Predicting crown rot in banana shipments

Submission for Choose your Own Project Data Science: Capstone

Johannes W.H. van der Waal

4th July 2021

Contents

Abstract

Organic bananas are grown without synthetic fertilizer or chemical pesticides. This is better for the environemnt and the health of consumers and banana workers. However, it makes export banana more susceptible to post-harvest decay, especially during the long (2-4 weeks) transit time in refrigerated containers to European markets. A notable form of post-harvest decay is rot of the cutting surface of the banana cluster, known as crown rot. Professionals in the industry have observed that the incidence of crown rot is associated with the time of the year and have the experience that the use of controlled atmosphere (CA) containers (with reduced O_2 and increased CO_2 content) can reduce the development of crown rot. In this article, I present an analysis of the problem based on a curated proprietary dataset with almost 7000 real shipments of organic bananas from Latin America to Europe over a time frame of more than 4 years. The analysis is performed by collecting historic weather data from a local weather station and analyze if weather has an association with the phenomenon. I conclude that this seems to be the case. I show that CA mitigates the problem when the growing temperature sums are higher and the transit time (residence time in container) longer. I develop machine learning models that can reasonably successfully predict the incidence of crwon rot on future

shipments. This allows to make informed decisions on whether or not to deploy CA based on cost-benefit analysis. A number of classification algorithms is tested. As the dataset is imbalanced, it requires syntehtic sampling to achieve a good balance of accuracy pover three prediciton classes

Contents

Introduction

Background of the problem

Organic bananas are grown without synthetic fertilizer or chemical pesticides. This is better for the environment and the health of consumers and banana workers. However, it makes export banana more susceptible to post-harvest decay, especially during the long (2-4 weeks) transit time in refrigerated containers to European markets. A notable form of post-harvest decay is rot of the cutting surface of the banana cluster, known as crown rot. The rot is caused by a complex of patheogenic funguses and bacteria, notably *Lasiodiplodia theobromae*, *Colletotrichum musae* and a complex of *Fusarium spp.*(Alvindia and Natsuaki 2008).

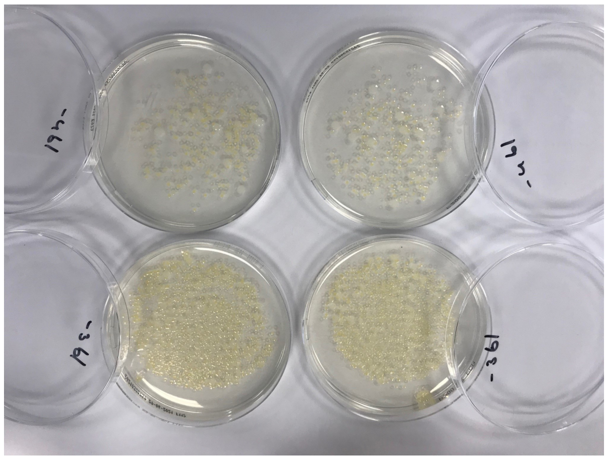


Figure 1: Crown rot causing fungi growing on petri dishes

The crown rot phenomenon causes considerable economic damage to the industry. It is a latent defect, meaning it cannot be detected during packaging. When the fruit is packed and stowed in a refrigerated container, the external appearance of the fruit is clean and green. Only during the sea voyage, under some poorly known conditions, the disease comes to development. It is only detected on arrival, when every container is subject to a rigorous quality control. The disease only develops to its fullest extent after ripening. Bananas are supposed to arrive green and unripe, and become ripe and edible only after 5-7 days ripening at slightly elevated temperature (18°C) under addition of a trace of ethylene gas, which is a plant ripening hormone. In light to moderate cases, it causes the rejection and wastage of a certain percentage of the fruit. In severe cases, it can lead to a significant reduction of the value of the cargo, and even infect healthy fruit because the affected bananas start to ripen and emit ethylene gas, which is a trigger for banana ripening. This means a loss to the producer of the fruit and also a reduction of the supply reliability to the customer. It is therefore important to detect and reduce the disease, if possible.

Professionals in the industry have observed that the incidence of crown rot is associated to a seasonal pattern and have the experience that the use of controlled atmosphere (CA) containers (with reduced O_2 and increased CO_2 content) can reduce the development of crown rot (Lassois and de Lapeyre de Bellaire 2014) (Wade, Kavanagh, and Sepiah 1993). The objective of this paper is to analyze factors associated with crown rot and its prevention or mitigation, and to develop a predictive model using machine learning algorithms.

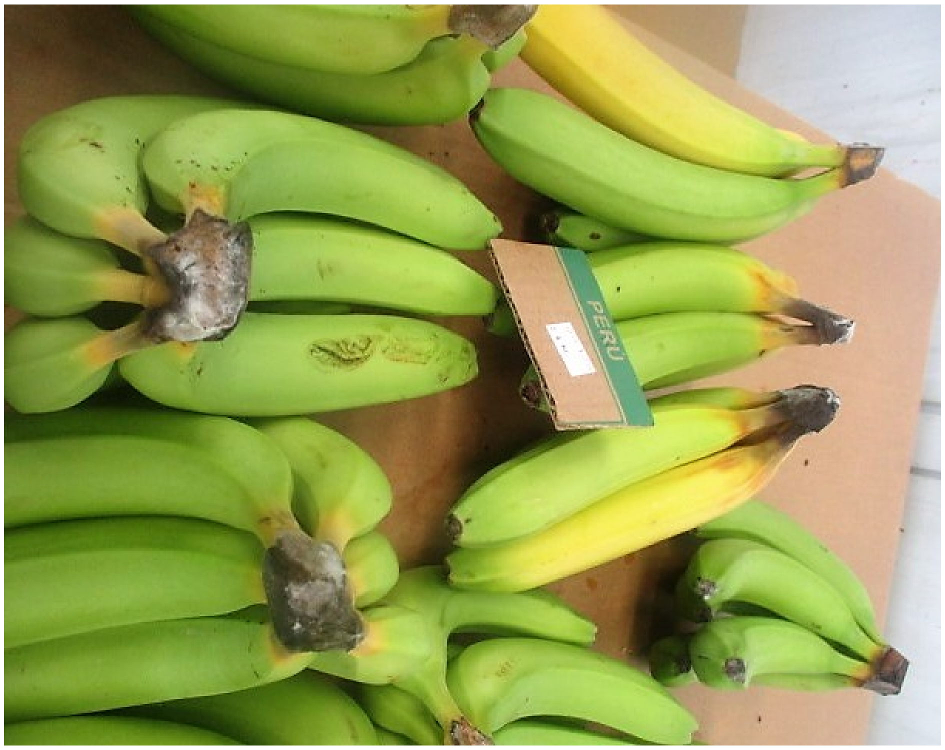


Figure 2: Crown rot on the cutting surfaces causing decay and premature ripening.

