Report: Tree Shadows

Massimo Palme

21/6/2020

1. INTRODUCTION

This work applies machine learning to an urban planning problem: the decision of where plant trees in a green infrastructure project considering the need to shadow buildings and reduce the energy needs for cooling.

Green infrastructure provides many benefits to the urban environment and to its inhabitants: capture CO_2, reduce noise and air pollution, release O_2, regulate the urban climate through evotranspiration processes, offer to users open spaces where they can walk, make sports, etcetera. Within the benefits provided by vegetation in cities, shadows produced by trees on buildings are quite important: many authors estimated cooling need reductions in summer time of 20-40% for buildings with façades protected by trees.

There are many theoretical studies on the shadow effect on cooling needs and consequent energy consumption of buildings; however the concrete application of these studies' results on real cases is still an emerging field of urban planning practice. Urban planners have to take decisions based on many different constraints, like the availability of space around buildings of various shapes, heights, and orientations. Moreover, they should assure the continuity of the Green Infrastructure (for example, to follow sidewalks or bike lanes). So, it will be very useful for urban planners a tool for supporting their decision process.

Such a tool should indicate an estimate of energy saving that could be obtained by planting a row of trees under the boundary conditions they can observe in a visit to the place: orientations, distance from building, building shape, type of tree to be planted.

To estimate the energy saving, a very common way is to conduct a building performance simulation. An expert of building performance simulation constructs a model for the building, then insert the boundary conditions and the climate files, to finally obtain a cooling need in Joules. By comparing the cases with and without the trees the energy saving can be obtained.

The problem is that such process is very long and involves the expert work of a building energy modeler. So, what if a machine learning process could predict the estimate of energy saving based on certain number of cases previously modeled?

If this is the case, a building performance simulation process conducted on a sample of cases could then be used to train a machine-learning algorithm. The algorithm will be a rapid and easy way to predict the energy saving for new urban configurations placed in similar climates.

In this work, a set of previously conducted simulations (123 cases) will be used to develop a machine-learning strategy and test the accuracy of predictions. Figures 1 and 2 show respectively some examples of concrete urban configurations and the geometrical interpretation of the relation between the building's façade and a tree.

I will not explain here the details of the building performance simulation model I designed (it is quite complicated and involves knowledge of building physics and specific simulation tool skills); I will just present the result of the simulations. Figure 3 show the energy saving grouped by trees types and averaged by building shapes. We can see that the average saving is around 20% but some cases present a very little saving (less than 5%) and other a very good saving (up to 50%).

