Projeto de Data Science

Modelagem Preditiva em IoT

Previsão de Uso de Energia

Fellipe Augusto Soares Silva

2020

Sumário

Lista de Figuras

Resumo

O presente projeto foi desenvolvido com a finalidade de implementar modelos preditivos para prever o consumo de energia com base nos dados de sensores IoT wireless de uma residência.

Foram desenvolvidos 3 modelos preditivos de aprendizagem de máquina (Machine Learning) supervisionada com base em dados históricos: lm, Random Forest e SVM. Estes dados foram coletados por sensores wireless por um período de 10 minutos por aproximadamente 5 meses, sendo que as condições de temperatura e umidade relativa da residência foram adquiridas por uma rede de sensores sem fio ZigBee.

Cada sensor wireless foi configurado para transmitir as condições de temperatura e umidade por um período de 3 minutos e em seguida foi calculado a média dos dados para períodos de 10 minutos. Os dados de consumo de energia foram registrados de 10 em 10 minutos e o tempo da estação meteorológica mais próxima, onde está localizado o Aeroporto de Chievres na Bélgica, foi baixado de um conjunto de dados públicos.

Com os dados fornecidos foram desenvolvidas análises exploratórias identificando períodos de maior consumo, ambientes com menos variações, outliers indicando possíveis problemas em eletrodomésticos; engenharia de recursos; identificação das variáveis mais relevantes; correlação; controle de underfitting e overfitting para o modelo de aprendizagem de máquina fornecer maior eficiência; treino e previsão do modelo; medidas de erro entre os valores previstos e os observados previamente através da métrica AUC; além de gráficos exemplificando cada etapa do processo de Data Science.

Por fim, ao analisar toda a problemática fornecida pelos dados, um novo algoritmo de Machine Learning foi implementado: K – Means. Com ele separamos os dados previstos de consumo em 9 categorias diferentes o que ajudou a entender períodos de maior consumo e consequentemente oferecer melhores insights ao usuário.

Palavras – Chave: Data Science, IoT, Internet of Things, Machine Learning, Linear Model, Random Forest, Support Vector Machine, K – Means, Análises Exploratórias, AUC, Area Under the Curve.

1 – Introdução

Este projeto foi desenvolvido em linguagem de programação R utilizando o RStudio[1] como plataforma de desenvolvimento e teste do script[2].

Foram utilizadas também algumas bibliotecas com pacotes essenciais para o desenvolvimento do código, como por exemplo o ggplot para criação de gráficos, o caret para o aprendizado de máquina, o dplyr para manipulação de dados, dentre outros que serão explicados nos capítulos a seguir.

Como todo projeto de Data Science, antes de iniciar a parametrização do modelo preditivo é preciso realizar a análise exploratória e conhecer o conjunto de dados, ou *dataset*. Durante este procedimento, algumas informações que poderiam apresentar problemas ao modelo preditivo foram transformadas em novos dados e com isso a análise do Business Intelligence apresentou diversas oportunidades de negócios, também evidenciadas no capítulo em questão.

Para concluir, serão apresentados todos os itens do script, comentados linha a linha e apresentando análises gráficas para complementar o entendimento do desenvolvimento deste projeto de Data Science.

O início da solução de qualquer problema de negócio está no entendimento do problema propriamente dito que será apresentado no capítulo a seguir.

[1] Rstudio é um software livre de ambiente de desenvolvimento integrado para R, uma linguagem de programação para gráficos e cálculos estatísticos.

[2] Script é um conjunto de instruções em código, ou seja, escritas em linguagem de computador para que o mesmo execute diversas funções no interior de um programa.

2 – Problema de Negócio

Internet das Coisas, ou apenas IoT, refere – se à uma rede de objetos físicos interconectados com a internet e capaz de reunir e transmitir dados. Dentre os objetos podemos ter eletrodomésticos, termostatos, carros, televisores, iluminação, wearebles, sensores de umidade, temperatura e pressão, dentre outros.

Neste projeto iremos desenvolver a análise de consumo de energia de eletrodomésticos de uma residência e realizar a construção de modelos preditivos para prever o consumo baseado em dados históricos. Teremos em nosso conjunto de dados medições de temperatura e umidade de uma rede sem fio de diferentes cômodos da residência, previsão do tempo de uma estação meteorológica próxima à residência, e energia consumida pelas luminárias.

IoT traz um conceito atual e transformador quando falamos de conexão entre objetos físicos através de sensores, chips e softwares. Neste projeto será possível observar como a análise exploratória pode trazer uma série de informações valiosas para os moradores/usuários desta ferramenta, como, momentos de maior consumo de energia baseado na variação da temperatura externa, ou também qual cômodo consome mais energia podendo inclusive identificar se em determinado mês esse consumo está ultrapassando os limites pré – estabelecidos indicando mal funcionamento de algum eletrodoméstico. As possibilidades são muitas.

Com a inserção de Machine Learning levamos o projeto para outro nível adicionando inteligência às análises, fornecendo ao usuário qualidade de vida e otimização de tarefas rotineiras. Não é o caso deste projeto mas poderíamos usar Machine Learning para reconhecer hábitos e preferências do usuário e começar um dia com a ativação automática da cafeteira, fornecer notificações da agenda do dia, notícias da manhã, informações sobre o clima, como foi a noite de sono, músicas para caminhar, informações em tempo real sobre preços de produtos de interesse, enfim, como disse as possibilidades são muitas, tudo depende do problema de negócio.

Para nosso problema de negócio vamos focar em sensores mais básicos de coleta de informação de temperatura de cômodos da residência, sensores externos e sensores de clima. Vejamos no próximo capítulo a estrutura do nosso dataset.

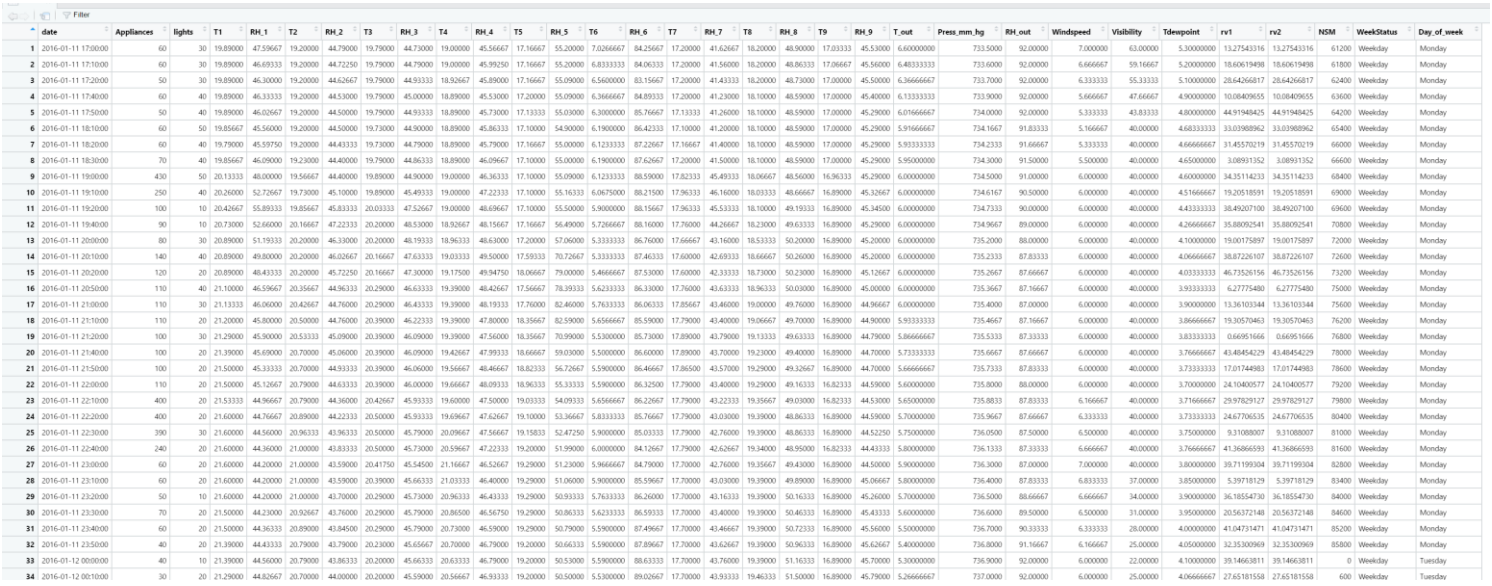
3 – Datasets utilizados

Para esse problema de negócio teremos os seguintes conjuntos de dados:

- train.csv;
- test.csv;

3.1 train / test

Este dataset está estruturado da seguinte maneira:



	date	Appliances	lights	T1	RH_1	T2	RH_2	T3	RH_3	T4	RH_4	T5	RH_5	T6	RH_6	T7	RH_7	T8	RH_8	T9	RH_9	T_out	Press_mm_hg	RH_out	Windspeed	Visibility	Tdewpoint	rv1	rv2	NSM	WeekStatus	Day_of_week
1	2016-01-11 17:00:00	60	30	19.89000	47.59667	19.20000	44.79000	19.79000	44.73000	19.00000	45.56667	17.16667	55.20000	7.026667	84.25667	17.20000	41.62667	18.20000	48.90000	17.03333	45.53000	6.60000000	733.5000	92.00000	7.000000	63.00000	5.300000000	13.27543316	13.27543316	61200	Weekday	Monday
2	2016-01-11 17:10:00	60	30	19.89000	46.69333	19.20000	44.72250	19.79000	44.79000	19.00000	45.99250	17.16667	55.20000	6.8333333	84.06333	17.20000	41.56000	18.20000	48.86333	17.06667	45.56000	6.48333333	733.6000	92.00000	6.666667	59.16667	5.200000000	18.60619498	18.60619498	61800	Weekday	Monday
3	2016-01-11 17:20:00	50	30	19.89000	46.30000	19.20000	44.62667	19.79000	44.93333	18.92667	45.89000	17.16667	55.09000	6.5600000	83.15667	17.20000	41.43333	18.20000	48.73000	17.00000	45.50000	6.36666667	733.7000	92.00000	6.333333	55.33333	5.100000000	28.64266817	28.64266817	62400	Weekday	Monday
4	2016-01-11 17:40:00	60	40	19.89000	46.33333	19.20000	44.53000	19.79000	45.00000	18.89000	45.53000	17.20000	55.09000	6.3666667	84.89333	17.20000	41.23000	18.10000	48.59000	17.00000	45.40000	6.13333333	733.9000	92.00000	5.666667	47.66667	4.900000000	10.08409655	10.08409655	63600	Weekday	Monday
5	2016-01-11 17:50:00	50	40	19.89000	46.02667	19.20000	44.50000	19.79000	44.93333	18.89000	45.73000	17.13333	55.03000	6.3000000	85.76667	17.13333	41.26000	18.10000	48.59000	17.00000	45.29000	6.01666667	734.0000	92.00000	5.333333	43.83333	4.800000000	44.91948425	44.91948425	64200	Weekday	Monday
6	2016-01-11 18:10:00	60	50	19.85667	45.56000	19.20000	44.50000	19.73000	44.90000	18.89000	45.86333	17.10000	54.90000	6.1900000	86.42333	17.10000	41.20000	18.10000	48.59000	17.00000	45.29000	5.91666667	734.1667	91.83333	5.166667	40.00000	4.683333333	33.03988962	33.03988962	65400	Weekday	Monday
7	2016-01-11 18:20:00	60	40	19.79000	45.59750	19.20000	44.43333	19.73000	44.79000	18.89000	45.79000	17.16667	55.00000	6.1233333	87.22667	17.16667	41.40000	18.10000	48.59000	17.00000	45.29000	5.93333333	734.2333	91.66667	5.333333	40.00000	4.666666667	31.45570219	31.45570219	66000	Weekday	Monday
8	2016-01-11 18:30:00	70	40	19.85667	46.09000	19.23000	44.40000	19.79000	44.86333	18.89000	46.09667	17.10000	55.00000	6.1900000	87.62667	17.20000	41.50000	18.10000	48.59000	17.00000	45.29000	5.95000000	734.3000	91.50000	5.500000	40.00000	4.650000000	3.08931352	3.08931352	66600	Weekday	Monday
9	2016-01-11 19:00:00	430	50	20.13333	48.00000	19.56667	44.40000	19.89000	44.90000	19.00000	46.36333	17.10000	55.09000	6.1233333	88.59000	17.82333	45.49333	18.06667	48.56000	16.96333	45.29000	6.00000000	734.5000	91.00000	6.000000	40.00000	4.600000000	34.35114233	34.35114233	68400	Weekday	Monday
10	2016-01-11 19:10:00	250	40	20.26000	52.72667	19.73000	45.10000	19.89000	45.49333	19.00000	47.22333	17.10000	55.16333	6.0675000	88.21500	17.96333	46.16000	18.03333	48.66667	16.89000	45.32667	6.00000000	734.6167	90.50000	6.000000	40.00000	4.516666667	19.20518591	19.20518591	69000	Weekday	Monday
11	2016-01-11 19:20:00	100	10	20.42667	55.89333	19.85667	45.83333	20.03333	47.52667	19.00000	48.69667	17.10000	55.50000	5.9000000	88.15667	17.96333	45.53333	18.10000	49.19333	16.89000	45.34500	6.00000000	734.7333	90.00000	6.000000	40.00000	4.433333333	38.49207100	38.49207100	69600	Weekday	Monday
12	2016-01-11 19:40:00	90	10	20.73000	52.66000	20.16667	47.22333	20.20000	48.53000	18.92667	48.15667	17.16667	56.49000	5.7266667	88.16000	17.76000	44.26667	18.23000	49.63333	16.89000	45.29000	6.00000000	734.9667	89.00000	6.000000	40.00000	4.266666667	35.88092541	35.88092541	70800	Weekday	Monday
13	2016-01-11 20:00:00	80	30	20.89000	51.19333	20.20000	46.33000	20.20000	48.19333	18.96333	48.63000	17.20000	57.06000	5.3333333	86.76000	17.66667	43.16000	18.53333	50.20000	16.89000	45.20000	6.00000000	735.2000	88.00000	6.000000	40.00000	4.100000000	19.00175897	19.00175897	72000	Weekday	Monday
14	2016-01-11 20:10:00	140	40	20.89000	49.80000	20.20000	46.02667	20.16667	47.63333	19.03333	49.50000	17.59333	70.72667	5.3333333	87.46333	17.60000	42.69333	18.66667	50.26000	16.89000	45.20000	6.00000000	735.2333	87.83333	6.000000	40.00000	4.066666667	38.87226107	38.87226107	72600	Weekday	Monday
15	2016-01-11 20:20:00	120	20	20.89000	48.43333	20.20000	45.72250	20.16667	47.30000	19.17500	49.94750	18.06667	79.00000	5.4666667	87.53000	17.60000	42.33333	18.73000	50.23000	16.89000	45.12667	6.00000000	735.2667	87.66667	6.000000	40.00000	4.033333333	46.73526156	46.73526156	73200	Weekday	Monday
16	2016-01-11 20:50:00	110	40	21.10000	46.59667	20.35667	44.96333	20.29000	46.63333	19.39000	48.42667	17.56667	78.39333	5.6233333	86.33000	17.76000	43.63333	18.96333	50.03000	16.89000	45.00000	6.00000000	735.3667	87.16667	6.000000	40.00000	3.933333333	6.27775480	6.27775480	75000	Weekday	Monday
17	2016-01-11 21:00:00	110	30	21.13333	46.06000	20.42667	44.76000	20.29000	46.43333	19.39000	48.19333	17.76000	82.46000	5.7633333	86.06333	17.85667	43.46000	19.00000	49.76000	16.89000	44.96667	6.00000000	735.4000	87.00000	6.000000	40.00000	3.900000000	13.36103344	13.36103344	75600	Weekday	Monday
18	2016-01-11 21:10:00	110	20	21.20000	45.80000	20.50000	44.76000	20.39000	46.22333	19.39000	47.80000	18.35667	82.59000	5.6566667	85.59000	17.79000	43.40000	19.06667	49.70000	16.89000	44.90000	5.93333333	735.4667	87.16667	6.000000	40.00000	3.866666667	19.30570463	19.30570463	76200	Weekday	Monday
19	2016-01-11 21:20:00	100	30	21.29000	45.90000	20.53333	45.09000	20.39000	46.09000	19.39000	47.56000	18.35667	70.99000	5.5300000	85.73000	17.89000	43.79000	19.13333	49.63333	16.89000	44.79000	5.86666667	735.5333	87.33333	6.000000	40.00000	3.833333333	0.66951666	0.66951666	76800	Weekday	Monday
20	2016-01-11 21:40:00	100	20	21.39000	45.69000	20.70000	45.06000	20.39000	46.09000	19.42667	47.99333	18.66667	59.03000	5.5000000	86.60000	17.89000	43.70000	19.23000	49.40000	16.89000	44.70000	5.73333333	735.6667	87.66667	6.000000	40.00000	3.766666667	43.48454229	43.48454229	78000	Weekday	Monday
21	2016-01-11 21:50:00	100	20	21.50000	45.33333	20.70000	44.93333	20.39000	46.06000	19.56667	48.46667	18.82333	56.72667	5.5900000	86.46667	17.86500	43.57000	19.29000	49.32667	16.89000	44.70000	5.66666667	735.7333	87.83333	6.000000	40.00000	3.733333333	17.01744983	17.01744983	78600	Weekday	Monday
22	2016-01-11 22:00:00	110	20	21.50000	45.12667	20.79000	44.63333	20.39000	46.00000	19.66667	48.09333	18.96333	55.33333	5.5900000	86.32500	17.79000	43.40000	19.29000	49.16333	16.82333	44.59000	5.60000000	735.8000	88.00000	6.000000	40.00000	3.700000000	24.10400577	24.10400577	79200	Weekday	Monday
23	2016-01-11 22:10:00	400	20	21.53333	44.96667	20.79000	44.36000	20.42667	45.93333	19.60000	47.50000	19.03333	54.09333	5.6566667	86.22667	17.79000	43.22333	19.35667	49.03000	16.82333	44.53000	5.65000000	735.8833	87.83333	6.166667	40.00000	3.716666667	29.97829127	29.97829127	79800	Weekday	Monday
24	2016-01-11 22:20:00	400	20	21.60000	44.76667	20.89000	44.22333	20.50000	45.93333	19.69667	47.62667	19.10000	53.36667	5.8333333	85.76667	17.79000	43.03000	19.39000	48.86333	16.89000	44.59000	5.70000000	735.9667	87.66667	6.333333	40.00000	3.733333333	24.67706535	24.67706535	80400	Weekday	Monday
25	2016-01-11 22:30:00	390	30	21.60000	44.56000	20.96333	43.96333	20.50000	45.79000	20.09667	47.56667	19.15833	52.47250	5.9000000	85.03333	17.79000	42.76000	19.39000	48.86333	16.89000	44.52250	5.75000000	736.0500	87.50000	6.500000	40.00000	3.750000000	9.31088007	9.31088007	81000	Weekday	Monday
26	2016-01-11 22:40:00	240	20	21.60000	44.36000	21.00000	43.83333	20.50000	45.73000	20.59667	47.22333	19.20000	51.99000	6.0000000	84.12667	17.79000	42.62667	19.34000	48.95000	16.82333	44.43333	5.80000000	736.1333	87.33333	6.666667	40.00000	3.766666667	41.36866593	41.36866593	81600	Weekday	Monday
27	2016-01-11 23:00:00	60	20	21.60000	44.20000	21.00000	43.59000	20.41750	45.54500	21.16667	46.52667	19.29000	51.23000	5.9666667	84.79000	17.70000	42.76000	19.35667	49.43000	16.89000	44.50000	5.90000000	736.3000	87.00000	7.000000	40.00000	3.800000000	39.71199304	39.71199304	82800	Weekday	Monday
28	2016-01-11 23:10:00	60	20	21.60000	44.20000	21.00000	43.59000	20.39000	45.66333	21.03333	46.40000	19.29000	51.06000	5.9000000	85.59667	17.70000	43.03000	19.39000	49.89000	16.89000	45.06667	5.80000000	736.4000	87.83333	6.833333	37.00000	3.850000000	5.39718129	5.39718129	83400	Weekday	Monday
29	2016-01-11 23:20:00	50	10	21.60000	44.20000	21.00000	43.70000	20.29000	45.73000	20.96333	46.43333	19.29000	50.93333	5.7633333	86.26000	17.70000	43.16333	19.39000	50.16333	16.89000	45.26000	5.70000000	736.5000	88.66667	6.666667	34.00000	3.900000000	36.18554730	36.18554730	84000	Weekday	Monday
30	2016-01-11 23:30:00	70	20	21.50000	44.23000	20.92667	43.76000	20.29000	45.79000	20.86500	46.56750	19.29000	50.86333	5.6233333	86.59333	17.70000	43.40000	19.39000	50.46333	16.89000	45.43333	5.60000000	736.6000	89.50000	6.500000	31.00000	3.950000000	20.56372148	20.56372148	84600	Weekday	Monday
31	2016-01-11 23:40:00	60	20	21.50000	44.36333	20.89000	43.84500	20.29000	45.79000	20.73000	46.59000	19.29000	50.79000	5.5900000	87.49667	17.70000	43.46667	19.39000	50.72333	16.89000	45.56000	5.50000000	736.7000	90.33333	6.333333	28.00000	4.000000000	41.04731471	41.04731471	85200	Weekday	Monday
32	2016-01-11 23:50:00	40	20	21.39000	44.43333	20.79000	43.79000	20.23000	45.65667	20.70000	46.79000	19.20000	50.66333	5.5900000	87.89667	17.70000	43.62667	19.39000	50.96333	16.89000	45.62667	5.40000000	736.8000	91.16667	6.166667	25.00000	4.050000000	32.35300969	32.35300969	85800	Weekday	Monday
33	2016-01-12 00:00:00	40	10	21.39000	44.56000	20.79000	43.86333	20.20000	45.66333	20.63333	46.79000	19.20000	50.53000	5.5900000	88.63333	17.70000	43.76000	19.39000	51.16333	16.89000	45.70000	5.30000000	736.9000	92.00000	6.000000	22.00000	4.100000000	39.14663811	39.14663811	0	Weekday	Tuesday
34	2016-01-12 00:10:00	30	20	21.29000	44.82667	20.70000	44.00000	20.20000	45.59000	20.56667	46.93333	19.20000	50.50000	5.5300000	89.02667	17.70000	43.93333	19.46333	51.50000	16.89000	45.79000	5.26666667	737.0000	92.00000	6.000000	25.00000	4.066666667	27.65181558	27.65181558	600	Weekday	Tuesday

Figura 1 - Dataset train / test

Como temos muitas variáveis (32) vou utilizar o Dicionário de Dados para explicar a estruturação de cada variável em nosso conjunto de dados, conforme segue no próximo capítulo.

