Metody Odkrywania Wiedzy

Dokumentacja końcowa projektu

„Predykcja zużycia energii na podstawie danych czujnikowych”

Krzysztof Belewicz
Paweł Pieńczuk

25 stycznia 2020

1. Opis projektu

Celem projektu było wyznaczenie całkowitego zużycia energii dla zadanej chwili czasu, tzn. sumy poborów sprzętów AGD (kolumna 'Appliances') i oświetlenia (kolumna 'Lights'). Zbiór danych został pozyskany z archiwum dostępnego na stronie: `https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction`. Pojęciem docelowym jest wartość całkowitej pobieranej mocy przez gospodarstwo domowe.

TODO

2. Opis danych

2.1. Charakterystyka danych

Dane wykorzystywane do eksperymentów zostały zebrane za pomocą sieci czujników w niewielkim domu w czasie 4.5 miesiąca. Składają się z:

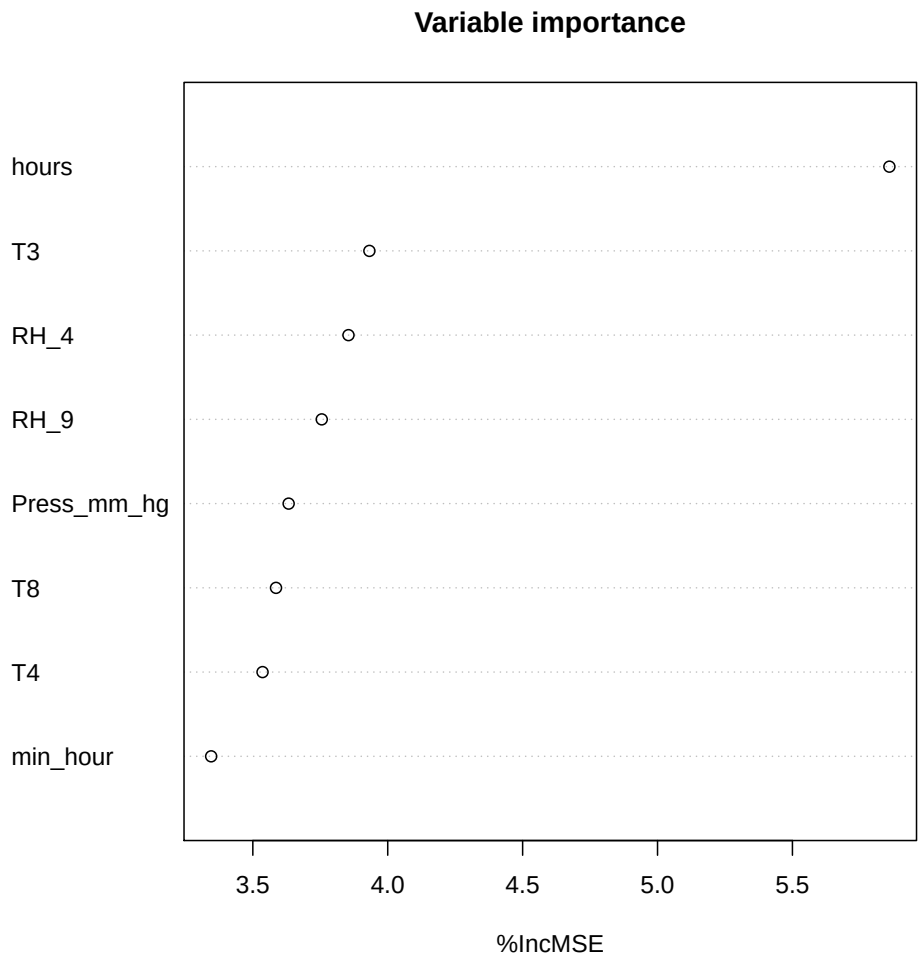
- daty i godziny pomiaru,
- poboru energii sprzętów domowych [Wh],
- poboru energii oświetlenia [Wh],
- pomiarów temperatury i wilgotności dla 8 różnych pomieszczeń ([$^\circ C$], [$\%$]),
- pomiarów temperatury i wilgotności dla zewnętrznej, północnej strony budynku ([$^\circ C$], [$\%$]),
- danych z pobliskiej stacji pogodowej:
  - temperatura powietrza [$^\circ C$],
  - temperatura punktu rosy [$^\circ C$],
  - ciśnienie atmosferyczne [$mm\,Hg$],
  - wilgotność [$\%$],
  - prędkość wiatru [m/s],
  - widoczność [km].