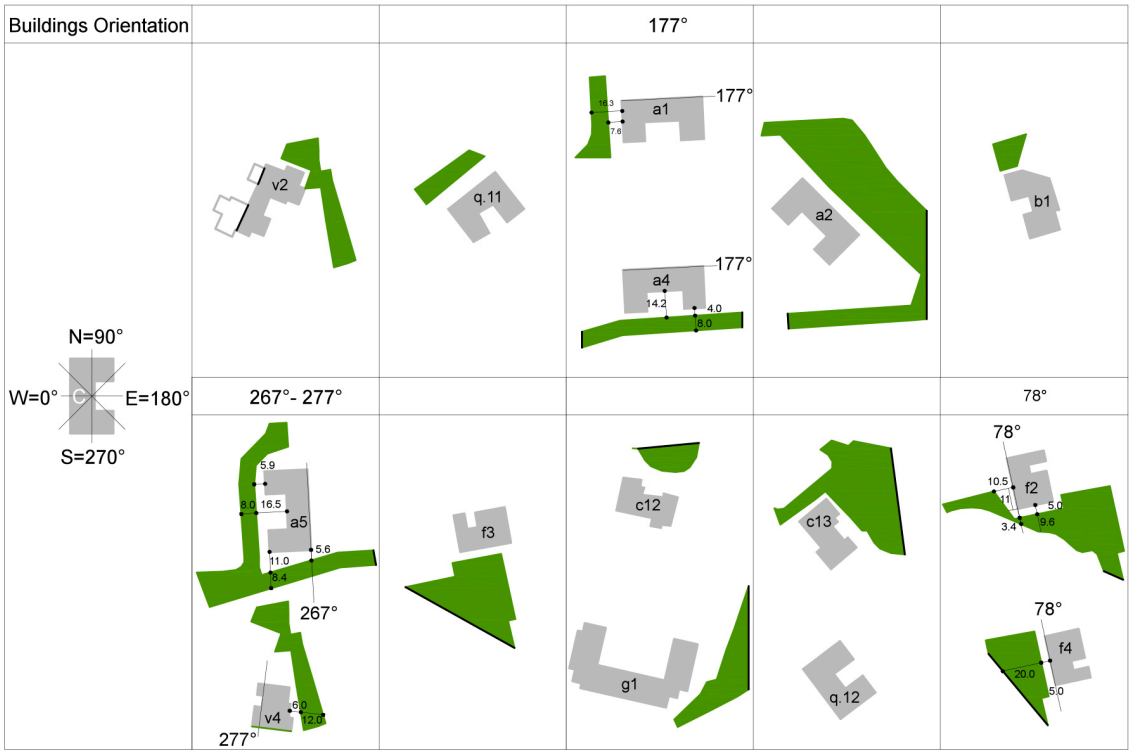

Figure 1: some of the configurations simulated

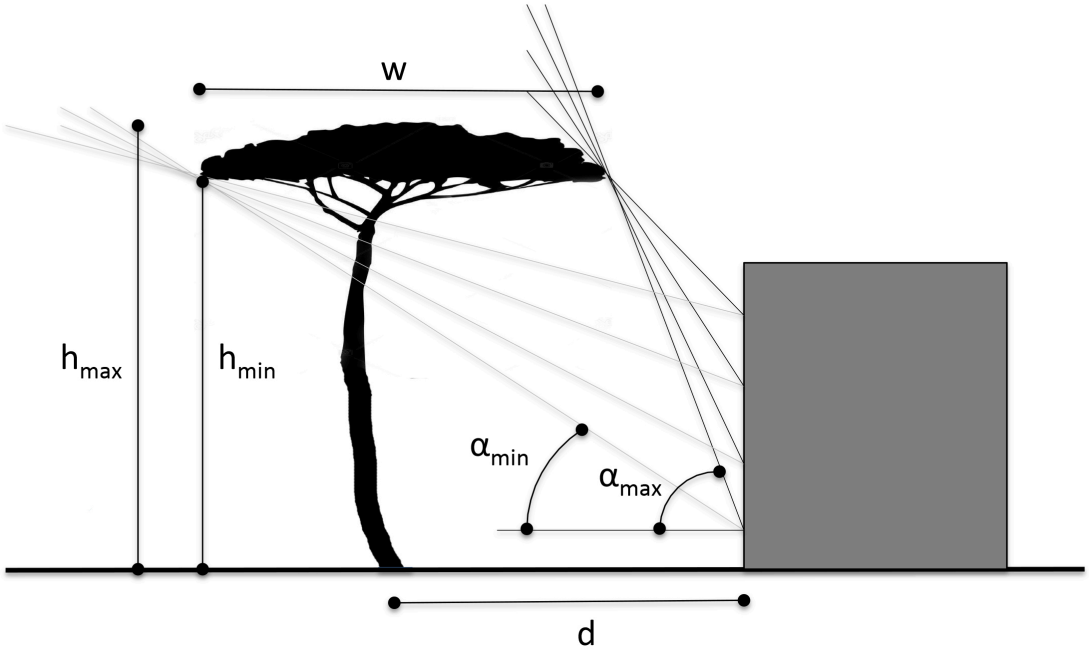

Figure 2: geometrical description of the relation between a tree and a building façade.