4 – Dicionário de Dados

Variável	Significado
date	Data no formato ano-mês-dia hora:minutos:segundos
Appliances	Consumo de energia em Wh (Watt-Hora). Esta é a variável objeto de estudo para previsões.
lights	Consumo de energia de luminárias (Wh)
T1	Temperatura na Cozinha (em Celsius)
RH1	Umidade Relativa na Cozinha (em %)
T2	Temperatura na Sala de Estar (em Celsius)
RH2	Umidade Relativa na Sala de Estar (em %)
T3	Temperatura na Lavanderia (em Celsius)
RH3	Umidade Relativa na Lavanderia (em %)
T4	Temperatura no Escritório (em Celsius)
RH4	Umidade Relativa no Escritório (em %)
T5	Temperatura no Banheiro (em Celsius)
RH5	Umidade Relativa no Banheiro (em %)
T6	Temperatura Externa Lado Norte (em Celsius)
RH6	Umidade Relativa Externa Lado Norte (em %)
T7	Temperatura na Sala de Passar Roupa (em Celsius)
RH7	Umidade Relativa na Sala de Passar Roupa (em %)
T8	Temperatura no Quarto do Adolescente (em Celsius)
RH8	Umidade Relativa no Quarto do Adolescente (em %)
T9	Temperatura no Quarto dos Pais (em Celsius)
RH9	Umidade Relativa no Quarto dos Pais (em %)
T_out	Temperatura Externa (Medidas da Estação de Chievres) (em Celsius)
Press_mm_hg	Pressão (Medidas da Estação de Chievres)
RH_out	Umidade Relativa Externa (Medidas da Estação de Chievres) (em %)

Windspeed	Velocidade do Vento (Medidas da Estação de Chievres) (em m/s)
Visibility	Visibilidade (Medidas da Estação de Chievres) (em km)
Tdewpoint	Ponto de Saturação (Medidas da Estação de Chievres) (em Celsius)
rv1	Variável Randômica
rv2	Variável Randômica
NSM	Número sequencial de medição
WeekStatus	Indicativo de Dia da Semana ou Final de Semana
Day_of_week	Indicativo de Segunda à Domingo

Figura 2 - Dicionário de Dados

Temos ao todo 20.000 observações em ambos datasets para cada uma das 32 variáveis apresentadas acima.

O conjunto de dados foi coletado por um período de 10 minutos por aproximadamente 5 meses, sendo que as condições de temperatura e umidade relativa da residência foram adquiridas por uma rede de sensores sem fio ZigBee[3].

Cada sensor wireless foi configurado para transmitir as condições de temperatura e umidade por um período de 3 minutos e em seguida foi calculado a média dos dados para períodos de 10 minutos.

Os dados de consumo de energia foram registrados de 10 em 10 minutos e o tempo da estação meteorológica mais próxima, onde está localizado o Aeroporto de Chievres na Bélgica, foi baixado de um conjunto de dados públicos.

Além disso temos também duas variáveis aleatórias (rv1 e rv2) que demonstraram ser pouco preditivas e foram removidas durante o processo de feature engineering, ou Engenharia de Atributos.

Nos capítulos seguintes daremos início à fase de análise exploratória utilizando técnicas de ciência de dados trazendo insights valiosos para os usuários e empresas que administram projetos de IoT.

A seguir vamos abordar as bibliotecas utilizadas.

[3] https://zigbeealliance.org/

5 – Bibliotecas utilizadas

Utilizar bibliotecas é imprescindível para o bom andamento de um projeto de Data Science. Neste projeto foram utilizadas as seguintes bibliotecas:

- readr[4]: biblioteca para leitura de datasets;
- dplyr[5]: facilitador de manipulação de dados;
- ggplot2[6]: biblioteca utilizada para plotar gráficos;
- RColorBrewer[7]: biblioteca para alterar cor dos gráficos;
- e1071[8]: biblioteca com diversas funções para análise de dados, incluindo algoritmos de Machine Learning
- caret[9]: outra biblioteca de Machine Learning;
- ROCR[10]: pacote para auxiliar na construção de métricas de avaliação;
- randomForest[11]: outro algoritmo de Machine Learning;
- lattice[12]: recurso de visualização de dados
- knitr[13]: pacote para relatórios dinâmicos
- car[14]: função para aplicação de comparações entre modelos de regressão
- MASS[15]: pacote estatístico para análise de multicolinearidade com stepAIC
- gains[16]: pacote auxiliar na avaliação de modelos como o SVM
- pROC[17]: apresenta análises de curva ROC
- multiROC[18]: pacote auxiliar de análise de curva ROC
- clustertend[19]: pacote estatístico para análise da tendência de clusterização de um conjunto de dados
- NbClust[20]: identifica o número ideal de cluster nos dados analisados
- ggfortify[21]: pacote auxiliar na plotagem de clusters
- cluster[22]: métodos para análise de clusters

[4] https://cran.r-project.org/web/packages/readr/index.html
[5] https://www.rdocumentation.org/packages/dplyr/versions/0.7.8
[6] https://www.rdocumentation.org/packages/ggplot2/versions/3.2.1
[7] https://www.rdocumentation.org/packages/RColorBrewer/versions/1.1-2/topics/RColorBrewer
[8] https://cran.r-project.org/web/packages/e1071/index.html
[9] http://topepo.github.io/caret/index.html
[10] https://cran.r-project.org/web/packages/ROCR/index.html
[11] https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/randomForest
[12] https://cran.r-project.org/web/packages/lattice/index.html
[13] https://cran.r-project.org/web/packages/knitr/index.html
[14] https://cran.r-project.org/web/packages/car/index.html
[15] https://cran.r-project.org/web/packages/MASS/index.html
[16] https://cran.r-project.org/web/packages/gains/index.html
[17] https://cran.r-project.org/web/packages/pROC/index.html
[18] https://cran.r-project.org/web/packages/multiROC/index.html
[19] https://cran.r-project.org/web/packages/clustertend/index.html
[20] https://cran.r-project.org/web/packages/NbClust/index.html
[21] https://cran.r-project.org/web/packages/ggfortify/index.html
[22] https://cran.r-project.org/web/packages/cluster/index.html

6 – Análise Exploratória

A análise exploratória é fundamental antes de realizar qualquer procedimento com os dados fornecidos pois é a partir dela que podemos entender onde temos melhores variáveis, onde temos problemas, onde temos oportunidades de melhoria, e dessa maneira podemos gerar uma série de conclusões importantes para dar continuidade no processo de aprendizado de máquina.

Vamos, inicialmente, explorar a relação de consumo de energia entre dias de semana e finais de semana.

6.1 Período de maior consumo de energia

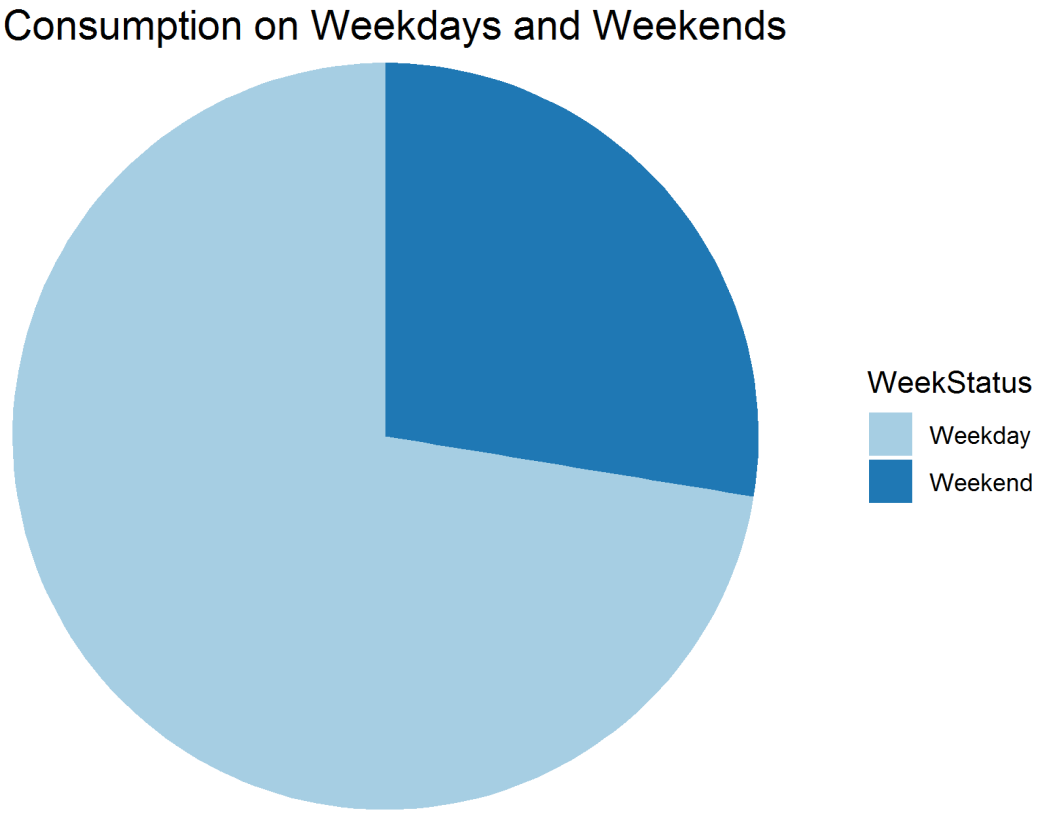


Figura 3 - Relação de Consumo de Energia

A figura acima demonstra maior consumo de energia nos dias de semana, vamos explorar então qual a quantidade consumida nestes períodos.

6.2 Consumo de energia em Wh por período

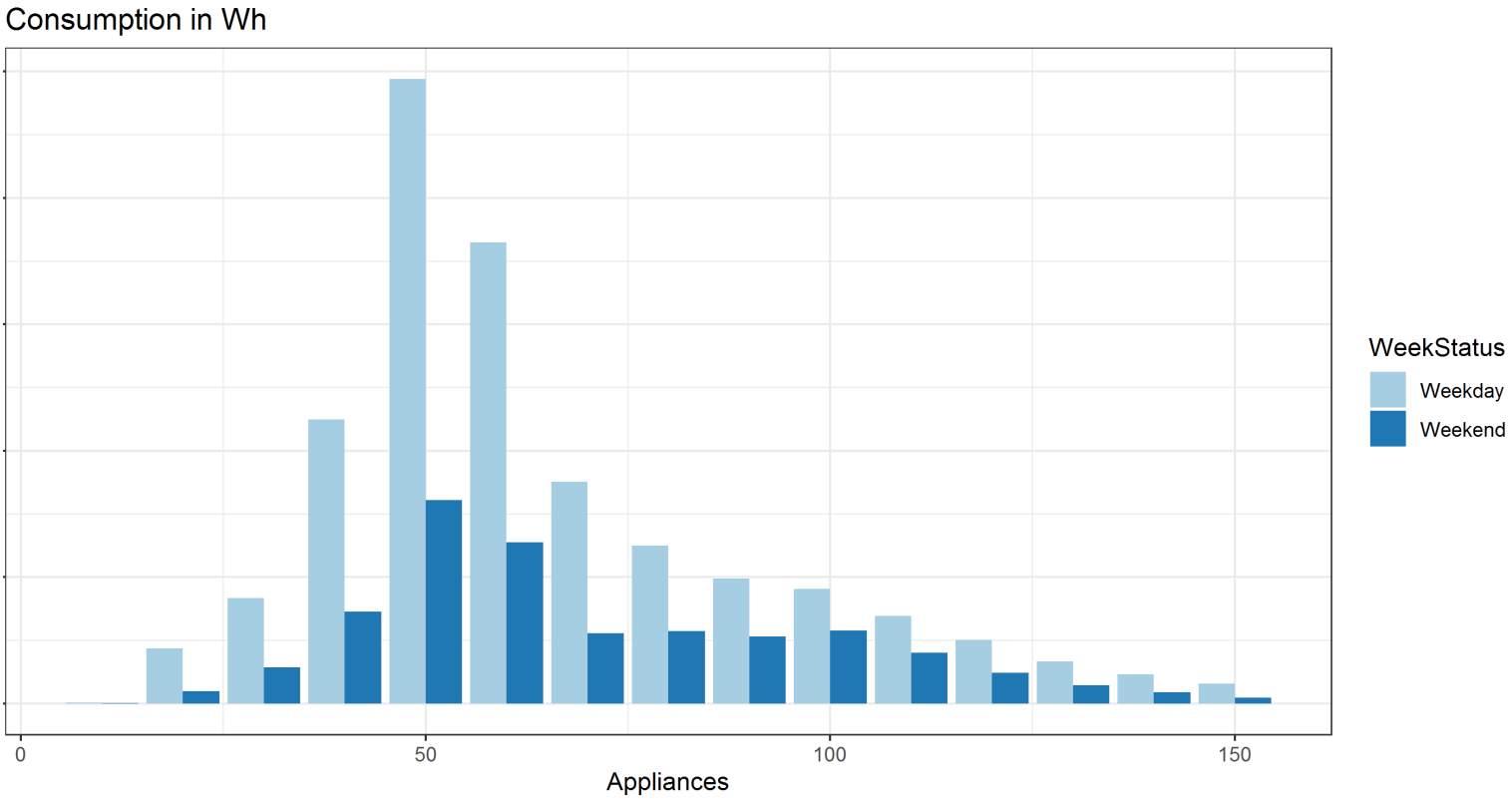


Figura 4 - Consumo em Wh

Como sabemos o maior consumo de energia ocorre em dias de semana, e de acordo com a figura acima, é possível observar que o maior consumo está em 50 Wh tanto para dias de semana quanto para finais de semana. Com essa informação é possível identificar padrões e analisar quais eletrodomésticos consomem mais energia na residência.

Como aqui não temos informações detalhadas sobre cada eletrodoméstico na residência, vou explorar como é a relação de consumo por ambiente onde temos sensores instalados, ou seja, qual a relação de consumo de energia com o aumento de temperatura nos ambientes mais utilizados como na cozinha e na sala de estar, por exemplo.

6.3 Relação Consumo x Temperatura (Living Room)

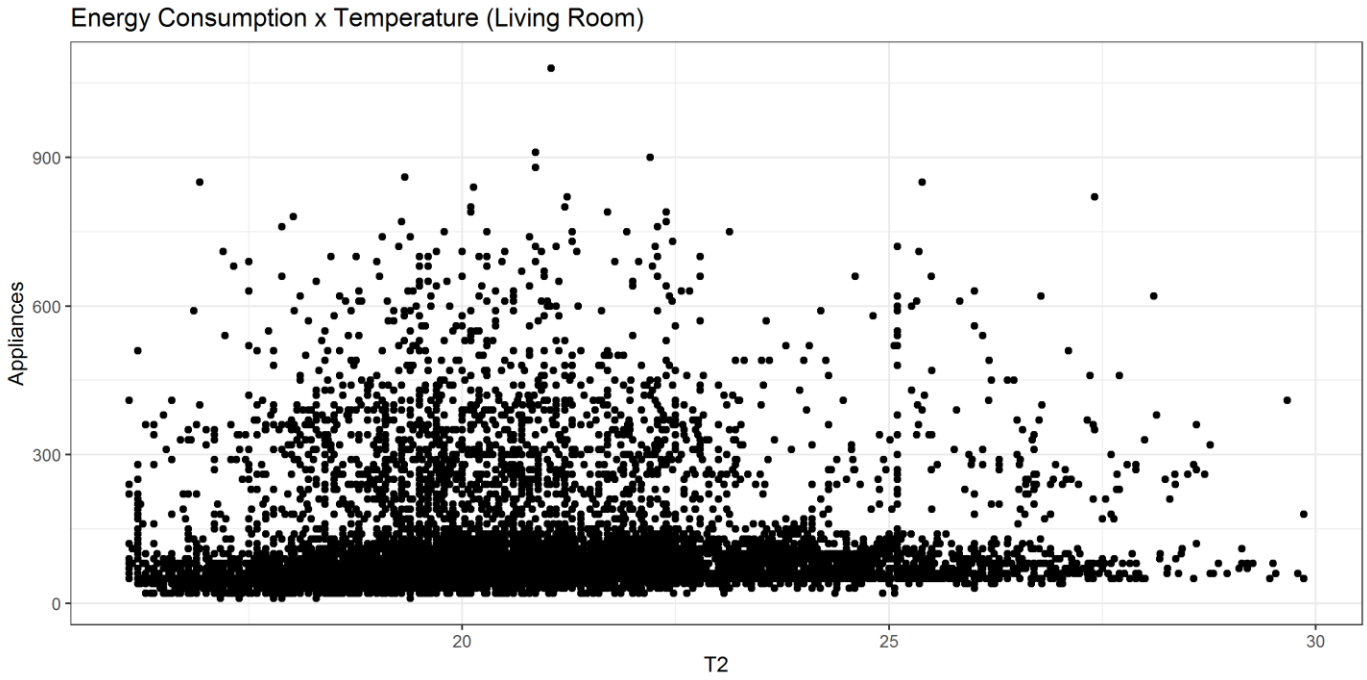


Figura 5 - Relação Consumo x Temperatura (Living Room) I

Podemos observar na figura acima que o aumento da temperatura na sala de estar não possui correlação positiva nem negativa com o aumento do consumo de energia, porém conseguimos identificar possíveis 3 separações nestes dados:

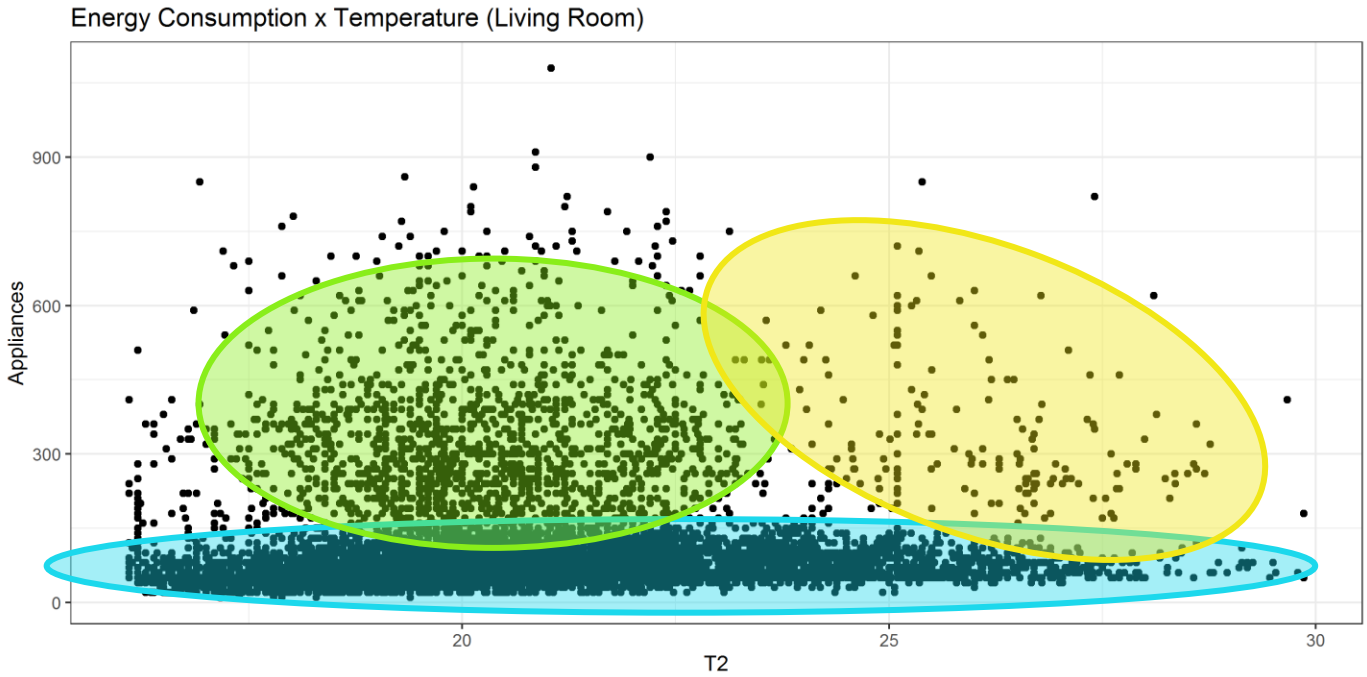


Figura 6 - Separação de Padrões de Consumo

Na cor azul temos um cluster com variação de temperatura entre -5ºC e 30ºC onde há maior concentração de consumo, ou seja, entre 0 e 150 Wh. É possível observar também que temos pontos de concentração no cluster verde, ou seja, em média a temperatura da sala de estar está mais próxima de 20ºC e em média este ambiente contribui para um consumo de aproximadamente 30 Wh.