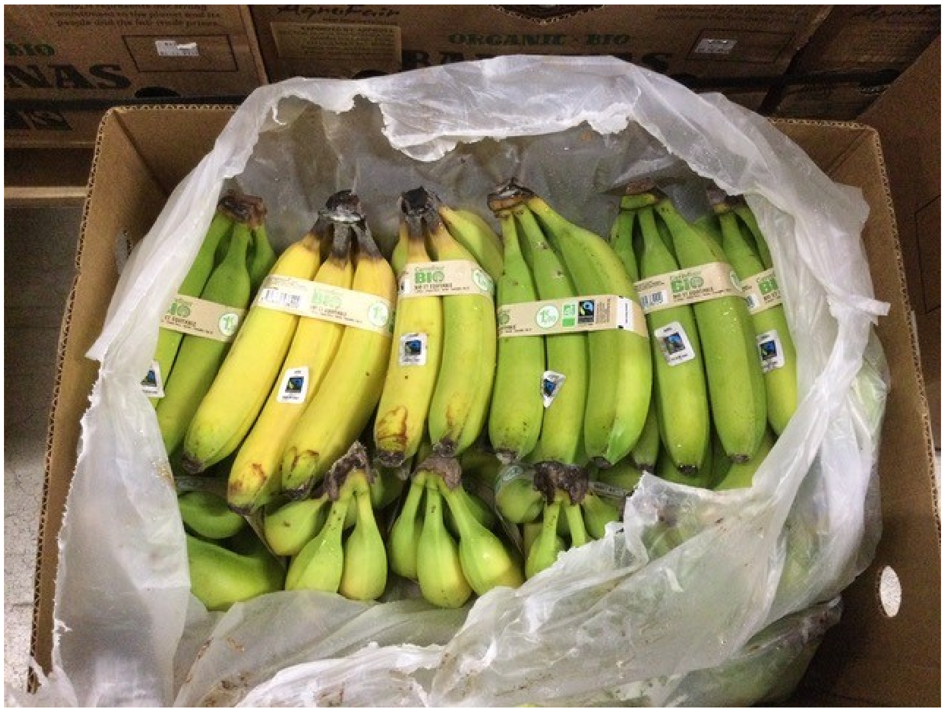


Figure 3: Box of organic bananas suffering from crown rot.

Dataset and experimental design

To achieve the objective, I use a proprietary dataset of almost 7000 data points consisting of real life quality survey data of as many container shipments of organic bananas from 2016 to the present. The raw dataset has many variables, such as producer, shipping line, port of destination. However, for the sake of simplification and anonymization, the dataset has been reduced to the most important variables, notable the packing date, the transit time and whether or not a CA container has been employed. The data is an excerpt of a larger dataset of over 20,000 rows. The data cleaning and preparation has been done outside this project, because of the proprietary nature of the data. In addition to the quality data, I need weather data. This are retrieved from meteostat.org. This is an open source platform that makes historic weather data availble for many locations.

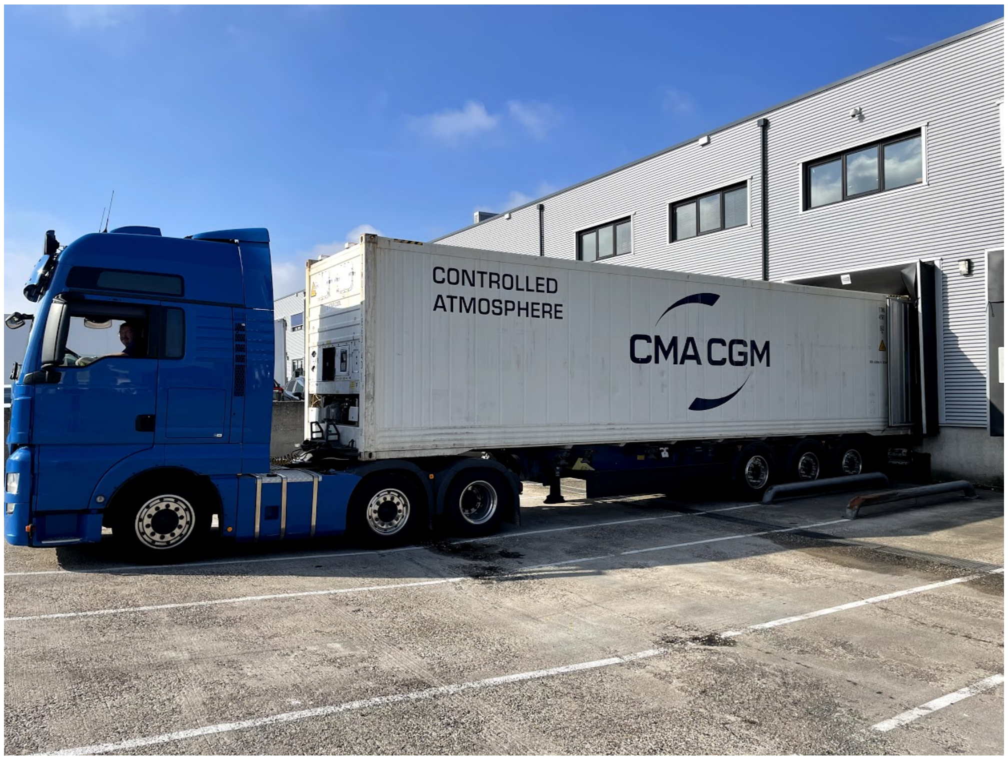


Figure 4: Controlled atmosphere container.

Analysis

Description of the dataset

```
#READING QUALITY AND SHIPMENT DATA
data <- read.csv("data.txt")
data <- data %>% select(-X) %>% mutate(dd = as.Date(dd), ca = as.factor(ca), pd = as.Date(pd))
```

The structure of the dataset is as follows:

```
summary(data)
```

```
##        dd                  ca             pd                  days
##  Min.   :2016-01-04   CA   :1440   Min.   :2015-12-08   Min.   :11.00
##  1st Qu.:2017-03-20   NO CA:5437   1st Qu.:2017-02-21   1st Qu.:26.00
##  Median :2018-07-30                Median :2018-07-02   Median :27.00
##  Mean   :2018-08-26                Mean   :2018-07-30   Mean   :27.41
##  3rd Qu.:2020-02-06                3rd Qu.:2020-01-08   3rd Qu.:29.00
##  Max.   :2021-06-28                Max.   :2021-06-04   Max.   :63.00
##        CR                 CRC                  C
##  Min.   :0.000000   Min.   :0.000000   Min.   :0.00000
##  1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.00000
##  Median :0.000000   Median :0.000000   Median :0.00000
##  Mean   :0.007565   Mean   :0.003565   Mean   :0.01469
##  3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.00800
##  Max.   :0.429907   Max.   :0.415842   Max.   :0.98020
```

ca : dummy variable indicating if consignment is carried under controlled atmosphere pd : pack date (a few days before date of departure of vessel) dd : discharge date (date of opening the container in Europe) X : a sequential number or row ID without meaning
days : the transit time, the time between pack date and date of discharge CR : the percentage of crown rot (mild) CRC : the percentage of crown rot (heavy) C : the summed and weighted compasite crown rot percentage (CR + 2 * CRC)

The table shows that there are no missing variables in the dataset. There are more pack dates than discharge dates. This can be understood, because the packing takes more time. Some days before the ship's departure are used to pack. On arrival, multiple containers are discharged on the same date. The average transit time is 27.4 days. The mean of CR and CRC are both below 1% (0.01). A composite crown rot percentage has been created attaching double weight to heavy crown rot. This is arbitrary. This data can be used as regression problem, or when crown rot percentages are transformed in classes, as a classification problem. Before developing such models, we first need to retrieve, process and attach weather data to this dataset. This is a separate task. Then we can analyze the data.

