Guia para utilizar os metamodelos que estimam o desempenho térmico de unidades habitacionais

Em acordo com a nova proposta para a NBR 15575

LabEEE - Rodolfo Kirch Veiga (rodolfoksveiga@gmail.com)

Florianópolis

Objetivo

Esse guia tem como objetivo descrever o procedimento para estimar o desempenho térmico de unidades habitacionais (UHs) uni e multifamiliares, seguindo as premissas da **nova proposta para NBR 15575**. As predições de desempenho serão realizadas na linguagem de programação **R** através de um modelo de aprendizado de máquinas (metamodelo) previamente treinado e testado pela equipe do **Laboratório de Eficiência Energética (LabEEE)** da **Universidade Federal de Santa Catarina (UFSC)**, em uma parceria com a empresa **Saint Gobain**.

Dependências

Para reproduzir o que foi descrito nesse documento é necessário que o usuário possua em sua máquina a linguagem R instalada e os seguintes arquivos:

1. models.rds
2. predict_outputs.r

Obs.: o uso da IDE RStudio pode facilitar a execução e o acompanhamento dos resultados dos códigos que seguem.

Características dos metamodelos

A base de dados gerada para treinar os metamodelos foi desenvolvida perturbando-se dois modelos de simulação base, referentes à: a) uma tipologia unifamiliar térrea, e; b) uma tipologia multifamiliar, composta por pavimento térreo, tipo e cobertura. As características de ambas as tipologias foram extraídas do levantamento de edificações residenciais brasileiras de baixa renda, realizado por Montes (2016).

A partir das perturbações e da implementação do método de amostragem *Latin Hypercube*, códigos escritos na linguagem Python geraram, de forma automatizada, **10000** modelos de simulação para a tipologia unifamiliar e **53760** para a tipologia multifamiliar. Para cada modelo de simulação foi gerado também seu respectivo

modelo de referência, como exige a nova proposta para a NBR 15575. Todas as simulações foram executadas no programa EnergyPlus versão 9.0.1, e através dos resultados das simulações foram calculados os seguintes dados de saída: a) percentual de horas ocupadas em que o ambiente encontra-se dentro de uma determinada faixa de temperatura (PHFT), e; b) somatório das cargas térmicas de refrigeração e aquecimento quando o ambiente está ocupado e encontra-se com temperatura fora da faixa pré-determinada (CgTT).

Detalhes sobre a modelagem e o cálculo dos dados de saída podem ser encontrados na nova proposta para NBR 15575.

Através da base de dados foram treinados e testados metamodelos utilizando as seguintes técnicas: *Linear Regression* (LM), *Support Vector Machine with Radial Kernel* (SVM), *Bayesian Neural Networks* (BRNN), *Boosted Tree* (BT) e *Extreme Gradient Boosting* (XGBT). No pre-processamento da base de dados, as variáveis de entrada quantitativas foram transformadas através da técnica *BoxCox* e as variáveis de entrada qualitativas foram transformadas em variáveis lógicas, também conhecidas como *Dummy Variables*. Todos os metamodelos foram treinados através do pacote Caret, da linguagem **R**.

A tabela abaixo trata-se a estatística descritiva dos metamodelos finais, ou seja, aqueles que apresentaram os resultados mais acurados na etapa de teste, i.e. menores erro médio absoluto (MAE) e raíz do erro quadrático médio (RMSE). Os metamodelos mais acurados foram alcançados através da técnica *Extreme Gradient Boosting*.

Tipologia	Output	MAE	RMSE	RSquared
Uni	PHFT	1.86	2.98	0.97
Uni	Dif. PHFT	1.92	3.10	0.97
Uni	CgTT	364.44	604.75	0.97
Uni	Dif. CgTT	332.38	543.47	0.97
Multi	PHFT	2.04	2.74	0.99
Multi	Dif. PHFT	1.98	2.71	0.98
Multi	CgTT	168.20	239.98	0.99
Multi	Dif. CgTT	186.20	258.06	0.98

Para predizer cada um dos dados de saída foi associado um metamodelo diferente. Entretanto, o usuário final não deve se apegar a esses detalhes, pois o código ofericido realizará todas as rotinas de cálculo necessárias e retornará uma tabela com os valores das variáveis de entrada fornecidas e dos respectivos valores das variáveis de saída.