A representação do cluster em amarelo pode indicar duas coisas, ou temos medições erradas, com ruídos, sensor descalibrado; ou temos eletrodomésticos apresentando sinais de problemas pois são evidentemente pontos outliers fugindo do padrão de consumo médio da residência. Não vou explorar outliers mas poderia certamente ser um ponto de atenção para fornecimento de informações úteis ao usuário com relação à necessidade de manutenção de equipamentos.

A fim de simplificar as informações dos gráficos anteriores deste tópico, vou reduzir as medições de temperaturas à apenas 1 casa decimal, conforme segue:

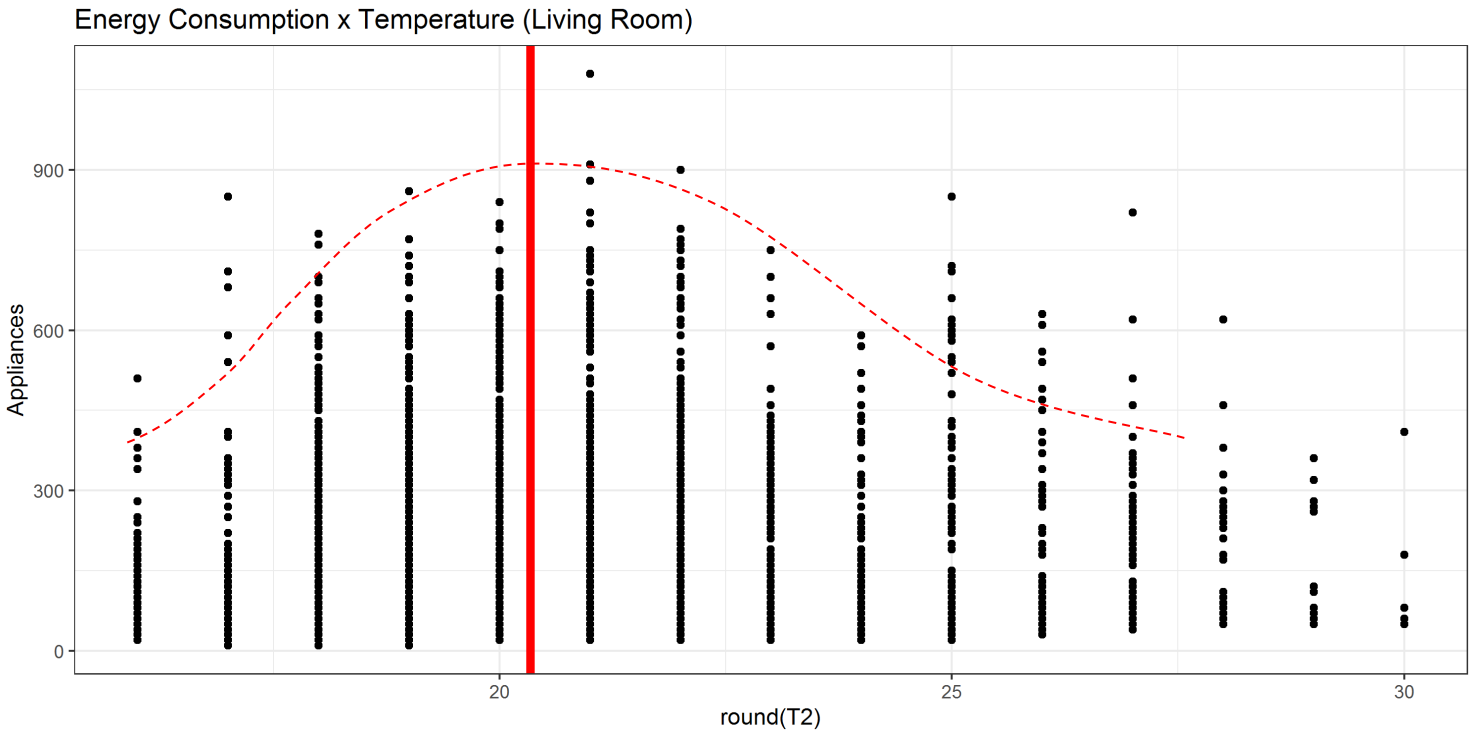


Figura 7 - Relação Consumo x Temperatura (Living Room) II

É possível observar na linha tracejada vermelha que agora temos um padrão mais observável onde a temperatura média (linha vermelha vertical contínua) de aproximadamente 20ºC indica maior consumo de energia. Em outras palavras, quando a temperatura da sala de estar está próxima de 20ºC haverá maior consumo de energia geral.

Como dito anteriormente, temos outliers em nosso conjunto de dados de medição, observe na próxima figura.

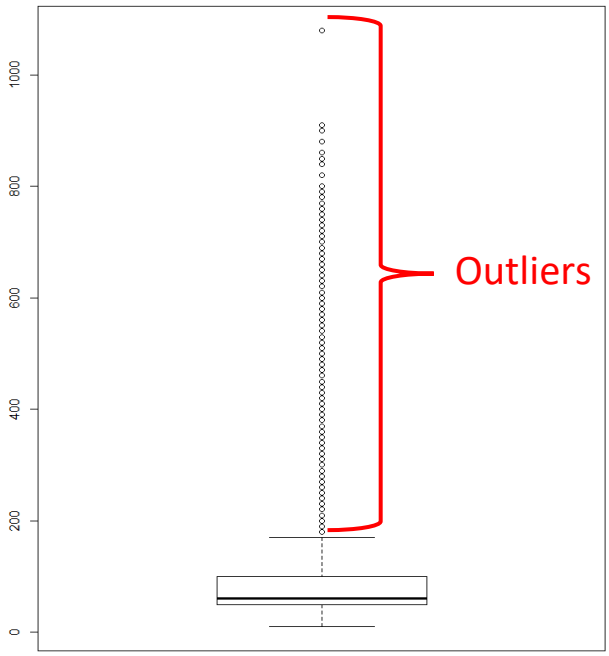


Figura 8 - Outliers (erros de medição ou problemas em eletrodomésticos)

Vamos estudar mais um ambiente de alta frequência de uso, a cozinha.

6.4 Relação Consumo x Temperatura (Kitchen)

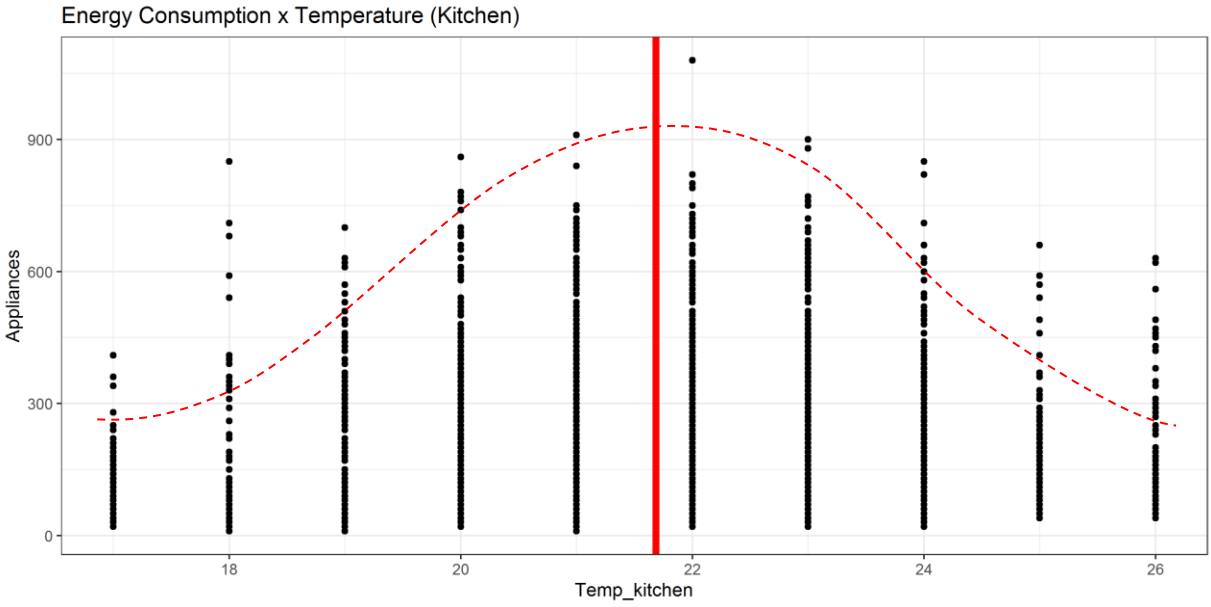


Figura 9 - Relação Consumo x Temperatura (Kitchen)

Assim como a análise da variação da temperatura na sala de estar, aqui na cozinha também não há correlação entre estas duas variáveis (nem positiva nem negativa), mas é possível observar que o aumento de temperatura indica o aumento do consumo de energia até próximo à temperatura média na cozinha, que é igual à aproximadamente 22ºC.

Por fim vamos analisar como a temperatura externa, ou seja, as alterações no clima afetam o consumo de energia geral.

6.5 Efeito da Alteração do Clima no Consumo de Energia

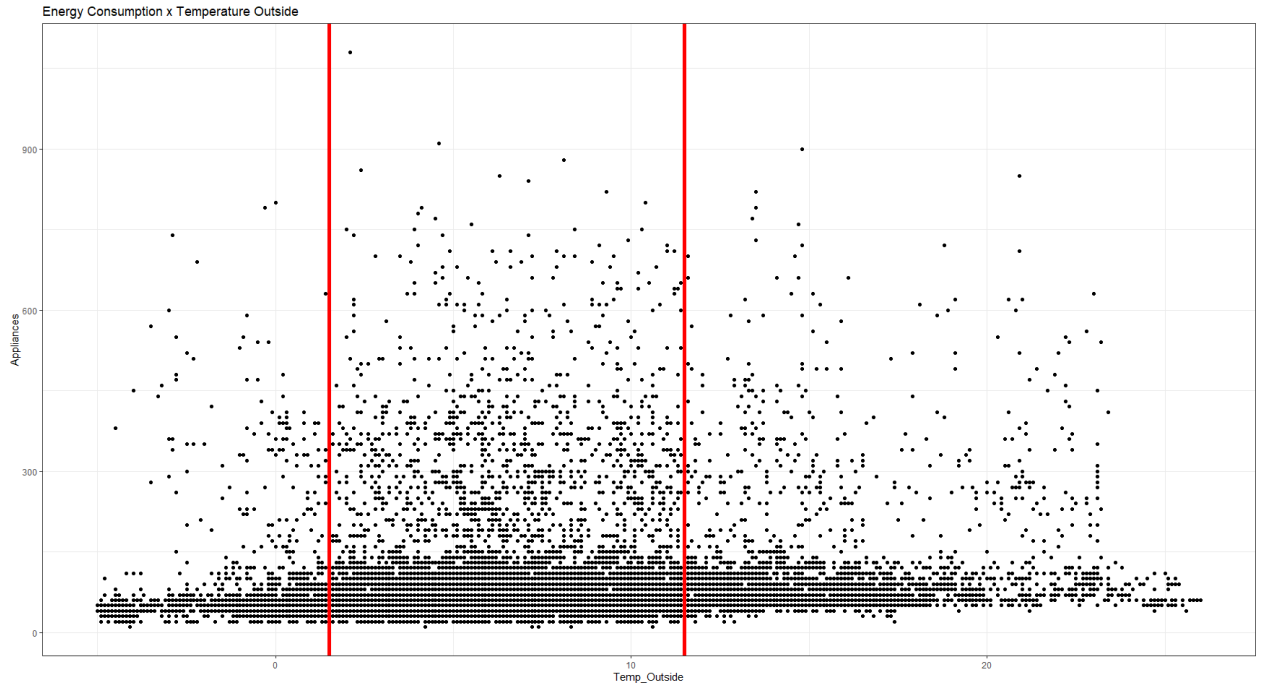


Figura 10 - Relação Consumo x Clima I

Observe que o maior consumo de energia está entre as faixas vermelhas na figura acima e na figura abaixo, ou seja, claramente quando o clima está mais frio (entre 1ºC e 11ºC), o consumo de energia geral da residência se torna maior.

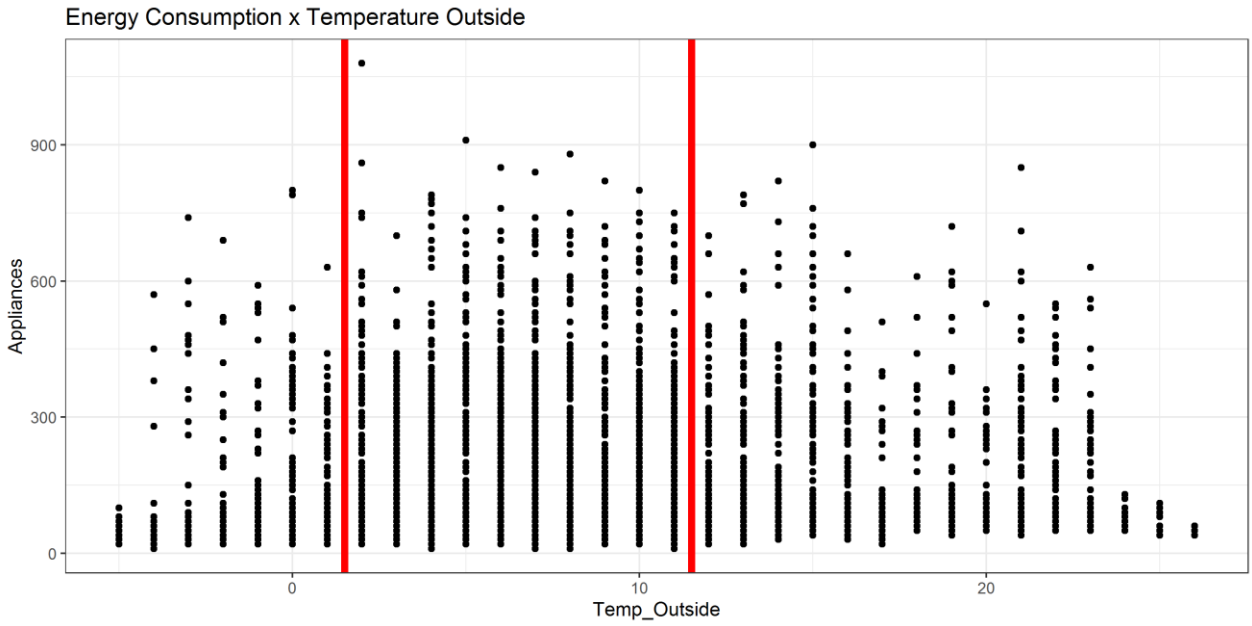


Figura 11 - Relação Consumo x Clima II

7 – Feature Engineering

Feature Engineering, ou Engenharia de Atributos é o processo de tratamento, adição e remoção de variáveis.

Esse processo consiste em descobrir quais colunas de dados criam os atributos mais úteis para melhorar a precisão do modelo de aprendizagem de máquina. Identificar atributos bons e ruins é parte importante dando reflexo no resultado final, outra possibilidade é adicionar variáveis relevantes com base nos dados fornecidos.

7.1 Distribuição das Variáveis

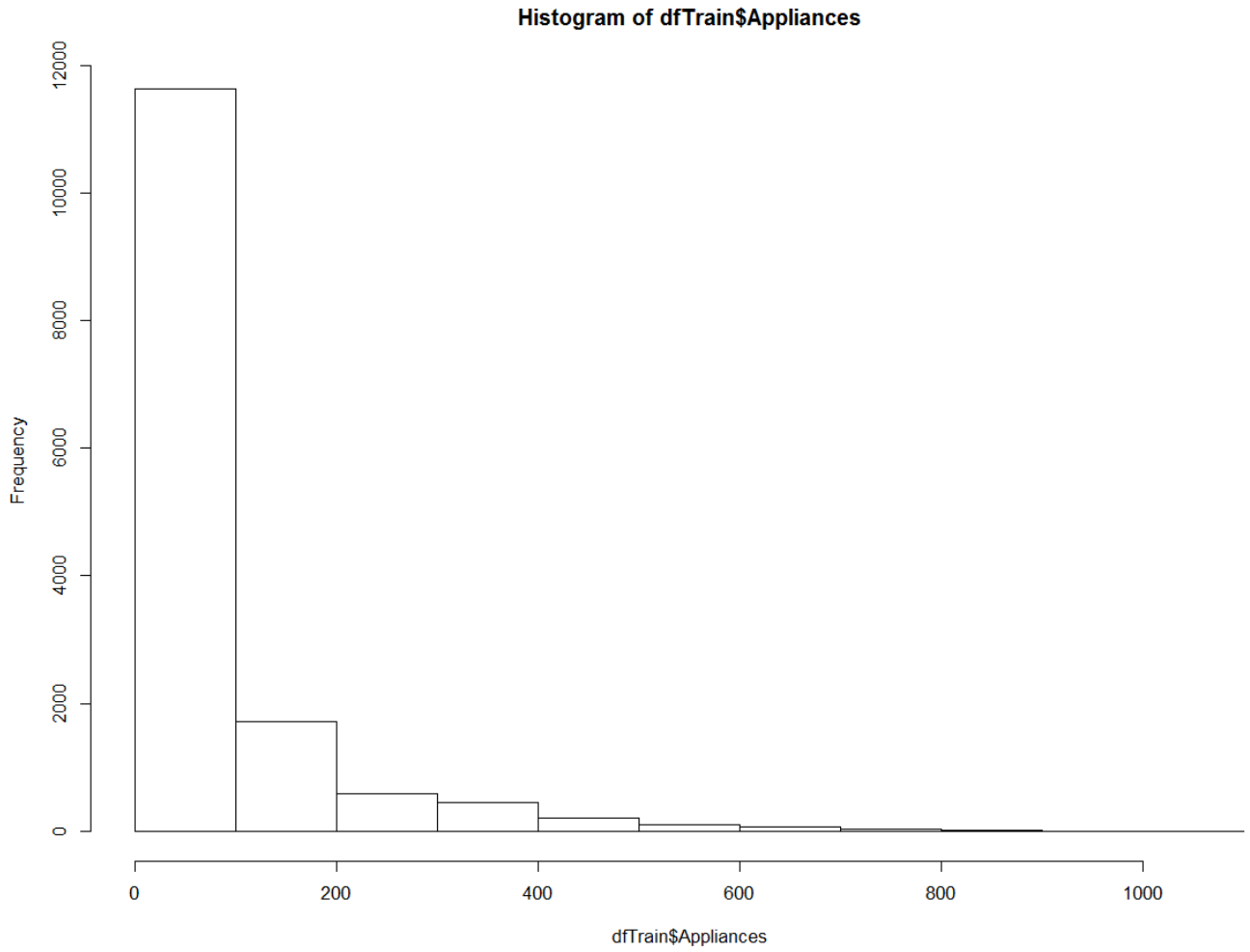


Figura 12 - Histograma Variável Target

Dentre os possíveis valores que a variável alvo de estudo para previsões (Appliances, Consumo de Energia) pode assumir, temos maior concentração entre 0 e 100 Wh.