Retrieving weather data

Professionals in the industry have observed a seasonal pattern in the crown rot incidence. The following boxplot of crown rot incidence per pack date month shows that this is also the case in the data. The month of April clearly has the highest incidence.The data has been plotted for CA and non-CA separately and seems to indicate that there is a marked difference in incidence.

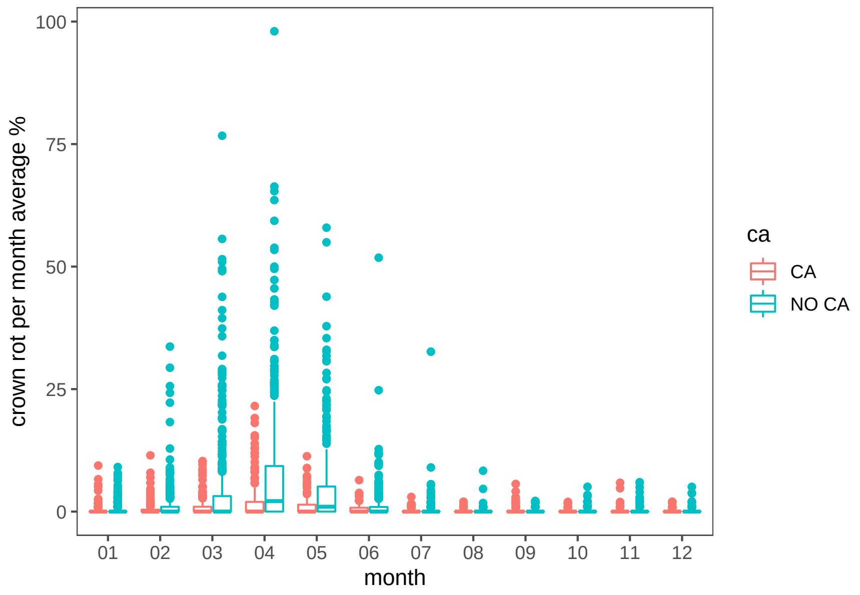


Figure 5: Boxplot of crown rot incidence per month

The seasonal pattern points in the direction of a weather related influence. For this reason, it is necessary to collect historic weather data. This can be retrieved from the application programming interface (API) of meteostat.org. The weather station closest to the banana plantations was station 84401 (Piura in Peru). The following code downloads the weather data from the API. The variable structure is appended from the "weather-structure."csv" file that I had constructed. Finally, only the date and temperature columns are retained. The precipitation field had been desirable, but this contained too many missing values to be useful. Moreover, the climate of Piura is very dry, with only 52 mm of rainfall per year. The water requirements of bananas are met by irrigation. Besides, the literature indicates that rainfall or humidity do not play a significant role (Lassois et al. 2010).

```r
#create a function for reading weather from meteostat API:

read_weather <- function(station){
  url <- paste("https://bulk.meteostat.net/hourly/",station,".csv.gz", sep= "")
  tempdir()
  tmp <- tempfile()
  curl_download(url, tmp)
  download.file(url, destfile = paste(station, ".wd", sep=""))
}

#define weather station of choice: 84401
station <- 84401


#read the weather from meteostat:84401
read_weather(station) #this function downloads the data in compressed form

#collect the data structure of weather report
names <- read.csv("weather-structure.csv", header=FALSE, sep=";")
vars <- unlist(names$V2)
type <-unlist(names$V3)

weather <- read.csv(paste(station, ".wd", sep = "")) #read and decompress the weather file
names(weather) <- vars # attach understandable variable names to the columns

# select the variables to be used and mutate some new variables

W_PE <- weather %>% select(date, temp) %>%
  mutate(date = as.Date(date))

summary(W_PE)
```

```
##       date                 temp
##  Min.   :1957-07-01   Min.   :-6.70
##  1st Qu.:1979-04-24   1st Qu.:20.70
##  Median :1988-05-22   Median :24.00
##  Mean   :1992-12-14   Mean   :24.17
##  3rd Qu.:2009-08-09   3rd Qu.:27.20
##  Max.   :2021-07-11   Max.   :43.00
##                       NA's   :1456
```

##Processing weather data

Next, I plot the weather data after I have calculated the mean temperatures per day (they are recorded 4-12 times per day).

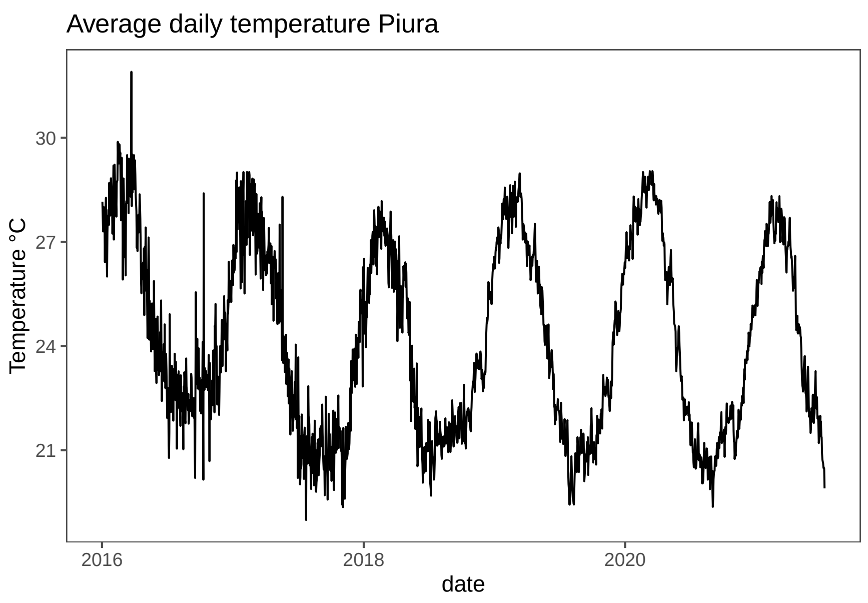


Figure 6: A plot of the mean temperatures per date for Piura, Peru

It is customary in agriculture to relate the development of crops to temperature sums: the temperature sum is the integral of the mean daily temperature above a certain crop-specific floor temperature over the time. The temperature sum is expressed as degree.days. In the case of bananas, the floor temperature is 13.5 °C. This is the temperature below which the banana plant does not grow. Not accidentally, it is also the temperature at which bananas are carried in refrigerated containers. At this temperature, the plant metabolism is reduced to the minimum. Below this temperature, the plant will die, viz. the bananas will be damaged by cold shock (Chillet et al. 2006), (Ganry and Meyer 1975). The degree.days are a parameter that is used to determine when a crop is ready to be harvested. In the banana industry in Peru, this is not done in this way, especially because the water supply by irrigation is a limiting factor to growth. Normally, bananas are harvested 10-12 weeks after flowering. For that reason I calculate the accumulated degree.days for a set time span of 10 weeks for every single pack date. This could also be 11 or 12, but this is not expected to change much. The results are plotted in a line plot, which shows that the rolling summation has a smoothing effect.The plot now indicates how many degree.days a bunch of bananas has received after a 10 week growing cycle.

```
#formula to calculate the cumulative sums based on span and cut-off of 13.5°C
span <- 10
summ_add_PE <- summ_PE %>% filter(date >=  (as.Date("2015-01-01")-span*7)) %>%
  mutate(grow_temp = (mean_temp - 13.5), cdd = cumsum(grow_temp)) %>% #the grow temperature is the temp
  mutate(add = cdd - lag(cdd, span*7)) %>% # the accumulated degree.days or temperature sum is calculat
  na.exclude() #exclude rows with NaN
```

The next step is now to join the calculated accumulated degree days to the quality dataset.

```
# join the quality data to the weather data to have a list of quality data with ADDs for every shipment
data_PE <- data %>% left_join(summ_add_PE, by = c('pd' = 'date'))
```

We can now proceed to the exploratory data analysis.