2.2. Przygotowanie danych

Każdy pomiar został uśredniony z 3 próbek wykonanych w równych odstępach co ok. 3,3 min. W ramach przygotowania danych, data i godzina pomiaru zostały rozdzielone na dwie oddzielne kolumny – ułatwi to późniejsze operacje na danych.

3. Opis algorytmów

TODO - ten chapter jest imo zawarty w 'konstrukcji i ocenie modeli'
ergo można usunąć, grafikę przenieść tam.
(chapter jest tutaj bo skopiowałem sprawko wstępne)
(grafika nie wiem czemu tutaj jest, tak wyszło)



4. Selekcja atrybutów

TODO jakoś ładnie opisać, to co niżej raczej dla inspiracji

Algorytm randomForest pozwala uszeregować atrybuty na dwa sposoby:

- przydatność atrybutu oceniania jest względem tego, jak bardzo model pogorszy się gdy tego atrybutu zabraknie,
- przydatność atrybutu określa zmniejszenie nieczystości po podziale w oparciu o ten atrybut.

Została wybrana pierwsza metoda ze względu na jej widoczny i bezpośredni związek z jakością całego modelu.

Najprostszym, zadowalającym rozwiązaniem jest zastosowanie pakietu randomForest w celu wyznaczenia predykcyjnej przydatności atrybutów, a następnie wybór (może w kilku wariantach) pewnej liczby najlepszych atrybutów (np. najlepsze 25%, 50% itp.). W tym przypadku zalecałbym użycie miary "mean decrease accuracy", co wymaga użycia w wywołaniu funkcji randomForest argumentu importance=TRUE, zaś w wywołaniu funkcji importance lub varImpPlot argumentu type=1. Inne bardziej zautomatyzowane podejścia można znaleźć w kilku pakietach wymienionych tutaj: `https://github.com/FrancisArgnR/R-FeatureSelection-Packages`

4.1. Wyniki selekcji atrybutów

TODO jakaś tabelka albo co + ew wnioski

5. Konstrukcja i ocena modeli

5.1. Metody konstrukcji modeli

Pojedyncze modele drzew regresji zazwyczaj cierpią z powodu wysokiej wariancji – jedną z metod jej redukcji jest tzw. Bagging (**B**ootstrap **agg**regat**ing**). Metoda ta polega na łączeniu i uśrednianiu wielu modeli drzew, co zmniejsza wariancję i redukuje zbytnie dopasowanie.

Bagging może zostać zrealizowany za pomocą pakietu *ipred* lub *caret*. *Ipred* jest z reguły prostszy w realizacji, jednakże stosowanie *caret* niesie za sobą kilka zalet. Znacznie prościej jest weryfikować krzyżowo wyniki – pomimo możliwości wykorzystania błędu OOB (Out-of-Bag) w *ipred*, weryfikacja krzyżowa daje dużo lepsze zrozumienie spodziewanego błędu. Dodatkowo możliwy jest dostęp do zmiennej odpowiedniości w wygenerowanych drzewach.

Do konstrukcji modeli klasyfikacji został wykorzystany pakiet rpart budujacy drzewo klasyfikacji oraz pakiet e1071, który umozliwia waidacje skrosna oraz automatyczny dobór parametrów w celu zmninimalizowania błedu.

5.2. Budowa modelu klasyfikacji z pakietu ipred

TODO - pakiety R, parametry (np kryteria stopu, etc)