7.2 Correlação das Variáveis

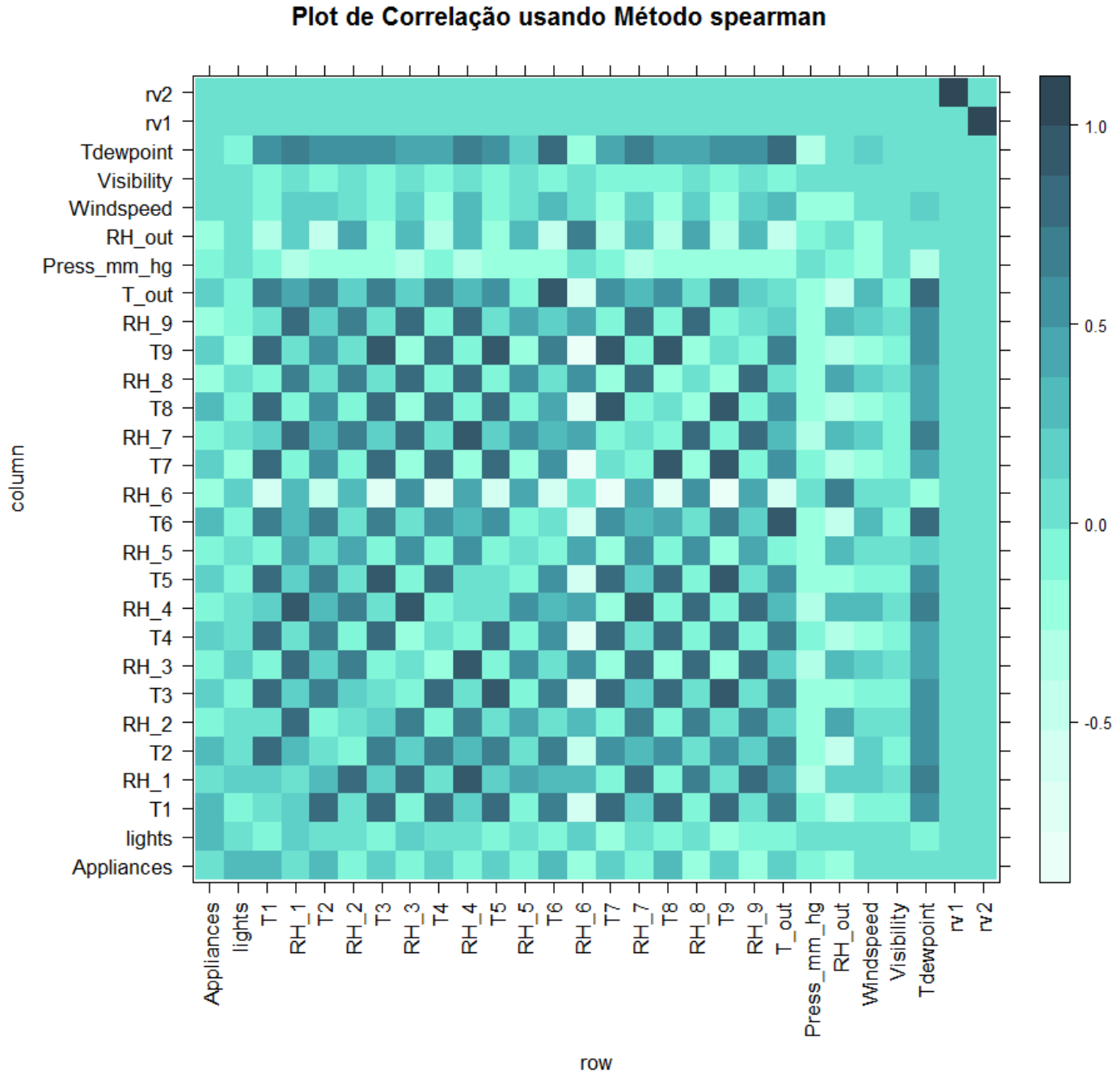


Figura 13 - Correlação entre Variáveis Preditoras

Observe que as variáveis rv1 e rv2 são altamente correlacionadas entre si e sem correlações explicativas com todas as outras, portanto decidi eliminar as mesmas para o estudo de Machine Learning. Outro processo executado foi o ajuste na escala dos dados e eliminação de valores nulos. Deste modo estamos prontos para o início dos processos de aprendizado de máquina.

8 – Machine Learning

Neste projeto foi explorado 3 modelos de Machine Learning sendo criados 5 tipos diferentes, a saber: lm, RandomForest, SVM (Support Vector Machine) e K-Means (apenas para apresentação dos resultados). Vejamos cada um em detalhe.

8.1 lm (Linear Model)

O primeiro modelo que iremos explorar será o lm (Linear Model) que carrega um série de Modelos Lineares.

Este é o modelo mais simples de aprendizado de máquina que podemos construir, onde através de uma fórmula com um número determinado de argumentos podemos treinar uma série de regressão e obter previsões em conjuntos de dados de teste.

Mas não se engane com simplicidade deste modelo pois o mesmo pode fornecer um série de informações estatísticas sobre o conjunto de dados treinado como valores residuais de erros, p-value, estimativa de melhores variáveis, dentre outras métricas.

Para nossos estudos, vou considerar a AUC como métrica de performance obtendo assim a acurácia de cada modelo a ser treinado.

8.1.1. AUC (Area Under the Curve)

Area Under the Curve é uma representação gráfica que ilustra o desempenho de um sistema classificador à medida que o seu limiar varia.

Para entender o conceito de AUC precisamos entender um conceito prévio, o ROC (Receiver Operaing Characteristic Curve). O ROC é exatamente a representação gráfica do AUC porém ele computa apenas os pontos da curva avaliando todos os limiares individualmente, o que pode ser ineficiente visto que queremos um resumo do desempenho do modelo. Portanto, o uso da Área Abaixo desta Curva ROC para medir exatamente a performance geral é mais indicado.

Benefícios da curva AUC:

- AUC é invariante de escala. Ele mede o quão bem as previsões são classificadas, em vez de seus valores absolutos;
- AUC é invariante do limiar de classificação. Ele mede a qualidade das previsões do modelo, independentemente do limite de classificação escolhido.

Desta maneira vamos utilizar esta métrica para análise da acurácia.

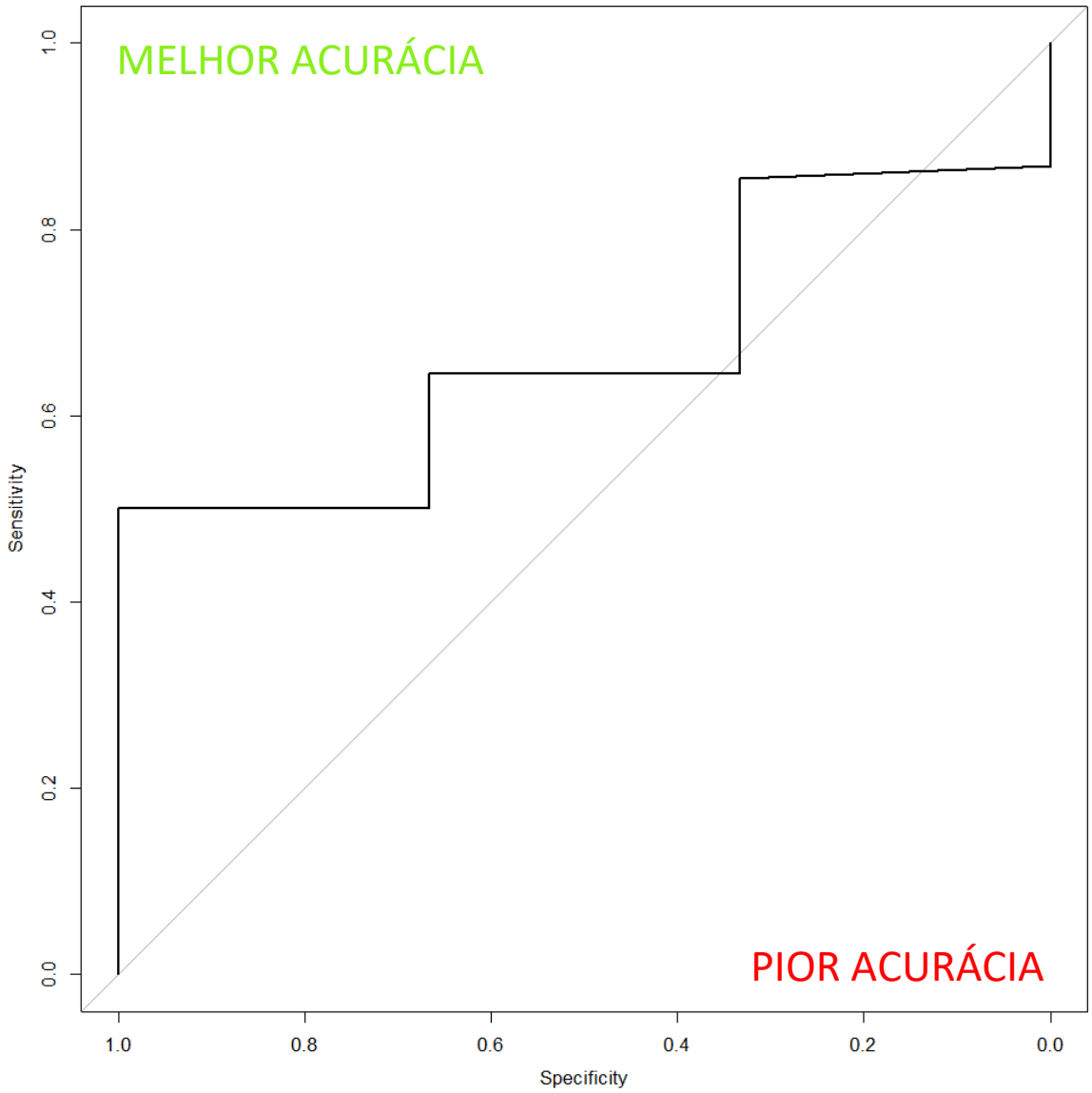


Figura 14 - AUC (lm)

Na figura acima, a linha na diagonal determina o limiar entre melhor e pior acurácia nos dados de teste após modelo ter sido treinado. Já a linha em formato de escada determina a performance do modelo, onde o ideal é que ela esteja o mais distante possível da linha diagonal e o mais perto possível de 1 na escala da Sensitividade, equivalente ao eixo y.

AUC -> 0.7125

Encontramos 71.25% de precisão neste modelo. Vamos agora estudar o Random Forest.

8.2 Random Forest

Com as variáveis selecionadas podemos treinar o modelo preditivo, porém um ponto fundamental é tentar identificar quando o modelo entra na zona de underfitting, quando encontra o menor erro (valor ideal) e quando chega na zona de overfitting. Um modelo Random Forest utiliza Árvores de Decisão para compor seu algoritmo e por ser um modelo poderoso é preciso ter cuidado para não cair em zonas onde o modelo aprendeu demais e oferece assim pior performance. Vejamos suas características.

Como o nome sugere, randomForest significa Floresta Aleatória, o que em Data Science podemos fazer analogia à 2 ou mais Árvores de Decisões onde cada árvore possui uma profundidade e decide entre suas 'folhas' qual o melhor caminho a ser percorrido.

No randomForest crescemos múltiplas árvores ao invés de uma única árvore. Mas como funciona o processo de classificação? Inicialmente para classificar um novo objeto baseado em atributos, uma árvore gera uma classificação para esse objeto (que é como se a árvore computasse votos para essa classe). Esse processo vai acontecendo para cada árvore presente na floresta e por fim, a floresta escolhe a classificação que tiver mais votos de todas as árvores da floresta. No caso de regressão, é considerado a média das saídas por árvores diferentes.

Para ilustrar o processo executado pelo randomForest, segue figura abaixo:

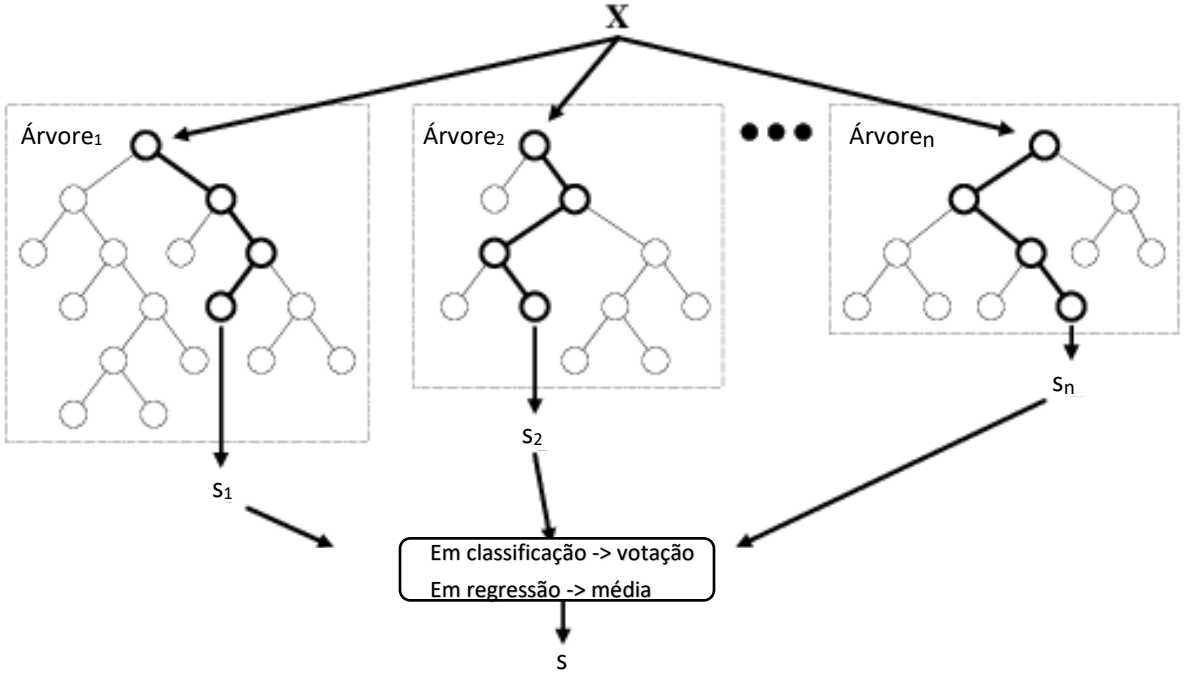


Figura 15 - Random Forest

Conforme figura acima, podemos ter n árvores e cada árvore pode ter quantas folhas desejar. É nesse momento que temos um problema, pois uma árvore rasa que foi treinada para

classificar um objeto pode não oferecer precisão pois aprendeu pouco, ou em outras palavras, teve um *underfitting*. No outro extremo temos o *overfitting* (sobreajuste, ou super treinamento), ou seja, se não for estabelecido um limite o modelo vai dar 100% de precisão no conjunto de treinamento pois ele acaba fazendo uma folha para cada observação.

Imagino que a pergunta agora seria: "Mas oferecer 100% de precisão não é bom?". A resposta é sim e não, pois é bom termos precisão, porém deste modo a precisão está apenas nos dados de treino e o meu objetivo é gerar um modelo de aprendizado de máquina imparcial onde qualquer dado possa ser previsto. No caso do *overfitting* quando eu apresentar novos dados (que são os dados de teste) o modelo vai falhar e retornar precisão insatisfatória.

Segue imagem ilustrando a problemática:

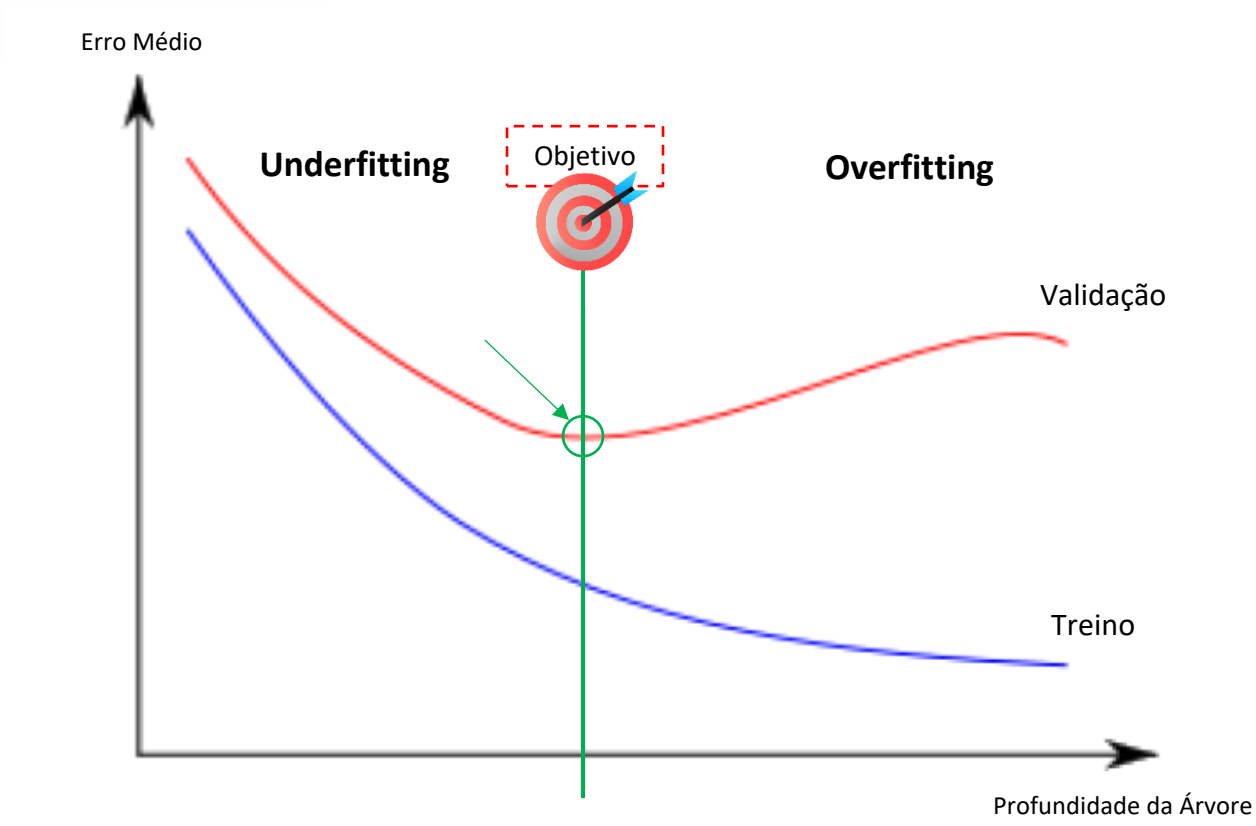


Figura 16 - Underfitting x Overfitting

Na figura acima temos 3 linhas com cores diferentes e cada uma representa uma informação importante:

- Linha Azul: representa o erro médio que os dados de treino fornecem de acordo com a profundidade da árvore. Quanto mais profundo, menor o erro visto que os dados de treino foram aprendidos quase que na totalidade.

- Linha Vermelha: representa o erro nos dados de teste (validação). Observe que o erro começa alto, diminui e em seguida aumenta novamente. Esse é um dos principais desafios enfrentados ao modelar árvores de decisão, encontrar o ponto ideal em que o erro seja o menor possível.
- Linha Verde: conforme é possível observar, a linha verde cruza exatamente no ponto ideal, onde o erro nos dados de teste (validação) seja o mínimo possível oferecendo assim melhor precisão ao modelo.

Consolidando na figura a seguir o objetivo em realizar modelagem preditiva controlando underfitting e overfitting:

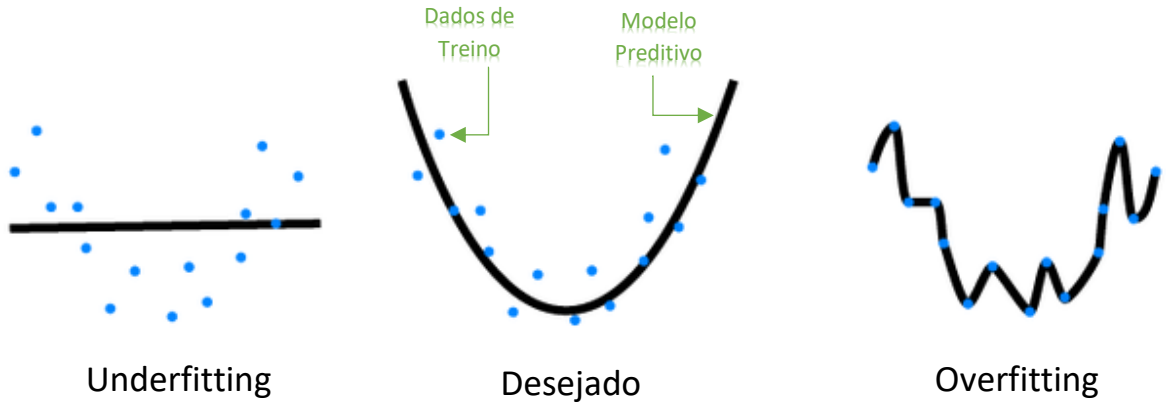


Figura 17 - Underfitting x Ideal x Overfitting

Conforme figura acima, o que desejamos é que o modelo tenha uma curva ideal evitando pouca precisão, mas que também não memorize 100% dos dados de treino falhando em novos e desconhecidos dados. Vejamos como o nosso modelo se comportou com os dados apresentados.

