')

#Variable importance
summary(gbm_occtrain)

#Partial dependence
plot.gbm(gbm_occtrain, 1, best_B) #Temperature
plot.gbm(gbm_occtrain, 2, best_B) #Humidity
plot.gbm(gbm_occtrain, 3, best_B) #Light
plot.gbm(gbm_occtrain, 4, best_B) #CO2
plot.gbm(gbm_occtrain, 5, best_B) #HumidityRatio

#Grid search
set.seed(23)
ctrl <- trainControl(method = 'cv', number = 5, verboseIter = T)
gbm_grid <- expand.grid(n.trees = c(1000, 1500, 2000),
                        interaction.depth = c(1, 2, 6),
                        shrinkage = c(0.01, 0.005, 0.001),
                        n.minobsinnode = 1)

#Model12
gbm_gridsearch <- train(Occupancy ~ ., data = train,
                        method = 'gbm',
                        distribution = 'bernoulli',
                        trControl = ctrl,
                        verbose = F,
                        tuneGrid = gbm_grid)
gbm_gridsearch
confusionMatrix(gbm_gridsearch)

# Prediction for Model12
gbm_pred <- predict(gbm_gridsearch, test)
confusionMatrix(gbm_pred, test$Occupancy)

#Model13
gbm_gridsearch_1 <- train(Occupancy ~ .-Humidity-HumidityRatio, data = train,
                        method = 'gbm',
```

```
                        distribution = 'bernoulli',
                        trControl = ctrl,
                        verbose = F,
                        tuneGrid = gbm_grid)
gbm_gridsearch_1
confusionMatrix(gbm_gridsearch_1)

# Prediction for Model13
gbm_pred_1 <- predict(gbm_gridsearch_1, test)
confusionMatrix(gbm_pred_1, test$Occupancy)
```

A-5: Final Model Accuracy Using "testing.csv"

```
#### 5. EVALUATE BEST MODELS ON TESTING.CSV ####

#best model from each section, apply same test on unseen test data
#finaltest from testing.csv

# Best Logistic Regression Model
#Model7

#Load unseen test data (finaltest: 2665 observations)

finaltestX_dropped <- finaltest[ , -c(2,5,6)]
finaltestX_dropped <- as.matrix(finaltestX_dropped)
head(finaltestX_dropped)

finaltestY <- finaltest[ , -c(1:5)]
finaltestY <- as.matrix(finaltestY)
head(finaltestY)

set.seed(23)
pi_hat_cv_dropped <- predict(modelCV_dropped, newx = finaltestX_dropped,
                             s = cv_dropped$lambda.min, type = 'response')

Y_hat_cv_dropped <- ifelse(pi_hat_cv_dropped >= 0.5, "1", "0")

N_finaltest <- length(finaltestY)
Error_cv_dropped <- sum((Y_hat_cv_dropped!=finaltestY)/N_finaltest)*100  #model7
round((Accuracy_cv_dropped <- 100 - Error_cv_dropped), 4)

(conf_modcv_dropped <- cbind(table(Y_hat_cv_dropped,
                                   finaltestY, dnn = c("predict", "true")),
                             Accuracy_cv_dropped))

#ROC
```

```
# Model7 (With Lasso)

pred_cv_dropped <- prediction(pi_hat_cv_dropped, finaltestY)

perf_cv_dropped  <- performance(pred_cv_dropped, 'tpr', 'fpr')

plot(perf_cv_dropped, colorize = FALSE, col = 'black')

lines(c(0,1), c(0,1), col = 'gray', lty = 4)

performance(pred_cv_dropped, measure = 'auc')@y.values[[1]]*100


#Best Classification Tree Model

#Model9

# Predict on final test set

yhat_4_finaltest <- predict(pr_tree_4, finaltest, type = 'class') #type:
classification

(c_mat <- table(yhat_4_finaltest, finaltest$Occupancy)) #predicted vs actaul

sum(diag(c_mat))/nrow(finaltest)*100 #classification accuracy %


#Best Bagging and Random Forest Model

#Model11


rf_pred <- predict(rf_occtrain_1, newdata = finaltest) #Model11


finalytest <- finaltest$Occupancy

table(rf_pred, finalytest, dnn = c('Predicted', 'True')) #Model11


#Misclassification rates

(rf_err <- mean(rf_pred != finalytest)) #Model11


#Classification Accuracy

round((Accuracy_rf <- 100 - rf_err*100), 4) #Model11


# Verify answer using built in caret confusion matrix:

confusionMatrix(rf_pred, finalytest) #Model11


#Gradient Boosted Trees

#Model13


gbm_pred_1 <- predict(gbm_gridsearch_1, finaltest)

confusionMatrix(gbm_pred_1, finaltest$Occupancy)
```

References

Benanchenhou, D., *Occupancy-Prediction,* (2017), GitHub repository: https://github.com/DalilaR/Occupancy-Prediction

Boehmke, B. and Greenwall, B., 2020. *Hands-On Machine Learning With R*. [ebook] Available at: <https://bradleyboehmke.github.io/HOML/> [Accessed 20 June 2020].

Bruce, P. and Bruce, A., 2018. *Practical Statistics For Data Scientists*. Beijing: O'Reilly Media.

Candenado, L. and Feldheim, V., 2016. Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. *Energy and Buildings*, 112, pp.28-39.

Candenado, L., *Occupancy-detection-data*, (2016), GitHub repository: https://github.com/LuisM78/Occupancy-detection-data

Cortes, C. and Mohr, M., 2004. AUC Optimization vs. Error Rate Minimization. *AT&T Labs – Research*,.

Hastie, T., James, G., Tibshirani, R. and Witten, D. 2017. *An Introduction To Statistical Learning*. New York: Springer.

Ray, S., 2016. *A Comprehensive Guide To Data Exploration*. [online] analyticsvidhya. Available at: <https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/> [Accessed 22 June 2020].

Stats.idre.ucla.edu. 2020. *Logit Regression | R Data Analysis Examples*. [online] Available at: <https://stats.idre.ucla.edu/r/dae/logit-regression/> [Accessed 25 June 2020].