Exploratory data analysis

Now that we have accumulated degree.days, it is possible to plot the incidence of crown rot against the degree.days and see if there is any association between the two.

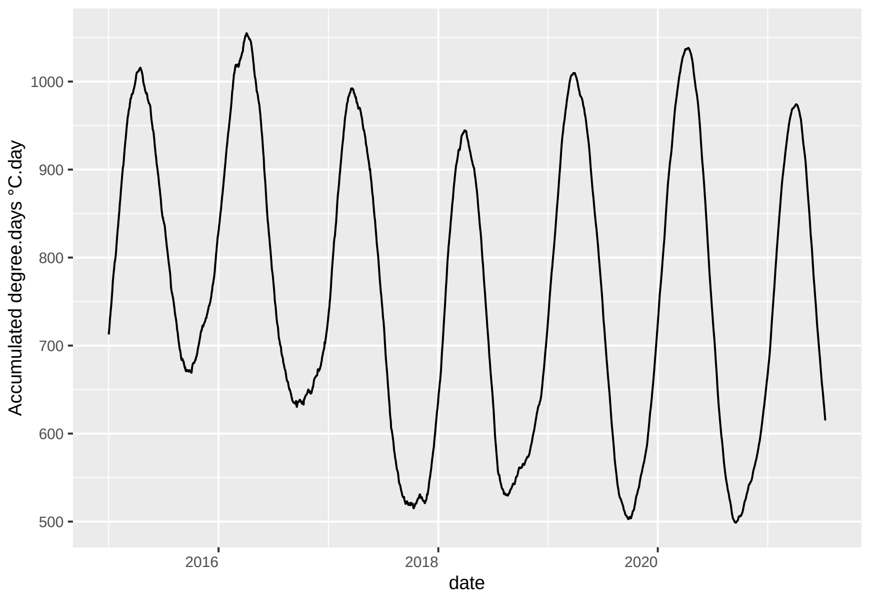


Figure 7: Plot of accumulated degree.days per date for Piura, Peru.

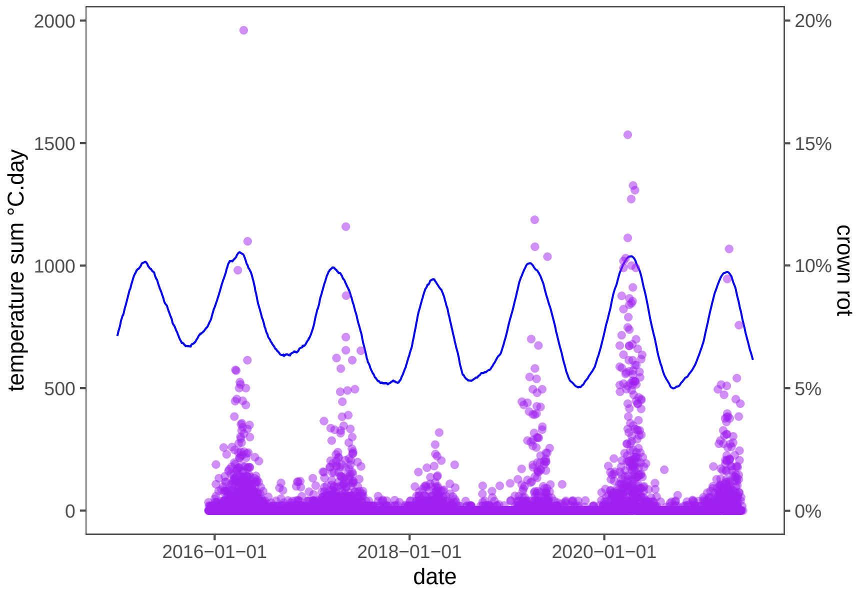


Figure 8: Accumulated degree days and crown rot.

The plot clearly shows that the crown rot incidence follows the weather pattern. Moer particularly, it increases with rising temperature sums. Apparently, the fruit becomes more vulnerable to fungal attack at the cutting faces when it has had more growing days. Perhaps this means that the physiological age of the fruit is older, in spite of the fact that the thickness of the fruit is not bigger (this conclusion cannot be drawn from the data though).

Now I want to explore any association between crown rot, degree.days and the use of CA containers. For that purpose, I plot the data in a heat map. The transit times and the degree.days are gatherd in bins.

```
#make a heatmap of crown rot versus transit time (days) and ADD
#define bins
lbl_days <- c("11-20", "21-30", "31-40", "41-50", "51-60", "61-70")

lbl_add <- c("401-500", "501-600", "601-700", "701-800", "801-900",
             "901-1000", "1001-1100","1101-1200")

#compute averages per bin
data_hm_PE <- data_PE %>% drop_na(add) %>% mutate(day_bins = cut(days, breaks = seq(10, 70, 10), includ
         add_bins = cut(add, breaks = seq(400,1200, 100), include.lowest = FALSE, label = lbl_add)) %>%
  group_by(day_bins, add_bins, ca) %>%
  dplyr::summarize(CR_avg = mean(C), n = n())
```

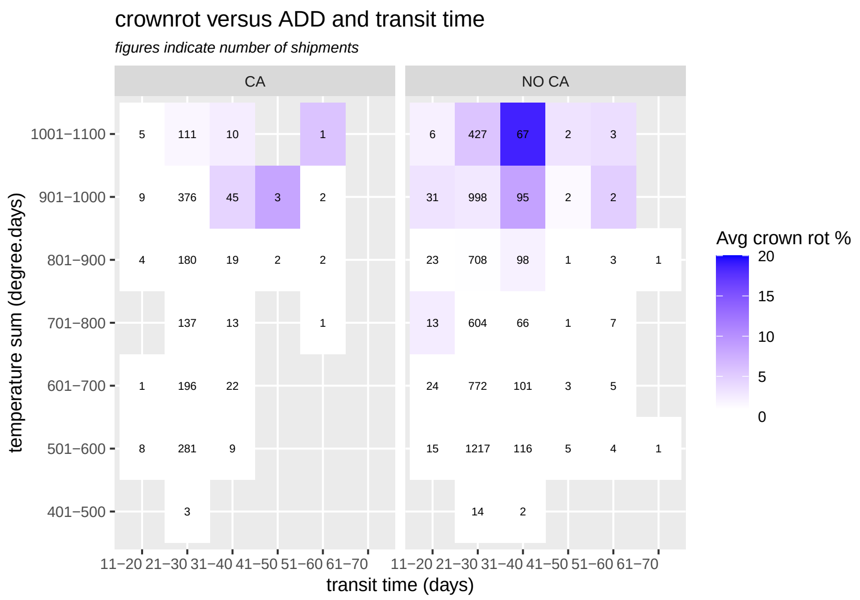


Figure 9: A heatmap of crown rot against ADD and transit time(days).