Variáveis de entrada e os valores adotados no treinamento do modelo

- `area` (m^2): área total do piso da UH
  - Unifamiliar (uni): 39, 58 e 77
  - Multifamiliar (multi): 34, 35, 51 e 52
- `ratio` (adim.): relação entre o comprimento e a profundidade do edifício - $ratio = lx/ly$
  - Uni: 0.5, 1 e 2.5
  - Multi: 0.5, 1 e 2
- `pav` (qualitativa): pavimento onde encontra-se a UH
  - Multi: “terreo” (térreo), “inter” (intermediário) e “cob” (cobertura)
- `solo` (qualitativa): exposição do pavimento térreo do edifício ao solo
  - Multi: “solo” (exposto ao solo) e “pilotis” (edifício sobre pilotis)
- `exp` (qualitativa) : exposição das paredes da UH

  - Multi: “canto” (duas fachadas expostas) e “meio” (uma fachada exposta)
- `pd` (m): pé-direito
  - Uni: 2.5 e 5
- `azi` (°): ângulo azimutal
  - Uni/Multi: 0, 90, 180 e 270
- `comp` (qualitativa): composição dos componentes construtivos
  - Uni/Multi: “ref” (concreto de 10 cm), “tv” (tjolo vazado), “tm10” (tijolo maciço de 10 cm), “tm20” (tijolo maciço de 20 cm), “sfar” (*steel frame* isolado com ar), “sfiso” (*steel frame* isolado com lã de vidro)
- `vid`: composição dos vidros
  - Uni/Multi: “simples_fs87” (vidro simples com 0.87 de FS), “simples_fs39” (vidro simples com 0.39 de FS), “duplo_fs87-87” (vidro duplo com FS de 0.87), “duplo_fs39-87” (vidro duplo com FS = 0.39 e 0.87)
- `abs` (adim.): absortância solar
  - Uni/Multi: 0.2, 0.6 e 0.8
- `paf` (%): percentual envidraçado das fachadas
  - Uni/Multi: 10, 30 e 50
- `fv` (adim.): fator de ventilação
  - Uni/Multi: 0.45 e 1
- `ven` (qualitative): possui veneziana?
  - Uni/Multi: “on” (possui veneziana) e “off” (não possui veneziana)
- `somb` (cm): comprimento do sombreamento horizontal sobre todas as fachadas
  - Uni: 0, 50 e 150
  - Multi: 0 e 120
- `tbsm` (°): média anual da temperatura de bulbo seco
  - Uni/Multi: 17.38, 19.04, 20.91, 22.95, 23.05, 23.92, 26.76, 26.82

Variáveis de saída

- `phft` (%): PHFT do modelo real
- `dif_phft` (%): diferença absoluta entre o PHFT do modelo real e do modelo de referência.
- `cgtt` (kWh/ano): carga térmica total
- `dif_cgtt` (kWh/ano): diferença absoluta entre a CgTT do modelo real e do modelo de referência.

Limitações

É importante ressaltar que os modelos de predição compreendem melhor os casos que os alimentaram durante o treinamento. Portanto, visando gerar resultados mais acurados, recomenda-se a adoção de variáveis de entrada com valores próximos àqueles fornecidos na etapa de treinamento.

Rotina de predição

Definição do ambiente de trabalho

Antes de qualquer operação é preciso carregar os pacotes que serão utilizados e os metamodelos desenvolvidos previamente.

```
  # setup environment
library(caret)
```

```
Loading required package: lattice

Loading required package: ggplot2
```

```
library(dplyr)
```

```

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(purrr)
```

```

Attaching package: 'purrr'

The following object is masked from 'package:caret':

    lift
```

```
# load predictive models
models = readRDS('result/models.rds')
```

As mesmas mensagens que apareceram acima, após o carregamento das bibliotecas, devem surgir no seu console caso as bibliotecas de fato tenham sido carregadas no seu ambiente de trabalho.

Definição das funções de predição

Em seguida serão definidas as funções que realizam as predições de forma automatizada.

Dado que, para cada tipologia e cada variável de saída há um metamodelo (totalizando 8 metamodelos), será definida a função `PredictOutput()`, que seleciona o metamodelo correspondente à variável de saída e à tipolgia desejadas e realiza as predições daquela variável de interesse Essa função possui 4 argumentos: a) `targ` refere-se à variável a ser predita ("phft", "dif_phft", "cgtt" ou "dif_cgtt"); b) dummies trata-se do conjunto de dados de entrada após as transformação das variáveis qualitativas em variáveis lógicas; c) `typo`: tipologia da edificação ("uni" ou "multi"); d) `models` é a lista com os modelos de predição treinados previamente. Após executada, `PredictOutput()` retorna um vetor com os valores da predições.

```
PredOutput = function(targ, dummies, typo, mls) {
  # associate the target and typology with the related model and execute the predictions
  out = capture.output({
    predictions = mls %>%
      pluck(typo, targ) %>%
      predict(newdata = dummies)
  })
  # return the predictions
  return(predictions)
}
```

Em seguida será definida a função `PredPerfUH()`. Essa função tem 2 argumentos: a) `inputs` trata-se de uma lista com o nome dos variáveis de entrada e seus respectivos valores ou do caminho da tabela/csv contendo os inputs; c) `models` novamente éa a lista contendo o conjunto de metamodelos que será utilizado para realizar as predições.