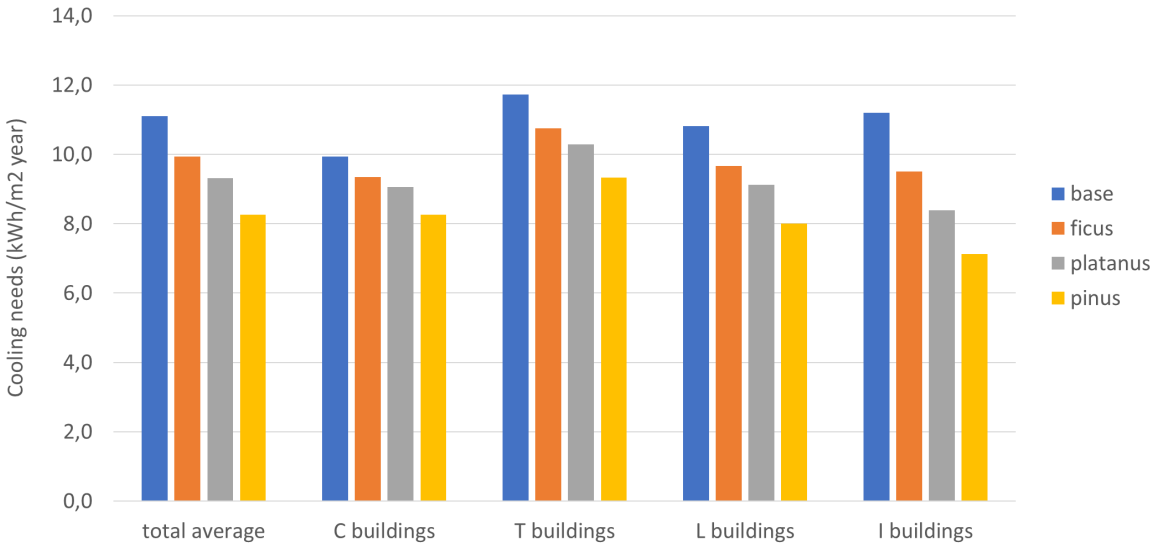

Figure 3: cooling needs for average building grouped by shape

Based on these results, now a machine-learning approach can be used to check if some algorithm can predict the saving in a useful way to urban planners.

2. METHODS

In this work, I will present first an exploratory analysis of data. Remember that these data are the results of building performance simulation I conducted by my own. At the same time, I will present the most important variables that participate to the evaluation, defining the boundary conditions of each configuration. These variables could be used as predictors in machine-learning approach.

After the exploratory analysis, I will propose some algorithms to be used in performing predictions through machine-learning strategies.

Table 1 resumes, for some sample configurations, the most important parameters that could be used as predictors and the final saving obtained by the building performance simulation. Predictors are:

a) The number of façades that can be shadowed according to space availability around building: not all configurations permits to shadow two or three façades, most of cases only permits to shadow one.

b) The plan shape of the building to be shadowed. In the simulation set, I divided the buildings in four groups: "C", "T", "L" and "I" shaped. All buildings have four floors (12 meters height).

c) The tree specie. Some trees are capable to better shadow the building, because of solar permeability (depending on the density of the leaves) and tree's geometry (height and width). I tested three species: *ficus benjamina*, *pinus pinaster* and *platanus occidentalis*.

d) The orientation of the building's main façade. The main façade is the façade with more windows and it is typically the largest façade of the building. Different orientation need of different shadowing strategy: if the building faces the Sun, there is a maximum angular altitude for the Sun projection, which depends on the latitude. If the main façade faces East or West, the situation in different because of the apparent Sun's movement: during some moments of the day, the Sun will stay directly over the building, while during other hours of the day it will stay in front of the façade (at sunrise and sunset).

e) The distance of the tree row from the façade. The trees are supposed to be placed in continuous rows at the minimum distance available following boundary conditions. The more the distance, the less the shadow that the tree will produce on the façade.

f) The deviation of the shadow. In some cases, it is impossible to directly shadow the main façade and the trees' row has to be placed in another orientation. This predictor is the angular deviation summarized in three possible configurations (0, 90, 180 degrees).

Configuration	Number of façades shadowed	Plan shape of building	Tree specie	Main orientation	Distance (m)	Deviation	Saving (%)
1	1	C	Ficus	South	7.6	90	10
2	1	C	Pinus	South	7.6	90	27
3	2	T	Pinus	South	7.2	0	46
4	1	L	Pinus	West	6.3	90	25
5	1	T	Platanus	E	3	0	18
6	3	I	Platanus	South	1.5	0	38
7	1	I	Platanus	South	22.7	0	10

Table 1: some configurations with possible predictors and simulation results

Exploratory analysis suggests that, among the predictors, the number of façades shadowed should have the deepest impact. Figure 4 shows a boxplot of the savings as a function of this predictor. We can notice also that the tree type has a considerable impact on the saving, as the figure 3 puts in evidence.