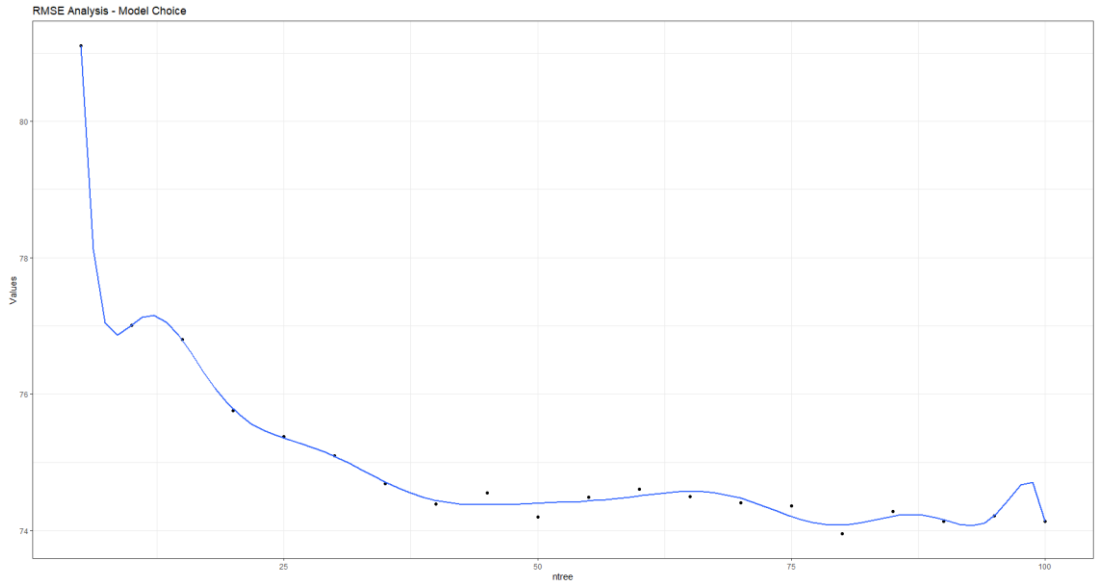


Figura 18 - Curva de Acurácia do modelo RandomForest

A figura 18 apresenta a variação na acurácia do modelo de modo que podemos observar estados de underfitting antes de 20 Árvores de Decisão com baixa precisão e estados de overfitting após 90 Árvores de Decisão também atuando precariamente na acurácia do modelo. Vamos utilizar o valor de 80 Árvores de Decisão como o ideal para a nossa base de conhecimento.

8.2.1. AUC (Area Under the Curve)

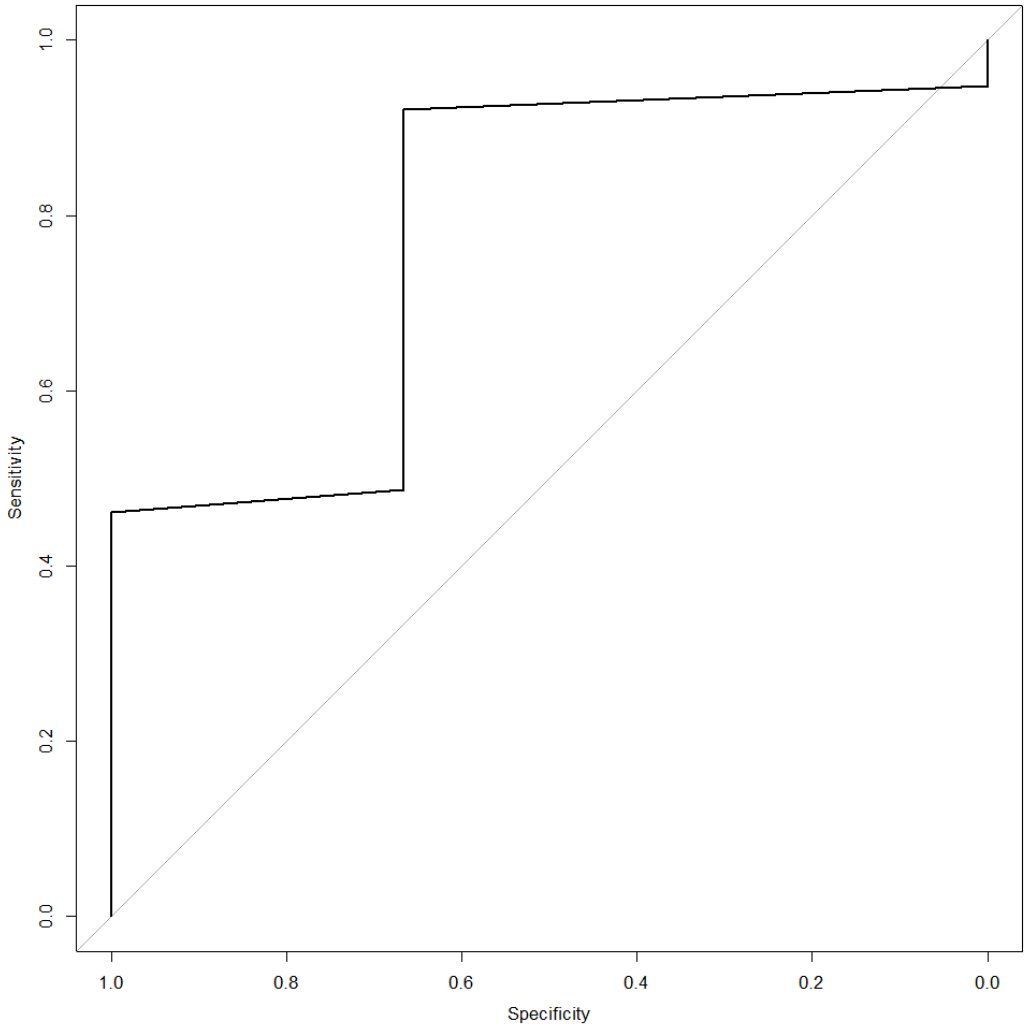


Figura 19 - AUC (RandomForest)

De acordo com a figura 19 é possível observar que a curva ROC se afastou da diagonal que divide os quadrantes se aproximando mais da Sensitivity 1.0, ou seja, este modelo apresentou melhor performance quando comparado ao anterior (lm).

AUC -> 0.7788

8.3 SVM (Support Vector Machine)

SVM, ou Support Vector Machine, é um conceito na ciência da computação para um conjunto de métodos de aprendizado supervisionado que analisam os dados e reconhecem padrões, usado para classificação e análise de regressão.

Utilizando uma função Kernel (função matemática), o SVM realiza a classificação de um determinado conjunto de dados mapeados em um espaço multidimensional, ou seja, o SVM utiliza hiperplanos como limites de decisão para classificação do ponto de dados.

Em problemas de classificação onde os dados não são linearmente separáveis o SVM executa um excelente trabalho de separação e classificação levando os dados para outros planos, separando os mesmos e classificando novos dados apresentados.

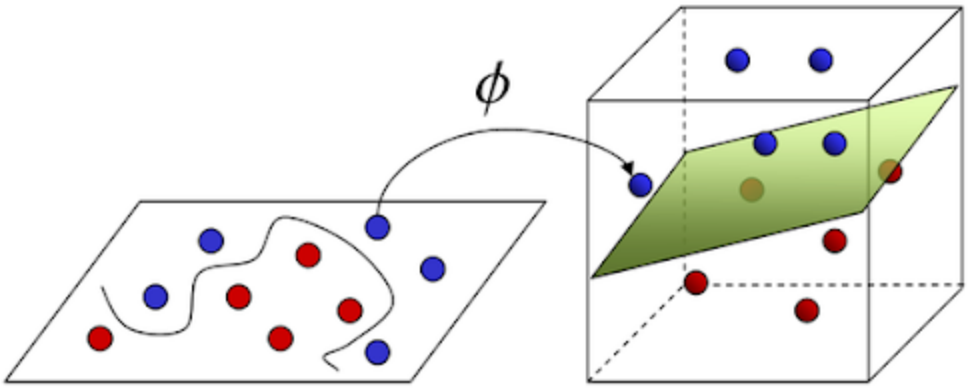


Figura 20 - Hiperplano SVM

Conforme figura acima, de maneira fantástica e através de funções matemáticas o SVM cria o hiperplano separador (em verde) possibilitando aplicações avançadas de aprendizado nos dados.

Benefícios do SVM:

- Funciona muito bem em domínios complicados, em que existe uma clara margem de separação;
- Eficaz em espaços dimensionais elevados;
- Eficaz nos casos em que o número de dimensões é maior do que o número de amostras;
- Em caso de outlier a SVM busca a melhor forma possível de classificação e, se necessário, desconsidera o outlier;

Desvantagens do SVM:

- Não funciona bem em conjuntos de dados muito grandes e nem com grandes quantidades de ruído, pois exige inversão de matriz - aumentando a complexidade computacional com até o cubo do volume de dados;
- Se as classes estiverem muito sobrepostas deve-se utilizar apenas evidências independentes (devido ao fato de não ser muito bom com dados com muitos ruídos);

Como é um modelo que exige mais configurações dentro dos hiperparâmetros de construção, criei 3 modelos SVM com diferentes configurações, porém apenas um deles ofereceu melhor performance. Inclusive, os tempos para executar cada SVM são absurdamente grandes e mostrou ser improdutivo no final. Vou apresentar as performances no capítulo de conclusões.

8.3.1. AUC (Area Under the Curve)

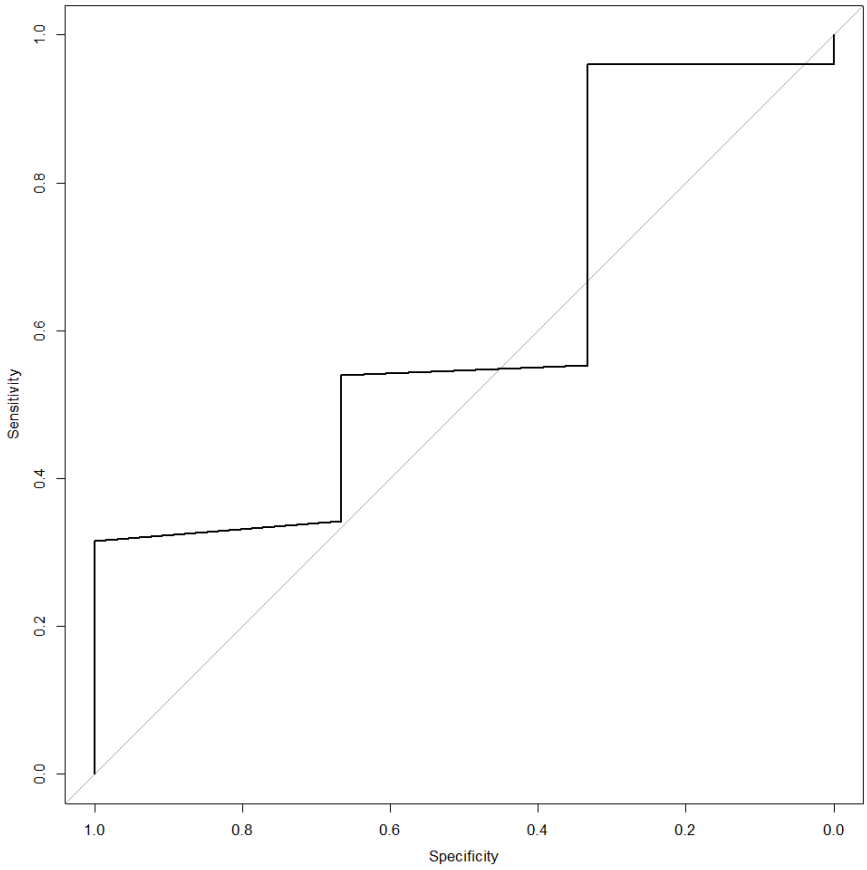


Figura 21 - AUC (SVM I)

AUC -> 0.7317

9 – Conclusões

Todo o projeto demorou em média 1 Hora e 20 Minutos para ser executado (considerando todos os modelos preditivos treinados e realizando previsões). O maior gargalo deste projeto está certamente nos modelos SVM, observe no gráfico a seguir o tempo gasto por modelo para ser executado.

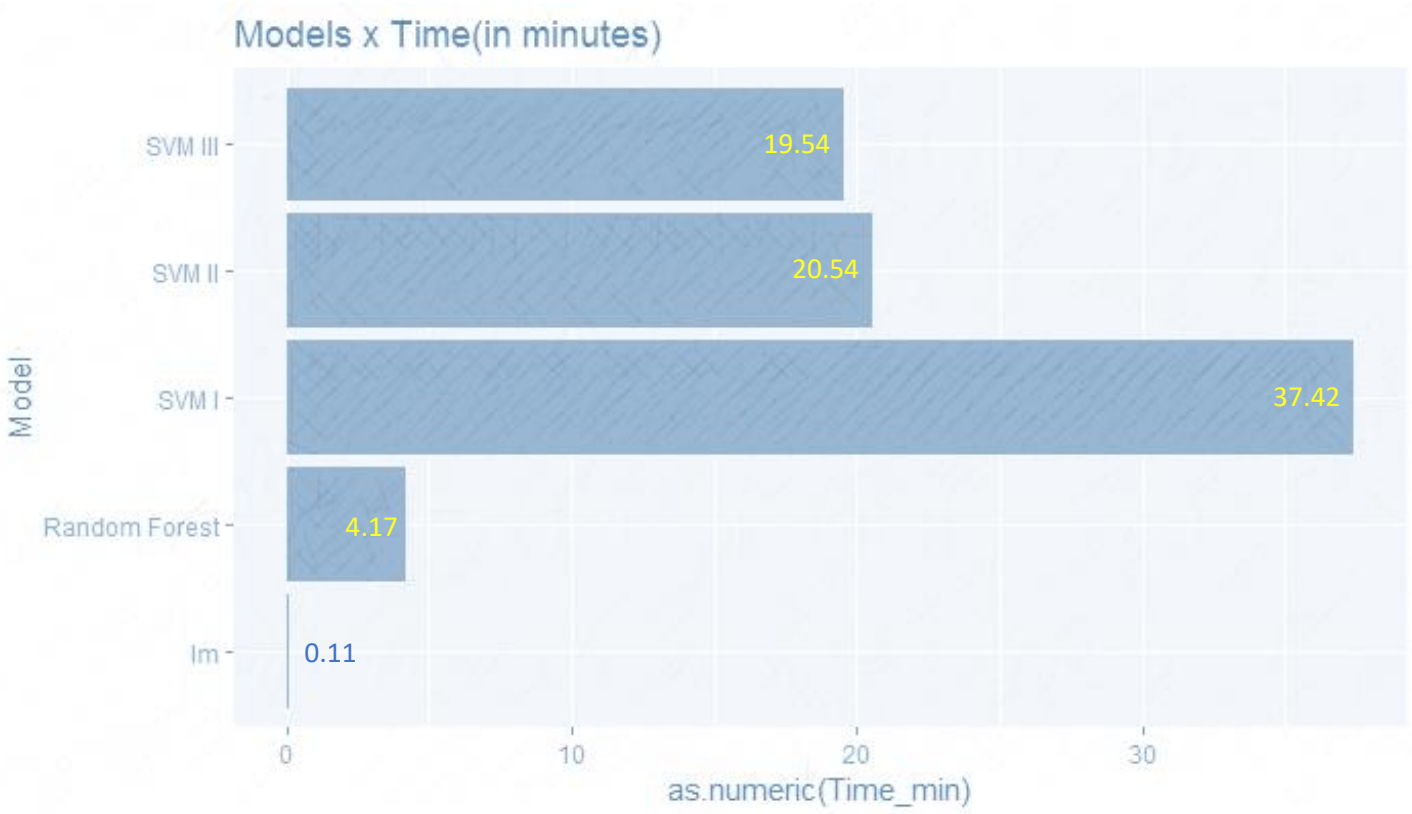


Figura 22 - Tempo de Treinamento de cada Modelo Preditivo

Os tempos de treinamento dos modelos SVM são altos devido ao alto nível de cálculos matemáticos realizados pelo modelo para ajustar os dados à diferentes hiperplanos, porém será que o alto custo de tempo comparado ao Random Forest, por exemplo, justifica melhor performance? Vejamos na apuração a seguir.

Para concluir a apresentação deste projeto vamos consolidar qual foi o modelo com melhor performance e como apresentar os resultados utilizando este modelo.

9.1 Apuração Modelos Preditivos

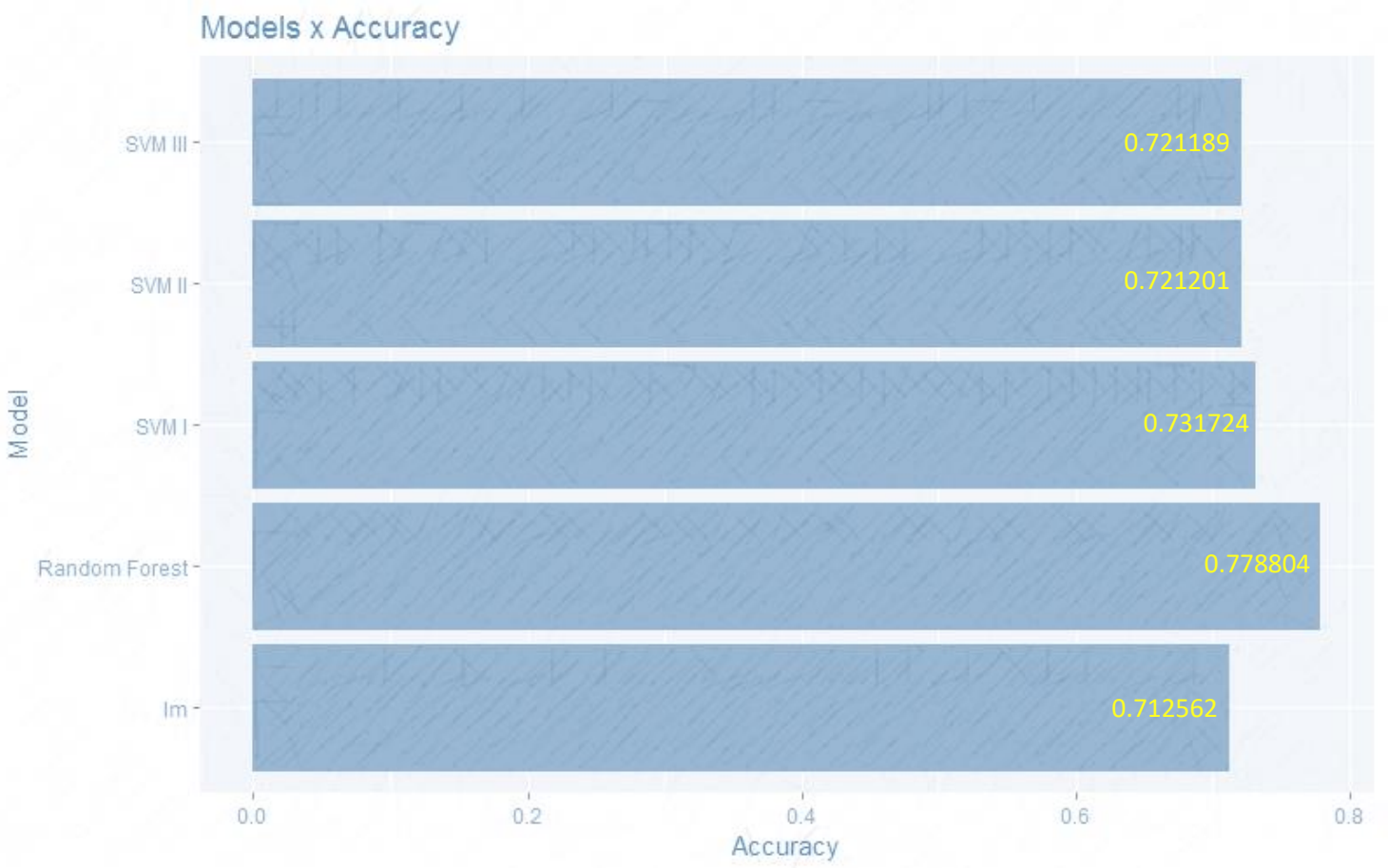


Figura 23 - Modelos x Acurácia

Conforme exposto nas análises individuais de cada modelo e confirmando acima, o RandomForest foi o campeão no quesito precisão em dados desconhecidos. Apesar de os modelos SVM serem modelos mais difíceis de serem ajustados devido à alta quantidade de parâmetros para serem configurados, ele não justifica o "tempo de treinamento versus custo de processamento versus acurácia". Certamente o SVM apresenta excelente performance, porém não o suficiente para superar o RandomForest para o conjunto de dados de nosso problema de negócio.

Uma observação importante pode ser feita quanto ao modelo "lm" que na sua "simplicidade" ofereceu quase o mesmo ganho de qualidade em previsões. Um algoritmo que pode oferecer também bons resultados mesmo sem grandes ajustes de hiperparâmetros, apenas sabendo analisar multicolinearidade e eliminando variáveis altamente correlacionadas.

9.2 Apuração AUC Modelos Preditivos

Consolidando o exposto no item 9.1 acima, vejamos como fica a AUC para todos os modelos.