The plot shows the average crown rot percentage per bin, split over with and without CA. The numbers indicate the number of shipments in the bin. The figures shows nicely that crown rot incidence increases with a degree.days and transit time increasing in combination. The use of CA seems to have a mitigating influence on the development of crown rot. To analyse these correlations more formally, we make a correlation matrix of the variables.

```
#explore correlation between variables

#make a correlation matrix
source("corstars.R") #this retrieves the function created by ....
```

Table 1: Pearson correlation matrix

	C	add	days
C			
add	0.31****		
days	0.11****	0.02	
CA	0.07****	-0.06****	0.00

```
data_PE %>% select(C, add, days, ca) %>% mutate(CA = as.numeric(ca)) %>% select(-ca) %>%
  drop_na() %>% corstars(.) %>% knitr::kable(caption = "Pearson correlation matrix")
```

The Pearson correlation table shows that there are significant correlations between crown rot percentage and degree.days, transit time and use of CA, but that these are generally weak (<0.5).

Before planning a machine learning approach, I first convert the crown rot percentage to a class variable. It is more practical to know if the risk of developing crown rot is low, medium or high, than to predict a percentage. For professionals, a low risk means no action, a medium or high risk could mean that it is sensible to apply a CA container or be prepared for crown rot damage. I use therefore two aforementioned classes, with cut-offs at 0.5% for medium and 5% for high. The following code creates the data for the machine learning. The table produces descriptive statistics.

```
#Create a categorical variable for CR risk"(low/medium/high:
data_ML <- data_PE %>% select(ca, days, C, add, pd) %>%
  mutate(C_risk = cut(C, breaks = c(0,0.005, 0.05, 1),
                      include.lowest = TRUE,
                      labels = c("low", "medium",  "high")),
         ca = as.factor(ca)) %>%
  select(-C)

#make a descriptive statistics table to inspect the data
library(skimr)
summary(data_ML)
```

```
##      ca            days             add                pd            
##  CA    :1440   Min.   :11.00   Min.   : 498.7   Min.   :2015-12-08  
##  NO CA:5437   1st Qu.:26.00   1st Qu.: 604.4   1st Qu.:2017-02-21  
##               Median :27.00   Median : 774.1   Median :2018-07-02  
##               Mean   :27.41   Mean   : 770.2   Mean   :2018-07-30  
##               3rd Qu.:29.00   3rd Qu.: 934.9   3rd Qu.:2020-01-08  
##               Max.   :63.00   Max.   :1055.0   Max.   :2021-06-04  
##    C_risk    
##  low   :5137  
##  medium:1236  
##  high  : 504  
##
##
##
```

The table shows us some important information: First there are no missing observations. Second, the dataset is imbalanced in the classes. That means that there are many “no” observations, and fewer “medium” and “high” observations. This influences the machine learning results as we shall see.

Machine learning results

Data partitioning

I partition the data in three sets: a train and a test set and a final hold-out validation set. The validation set contains the data for 2021. The previous years serve as train and test set, with the aim of doing meaningful predictions on the current season.

```
#splitting the data in a train and a test set and a hold-out set for validation
validation <- data_ML %>% filter(pd >= "2021-01-01") %>% select(-pd) %>% mutate(ca = as.integer(ca))
traintest <- data_ML %>% filter(pd < "2021-01-01") %>% drop_na() %>% select(-pd) %>% mutate(ca = as.inte

#create a data partition
set.seed(1, sample.kind = "Rounding")
trainIndex <- createDataPartition(traintest$C_risk, p = .5,
                                  list = FALSE,
                                  times = 1)

train <- traintest[trainIndex,]
test <- traintest[-trainIndex,]
```

##Pre-processing and tune controls The data are automatically scaled and centered by using the "preProcess = c("center","scale")" feature of the train command in the caret package. The tune control features of caret are used to automatically determine the optimal tuning setting, such as the k for the k nearest neighbours, the mtry for the random forest and the complexity parameter to prune the trees in the decision tree.

A first machine learning test model

With this data, I try a first machine learning model to get an idea of the performance. This is a recursive partitioning decision tree model, which also allows to nicely visualize the model.The complexity parameter cp is set in such a way that the model is pruned to just a few branches, to ensure a sufficiently abstract model and avoid overfitting. Automatic 10-time cross-validation is applied.

```
# train a decision tree for test
library(rpart)
set.seed(2, sample.kind = "Rounding")
model1 <- train(C_risk ~ ., data = train,
                method = "rpart",
                preProcess = c("center", "scale"),
                tuneLength = 10,
                trControl = trainControl(method = "cv"))

y_hat <- predict(model1, newdata = test)
cfm_r <- confusionMatrix(y_hat, test$C_risk, dnn = c("Prediction", "Reference"))
cfm_r
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction   low medium high
##     low     2342    486  137
##     medium    24      9    4
##     high      37     49   75
##
## Overall Statistics
##
##                Accuracy : 0.767
##                  95% CI : (0.7519, 0.7816)
##     No Information Rate : 0.7597
##     P-Value [Acc > NIR] : 0.1747
##
##                   Kappa : 0.1748
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: low Class: medium Class: high
## Sensitivity              0.9746      0.016544     0.34722
## Specificity              0.1803      0.989309     0.97082
## Pos Pred Value           0.7899      0.243243     0.46584
## Neg Pred Value           0.6919      0.828855     0.95303
## Prevalence               0.7597      0.171989     0.06829
## Detection Rate           0.7404      0.002845     0.02371
## Detection Prevalence     0.9374      0.011698     0.05090
## Balanced Accuracy        0.5774      0.502927     0.65902
```

```
#visualize the decision tree
library(rpart.plot)
set.seed(3, sample.kind = "Rounding")
model.rpart1 <- rpart(C_risk ~ ., data = train, method = "class",control = rpart.control(cp = 0.01))
rpart.plot(model.rpart1)
```