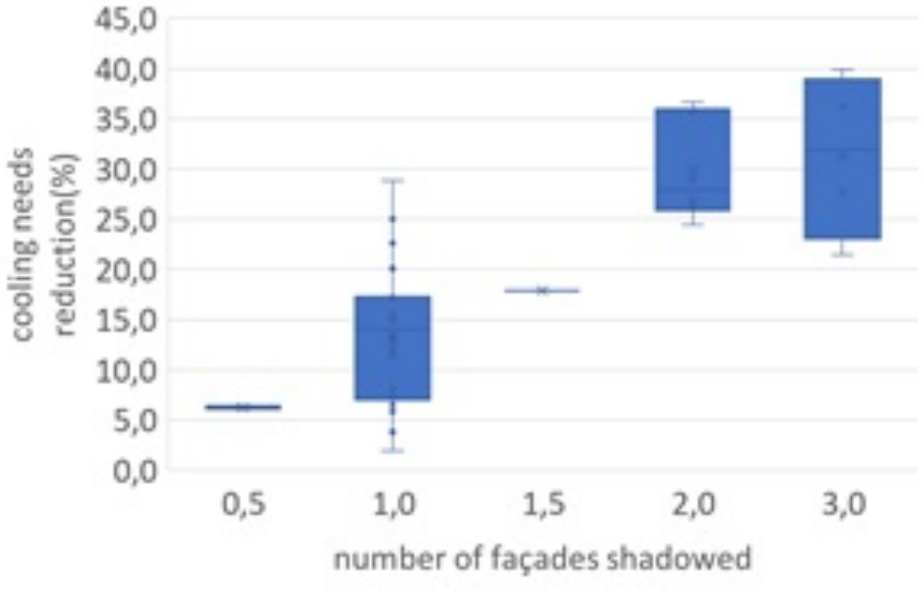

Figure 4: cooling reduction as a function of the number of façades shadowed

Now, to start the machine learning approach, I decided to use all the six predictors to better calibrate the results. Respect to algorithms, I used the Knn, the Random Forest, the Glm and the Loess algorithms. I also used an ensemble of the four algorithms.

First step is to check the RMSE that we obtain predicting the energy saving. I expected a considerable RMSE, because the difficult to predict an exact energy consumption reduction, that depends on many variables.

Remember that the idea is to create a tool useful for urban planners, so maybe the prediction of an exact energy saving is not the real goal. A good result could be the establishment of a classification method that indicates if the saving will be (or not) acceptable. This result can be used by planners in a cost-benefit analysis, which involve other variables, not only the cooling needs of buildings.

Following this idea, the second step is to define a criterion to accept or reject the contribution of the trees to energy saving. I decided to use the 15% of saving as the threshold for acceptance of the benefit. So, if the cooling needs reduction is more than 15%, the use of trees in that configuration will be classified as "acceptable".

To do this in the machine-learning approach, it is sufficient to assign the prediction to the correspondent category and check with the classification of the real data (from building performance simulation) of the test set.

Finally, I tried a better classification in five categories instead of two: the ranges are less than 5% saving ("very low"), 5-15% saving ("low"), 15-25% saving ("medium"), 25-35% saving ("high") and more than 35% ("very high").

To do this in the machine-learning approach it is sufficient to assign the predictions to one of the five categories and check with the real data, as in the prior case.

3. RESULTS

Here is the code I used, with result discussion by steps.

First, install the packages if needed:

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cra
n.us.r-project.org")

if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-p
roject.org")

if(!require(data.table)) install.packages("data.table", repos = "http://c
ran.us.r-project.org")

library(tidyverse)
library(caret)
```

```
library(data.table)
library(tibble)
library(pillar)

library(readr)
```

Then, read the data. There is a table of predictors and results (savings), plus two vectors of categories (the first is the Y/N categorization and the second is the five-class categorization of real data).

```
dat <- read_csv("Shadows.csv")

data<-data.frame(dat)
v<-read.csv("z.csv",header=FALSE)
rep<-read.csv("rep.csv",header=FALSE)
```

Now, generate train and test set. We choose a 25% of cases to be in the test set. More than 25% is difficult because of the low number of data available. Less than 25% could lead to not very credible results.