Primeiramente, a função `PredPerfUH()` confere se uma lista com as variáveis de entrada e os seus valores foi definida através do argumento `inputs`. Se uma lista foi inserida com mais de um valor para uma das variáveis de entrada, ela é transformada em um *grid* com os valores listados. Caso não tenha sido inserida uma lista, a rotina confere se foi inserido um caminho para a tabela/csv. Caso nenhum dos dois tenha sido inserido, o código gera um erro. Nesse caso, é preciso conferir o valor que definido para o argumento `inputs`, pois ele provavelmente deve estar errado. Após checar a variável `inputs`, o número de variáveis de entrada é avaliado para que seja definida a tipologia de interesse, sendo que **a tipologia unifamiliar exige valores de 12 variáveis de entrada**, enquanto **a tipologia multifamiliar necessita de 14 parâmetros de entrada**. As transformações das variáveis lógicas são realizadas para que as predições das 4 variáveis de saída ("phft", "dif_phft", "cgtt" e "dif_cgtt") sejam estimadas em um *loop* sobre a função `PredictOutput()`, descrita anteriormente. Por fim, `PredPerfUH()`, retorna uma tabela com os valores das variáveis de entrada e os respectivos variáveis de saída.

```
PredPerfUH = function(inputs, mls = models) {
  # fit inputs into a grid data frame if no table was provided
  if (is.list(inputs)) {
    inputs = expand.grid(inputs)
  } else if (is.character(inputs)) {
    inputs = read.csv(inputs)
  } else {
    stop(paste0('Confira se o argumento "inputs" foi associado a uma lista com as ',
                'variáveis de entrada e os seus valores ou o caminho da tabela/csv ',
                'na sua máquina.  Qualquer valor diferente disso não é aceito.'))
  }
  # check if the number of inputs is correct and define the typology
  ncols = ncol(inputs)
  if (ncols == 12) {
    message('Variáveis de entrada referentes à tipologia unifamiliar.')
    typo = 'uni'
  } else if (ncols == 14) {
    message('Variáveis de entrada referentes à tipologia multifamiliar.')
    typo = 'multi'
  } else {
    stop('Reveja as suas variáveis de entrada. Me parece que elas estão incorretas..!')
  }
  # create dummy variables
  dummies = mls[[c(typo, 'dummy_model')]] %>%
    predict(newdata = mutate(inputs, targ = NA)) %>%
```

```
    data.frame()
  # predict the outputs
  targs = c('phft', 'dif_phft', 'cgtt', 'dif_cgtt')
  predictions = targs %>%
    sapply(PredOutput, dummies, typo, mls) %>%
    as.data.frame() %>%
    round(1)
  # pos-process
  if (nrow(inputs) == 1) {
    # adjust the data into proper dataframe
    data = predictions %>%
      rename(prediction = '.') %>%
      t() %>%
      as.data.frame()
  } else {
    # bind inputs and outputs to build the final data frame
    data = cbind(inputs, predictions)
  }
  # return the data
  return(data)
}
```

Exemplos de aplicação do código

Agora que as bibliotecas e modelos já foram carregadas no ambiente de trabalho e as funções necessárias já foram definidas, basta chamar a função `PredPerfUH()` preenchendo os argumentos necessários da maneira correta.

1. Utilizando uma lista de variáveis de entrada no argumento `inputs`

Defina uma lista de variáveis de entrada e chame a função `PredPerfUH()`. Associe o argumento `inputs` à lista pré-definida no seu ambiente de trabalho.

```
inputs = list(
  area = 39,
  ratio = 0.5,
  pd = 2.5,
  azi = 180,
  comp = 'tv',
  vid = 'simples_fs87',
  abs = 0.4,
  paf = 30,
  fv = 0.45,
  ven = 'off',
  somb = 50,
  tbsm = 23.05
)
data = PredPerfUH(inputs)
```