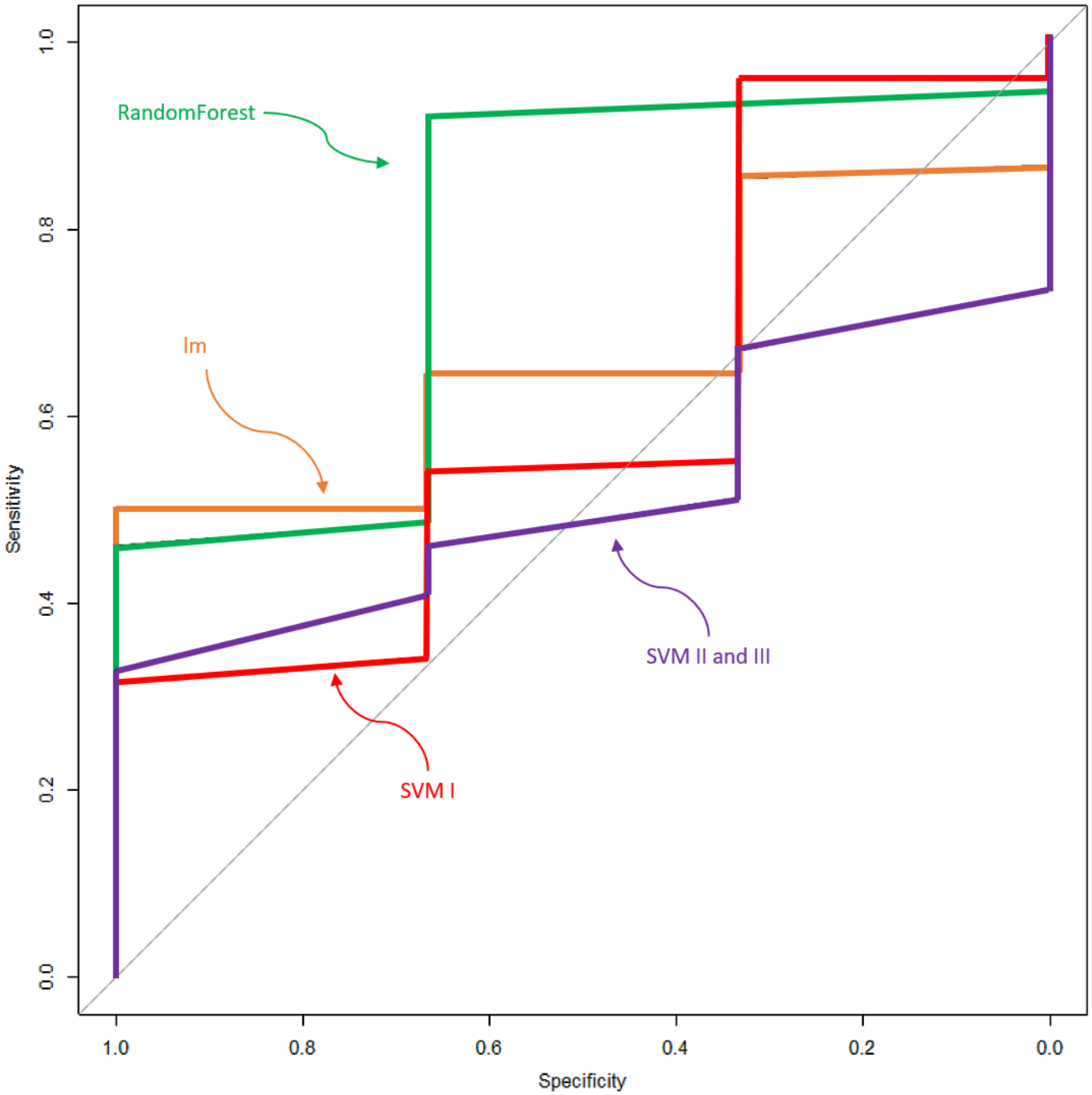


Figura 24 - AUC (Todos os Modelos)

Como dito anteriormente, o modelo RandomForest oferece melhor performance geral, vamos utilizar ele para apresentação de novos dados e análise de níveis de consumo.

9.3 Previsões em Novos Dados

Ao apresentar novos e desconhecidos dados, obtemos a seguinte previsão de consumo de energia:

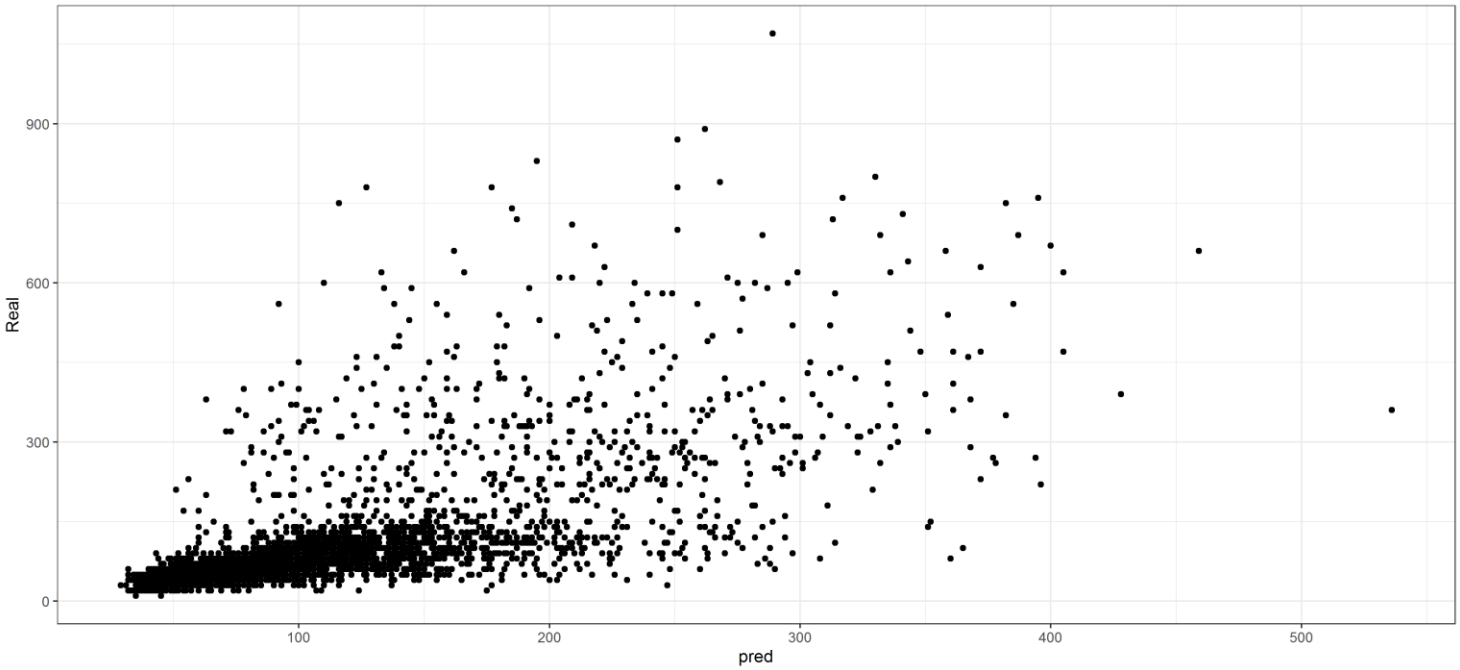


Figura 25 - Previsto x Real

Na figura acima é possível constatar que a maior parte das previsões se concentram em valores entre 0 e 150 Wh, um bom indicativo de que nosso modelo está ajustado aos dados treinados pois conforme vimos na análise exploratória esta é a média de consumo de energia real nesta residência, ou seja, os valores previstos seguem um padrão próximo aos valores reais.

Para gerar mais valor à análise de consumo, vou utilizar o algoritmo de Machine Learning K-Means que vai auxiliar na separação automática dos dados em 9 diferentes clusters.

9.4 K-Means nos Valores Previstos

K – Means é um algoritmo de aprendizagem não – supervisionada que realiza clustering nos dados particionando cada observação entre os K grupos de acordo com o cálculo da média de distância entre centros.

Baseado no diagrama de Voronoi[23], o K – Means encontra similaridades entre o dados realizando os seguintes passos:

[23] https://en.wikipedia.org/wiki/Voronoi_diagram

- Atribuição de Valores para Centróides: aleatoriamente, baseado no conjunto de dados, o algoritmo determina valores iniciais para os k centróides. Em nosso caso será k = 9;
- Cálculo das distâncias dos dados: com os k centróides, o algoritmo calcula a distância entre cada ponto de dado fornecido e os k centróides determinados anteriormente e utiliza estes valores para a classificação no próximo passo;
- Atribuição de Classes aos Dados: de acordo com o valor determinado inicialmente dos centróides o algoritmo vai classificar cada ponto de dado de acordo com a menor distância encontrada para um dos k centróides, assim teremos todos os pontos de dados incorporados à algum dos k clusters. Porém, para melhoria e otimização das classificações, o K – Means continua se ajustando, ou seja, novos cálculos são realizados conforme próximo passo;
- Cálculo de Novos Centróides: a fim de melhorar a classificação, o K – Means realiza novos cálculos de distância porém agora entre os pontos pertencentes à cada classe K, ajustando o valor dos centróides e possivelmente reclassificando novos pontos de acordo com o processo acima;
- Recálculos e Ajustes: com a informação anterior o algoritmo volta ao segundo passo realizando novos cálculos de valores dos centróides e ajustando os pontos de dados repetindo várias vezes este processo até que cada ponto de dado pertença a apenas uma das K classes determinadas.

Vejamos como o algoritmo determinou os centróides automaticamente:

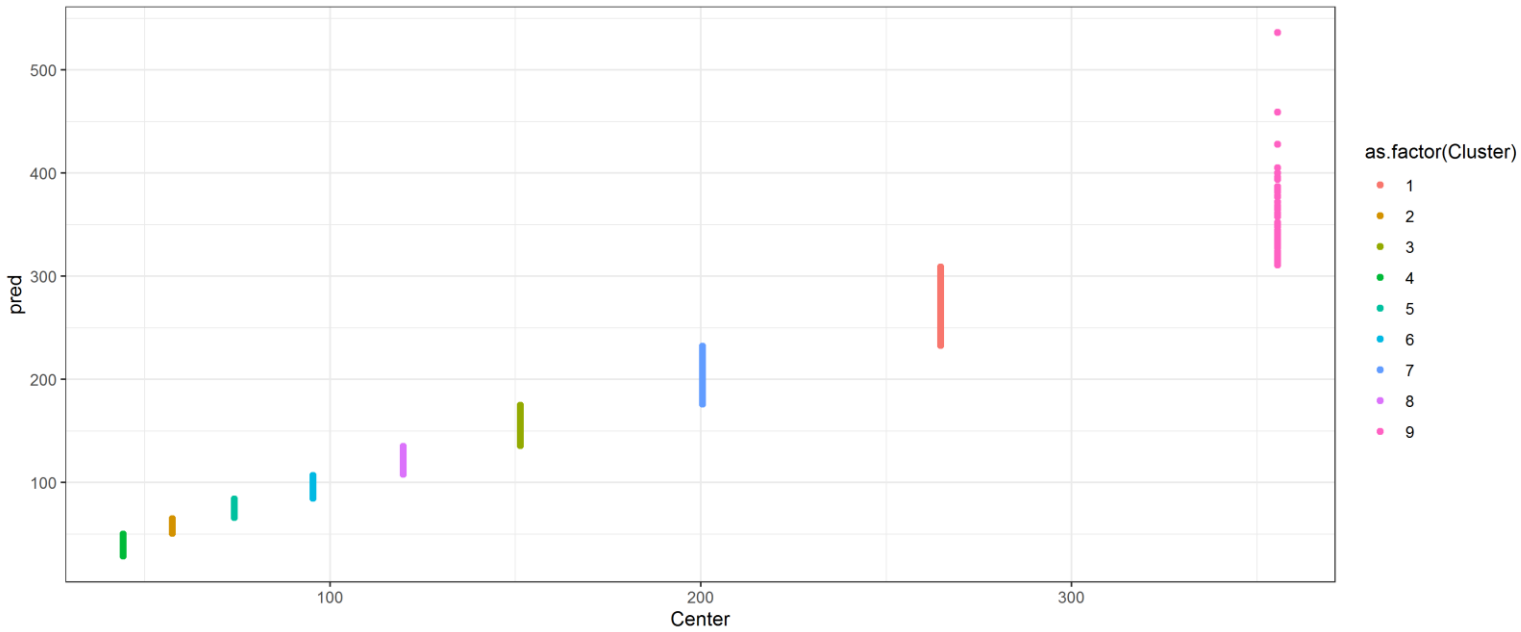


Figura 26 – Range do Cluster com K - Means

Na figura 26 temos os k = 9 clusters classificados automaticamente pelo algoritmo e os ranges de consumo determinado por ele. Pegue por exemplo o cluster na cor violeta e observe que ele classificou neste cluster o consumo previsto entre aproximadamente 100 e 150 Wh. Vamos inserir os valores dos centróides ao conjunto de dados previstos:

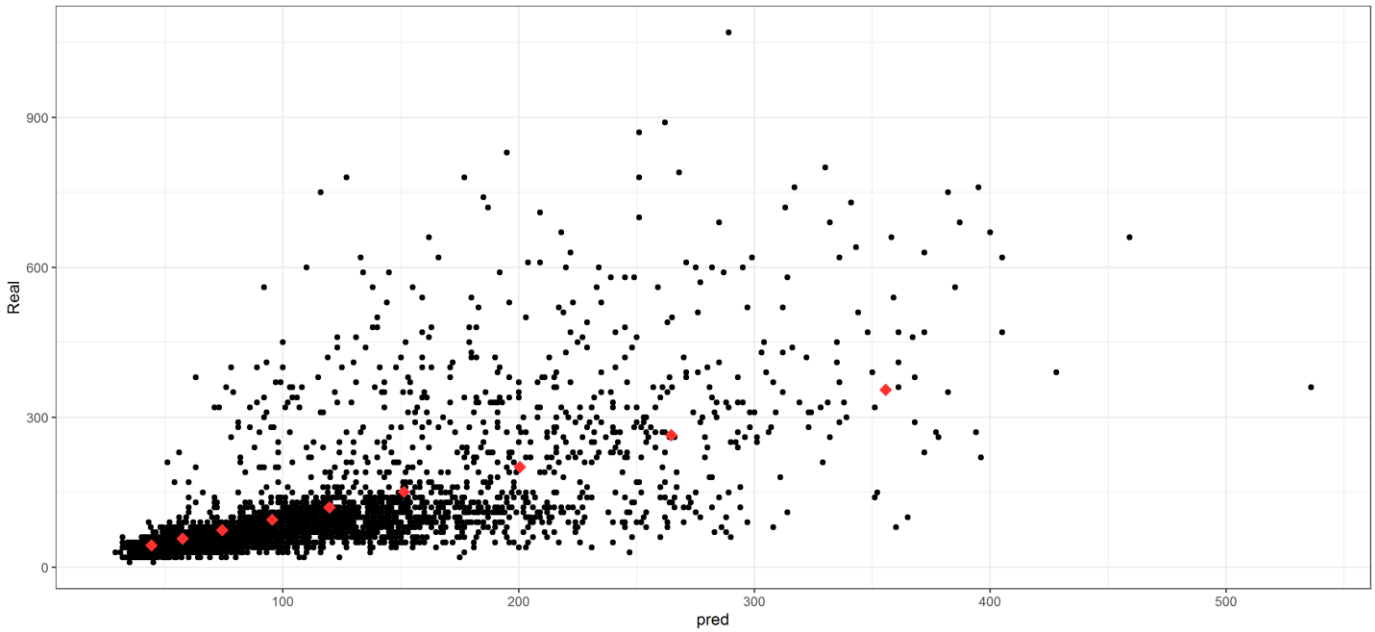


Figura 27 - K centróides nos dados previstos

Na figura 27 os centróides calculados pelo algoritmo foram inseridos aos valores previstos, mas ainda assim a visualização pode melhorar, com a inserção dos clusters à imagem podemos realizar a segmentação de consumo:

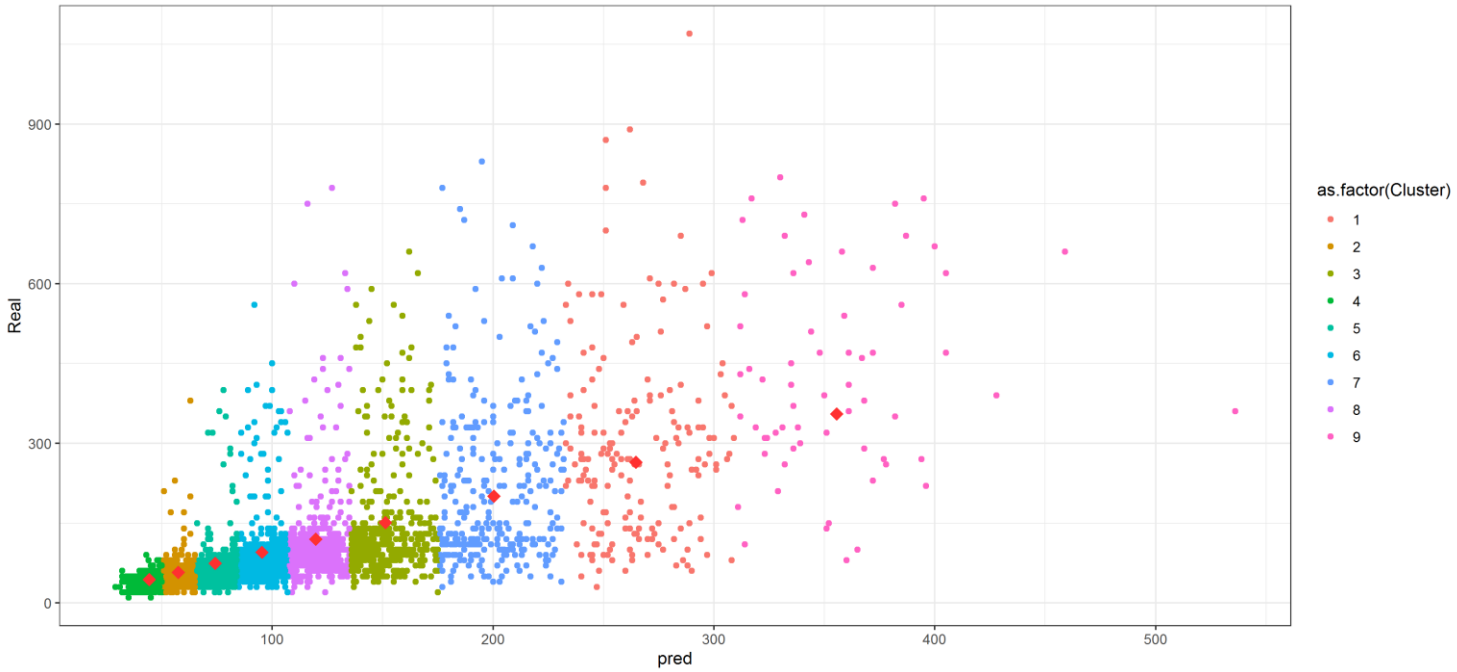


Figura 28 - Clusters, Centróides e Dados Previstos

9.5 Considerações Finais

Realizar a clusterização nos ajuda a entender períodos de maior consumo e consequentemente oferecer melhores insights ao usuário. Na clusterização da figura 28 é possível notar que os 6 primeiros clusters são bem definidos enquanto os 3 restantes possuem bastante valores dispersos.

Como sabemos, durante a análise exploratória identificamos o range de valores de maior consumo como sendo entre 0 e 150Wh, demonstrado com sucesso na clusterização dos valores previstos onde a maior concentração de valores está ocupando o mesmo range.

A análise exploratória mostrou também quão importante é o estudo dos dados pois foi nela que observamos correlações e variáveis que forneciam melhores oportunidades para modelos preditivos robustos.

Com as melhores variáveis escolhidas e com a eliminação de variáveis altamente correlacionadas, construímos 3 tipos de modelos preditivos: lm, RandomForest e SVM. Cada modelo teve a sua performance estudada através da métrica AUC e do tempo de execução sendo o RandomForest o modelo campeão pois forneceu não apenas melhor acurácia mas também um tempo de treinamento aceitável. Fato é que os modelos SVM são poderosíssimos, porém levam muito tempo para se ajustarem (em média 30 minutos neste projeto), enquanto que modelos lm são simples e rápidos (em torno de 0.11 segundos para treinar neste projeto) e ofereceu performance muito próxima ao SVM.

Com IoT as possibilidades são muitas, mas tudo depende da correta estruturação do problema. Para nosso problema de negócio deste projeto obtivemos acurácia máxima de 77.88% entregando ao final clusters de análise de consumo em Wh segmentando o perfil dos residentes desta moradia oferecendo assim níveis de entendimento para cada ambiente estudado como cozinha, sala de estar, quartos, banheiro, área externa e inclusive como o clima afeta o consumo geral de energia.

Apesar de possuirmos boa quantidade de dados, a maior parte deles foi estimada através de cálculos matemáticos durante a coleta, portanto, como melhoria poderíamos realizar muito mais coletas de dados dos sensores em tempo real (sem estimativas) e com mais frequência armazenando em um datalake passando a tratar em um cluster hadoop a grande quantidade de dados ingeridas, assim o problema passaria para uma análise de big data fornecendo em tempo real previsões ao usuário e companhias de energia.