```
set.seed(3, sample.kind = "Rounding")

test_index<-createDataPartition(data$saving,times=1,p=0.25,list=FALSE)
test_set <- data[test_index, ]
train_set <- data[-test_index, ]

test_v<-v[test_index, ]
test_rep<-rep[test_index, ]
```

Train different algorithms (Knn, Random forest, Glm and Loess)

```
train_knn<-train(saving~.,method="knn",data=train_set)
y_hat_knn<-predict(train_knn,test_set,type="raw")

train_rf<-train(saving~.,method="rf",data=train_set)
y_hat_rf<-predict(train_rf,test_set,type="raw")

train_gamLoess<-train(saving~.,method="gamLoess",data=train_set)

y_hat_gamLoess<-predict(train_gamLoess,test_set,type="raw")

train_glm<-train(saving~.,method="glm",data=train_set)

y_hat_glm<-predict(train_glm,test_set,type="raw")
```

Construct an ensemble and check the RMSE

```
y_hat_resp_ensamble<-(y_hat_gamLoess+y_hat_glm+y_hat_knn+y_hat_rf)/4

RMSE_5<-sqrt(mean((y_hat_resp_ensamble-test_set$saving)^2))
```

```
RMSE_2<-sqrt(mean((y_hat_rf-test_set$saving)^2))
RMSE_1<-sqrt(mean((y_hat_knn-test_set$saving)^2))
RMSE_3<-sqrt(mean((y_hat_glm-test_set$saving)^2))
RMSE_4<-sqrt(mean((y_hat_gamLoess-test_set$saving)^2))
mean(test_set$saving)

options(pillar.sigfig=3)

rmse_results <- tibble(method = c("Knn","Rf","Glm","Loess","Ensamble"), R
MSE = c(RMSE_1,RMSE_2,RMSE_3,RMSE_4,RMSE_5))

rmse_results
```

Results show that the mean saving is 16.55%. The RMSE is listed in the table for each algorithm and for the ensemble.

```
## [1] 16.54545

## # A tibble: 5 x 2
##   method    RMSE
##   <chr>    <dbl>
## 1 Knn       9.13
## 2 Rf        5.93
## 3 Glm       5.36
## 4 Loess     5.20
## 5 Ensamble  5.31
```

RMSE is more than the 30% of the mean value of saving. It will be difficult to predict the saving with a reasonable confidence interval. However, a practical tool to be used in planning does not need to present a numerical value as a result. It could be very useful a tool that helps planners to decide weather or not to plant a row of trees in each configuration. We established a threshold value of 15%: under this value it is not recommended to plant the trees. Now, we can use machine-learning approach to predict if the saving will be more or less than the threshold, and check the accuracy.

Try Knn first:

```
y_hat_1<-ifelse(y_hat_knn>15,"Y","N")
y_hat_1<-as.factor(y_hat_1)
confusionMatrix(y_hat_1,test_v)

accuracy_1<-confusionMatrix(y_hat_1,test_v)$overall["Accuracy"]
```

Result:

```
## Confusion Matrix and Statistics (Knn)
##
##           Reference
## Prediction  N  Y
##          N 13  1
##          Y  5 14
##
##                Accuracy : 0.8182
##                  95% CI : (0.6454, 0.9302)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.001007
##
##                   Kappa : 0.6413
##
##  Mcnemar's Test P-Value : 0.220671
##
##             Sensitivity : 0.7222
##             Specificity : 0.9333
##          Pos Pred Value : 0.9286
##          Neg Pred Value : 0.7368
##              Prevalence : 0.5455
##          Detection Rate : 0.3939
##    Detection Prevalence : 0.4242
##       Balanced Accuracy : 0.8278
##
##        'Positive' Class : N
##
```

So Knn predicts the Y/N selection with a 82% of accuracy. Try the other algorithms and the ensemble:

```
y_hat_2<-ifelse(y_hat_rf>15,"Y","N")
y_hat_2<-as.factor(y_hat_2)
confusionMatrix(y_hat_2,test_v)

accuracy_2<-confusionMatrix(y_hat_2,test_v)$overall["Accuracy"]

y_hat_3<-ifelse(y_hat_gamLoess>15,"Y","N")
y_hat_3<-as.factor(y_hat_3)
confusionMatrix(y_hat_3,test_v)

accuracy_3<-confusionMatrix(y_hat_3,test_v)$overall["Accuracy"]

y_hat_4<-ifelse(y_hat_glm>15,"Y","N")
y_hat_4<-as.factor(y_hat_4)
confusionMatrix(y_hat_4,test_v)
```