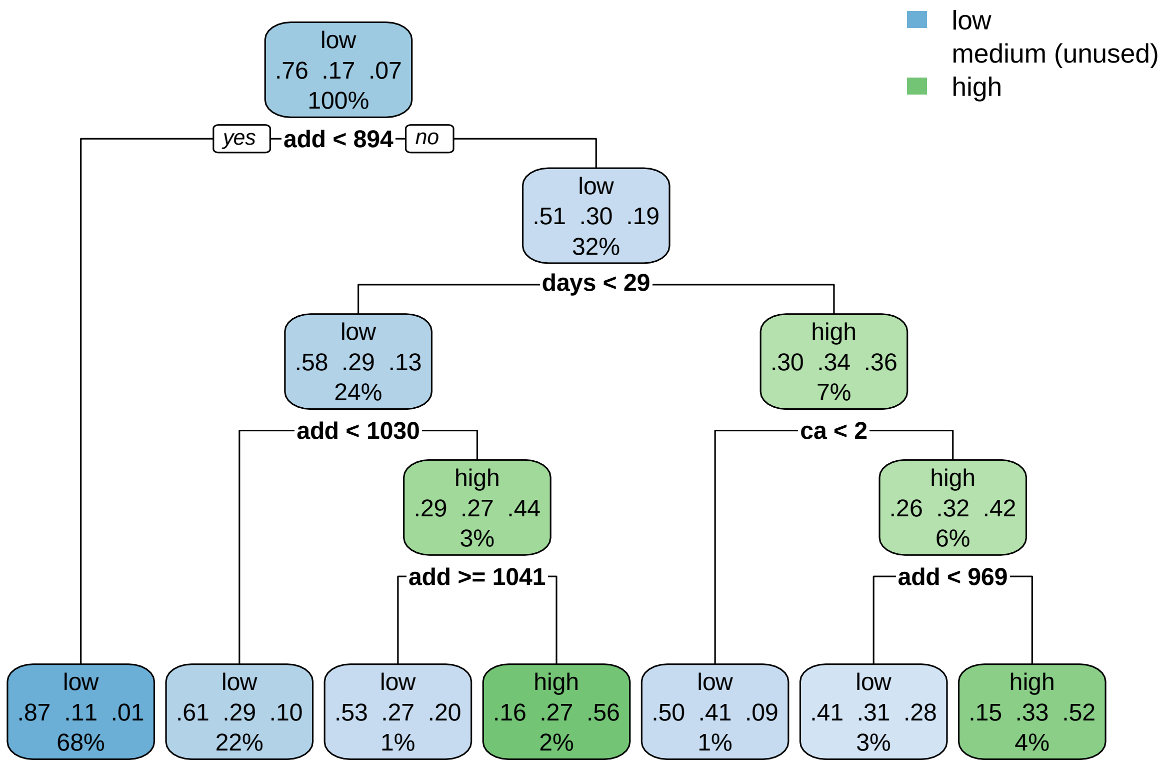


Figure 10: Decision tree.

The contingency table shows that the prediction has an accuracy of .The table shows however that the sensitivity is high, in other words, it is capable of predicting true positives ("low") well, but the sensitivity is low, it is not good at predicting true negatives ("medium"'s and "high"'s). It is especially not good at predicting "medium"'s. The decision tree is informative: it gives the decision rules for deciding the probability of an observation falling in one of the classes. The probability of predicting lows is good, with good sensitivity (TP/(TP+FN)), but there are too many missed mediums and highs or false negatives.The specificity is high (TP / (TN + FP)). Missing these is costly.A reason for the poor predictive capability of the positives (the mediums and highs) is that the dataset is imbalanced. There are many "low" values (majority class) and few "medium" and "high" (minority class) and this biases the algorithm towards predicting lows.

Dealing with an imbalanced dataset

To addres the imbalance, I will make use of synthetic minority oversampling technique (SMOTE) (Chawla et al. 2002). It means that the algorithm artificially generates more minority observations using a distance-based algorithm. It can also randomly reduce majority class observations, to produce a balanced dataset where all classes have equal proportions. I use the smotefamily package in R on the train dataset. I reduce the majority class by 0.5 and increase the minority classes by 2 and 7 respectively.

```
library(UBL)
train.smote <- SmoteClassif(C_risk ~ ., train, list("low"=0.5, "medium" = 2, "high"=7), k = 5, repl = FA
             dist = "Euclidean")

#the table show the dataset is now balanced over the classes (using under and oversampling)
table(train.smote$C_risk)
```

```
##
##    low medium   high
##   1202   1088   1512
```

```
summary(train.smote)
```

```
##        ca             days           add            C_risk
##  Min.   :1.000   Min.   :15.00   Min.   : 498.8   low   :1202
##  1st Qu.:2.000   1st Qu.:26.00   1st Qu.: 729.9   medium:1088
##  Median :2.000   Median :27.79   Median : 926.1   high  :1512
##  Mean   :1.834   Mean   :27.94   Mean   : 862.2
##  3rd Qu.:2.000   3rd Qu.:29.00   3rd Qu.: 995.1
##  Max.   :2.000   Max.   :60.00   Max.   :1055.0
```

Testing the balanced dataset

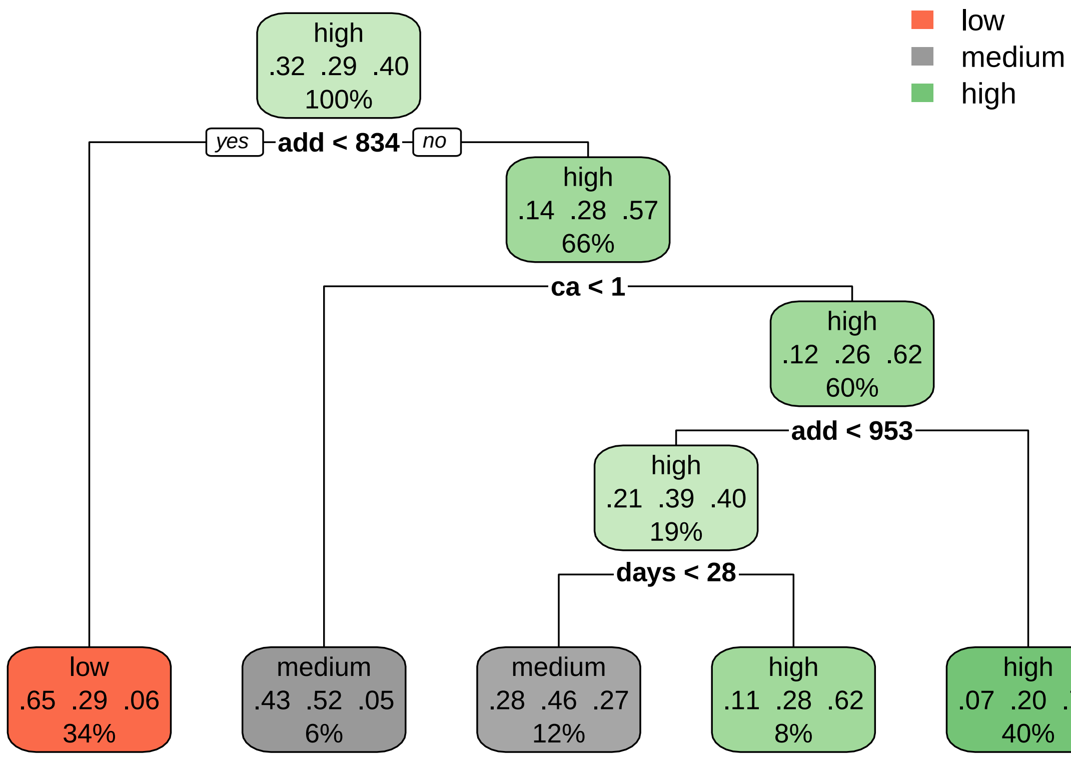
The SMOTE-balanced dataset is now used to train the decision tree again and display the decision tree.

```
# test the smoted dataset on the decision tree:
model_rpart <- train(C_risk ~ ., data = train.smote,
                method = "rpart",
                preProcess = c("center", "scale"),
                tuneLength = 10,
                trControl = trainControl(method = "cv"))

y_hat_rpart <- predict(model_rpart, newdata = test)
cfm_rpart <- confusionMatrix(y_hat, test$C_risk, dnn = c("Prediction", "Reference"))
cfm_rpart
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  low medium high
##     low    2342    486  137
##     medium   24      9    4
##     high     37     49   75
##
## Overall Statistics
##
##                Accuracy : 0.767
##                  95% CI : (0.7519, 0.7816)
##     No Information Rate : 0.7597
##     P-Value [Acc > NIR] : 0.1747
##
##                   Kappa : 0.1748
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: low Class: medium Class: high
## Sensitivity              0.9746      0.016544     0.34722
## Specificity              0.1803      0.989309     0.97082
## Pos Pred Value           0.7899      0.243243     0.46584
## Neg Pred Value           0.6919      0.828855     0.95303
## Prevalence               0.7597      0.171989     0.06829
## Detection Rate           0.7404      0.002845     0.02371
## Detection Prevalence     0.9374      0.011698     0.05090
## Balanced Accuracy        0.5774      0.502927     0.65902
```

```r
#we can visualize the decision tree
library(rpart.plot)
#we can use priors of the distribution to improve the model and define minimum splits to reduce detail
model.rpart2 <- rpart(C_risk ~ ., data = train.smote, method = "class", parms = list(split = "ca"),
                      control = rpart.control(minsplit = 50))
rpart.plot(model.rpart2)
```