Código Fonte

```
# IoT Forecasting - Energy Use Forecasting ----

#### Defined the Business Problem #### ----------------------------------------------------------------

# This IoT project aims to create predictive models for forecasting energy consumption of

# home appliances. The data used includes measurements of temperature and humidity sensors in a wireless network,

# weather forecast for an airport station and energy used by luminaires.

# In this machine learning project you must perform data filtering to remove parameters

# non-predictive and select the best features (best features) for forecasting. The data set was

# collected over a period of 10 minutes for about 5 months. The temperature and humidity conditions of the house were

# monitored with a wireless ZigBee sensor network.

# Each wireless node transmitted temperature and humidity conditions for about 3 min. Then, the average of

# data was calculated for 10-minute periods. Energy data was recorded every 10 minutes with meters

# of bus power m. Weather from the nearest weather station to the airport (Chievres Airport, Belgium)

# was downloaded from a public Reliable Prognosis dataset (rp5.ru) and merged with the experimental

# datasets using the date and time column. Two random variables were included in the data set to test

# regression models and filter out non-predictive attributes (parameters).

# Your job now is to build a predictive model that can predict energy consumption based on data

# of IoT sensors collected. We recommend using RandomForest for the selection of attributes and SVM, Multilinear Logistic

# Regression or Gradient Boosting for the predictive model
```

```
# Obs: If you have problems with accenting, see this link.:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

## Data Dictionary ----------------------------------------------------------------------------
# date -> time year-month-day hour:minute:second
# Appliances(TARGET) -> energy use (in Wh)
# lights -> energy use of light fixtures in the house (in Wh)
# T1 -> Temperature in kitchen area (in Celsius)
# RH1 -> Humidity in kitchen area (in %)
# T2 -> Temperature in living room area (in Celsius)
# RH2 -> Humidity in living room area (in %)
# T3 -> Temperature in laundry room area (in Celsius)
# RH3 -> Humidity in laundry room area (in %)
# T4 -> Temperature in office room (in Celsius)
# RH4 -> Humidity in office room (in %)
# T5 -> Temperature in bathroom (in Celsius)
# RH5 -> Humidity in bathroom (in %)
# T6 -> Temperature outside the building (north side) (in Celsius)
# RH6 -> Humidity outside the building (north side) (in %)
# T7 -> Temperature in ironing room (in Celsius)
# RH7 -> Humidity in ironing room (in %)
# T8 -> Temperature in teenager room 2 (in Celsius)
# RH8 -> Humidity in teenager room 2 (in %)
# T9 -> Temperature in parents room (in Celsius)
# RH9 -> Humidity in parents room (in %)
# T_out -> Temperature outside (from Chievres weather station) (in Celsius)
# Press_mm_hg -> Pressure (from Chievres weather station) (in mm Hg)
# RH_out -> Humidity outside (from Chievres weather station) (in %)
# Windspeed -> Wind speed (from Chievres weather station) (in m/s)
```

```
# Visibility -> Visibility (from Chievres weather station) (in km)

# Tdewpoint -> Dew Point Temperature (from Chievres weather station) (in °C)

# rv1 -> Random variable 1, nondimensional

# rv2 -> Random variable 2, nondimensional

# NSM -> ? (Appears to be a sequencial number of sensor measure)

# WeekStatus -> Weekday or Weekend

# Day_of_week -> Monday to Sunday

# *** RH = Relative Humidity


# Highlights:

# The Appliances energy consumption prediction in a low energy house is the dataset content

# "Weather Data" from a nearby station was found to improve the prediction!

# "Pressure", "Air Temperature" and "Wind Speed" are important parameters in the prediction!

# Data from a WSN (Wireless Sensor Network) that measures "Temperature" and "Humidity" increase
the prediction accuracy!

# From the WSN (Wireless Sensor Network), the kitchen, laundry and living room data ranked high in
importance!


## Directory ----------------------------------------------------------------------------------

# Configuring the working directory

# Quote the working directory you are using on your computer.


# SET WORKING DIRECTORY

working_directory <- "C:/FCD/4MachineLearning/Cap22/R2"

results_directory <- "C:/FCD/4MachineLearning/Cap22/R2/Results1"

setwd(working_directory)

# GETTING CURRENT DIRECTORY

getwd()
```

```
# Dataframe to store results
dfResults <- data.frame(Model = character(),
                        RMSE = numeric(),
                        Accuracy = numeric(),
                        Time_min = numeric())


## Library ----------------------------------------------------------------------------------

# IMPORTING NECESSARY LIBRARIES - ##ATTENTION## MANDATELY CANNOT CONTAIN SPACE IN THE USER'S NAME TO BE ABLE TO INSTALL SOME PACKAGES, AS IS THE CASE OF caret
# THEREFORE I HAD TO TERMINATE RStudio, OPEN THE USER fellipe.silva AND INSTALL THERE, THEN RETURN HERE AND USE NORMALLY. THERE WILL NOT GET AN ERROR !!!!!!!!!


#install.packages("gains")
#install.packages("pROC")
#install.packages("ROSE")
#install.packages("mice")
#install.packages("knitr")
#install.packages("car")
#install.packages("gains")
#install.packages("pROC")
#install.packages("multiROC")
#install.packages("clustertend")
#install.packages("NbClust")
#install.packages("ggfortify")
# Another method for reading the variables
library(readr)
# Package to manipulate data
library(dplyr)
# Graphical analysis package
```

```
library(ggplot2)

# Package to plot correlation matrix

library(lattice)

# Package that has the kable function

library(knitr)

# Package that has the kable function

library(car)

# Package to help in the analysis of Multicollinearity with stepAIC

library(MASS)

# Package for working with machine learning (model building; optimization of hoper parameters with GridSearchCV; train_test_split)

library(caret)

# randomForest to extract the most relevant variables

library(randomForest)

# It also offers the SVM algorithm

library(e1071)

# Helps to interpret SVM models

library(gains)

# To build the ROC curve

library(pROC)

# Complement to the previous one to assist in the construction of evaluation metrics, such as confusion matrix for example.

library(ROCR)

# Metrics with ROC curve

library(multiROC)

# Package to analyze if data is clusterable with "hopkins"

library(clustertend)

# Package to determine the number of clusters in a data set

library(NbClust)
```

```
# Cluster plot package
library(ggfortify)
# Cluster plot package
library(cluster)
# Graphic package
library(RColorBrewer)

## Datasets --------------------------------------------------------------------------------
# dfTrain <- read_csv("projeto8-training.csv") # Load date in POSIXct format, but I won't use date here
dfTrain <- read.csv("projeto8-training.csv")
dfTest <- read.csv("projeto8-testing.csv")
str(dfTrain)
str(dfTest)
#View(dfTrain)
# 14.803 rows, 32 columns

## Exploratory Analysis -------------------------------------------------------------------
summary(dfTrain$Appliances)

# ATENÇÃO:
# The error below happens in the plots of the ggplot type geom_bar (), following explanation:
#Error: stat_count() must not be used with a y aesthetic.
#This error comes due to the mapping of elements in aes() while plotting barplot.
# geom_bar by default takes a count of values as y in a bar plot. So use stat="identity" within barplot()
to avoid the default and use fields as used in mapping function.
# ggplot(data,aes(field1,field2)) +geom_plot(stat="identity")

df <- dfTrain
df$date <- as.Date(dfTrain$date)
```

```
str(df)

# 01-Higher consumption happens on weekdays or weekends?
analysis <- df %>%
  count(WeekStatus) %>%
  as.data.frame()
#View(analysis)
# Pie Chart with ggplot()
ggplot() +
  geom_bar(data = analysis,
           aes(x = "", y = n, fill = WeekStatus),
           stat = "identity") +
  coord_polar("y",
              start = 0) +
  scale_fill_brewer(palette = "Paired") +
  theme_void() +
  theme(axis.text.x = element_blank()) +
  ggtitle("Consumption on Weekdays and Weekends")
# Weekday has higher consumption
setwd(results_directory)
ggsave("1-Consumption on Weekdays and Weekends.png", width = 10, height = 5)
setwd(working_directory)

# 02-How is the consumption by Wh and WeekStatus (Weekday or Weekend)
analysis <- df %>%
  group_by(Appliances, WeekStatus) %>%
  count(Appliances) %>%
  filter(Appliances <= 150) %>%
  as.data.frame()
```

```
#View(analysis)
# Bar Plot with ggplot()
ggplot() +
  geom_bar(data = analysis,
           aes(x=Appliances, y=n, fill = WeekStatus),
           stat = "identity",
           position = "dodge") +
  theme_bw() +
  scale_fill_brewer(palette = "Paired") +
  ggtitle("Consumption in Wh")
# It is possible to observe that in addition to the greatest consumption being on weekdays,
# the largest Wh is 50 for Weekday and Weekend. So we have an interesting pattern
# where you can check which appliances have the highest consumption.
setwd(results_directory)
ggsave("2-Consumption in Wh.png", width = 10, height = 5)
setwd(working_directory)


# 03-Is there a relationship between energy consumption and the average temperature rise in the
kitchen?
analysis <- df %>%
  count(Temp_kitchen = round(T1), Appliances) %>%
  as.data.frame()
#View(analysis)
ggplot() +
  geom_point(data = analysis, aes(x = Temp_kitchen, y = Appliances ), stat = "identity") +
  geom_vline(xintercept = mean(df$T1), linetype = 1, color = "red", size = 2) +
  theme_bw() +
  scale_fill_brewer(palette = "Paired") +
  ggtitle("Energy Consumption x Temperature (Kitchen)")
```

```
# There is no correlation between these two variables (neither positive nor negative), but it is possible to observe

# that the temperature rise indicates an increase in energy consumption up to close to the average temperature

# in the kitchen, which is equal to mean (df$T1) = 21.68417

setwd(results_directory)

ggsave("3-Energy Consumption x Temperature (Kitchen).png", width = 10, height = 5)

setwd(working_directory)


# 04-Is there a relationship between energy consumption and the average temperature rise in the living room?

analysis <- df %>%

  count(T2, Appliances) %>%

  as.data.frame()

#View(analysis)

ggplot() +

  geom_point(data = analysis, aes(x = T2, y = Appliances ), stat = "identity") +

  theme_bw() +

  scale_fill_brewer(palette = "Paired") +

  ggtitle("Energy Consumption x Temperature (Living Room)")

# There is no correlation between these two variables (neither positive nor negative), but it is possible to observe

# that the temperature rise indicates an increase in energy consumption up to close to the average temperature

# in the living room too, which is equal to mean (df$T2) = 20.34251

setwd(results_directory)

ggsave("4-Energy Consumption x Temperature (Living Room).png", width = 10, height = 5)

setwd(working_directory)


ggplot() +
```

```
  geom_point(data = analysis, aes(x = round(T2), y = Appliances ), stat = "identity")  +
  geom_vline(xintercept = mean(df$T2), linetype = 1, color = "red", size = 2) +
  theme_bw() +
  scale_fill_brewer(palette = "Paired") +
  ggtitle("Energy Consumption x Temperature (Living Room)")
setwd(results_directory)
ggsave("4-Energy Consumption x Temperature (Living Room)B.png", width = 10, height = 5)
setwd(working_directory)

# 05-How does the outside temperature (T_out) affect energy consumption?
analysis <- df %>%
  count(Temp_Outside = round(T_out, digits = 0), Appliances) %>%
  as.data.frame()
#View(analysis)
ggplot() +
  geom_point(data = analysis, aes(x = Temp_Outside, y = Appliances ), stat = "identity")  +
  geom_vline(xintercept = 1.5, linetype = 1, color = "red", size = 2)  +
  geom_vline(xintercept = 11.5, linetype = 1, color = "red", size = 2) +
  theme_bw() +
  scale_fill_brewer(palette = "Paired") +
  ggtitle("Energy Consumption x Temperature Outside")
# The highest energy consumption occurs when the outside temperature is between 0 and 10 ºC
setwd(results_directory)
ggsave("5-Energy Consumption x Temperature Outside.png", width = 10, height = 5)
setwd(working_directory)

# So, let's make some adjustments to the dataset and continue with the Machine Learning models

# Feature Engineering ------------------------------------------------------------------------------
```

```
# Removing non-relevant variables

dfTrain <- subset(dfTrain, select = -c(date, NSM, WeekStatus, Day_of_week))

dfTest <- subset(dfTest, select = -c(date, NSM, WeekStatus, Day_of_week))

str(dfTrain)

#or by number

#non.vars <- c(1, 30, 31, 32)

#dfTrain <- dfTrain[ , -c(non.vars)]

#str(dfTrain)


# Missing Values?

sum(is.na(dfTrain)) # There is no missing values, but if there was, the next command helps identify wich column is N/A

sum(is.na(dfTest))  # There is no missing values, but if there was, the next command helps identify wich column is N/A

# sapply function applies "function(x)sum(is.na(x))" to each column in the dataset

sapply(dfTrain, function(x)sum(is.na(x)))


# Target feature frequency

as.data.frame(table(dfTrain$Appliances))

#or in another way

table(dfTrain$Appliances)

prop.table(table(dfTrain$Appliances)) * 100


# Visual Analysis

# BoxPlot e Histogram

setwd(results_directory)

png('6-Boxplot.png', width = 800, height = 1000, res = 100)

boxplot(dfTrain$Appliances)
```

```
dev.off()

png('7-Histogram.png', width = 1200, height = 900, res = 100)

hist(dfTrain$Appliances)

dev.off()

setwd(working_directory)

# We have possible outliers, but these outliers can be wrong collections or people spending more than the average.

# Most frequency energy use (in Wh) is between 0 - 100


# Scaling Variables

# Multiple Linear Regression Models expect to receive standard numerical variables.

str(dfTrain)

# As there is no categorical variables, let's scale all of them (mainly the "Press_mm_hg")

# Function (scale.features) for normalization of numerical variables. Both Linear Regression and Logistic Regression DEMANDS that

# numeric variables are on the same scale.

# -> The scale transformation calculates the standard deviation for an attribute and divides

# each value by this standard deviation.

# -> The "center" transformation averages an attribute and subtracts it from each value.

scale.features <- function(df, variables){

 for (variable in variables){

  df[[variable]] <- scale(df[[variable]], center = T, scale = T)

 }

 return(df)

}


# Getting all cols (except "Appliances" the target variable)

cols <- names(dfTrain[-1])

# Calling our function created above with the cols to scale
```

```
dfTrain_Scaled <- scale.features(dfTrain,cols)

dfTest_Scaled <- scale.features(dfTest,cols)

#View(dfTrain_Scaled)

#or simply this way (just because all columns are numericals)

#dfTrain_Scaled <- scale(dfTrain, center = T, scale = T)


# Correlation

# We do the colinearity calculation EVERYTIME we are working with Regression, and especially when we are working with multiple linear correlation because there is

# another associated issue that is multicollinearity, that is, I can have 2 or 3 input variables that have the same level of information and it will affect how the

# model is built. Eventually I have to calculate the correlation in all variables, identify if I have multicollinearity, if there is one I can remove one of these

# variables and then I create the regression model


# Defining cols to correlation analysis

cols <- names(dfTrain_Scaled)


# CORRELATION METHODS - CORRELATION IS THE WAY TO FIND THE MOST RELEVANT VARIABLES TO CONTINUE

# Pearson - coefficient used to measure the degree of relationship between two variables with a linear relationship

# Spearman - non-parametric test, to measure the degree of relationship between two variables

# Kendall - non-parametric test, to measure the dependence force between two variables


# Vector with correlation methods

meth <- c("pearson", "spearman")


# Applying the correlation methods with the color () function

# lapply -> PERFORM A LOOP FOR LISTS OR VECTORS, OR BETTER, APPLY A FUNCTION TO A LIST OR VECTOR
```

```
cors <- lapply(meth, function(method)(cor(dfTrain_Scaled[, cols], method = method)))


#head(cors)


# Preparing the correlation plot


# Preparing colors to plot - https://mycolor.space/

# Level Colors

col.l <- colorRampPalette(c('#EBFFF9', '#D6FFF3', '#B7FFE9', '#8BFFDC', '#65D9CD', '#4CB3B8', '#3F8D9C',
'#38697C', '#2F4858'))(90)


# ADD ZEROS TO DIAGONALS

# levelplot -> DESIGN LEVEL COLORS TO THE GRAPHIC

plot.cors <- function(x, labs){

  diag(x) <- 0.0

  plot( levelplot(x,

                  main = paste("Plot de Correlação usando Método", labs),

                  scales = list(x = list(rot = 90), cex = 1.0),

                  col.regions=col.l))

}


# Correlation Map

setwd(results_directory)

png('8-Correlation_Map.png', width = 900, height = 900, res = 100)

Map(plot.cors, cors, meth)

dev.off()

setwd(working_directory)
```

```
# Looking at the above map, there are some features high correlated, as it's difficult to analyze all of them, I'll use a

# Machine Learning Model to extract the most important features.


# Machine Learning Process I (Important Variables) ----------------------------------------------------------

# Multiple Linear Regression(model_v1) ------------------------------------------------------------------------

# First I'll use all features however, as mentioned above, some of them are not helping the ML Process

# We want to know which combination of predictors will best predict energy efficiency.

# Which predictors increase our accuracy by a statistically significant amount?

# We can guess some of the graph's trends, but I really want to run a test

# statistician to determine which predictors are significant and to determine the optimal formula

# for the forecast.

set.seed(987)

model_v1 <- lm(Appliances ~ ., data = dfTrain_Scaled)

summary(model_v1)


# Resulting in:

# - R2 very low = 0.1694 (reference is 0 to 1, where the bigger the better)

# - High Std.Error = 0.9121 (the lower the better)

# - optimal p-value when working with all variables (p-value <0.05 indicates that the explanatory variable helps to explain the target variable, that is, it is a good predictor)


# So let's look at multicollinearity to find the best match


# Stepwise Selection Method (Stepwise Selection Method) ----------------------------------------

# Among the available methods, we will perform the gradual selection to help us select

# a subset of variables that best explain "Appliances".


# The Akaike information criterion (AIC) is an estimator of forecast error and therefore estimates the
```

```
# Relative quality of statistical models for a given data set.

# Given a collection of models for the data, AIC estimates the quality of each model, in relation to

# each of the other models. Thus, the AIC provides a means for selecting models.

# The AIC coefficient is based on information theory. When a statistical model is used to

# represent the process that generated the data, the representation will almost never be accurate; therefore,

# some information will be lost using the model to represent the process. AIC estimates

# the relative amount of information lost by a given model: the less information

# a model loses, the higher the quality of that model.

# Briefly, the AIC method will perform the construction of several models with different combinations of predictor variables

# and compare the performance of each model. The best model will be the one with the best variables (attributes).

#? stepAIC

# direction -> can be forward, backforward, or both. This hyperparameter is the direction of the steps that the algorithm will take,

#           both it goes both forward (when the model improves) and backwards (when the model gets worse, taking steps

#           backwards in order to return to a model that was better)

# trace -> FALSE to not print the results on the screen.

step <- stepAIC(model_v1, direction = "both", trace = FALSE)

summary(step)

summary(step)$coeff

summary(step)$r.squared

# - R2 very low = 0.1692 (reference is 0 to 1, where the bigger the better)

# We thus find the most relevant variables
```

```
# The variables are: lights + RH_1 + T2 + RH_2 + T3 + RH_3 + T6 + RH_6 + RH_7 + T8 + RH_8 + T9 +
RH_out + Windspeed


# Multiple Linear Regression(model_v2) ----------------------------------------------------------------------

# New Model (lm) to analyze R2, std err and p-value with the most relevant variables

set.seed(456)

model_v2 <- lm(Appliances ~