5.3. Ocena modeli

TODO opisać że wybraliśmy np:
CC - współczynnik korelacji liniowej Pearsona

$$\frac{cov(P,A)}{var(P) \cdot var(A)}$$

TODO tutaj ładnie się wpasuje co po kolei w kodzie poszło

6. Wnioski

TODO

Plik: main.r

```
# main v1.1
#setwd("E:/Documents/Studies/MOW/MOW_project")
#setwd("Z:/Offtop/MOW/Project/MOW_project")
setwd("F:/GitHub/MOW_project")
setwd(".")

library(rsample)      # data splitting
library(dplyr)        # data wrangling
library(ipred)        # bagging
library(caret)        # bagging
library(dmr.regtree)

rm(list = ls())

# DATA COLLECTION AND ORGANIZATION GOES HERE

source('R/data_org.R', echo=TRUE)

# FEATURE SELECTION GOES HERE

source('R/feature_selection.R')
source('R/simple.filter.R')

res <- feature_selection(test_data = test_data,
                                type = "rf",
                                part = 0.25,
                                trees_num=20)

# CREATING MODELS GOES HERE

# EVALUATION PROCEDURES GOES HERE

source("R/model_eval.R")

ctrl <- trainControl(method = "cv", number = 10)
args_t <- c(method="treebag",trControl=ctrl)

tempdataframe <- model_eval(test_data = test_data,
                            fun = rpart,
                            formula = as.character(res$attr_part[1]),
                            crossval_number = 10
                            )
tempdataframe2 <- model_eval(test_data = test_data,
                             fun = lm,
                             formula = as.character(res$attr_part[1]),
                             crossval_number = 10
                            )
tempdataframe3 <- model_eval(test_data = test_data,
                             fun = bagging,
                             formula = as.character(res$attr_part[1]),
                             crossval_number = 10
)
tempdataframe4 <- model_eval(test_data = test_data,
                             fun = train,
                             formula = as.character(res$attr_part[1]),
                             crossval_number = 10,
                             args=args_t
)
```

Plik: data_org.R

```r
#data_org.R
# data collecting and organization

training_data <- read.csv("training.csv")
complete_data <- read.csv("energydata_complete.csv")

month = as.numeric(substring(complete_data$date,6,7))
day = as.numeric(substring(complete_data$date,9,10))
hours = as.numeric(substring(complete_data$date,12,13))
minutes = as.numeric(substring(complete_data$date,15,16))
day_mon = day + 30 * month
min_hour = minutes + 60 * hours

test_data <- data.frame(
  Appliances = complete_data$Appliances,
  month = substring(complete_data$date,6,7),
  day = substring(complete_data$date,9,10),
  hours = substring(complete_data$date,12,13),
  minutes = substring(complete_data$date,15,16),
  day_mon = day_mon,
  min_hour = min_hour,
  T1 = complete_data$T1,
  T2 = complete_data$T2,
  T3 = complete_data$T3,
  T4 = complete_data$T4,
  T5 = complete_data$T5,
  T6 = complete_data$T6,
  T7 = complete_data$T7,
  T8 = complete_data$T8,
  T9 = complete_data$T9,
  RH_1 = complete_data$RH_1,
  RH_2 = complete_data$RH_2,
  RH_3 = complete_data$RH_3,
  RH_4 = complete_data$RH_4,
  RH_5 = complete_data$RH_5,
  RH_6 = complete_data$RH_6,
  RH_7 = complete_data$RH_7,
  RH_8 = complete_data$RH_8,
  RH_9 = complete_data$RH_9,
  T_out = complete_data$T_out,
  RH_out = complete_data$RH_out,
  Press_mm_hg = complete_data$Press_mm_hg,
  Windspeed = complete_data$Windspeed,
  Visibility = complete_data$Visibility,
  Tdewpoint = complete_data$Tdewpoint,
  rv1 = complete_data$rv1,
  rv2 = complete_data$rv2
)

test_data <- test_data[1:1000,]

month = as.numeric(substring(training_data$date,6,7))
day = as.numeric(substring(training_data$date,9,10))
hours = as.numeric(substring(training_data$date,12,13))
minutes = as.numeric(substring(training_data$date,15,16))

day_mon = day + 30 * month
min_hour = minutes + 60 * hours

```

```
vars_train <- data.frame(
  Appliances = training_data$Appliances,
  month = substring(training_data$date,6,7),
  day = substring(training_data$date,9,10),
  hours = substring(training_data$date,12,13),
  minutes = substring(training_data$date,15,16),
  day_mon = day_mon,
  min_hour = min_hour,
  T1 = training_data$T1,
  T2 = training_data$T2,
  T3 = training_data$T3,
  T4 = training_data$T4,
  T5 = training_data$T5,
  T6 = training_data$T6,
  T7 = training_data$T7,
  T8 = training_data$T8,
  T9 = training_data$T9,
  RH_1 = training_data$RH_1,
  RH_2 = training_data$RH_2,
  RH_3 = training_data$RH_3,
  RH_4 = training_data$RH_4,
  RH_5 = training_data$RH_5,
  RH_6 = training_data$RH_6,
  RH_7 = training_data$RH_7,
  RH_8 = training_data$RH_8,
  RH_9 = training_data$RH_9,
  RH_out = training_data$RH_out,
  T_out = training_data$T_out,
  Press_mm_hg = training_data$Press_mm_hg,
  Windspeed = training_data$Windspeed,
  Visibility = training_data$Visibility,
  Tdewpoint = training_data$Tdewpoint,
  rv1 = training_data$rv1,
  rv2 = training_data$rv2
)
```