The confusion matrix of the algorithm applied to the balanced dataset shows that the accuracy was reduced from 0.7669934 to 0.7669934, but that the kappa value changed from `cfm_r$overall[2]` to 0.1747537. The kappa value is a measure of how well the algorithm is better capable of indicating the performance of the algorithm. It removes the possibility of the algorithm making a correct random guess.

The decision tree gives interesting information about the probabilities of an observation ending up in one of the classes, depending on decision parameters, such as accumulated degreedays, transit time and use of CA. It shows that the use of CA tends to reduce a high risk to a medium or low risk in situations where the accumulated degreedays are higher than 968 and the transit time is less than 30 days. This conforms to the heatmap of Figure @ref(fig:heatmap). ## Various classification algorithms algorithms

With the synthetic balanced dataset, I will now experiment with a number of classification algorithms and compare these with each other.

The models take a while to calculate on an average laptop computer. After the calculation, the results are collected in a dataframe.

```
# make a table of the results of the models

results_models <- data.frame(cfm_knn$overall, cfm_mlp$overall, cfm_nnet$overall,
                             cfm_rda$overall, cfm_rf$overall, cfm_rpart$overall,
                             cfm_svm$overall, cfm_treebag$overall, cfm_glmnet$overall)
names <- c("k-nearest neighbors", "multilayer perceptron", "neural network", "regularized discriminant a
           "recursive partitioning & regression trees", "support vector machine", "bootstrap aggregated
results_models<- results_models[1:2,]
colnames(results_models) <- names

df <- cbind(parameter = rownames(results_models), as_tibble(results_models)) %>%
  pivot_longer(cols = -"parameter", names_to = c("algorithm"), values_to = "value")
```

The results are plotted in two graphs, one for Accuracy and one for the kappa value.

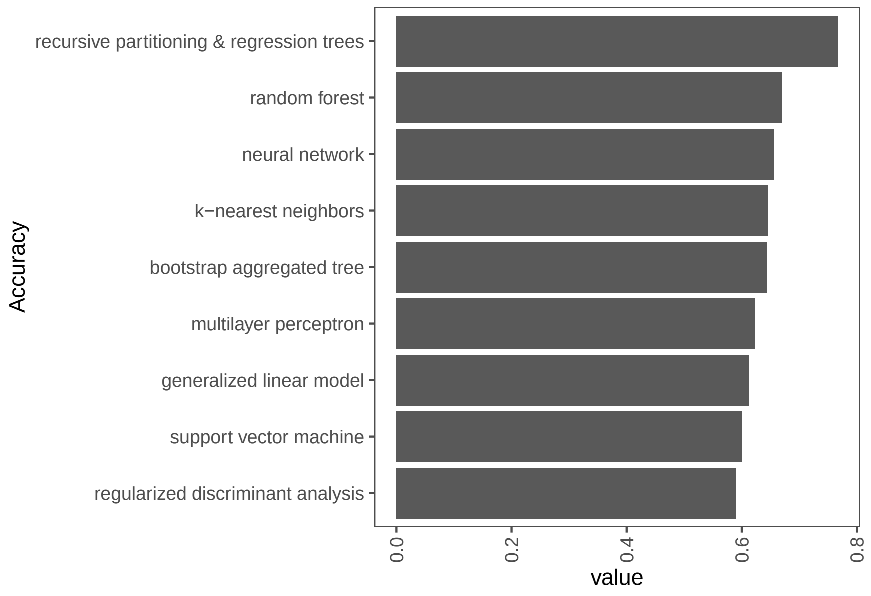


Figure 11: Results of classification algorithms: Accuracy

Figures @ref(fig:result_plot1) and @ref(fig:result_plot2) show that different algorithms have different performance. The accuracy is generally not impressive. This is in part due to the choice for a three-class dependent variable. It is also due to incomplete predictors, introducing quite some residual error into the model. In the extended dataset, there are more predictor variables, like producer, measured fruit size, shipping line and arrival temperature. In some cases, it is known that an instance classified as "CA" is in fact a non-CA when the CA feature was ordered, but had not functioned properly.

##Final hold-out test.

The validation sample that was kept behind, is now used for the final hold-out test on the best performing algorithm (judged by kappa).

```
# final hold out test on validation sample with neural network
#
y_hat_nnet_v <- predict(model_nnet, newdata = validation)
cfm_nnet_v <- confusionMatrix(y_hat_nnet_v, validation$C_risk, dnn = c("Prediction", "Reference"))
cfm_nnet_v
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low medium high
##     low    167     22    4
##     medium 105     63   14
##     high    58     63   54
##
## Overall Statistics
##
##                Accuracy : 0.5164
##                  95% CI : (0.4737, 0.5589)
##     No Information Rate : 0.6
```

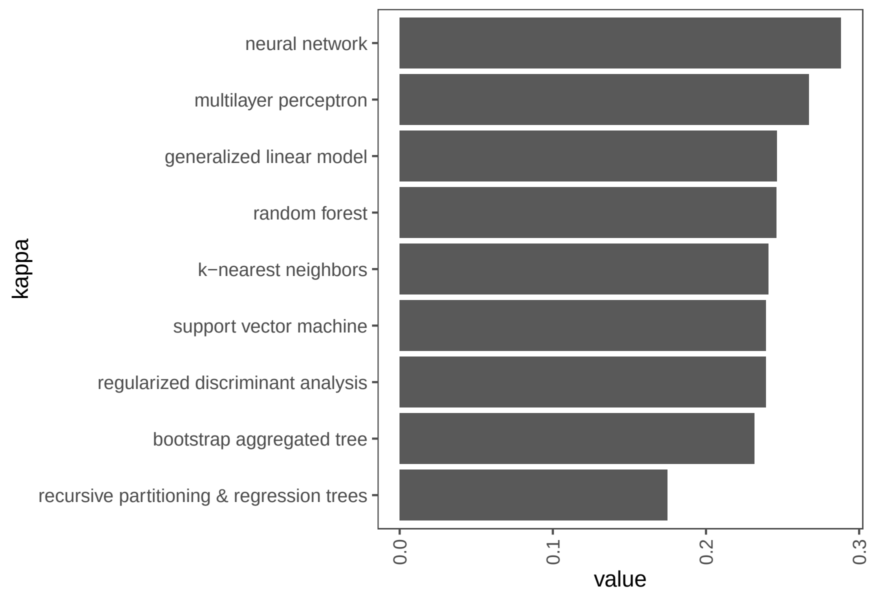


Figure 12: Results of classification algorithms: Kappa

```
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2658
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: low Class: medium Class: high
## Sensitivity              0.5061        0.4257     0.75000
## Specificity              0.8818        0.7040     0.74686
## Pos Pred Value           0.8653        0.3462     0.30857
## Neg Pred Value           0.5434        0.7690     0.95200
## Prevalence               0.6000        0.2691     0.13091
## Detection Rate           0.3036        0.1145     0.09818
## Detection Prevalence     0.3509        0.3309     0.31818
## Balanced Accuracy        0.6939        0.5648     0.74843
```

```
confusion_matrix <- as.data.frame(table(pred = y_hat_nnet_v, ref = validation$C_risk))
```