```r
accuracy_4<-confusionMatrix(y_hat_4,test_v)$overall["Accuracy"]

y_hat_ensamble<-ifelse(y_hat_resp_ensamble>15,"Y","N")
y_hat_ensamble<-as.factor(y_hat_ensamble)
confusionMatrix(y_hat_ensamble,test_v)

accuracy_5<-confusionMatrix(y_hat_ensamble,test_v)$overall["Accuracy"]

accuracy_results<-tibble(method = c("Knn","Rf","Glm","Loess","Ensamble"),
Accuracy = c(accuracy_1,accuracy_2,accuracy_3,accuracy_4,accuracy_5))

accuracy_results
```

Result:

```
## # A tibble: 5 x 2
##   method   Accuracy
##   <chr>       <dbl>
## 1 Knn         0.818
## 2 Rf          0.909
## 3 Glm         0.939
## 4 Loess       0.909
## 5 Ensamble    0.970
```

We see that most of the algorithms can predict the option Y/N with more than a 90% of accuracy. Best performance is the ensamble with a 97% of accuracy.

The Random forest approach is very interesting because we can see which predictors are the most important in selection: the number of fachades shaded (used the 100% of times), the tree type (pinus permits to shade better the buildings), and the distance from the fachade (the bigger the distance, the lower the shadow). Then the angular desviation between the row of trees and the fachade and the plan shape of the building (buildings with "I" shape performe in general better than others, due to the fact that is esaier to shadow the main fachade).

```
## rf variable importance
##
##                            Overall
## numbero.fachades.shadowed 100.000
## tree.typepinus             62.645
## distance                   41.593
## desviation                 31.619
## plan.shapeI                21.463
## tree.typeplatanus           5.887
```

```
## plan.shapeL                 4.872
## plan.shapeT                 2.780
## orientationS                1.101
## orientationW                1.015
## orientationN                0.000
```

Another interesting result of Rf algorithms is the sensitivity (100%). In the urban planning problem sensitivity should be priviledged respect to specifcity because the most important thing is not to choose as favourable (Y) a case that it is really not (N). So Rf approach assure that the minimum of 15% energy saving is obtained.

```
## Confusion Matrix and Statistics (Random Forest)
##
##           Reference
## Prediction  N  Y
##          N 18  3
##          Y  0 12
##
##                Accuracy : 0.9091
##                  95% CI : (0.7567, 0.9808)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 7.304e-06
##
##                   Kappa : 0.8136
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.8000
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5455
##          Detection Rate : 0.5455
##    Detection Prevalence : 0.6364
##       Balanced Accuracy : 0.9000
##
##        'Positive' Class : N
##
```

The ensamble has also a sensitivity of 100% and a better specificity. It actually predict correctly 32 cases of 33.

```
## Confusion Matrix and Statistics (Ensemble)
##
##           Reference
## Prediction  N  Y
##          N 18  1
##          Y  0 14
##
```

```
##                Accuracy : 0.9697
##                  95% CI : (0.8424, 0.9992)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 5.86e-08
##
##                   Kappa : 0.9385
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9333
##          Pos Pred Value : 0.9474
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5455
##          Detection Rate : 0.5455
##    Detection Prevalence : 0.5758
##       Balanced Accuracy : 0.9667
##
##        'Positive' Class : N
```

Now, we can also try to establish more than two classes. For example, we can decide that for savings less than 5% there is a "very low" saving, for savings between 5 and 15% a "low" saving, for savings between 15 and 25% a "medium" saving, for savings between 25 and 35% a "high" saving and for savings more than 35% a "very high" saving.

Now check the accuracy of the algorithms for this 5 classes classification:

```
y_hat_resp_1<-ifelse(y_hat_knn<5,"very low",ifelse(y_hat_knn<15,"low",ife
lse(y_hat_knn<25,"medium",ifelse(y_hat_knn<35,"high","very high"))))
y_hat_resp_1<-as.factor(y_hat_resp_1)
confusionMatrix(y_hat_resp_1,test_rep)

accuracy_resp_1<-confusionMatrix(y_hat_resp_1,test_rep)$overall["Accuracy
"]

y_hat_resp_2<-ifelse(y_hat_rf<5,"very low",ifelse(y_hat_rf<15,"low",ifels
e(y_hat_rf<25,"medium",ifelse(y_hat_rf<35,"high","very high"))))
y_hat_resp_2<-as.factor(y_hat_resp_2)
confusionMatrix(y_hat_resp_2,test_rep)

accuracy_resp_2<-confusionMatrix(y_hat_resp_2,test_rep)$overall["Accuracy
"]

y_hat_resp_3<-ifelse(y_hat_glm<5,"very low",ifelse(y_hat_glm<15,"low",ife
lse(y_hat_glm<25,"medium",ifelse(y_hat_glm<35,"high","very high"))))
y_hat_resp_3<-as.factor(y_hat_resp_3)
confusionMatrix(y_hat_resp_3,test_rep)
```

```
accuracy_resp_3<-confusionMatrix(y_hat_resp_3,test_rep)$overall["Accuracy
"]

y_hat_resp_4<-ifelse(y_hat_gamLoess<5,"very low",ifelse(y_hat_gamLoess<15
,"low",ifelse(y_hat_gamLoess<25,"medium",ifelse(y_hat_gamLoess<35,"high",
"very high"))))
y_hat_resp_4<-as.factor(y_hat_resp_4)
confusionMatrix(y_hat_resp_4,test_rep)

accuracy_resp_4<-confusionMatrix(y_hat_resp_4,test_rep)$overall["Accuracy
"]

y_hat_resp_5<-ifelse(y_hat_resp_ensamble<5,"very low",ifelse(y_hat_resp_e
nsamble<15,"low",ifelse(y_hat_resp_ensamble<25,"medium",ifelse(y_hat_resp
_ensamble<35,"high","very high"))))
y_hat_resp_5<-as.factor(y_hat_resp_5)
confusionMatrix(y_hat_resp_5,test_rep)

accuracy_resp_5<-confusionMatrix(y_hat_resp_5,test_rep)$overall["Accuracy
"]

accuracy_resp_results<-tibble(method = c("Knn","Rf","Glm","Loess","Ensamb
le"), Accuracy_cat = c(accuracy_resp_1,accuracy_resp_2,accuracy_resp_3,ac
curacy_resp_4,accuracy_resp_5))

accuracy_resp_results
```

Result:

```
## # A tibble: 5 x 2
##   method   Accuracy_cat
##   <chr>           <dbl>
## 1 Knn             0.485
## 2 Rf              0.606
## 3 Glm             0.636
## 4 Loess           0.636
## 5 Ensamble        0.758
```

In this case, we obtain accuracy of 76% with the ensamble, not bad but it should be further improved. We will need more data (we have now only 123) to do that.

4. CONCLUSION

With this work I developed a machine-learning strategy to predict the energy saving that buildings can achieve when rows of trees are used to shadow the façades. I used my own building performance simulation results of a set of 123 cases. I used the 75% of these cases to train machine-learning algorithms (knn, random forest, glm and loess) and the 25% to test the accuracy of predictions.

The RMSE of exact energy saving prediction is quite high (rounding 30%); nevertheless, a categorical prediction (more or less than 15%) is very accurate. The best result is achieved by the ensemble, with a 97% accuracy. A prediction that uses five categories is not so accurate, achieving a 76% as the best performance (always the ensemble performs better).

Among algorithms, Glm and Loess perform slightly better than Random Forest and Knn. However, Random forest is interesting because its high sensitivity. As observed in the result section, sensitivity is important in practice, guaranteeing that real case will not underperform respect to prediction.

The analysis of the variable importance in random forest prediction, confirms also the hypothesis that the number of façades shadowed and the tree specie are the most important predictors to be used.

Future development of this work will include:

a) The simulation of more configurations to improve the performance of the machine-learning;

b) The check of the accuracy in similar projects (with other emplacements but similar climates)

c) The development of a climate-sensitive strategy (which should include latitude and climatic data as new predictors).

Final goal should be the development of a user-friendly tool to be used in supporting urban-planning.