`Variáveis de entrada referentes à tipologia unifamiliar.`

Imprima os resultados para conferir se tudo saiu como o planejado.

```
  print(data)
```

	phft	dif_phft	cgtt	dif_cgtt
prediction	64.2	7.3	3747.5	-666.6

Lembre-se que também é possível definir variações na lista de variáveis de entrada e o código criará um *grid*, como no exemplo abaixo.

```
inputs = list(
  area = 39,
  ratio = 0.5,
  pd = 2.5,
  azi = c(0, 180),
  comp = 'tv',
  vid = 'simples_fs87',
  abs = c(0.2, 0.8),
  paf = c(10, 50),
  fv = 0.45,
  ven = 'off',
  somb = 50,
  tbsm = 23.05
)
data = PredPerfUH(inputs)
```

```
Variáveis de entrada referentes à tipologia unifamiliar.
```

Imprima os resultados e observe o *grid* foi gerado. o *grid* é na verdade uma simples combinação de todas as variáveis contra todas.

```
  print(data)
```

area	ratio	pd	azi	comp	vid	abs	paf	fv	ven	somb	tbsm	phft	dif_phft	cgtt	dif_cg
39	0.5	2.5	0	tv	simples_fs87	0.2	10	0.45	off	50	23.05	72.7	17.9	2059.2	-2532
39	0.5	2.5	180	tv	simples_fs87	0.2	10	0.45	off	50	23.05	73.3	18.9	1725.5	-2878
39	0.5	2.5	0	tv	simples_fs87	0.8	10	0.45	off	50	23.05	48.3	-7.6	5272.7	610.3
39	0.5	2.5	180	tv	simples_fs87	0.8	10	0.45	off	50	23.05	48.7	-7.2	4817.0	287.0
39	0.5	2.5	0	tv	simples_fs87	0.2	50	0.45	off	50	23.05	77.8	22.0	2611.2	-1882
39	0.5	2.5	180	tv	simples_fs87	0.2	50	0.45	off	50	23.05	78.7	22.5	2449.0	-2197
39	0.5	2.5	0	tv	simples_fs87	0.8	50	0.45	off	50	23.05	62.5	6.5	4720.4	199.7
39	0.5	2.5	180	tv	simples_fs87	0.8	50	0.45	off	50	23.05	62.6	7.4	4394.8	-152.

2. Inserção do caminho da tabela/csv através do argumento `inputs`

Defina o caminho da tabela/csv na sua máquina e chame a função `PredPerfUH()`, associando o caminho pré-definido ao argumento `inputs`.

```
  inputs = './inputs.csv'
  data = PredPerfUH(inputs)
```

Variáveis de entrada referentes à tipologia multifamiliar.

```
nrow(data)
```

```
[1] 27648
```

Repare que a tabela de *inputs* inserida nesse exemplo possui 27668 observações (linhas), portanto, apenas as 20 primeiras linhas serão impressas.

area	ratio	pav	solo	exp	azi	comp	vid	abs	paf	fv	ven	somb	tbsm	phft	dif_phft
34	0.5	terreo	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	88.9	-6.7
52	0.5	terreo	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	86.7	-5.4
34	1.0	terreo	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	89.2	-6.8
52	1.0	terreo	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	86.7	-6.8
34	2.0	terreo	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	85.2	-12.3
52	2.0	terreo	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	83.0	-11.5
34	0.5	inter	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	81.3	-9.9
52	0.5	inter	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	78.6	-9.6
34	1.0	inter	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	81.0	-9.4
52	1.0	inter	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	79.2	-9.9
34	2.0	inter	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	78.0	-12.4
52	2.0	inter	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	76.9	-12.3
34	0.5	cob	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	67.1	-15.0
52	0.5	cob	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	61.5	-15.6
34	1.0	cob	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	65.9	-15.3
52	1.0	cob	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	62.0	-15.9
34	2.0	cob	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	64.7	-18.0
52	2.0	cob	solo	canto	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	61.4	-18.0
34	0.5	terreo	solo	meio	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	91.6	-5.6
52	0.5	terreo	solo	meio	0	ref	simples_fs87	0.2	10	0.45	on	0	17.38	90.6	-4.0

Lembre-se sempre de checar se você está inserindo todas as variáveis relativas à tipologia de interesse (uni ou multifamiliar) e se os valores que foram inseridos estão de acordo com aqueles utilizados na etapa de treinamento do metamodelo, descrita no item "Características dos metamodelos".

Salvando os resultados

uma das maneiras mais simples de salvar os resultados de um *dataframe* no **R** é através da função `write.csv()`. Primeiro defina o caminho onde quer salvar os arquivos na sua máquina e, por fim, chame a função `write.csv()`, inserindo os resultados das predições no primeiro argumento e o caminho pré-definido no segundo argumento. A função `write.csv()` conta com outros argumentos, que podem ser listados utilizando `args(write.csv)`. A documentação da função `write.csv()` pode ser encontrada nesse link.

```
data_path = '/Home/RodolfoVeiga/Desktop/'
write.csv(data, data_path)
```