Plik: feature_selection.R

```r
# feature_selection.R

library(randomForest)
library(rpart)
library(rpart.plot)  # plotting regression trees
library(Metrics)     # RMSE
library(dmr.disc)
library(dmr.stats)

#' @title Feature Selection
#'
#' @param test_data  - data to perform feature ranking
#' @param type - "rf" - randomForest IMPORTANCE-based ranking (MSE)(default),
#' "bootstrap" - bootstraped R2-based (coef. of determination) ranking
#' "simple" - simple filter based algorithms
#' @param part - part of attributes returned by feature selection, 0 < part <=1 (default=0
#' @param trees_num - numbers of trees (randomForest) or bootstrap sets (bootstrap)
#'
#' @return formula with selected attributes
#'
#'
#'
feature_selection <- function(test_data,type="rf",part=1,trees_num=10){

  if ( !is.data.frame(test_data) ){
    message("Pass data frame format")
    stop()
  }
  if (trees_num<1) {
    message("Trees number should be >= 1")
    stop()
  }
  if (part>1|part<=0){
    message("Parts of atrributes must be >0 and <=1")
    stop()
  }

  print("FEATURE SELECTION")
  count <- ceiling((ncol(test_data)-1)*part)

  if (type=="rf"){
    full_model_RF <- randomForest(formula = Appliances ~ .,
                          data = test_data,
                          importance = TRUE,
                          ntree=trees_num)
    importance_RF <- data.frame(importance(full_model_RF, type=1), "k"=1:(ncol(test_data)-
    wyniki <- data.frame("importance"=importance_RF[order(importance_RF[,1],decreasing = T
                               "k"=importance_RF$k[order(importance_RF[,1],decreasing = TRU
                               "attr"=rownames(importance_RF)[order(importance_RF[,1], decr

    # plotting most important parameters
    varImpPlot(x=full_model_RF,
             n.var=count,
             type=1,
             main="Variable importance")

    # passing most important attributes from feature selection
    count <- ceiling((ncol(test_data)-1)*part)
    attr_part <- paste(wyniki$attr[2:count], collapse = "+")
```

```
60      attr_part <- paste("Appliances", "~", attr_part)
61      out <- data.frame(attr_part,wyniki)
62      return(out)
63    }
64    else if(type=="bootstrap"){
65
66      r2 <- function(pred.y, true.y)
67      { 1 - length(true.y)*mse(pred.y = pred.y,true.y=true.y)/((length(true.y)-1)*var(true.y
68
69      N=trees_num
70
71      wyniki <- data.frame(matrix(0,ncol=3,nrow=(ncol(test_data)-2)))
72
73      # bootstrap
74      for (i in 1:N)
75      {
76        wyniki_tmp <- data.frame("quality"=double(), "quality_perm"=double(), "quality_diff"
77
78        n <- sample(nrow(test_data), nrow(test_data), replace=TRUE)
79        data_train <- test_data[n,]
80        data_test <- test_data[-n,]
81
82        # tworzenie modelu na wszystkich atrybutach
83        model <- rpart(formula=Appliances~.,
84                       data=data_train,
85                       method='anova'
86        )
87        pred_full <- predict(model, data_test)
88
89        quality <- r2(pred.y = pred_full, true.y = data_test$Appliances)
90
91        # badanie wplywu permutacji
92        for (k in 2:(ncol(test_data)-1))
93        {
94          # permutacja k-tego atrybutu
95          data_test_perm <- data_test
96          data_test_perm[,k] <- sample(data_test_perm[,k])
97
98          # predykcja dla atrybutow o spermutowanych wartosciach
99          pred_perm <- predict(model, data_test_perm)
100
101         quality_perm <- r2(pred.y = pred_perm, true.y = data_test_perm$Appliances)
102
103         quality_diff <- quality - quality_perm
104         wyniki_tmp <- rbind(wyniki_tmp, c(quality, quality_perm, quality_diff))
105       }
106       # dodanie wynikow z i-tej iteracji
107       wyniki <- wyniki + wyniki_tmp
108     }
109
110     wyniki <- wyniki/N
111
112     wyniki <- data.frame(2:k, colnames(test_data)[2:k], wyniki[,3])
113     wyniki <- wyniki[order(wyniki[,3], decreasing = TRUE),]
114     colnames(wyniki) <- c("k", "attr", "R2_diff")
115
116     count <- ceiling((ncol(test_data)-1)*part)
117     message('Najwa?niejsze␣', part*100, '%␣atrybut?w␣to:\n', paste(wyniki$attr[1:count],
118     attr_part <- paste(wyniki$attr[2:count], collapse = "+")
119     attr_part <- paste("Appliances", "~", attr_part)
120
```

```
121     out <- data.frame(attr_part,wyniki)
122     return(out)
123   }
124
125   else if (type=="simple"){
126     # Symmetric uncertainty - symunc
127     # discnm.eqfreq - equal-frequency discretization
128
129     res <- simple.filter(formula = Appliances~.,
130                          data = discnm.eqfreq(~.,test_data,50),
131                          dd=symunc)
132
133     names_t <- names(res)
134     attr_part <- paste(names_t[1:count], collapse = "+")
135     attr_part <- paste("Appliances", "~", attr_part)
136     out <- data.frame(attr_part,res)
137     return(out)
138   }
139   else{
140     message("Wrong type of feature selection: \"bootstrap\" \"rf\" or \"simple\" ")
141     stop()
142   }
143 }
```