The follwing graph visualises the confusion matrix:

Figure @ref(fig:cfm) visualises the confusion matrix. The model on a real sample achieves an accuracy of 0.5163636 and a kappa value of 0.265835. This is perhaps not impressive, but better than a random guess with an estimated accuracy of about 0.33 (3 classes). Besides, it is important that highs are as much as possible classified as highs, and not as lows, but if it is classified as a medium, it is less problematic. Of course, one could argue, that in that case it would be more logical to design it as a two-class problem. However, I have chosen for this approach to achieve a balance between reasonable accuracy of predicting mediums and highs, and not missing too many, as they can come with high costs, on the one hand, as well as not classifying lows incorrectly as medium or high and having to bear the additional cost of 1000 USD per container for operating under CA. If the economic relation between medium and high CR incidence was clearer, this could help inform such a decision.

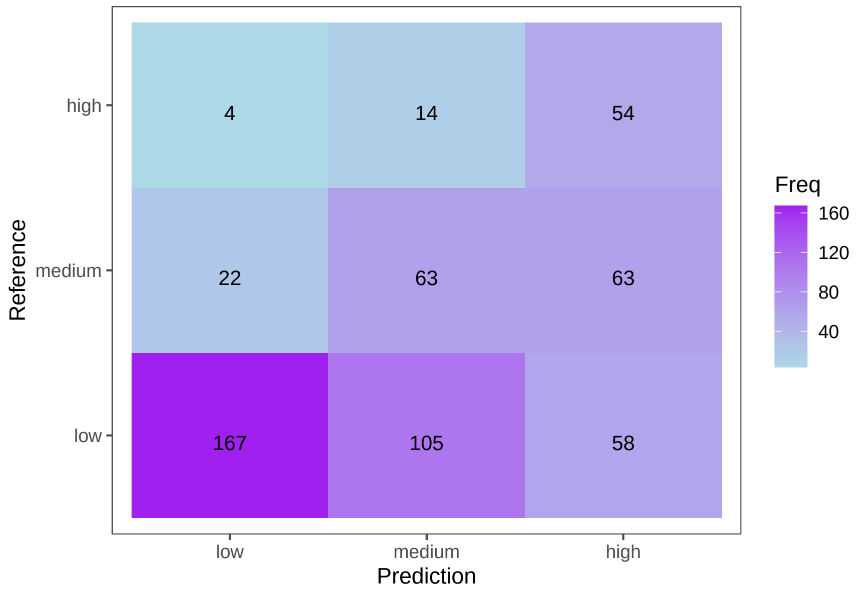


Figure 13: Heatmap of confusion matrix on perceptron model on final hold-out sample.

Finaly, it is interesting to know if the choice for balancing the data really results in a better prediction when using the selected algorithm. I do this by training a model on the imbalanced dataset, and applying to the validation set.

The confusion matrix shows that indeed, using the unbalanced dataset results in the feared effect: a high accuracy on the low class, but a very poor accuracy on the medium and high classes.

Conclusion

In this report, I have analyzed the incidence of the phytosanitary problem of crown rot on organic bananas. This problem has a seasonal appearance, and seems to be related to the weather, especially the temperature and the duration of transport in refrigerated containers. The use of a controlled atmosphere is also of influence. For the analysis, I used a proprietary dataset of 6327 instances of real banana shipments over a few years. A visual exploration of the data showed that there was indeed an association of crown rot with temperature or rather accumulated degree days, a measure often used in agriculture to calculate the physiological development of crops. The dataset was split into three classes, defining low, medium and high incidence. The dataset was imbalanced, in that there were many more instances of low than of the other classes. This makes training a model difficult, in that it biases the model towards the most frequent class. The accuracy for this class is then high, as well as the overall accuracy, but the model fails then to predict the instances that are really to be avoided and need to be correctly predicted, without predicting too many false positives on the low class. Using synthetic over and under sampling (SMOTE) I correct the imbalance in the dataset. This results in a better functioning prediction. I then test a number of classification algorithms, and select the one that has a good balance between overall accuracy and accuracy on predicting the high and medium classes, as expressed by the kappa ratio. Finally, I apply the selected model to the validation set - real consignments shipped since the beginning of this year - and find an acceptable predictive accuracy. A last test on the imbalance dataset reveals that indeed balancing is necessary to achieve useful results. The model could be used in real life for employing CA transport for at-risk consignments, or any other meaningful interventions, thus reducing costs for small banana producers. Better weather data, e.g. with rainfall and humidity data, and more predictive variables, could enhance the model. The approach could also be applied to other quality problems in the banana industry and to other crops.

References

Alvindia, Dionisio G., and Keiko T. Natsuaki. 2008. “Evaluation of fungal epiphytes isolated from banana fruit surfaces for biocontrol of banana crown rot disease.” *Crop Protection* 27 (8): 1200–1207. https://doi.org/10.1016/j.cropro.2008.02.007.

Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. “SMOTE: Synthetic Minority Over-sampling Technique.” *Journal of Artificial Intelligence Research* 16 (2): 321–57. https://doi.org/10.1002/eap.2043.

Chillet, M, O Hubert, M J Rives, and L de Lapeyre de Bellaire. 2006. “Effects of the physiological age of bananas on their susceptibility to wound anthracnose due to Colletotrichum musae.” *Plant Disease* 90 (9): 1181–85.

Ganry, Jacky, and Jean-Paul Meyer. 1975. “Recherche d’une loi d’action de la température sur la croissance des fruits du bananier.” *Fruits* 30 (6): 375–92.

Lassois, Ludivine, and Luc de Lapeyre de Bellaire. 2014. “Crown rot disease of bananas.” In *Postharvest Decay*, 103–30. Academic Press. https://doi.org/10.1016/B978-0-12-411552-1.00003-X.

Lassois, Ludivine, M Haïssam Jijakli, Marc Chillet, and Luc de Lapeyre de Bellaire. 2010. “Crown Rot of Bananas: Preharvest Factors Involved in Postharvest Disease Development and Integrated Control Methods.” *Plant Disease* 94 (6): 648–58. http://10.0.4.70/PDIS-94-6-0648%20https://search.ebscohost.com/login.aspx?direct=true&db=afh&AN=50936225&site=ehost-live.

Wade, N. L., E. E. Kavanagh, and M. Sepiah. 1993. “Effects of modified atmosphere storage on banana postharvest diseases and the control of bunch main-stalk rot.” *Postharvest Biology and Technology* 3 (2): 143–54. https://doi.org/10.1016/0925-5214(93)90006-O.