Plik: simple.filter.R

```
1
2  library(dmr.stats)
3  library(dmr.util)
4
5  dd.chi2 <- function(a1, a2) 1-chisq.test(a1, a2)$p.value
6  cd.kruskal <- function(a1, a2) 1-kruskal.test(a1, a2)$p.value
7  cc.spearman <- function(a1, a2) 1-cor.test(a1, a2, method="spearman")$p.value
8
9  #' @title Simple attribute filter
10 #'
11 #' @param formula
12 #' @param data
13 #' @param dd
14 #' @param cd
15 #' @param cc
16 #'
17 #' @return
18 #'
19 #'
20 #'
21 simple.filter <- function(formula, data, dd=dd.chi2, cd=cd.kruskal, cc=cc.spearman)
22 {
23   attributes <- x.vars(formula, data)
24   target <- y.var(formula)
25   utility <- function(a)
26   {
27     unname(switch(attr.type(data[[a]], data[[target]]),
28                   dd = dd(data[[a]], data[[target]]),
29                   cd = cd(data[[a]], data[[target]]),
30                   dc = cd(data[[target]], data[[a]]),
31                   cc = cc(data[[a]], data[[target]])))
32   }
33   sort(sapply(attributes, utility), decreasing=TRUE)
34 }
```

Plik: model_eval.R

```r
# model evaluation

library(dmr.claseval)

#' @title Model Evaluation
#'
#' @param test_data - data to perform cross-validation
#' @param fun - regression algorithm (e.g. train,bagging,lm) default=rpart
#' @param formula - regression formula
#' @param crossval_number - number of cross-validations, default=10
#' @param args - passed arguments into regression algorithm, default=NULL
#'
#' @return data frame with measures in following order: MSE, RRSE, MAE, RMSE, RAE, COR, RS
#'
#'
model_eval <- function(test_data,fun=rpart,formula,crossval_number=10,args=NULL){

  if ( !is.data.frame(test_data) ){
    message("Pass data frame format")
    stop()
  }
  if ( !is.character(formula)) {
    message("Pass string into attr parameter")
    stop()
  }
  if (crossval_number<1){
    message("Cross validations number must be greater than 0")
    stop()
  }
  if (crossval_number==1){
    warning("No cross-validation")
  }
  temp_model <- crossval(fun,
                         as.formula(formula),
                         test_data,
                         k=crossval_number,
                         args = args)
  MSE <- mse(temp_model$pred, temp_model$true)
  RRSE <- rrse(temp_model$pred, temp_model$true)
  MAE <- mae(temp_model$pred, temp_model$true)
  RMSE <- rmse(temp_model$pred, temp_model$true)
  RAE <- rae(temp_model$pred, temp_model$true)
  COR <- cor(temp_model$pred, temp_model$true, method="pearson")
  RSE <- rse(temp_model$pred, temp_model$true)

  out <- data.frame(MSE,RRSE,MAE,RMSE,RAE,COR,RSE)
  return(out)
}
```