lights+RH_1+T2+RH_2+T3+RH_3+T6+RH_6+RH_7+T8+RH_8+T9+Press_mm_hg+RH_out+Windspeed,

        data = dfTrain_Scaled)

summary(model_v2)


# - R2 very low = 0.1647 (reference is 0 to 1, where the bigger the better)


# Detecting Collinearity

# A major problem with multivariate regression is collinearity. If two or more variables

# predictive are highly correlated and both are inserted into a regression model,

# increases the true standard error and you get very unstable slope estimates.

# We can assess collinearity by the Variance Inflation Factor (VIF). Let's look at the

# inflation factors of variation if we cast all variables into the model.

#?vif

# kable -> formats the result into a friendly table

# vif -> calculates the variance inflation and also the variance factors in a generalized way for linear
models

kable(vif(model_v2), align = 'c')

# Interpreting:

# - Values for VIF greater than 10 are considered large.

# - We must also pay attention to VIF values between 5 and 10.


# Indications that RH_6, Press_mm_hg are collinear, I will eliminate and make a new model
```

```
# Multiple Linear Regression(model_v3) ----------------------------------------------------------------------------

# New Model (lm) to analyze R2, std err and p-value with the most relevant variables

time_init <- Sys.time()

set.seed(74)

model_v3 <- lm(Appliances ~
lights+RH_1+T2+RH_2+T3+RH_3+T6+RH_7+T8+RH_8+T9+RH_out+Windspeed,

        data = dfTrain_Scaled)

summary(model_v3)


# - R2 very low = 0.1645 (reference is 0 to 1, where the bigger the better)


kable(vif(model_v3), align = 'c')


# How is correlation now:

# Defining cols to correlation analysis

cols <-
c("lights","RH_1","T2","RH_2","T3","RH_3","T6","RH_6","RH_7","T8","RH_8","T9","Press_mm_hg","RH_
out","Windspeed")

meth <- c("pearson", "spearman")

cors <- lapply(meth, function(method)(cor(dfTrain_Scaled[, cols], method = method)))

Map(plot.cors, cors, meth)

# Now without RH_1 and RH_3

cols <-
c("lights","RH_1","T2","RH_2","T3","RH_3","T6","RH_7","T8","RH_8","T9","RH_out","Windspeed")

meth <- c("pearson", "spearman")

cors <- lapply(meth, function(method)(cor(dfTrain_Scaled[, cols], method = method)))

Map(plot.cors, cors, meth)


# Forecasts in the final (lm) model (model_v3)

str(dfTest_Scaled)
```

```
# Adjusting dfTest_Scaled with just the columns to predict

dfTest_Scaled_Pred <-
dfTest_Scaled[,c("Appliances","lights","RH_1","T2","RH_2","T3","RH_3","T6","RH_7","T8","RH_8","T9","
RH_out","Windspeed")]

#View(dfTest_Scaled_Pred)

# Now making predictions

pred <- round(predict(model_v3, dfTest_Scaled_Pred, type = 'response'))

expe <- as.numeric(dfTest_Scaled$Appliances)

#View(pred)


# Score RMSE of this model

rmse <- RMSE(pred, expe)

# RMSE = 93.96


# Score AUC of this model

curve_roc <- multiclass.roc(response = dfTest_Scaled$Appliances, predictor = pred)

class(dfTest_Scaled$Appliances)

class(pred)


accuracy <- curve_roc$auc

# Multi-class area under the curve: 0.7126 (That means, 71.26% Accuracy)


# Plot AUC

setwd(results_directory)

png('9-AUC(lm).png', width = 1000, height = 1000, res = 100)

plot(roc(dfTest_Scaled$Appliances, pred))

dev.off()

setwd(working_directory)
```

```
time.end  <- Sys.time()

# Time Difference from beggining to end
duration <- round(time.end - time_init, 2)

# I've got poor results using this "lm" algorithm. So, I'll try randomForest.

# Combining the above Results
model <- "lm"
dfResults1 <- data.frame(model, rmse, accuracy, round(duration/60, digits = 2))
colnames(dfResults1) <- c("Model", "RMSE", "Accuracy", "Time_min")
dfResults <- rbind(dfResults, dfResults1)



# Machine Learning Process II (randomForest) ----------------------------------------------------------------------
# Assessing the importance of all variables
# CREATING A MODEL WITH randomForest AND AFTER EXTRACTING THE MOST RELEVANT VARIABLES
time_init <- Sys.time()
set.seed(12)
model_var_imp <- randomForest(Appliances ~ . ,
                              data = dfTrain_Scaled,
                              ntree = 85,
                              nodesize = 10,
                              importance = TRUE)
varImpPlot(model_var_imp)
# Most important features:
lights+RH_1+T2+RH_2+T3+RH_3+T6+RH_7+T8+RH_8+T9+RH_out+Windspeed

# Adjusting dfTest_Scaled with just the columns to predict
```

```
dfTest_Scaled_Pred <-
dfTest_Scaled[,c("Appliances","lights","RH_1","T2","RH_2","T3","RH_3","T6","RH_7","T8","RH_8","T9","
RH_out","Windspeed")]

#View(dfTest_Scaled_Pred)


# New Model with the most important features analyzed by randomForest algorithm

# I'll analyse overfitting either

set.seed(178)

model4 <- function(n){

  model_v4 <- randomForest(Appliances ~
lights+RH_1+T2+RH_2+T3+RH_3+T6+RH_7+T8+RH_8+T9+RH_out+Windspeed,

                           data = dfTrain_Scaled,

                           ntree = n,

                           nodesize = 10)

  predicted4 <- round(predict(model_v4, newdata = dfTest_Scaled_Pred), digits = 0)

  expected4 <- dfTest_Scaled$Appliances


  return(RMSE(predicted4, expected4))

}


# Constructing a table to store RMSE values for analysis

tabRMSE <- data.frame(ntree = seq(5,100,5))

Result <- c()


# Control function

for (i in tabRMSE$ntree) {

  Result <- append(Result, model4(i))

}


# Merging Results and Analyzing Results
```

```
tabRMSE <- cbind(tabRMSE, Result)


# Graphical Analysis
colnames(tabRMSE) <- c('ntree', 'ResultRMSE')


setwd(results_directory)
png('10-RMSE Analysis.png', width = 2000, height = 900, res = 100)
ggplot(tabRMSE, aes(x = ntree, y = ResultRMSE)) +
  geom_point() +
  stat_smooth(method = 'lm', formula = y ~ poly(x,13), se= FALSE) +
  labs(title = "RMSE Analysis - Model Choice", x = "ntree", y = 'Values') + guides(color = 'none') +
theme_bw()
dev.off()
setwd(working_directory)


# Optimum number of trees
set.seed(374)
ntree = 80 # AUTOMATIZAR ESTA LINHA ATRAVÉS DA TABELA tabRMSE
model_v4 <- randomForest(Appliances ~
lights+RH_1+T2+RH_2+T3+RH_3+T6+RH_7+T8+RH_8+T9+RH_out+Windspeed,
                         data = dfTrain_Scaled,
                         ntree = ntree,
                         nodesize = 10)
print(model_v4)


# Making Predictions
pred <- round(predict(model_v4, newdata = dfTest_Scaled_Pred), digits = 0)
expe <- as.numeric(dfTest_Scaled$Appliances)
```

```
# Score RMSE of this model
rmse <- RMSE(pred, expe)
# RMSE = 74.10

# Score AUC of this model
curve_roc <- multiclass.roc(response = dfTest_Scaled$Appliances, predictor = pred)
class(dfTest_Scaled$Appliances)
class(pred)

accuracy <- curve_roc$auc
# Multi-class area under the curve: 0.7788 (That means, 77.88% Accuracy)

# Plot AUC
setwd(results_directory)
png('11-AUC(Random Forest).png', width = 1000, height = 1000, res = 100)
plot(roc(dfTest_Scaled$Appliances, pred))
dev.off()
setwd(working_directory)

time.end  <- Sys.time()

# Time Difference from beggining to end
duration <- round(time.end - time_init, 2)

# Combining the above Results
model <- "Random Forest"
dfResults1 <- data.frame(model, rmse, accuracy, duration)
colnames(dfResults1) <- c("Model", "RMSE", "Accuracy", "Time_min")
dfResults <- rbind(dfResults, dfResults1)
```

```
# This is the lowest RMSE I could achieve using an randomForest model. Now I'll try an SVM.


# Machine Learning Process III (SVM) ------------------------------------------------------------

# IF I KEEP AS A FACTOR, THEN SVM WILL REALIZE A REGRESSION AND NOT A CLASSIFICATION PROBLEM, THEREFORE

# PLACING THE TARGET VARIABLE AS A FACTOR INDICATES FOR THE SVM MODEL WHAT TYPE OF PROBLEM IT HAS TO PERFORM (REGRESSION

# OR CLASSIFICATION)


# -----> First version of the SVM model - Standard Version with Radial Kernel (RBF) - Takes a little while

# The algorithm chooses the type of SVM according to the data type of the target variable, as stated above

#?svm

time_init <- Sys.time()

set.seed(357)

model_v5 <- svm(Appliances ~ lights+RH_1+T2+RH_2+T3+RH_3+T6+RH_7+T8+RH_8+T9+RH_out+Windspeed,

                data = dfTrain_Scaled,

                na.action = na.omit,

                scale = TRUE)

summary(model_v5)

print(model_v5)


# Making predictions with the model

# Adjusting dfTest_Scaled with just the columns to predict

dfTest_Scaled_Pred <- dfTest_Scaled[,c("Appliances","lights","RH_1","T2","RH_2","T3","RH_3","T6","RH_7","T8","RH_8","T9","RH_out","Windspeed")]

#View(dfTest_Scaled_Pred)
```

```
# Making Predictions
pred <- round(predict(model_v5, newdata = dfTest_Scaled_Pred), digits = 0)
expe <- as.numeric(dfTest_Scaled$Appliances)

# Score RMSE of this model
rmse <- RMSE(pred, expe)
# RMSE = 91.27

# Score AUC of this model
curve_roc <- multiclass.roc(response = dfTest_Scaled$Appliances, predictor = pred)
class(dfTest_Scaled$Appliances)
class(pred)

accuracy <- curve_roc$auc
# Multi-class area under the curve: 0.7317 (That means, 73.17% Accuracy)

# Plot AUC
setwd(results_directory)
png('12-AUC(SVM I).png', width = 1000, height = 1000, res = 100)
plot(roc(dfTest_Scaled$Appliances, pred))
dev.off()
setwd(working_directory)

time.end  <- Sys.time()

# Time Difference from beggining to end
duration <- round(time.end - time_init, 2)

# This model took 40 min to train.
```

```
# Hoping to improve Accuracy, I will continue with SVM but I will do a GridSearch to find the best one
# parameter setting.

# Combining the above Results
model <- "SVM I"
dfResults1 <- data.frame(model, rmse, accuracy, duration)
colnames(dfResults1) <- c("Model", "RMSE", "Accuracy", "Time_min")
dfResults <- rbind(dfResults, dfResults1)

# -----> Second version of the SVM model - Version with Linear Kernel and GridSearch - It takes a good few minutes

# Let's do a grid search (Grid Search) to adjust hyperparameters and use a linear kernel.
# But here we will not keep the cost higher than 2, so that outliers do not affect extensively
# the creation of decision limits and, therefore, lead to overfitting.
#?tune
time_init <- Sys.time()
set.seed(182)
# ranges -> cost parameter for this case. 5 models will be created.
# modelo_grid1 -> will be the 5 models, then I will extract the best
model_grid1 <- tune(svm,
                    Appliances ~ lights+RH_1+T2+RH_2+T3+RH_3+T6+RH_7+T8+RH_8+T9+RH_out+Windspeed,
                    data = dfTrain_Scaled,
                    kernel = 'linear',
                    ranges = list(cost = c(0.05, 0.1, 0.5, 1, 2)))

summary(model_grid1)

# Best model parameters
```

```
model_grid1$best.parameters

# cost = 2


# Extracting the Best Model

model_grid1$best.model

model_v6 <- model_grid1$best.model

summary(model_v6)


# Making Predictions

dfTest_Scaled_Pred <-
dfTest_Scaled[,c("Appliances","lights","RH_1","T2","RH_2","T3","RH_3","T6","RH_7","T8","RH_8","T9","
RH_out","Windspeed")]


# Making Predictions

pred <- round(predict(model_v6, newdata = dfTest_Scaled_Pred), digits = 0)

expe <- as.numeric(dfTest_Scaled$Appliances)


# Score RMSE of this model

rmse <- RMSE(pred, expe)

# RMSE = ???


# Score AUC of this model

curve_roc <- multiclass.roc(response = dfTest_Scaled$Appliances, predictor = pred)

class(dfTest_Scaled$Appliances)

class(pred)


accuracy <- curve_roc$auc

# Multi-class area under the curve: 0.7212 (That means, 72.12% Accuracy)
```

```
# Plot AUC
setwd(results_directory)
png('13-AUC(SVM II).png', width = 1000, height = 1000, res = 100)
plot(roc(dfTest_Scaled$Appliances, pred))
dev.off()
setwd(working_directory)

time.end  <- Sys.time()

# Time Difference from beggining to end
duration <- round(time.end - time_init, 2)

# It is offering a slightly worse performance than the previous one. This model took 20 minutes to train.
# Therefore, I will not consider this model_v6

# Combining the above Results
model <- "SVM II"
dfResults1 <- data.frame(model, rmse, accuracy, duration)
colnames(dfResults1) <- c("Model", "RMSE", "Accuracy", "Time_min")
dfResults <- rbind(dfResults, dfResults1)

# -----> Third version of the SVM model - Version with Polynomial Kernel and GridSearch - It takes a few minutes

# Let's do a grid search (Grid Search) to adjust hyperparameters and use Polynomial Kernel.
# But here we will not keep the cost higher than 2, so that outliers do not affect extensively
# the creation of decision limits and, therefore, lead to overfitting.
time_init <- Sys.time()
set.seed(182)
```

```
model_grid2 <- tune(svm,
                    Appliances ~ lights+RH_1+T2+RH_2+T3+RH_3+T6+RH_7+T8+RH_8+T9+RH_out+Windspeed,
                    data = dfTrain_Scaled,
                    kernel = 'polynomial',
                    ranges = list(cost = c(1,2), degree = c(2,3,4)))

summary(model_grid2)

# Best model parameters
model_grid2$best.parameters
# cost = 2
# degree = 4

# Extracting the Best Model
model_grid2$best.model
model_v7 <- model_grid1$best.model
summary(model_v7)

# Making Predictions
dfTest_Scaled_Pred <-
dfTest_Scaled[,c("Appliances","lights","RH_1","T2","RH_2","T3","RH_3","T6","RH_7","T8","RH_8","T9","
RH_out","Windspeed")]

# Making Predictions
pred <- round(predict(model_v7, newdata = dfTest_Scaled_Pred), digits = 0)
expe <- as.numeric(dfTest_Scaled$Appliances)

# Score RMSE of this model
rmse <- RMSE(pred, expe)
```

```
# Score AUC of this model
curve_roc <- multiclass.roc(response = dfTest_Scaled$Appliances, predictor = pred)
class(dfTest_Scaled$Appliances)
class(pred)

accuracy <- curve_roc$auc
# Multi-class area under the curve: 0.7212 (That means, 72.12% Accuracy)

# Plot AUC
setwd(results_directory)
png('14-AUC(SVM III).png', width = 1000, height = 1000, res = 100)
plot(roc(dfTest_Scaled$Appliances, pred))
dev.off()
setwd(working_directory)

time.end  <- Sys.time()

# Time Difference from beggining to end
duration <- round(time.end - time_init, 2)

# Did this model take 21 min to be trained.
# SVM model_v5 gave the best performance (73.12%):
# - To show the results in a better way I'll cluster energy comsuption in 9 different clients using a Machine Learning Model (K-Means).
#   In this way I'll be able to analyze groups of energy instead of lots of different energy use.

# Combining the above Results
model <- "SVM III"
```

```
dfResults1 <- data.frame(model, rmse, accuracy, duration)

colnames(dfResults1) <- c("Model", "RMSE", "Accuracy", "Time_min")

dfResults <- rbind(dfResults, dfResults1)


# Final Results ----------------------------------------------------------------------------------------

# Storing Models Results in csv format:

setwd(results_directory)

write.csv(dfResults, file = 'Final_Results.csv')


# ATTENTION:

# The error below happens in the geom_bar () ggplot plots, explanation:

#"Error: stat_count() must not be used with a y aesthetic."

#This error comes due to the mapping of elements in aes() while plotting barplot.

# geom_bar by default takes a count of values as y in a bar plot. So use stat="identity" within barplot()
to avoid the default and use fields as used in mapping function.

# ggplot(data,aes(field1,field2)) + geom_plot(stat="identity")


png('15-Final_Results(Accuracy).png', width = 800, height = 500, res = 100)

# Looking at Accuracy by Model

ggplot() +

  geom_bar(data = dfResults,

           aes(x = Model, y = Accuracy),

           stat = 'identity') +

  coord_flip() +

  ggtitle("Models x Accuracy")

dev.off()


png('16-Final_Results(Time).png', width = 700, height = 400, res = 100)

# Looking at Time by Model
```

```
ggplot() +
  geom_bar(data = dfResults,
           aes(x = Model, y = as.numeric(Time_min)),
           stat = 'identity') +
  coord_flip() +
  ggtitle("Models x Time(in minutes)")
dev.off()


setwd(working_directory)


# Clustering ----------------------------------------------------------------------------------------------------
# Chosen model: model_v4 (Random Forest)
# Making Predictions
pred <- round(predict(model_v4, newdata = dfTest_Scaled_Pred), digits = 0)
expe <- as.numeric(dfTest_Scaled$Appliances)


# Determining k as the object of study and joining the predictions made by the Random Forest model_v4
to the test dataframe
k <- as.data.frame(cbind(pred, expe))
colnames(k) <- c('pred', 'Real')
#View(k)
# Wh filter
#k <- k %>%
#      filter(Real <= 150)


# Expected vs. Predicted
ggplot() +
  geom_point(data = k, aes(x = pred, y = Real )) +
  theme_bw()
```

```
setwd(results_directory)
ggsave("17-Predicted x Expected.png", width = 13, height = 6)
setwd(working_directory)

# Creating model_k to perform Machine Learning and calculate clusters
model_k <- kmeans(k$pred, 9)
print(model_k)

# Checking the size of the clusters, that is, the number of customers in each of the 9 clusters
model_k$size

# Checking the center of the clusters
model_k$centers

# Cluster Map (k) - that is, joining Wh to its respective Cluster
k$Cluster <- model_k$cluster

# Center Map (k) - that is, joining the Cluster to its respective Center
lin = 0
for (i in k$Cluster) {
  lin = lin + 1
  #print(lin)
  if (i == 1) {k$Center[lin] <- model_k$centers[1]}
  if (i == 2) {k$Center[lin] <- model_k$centers[2]}
  if (i == 3) {k$Center[lin] <- model_k$centers[3]}
  if (i == 4) {k$Center[lin] <- model_k$centers[4]}
  if (i == 5) {k$Center[lin] <- model_k$centers[5]}
  if (i == 6) {k$Center[lin] <- model_k$centers[6]}
  if (i == 7) {k$Center[lin] <- model_k$centers[7]}
```

```
  if (i == 8) {k$Center[lin] <- model_k$centers[8]}
  if (i == 9) {k$Center[lin] <- model_k$centers[9]}
}

#View(k)

# Range of each Cluster and distances between its Centers
ggplot() +
  geom_point(data = k, aes(x = Center, y = pred, color = as.factor(Cluster)), stat = "identity") +
  theme_bw()
setwd(results_directory)
ggsave("18-Cluster Center x Cluster Range.png", width = 12, height = 5)
setwd(working_directory)

# Evolution of the Cluster
ggplot() +
  geom_point(data = k, aes(x = pred, y = pred, color = as.factor(Cluster) ), stat = "identity") +
  theme_bw()

# Expected vs. Real + Cluster Centers
ggplot() +
  geom_point(data = k, aes(x = pred, y = Real )) +
  geom_point(aes(x=model_k$centers[1], y=model_k$centers[1]), shape = 18, size = 3.7,
colour="firebrick1") +
  geom_point(aes(x=model_k$centers[2], y=model_k$centers[2]), shape = 18, size = 3.7,
colour="firebrick1") +
  geom_point(aes(x=model_k$centers[3], y=model_k$centers[3]), shape = 18, size = 3.7,
colour="firebrick1") +
  geom_point(aes(x=model_k$centers[4], y=model_k$centers[4]), shape = 18, size = 3.7,
colour="firebrick1") +
```

```
  geom_point(aes(x=model_k$centers[5], y=model_k$centers[5]), shape = 18, size = 3.7,
colour="firebrick1") +

  geom_point(aes(x=model_k$centers[6], y=model_k$centers[6]), shape = 18, size = 3.7,
colour="firebrick1") +

  geom_point(aes(x=model_k$centers[7], y=model_k$centers[7]), shape = 18, size = 3.7,
colour="firebrick1") +

  geom_point(aes(x=model_k$centers[8], y=model_k$centers[8]), shape = 18, size = 3.7,
colour="firebrick1") +

  geom_point(aes(x=model_k$centers[9], y=model_k$centers[9]), shape = 18, size = 3.7,
colour="firebrick1") +

  theme_bw()

setwd(results_directory)

ggsave("19-Predicted x Expected (with Cluster Centers).png", width = 13, height = 6)

setwd(working_directory)


# Cluster of each consumption group

ggplot() +

  geom_point(data = k, aes(x = pred, y = Real, color = as.factor(Cluster) ), stat = "identity") +

  theme_bw()

setwd(results_directory)

ggsave("20-Predicted x Expected Clustered (with Cluster Centers).png", width = 13, height = 6)

setwd(working_directory)


# Cluster of each consumer group + Cluster Centers

ggplot() +

  geom_point(data = k, aes(x = pred, y = Real, color = as.factor(Cluster) ), stat = "identity") +

  geom_point(aes(x=model_k$centers[1], y=model_k$centers[1]), shape = 18, size = 3.7,
colour="firebrick1") +

  geom_point(aes(x=model_k$centers[2], y=model_k$centers[2]), shape = 18, size = 3.7,
colour="firebrick1") +

  geom_point(aes(x=model_k$centers[3], y=model_k$centers[3]), shape = 18, size = 3.7,
colour="firebrick1") +
```

```
  geom_point(aes(x=model_k$centers[4], y=model_k$centers[4]), shape = 18, size = 3.7,
colour="firebrick1") +
  geom_point(aes(x=model_k$centers[5], y=model_k$centers[5]), shape = 18, size = 3.7,
colour="firebrick1") +
  geom_point(aes(x=model_k$centers[6], y=model_k$centers[6]), shape = 18, size = 3.7,
colour="firebrick1") +
  geom_point(aes(x=model_k$centers[7], y=model_k$centers[7]), shape = 18, size = 3.7,
colour="firebrick1") +
  geom_point(aes(x=model_k$centers[8], y=model_k$centers[8]), shape = 18, size = 3.7,
colour="firebrick1") +
  geom_point(aes(x=model_k$centers[9], y=model_k$centers[9]), shape = 18, size = 3.7,
colour="firebrick1") +
  theme_bw()
setwd(results_directory)
ggsave("21-Predicted x Expected Clustered (with Cluster Centers).png", width = 13, height = 6)
setwd(working_directory)

# In order not to pollute the plot area much, I will reduce the dimensionality and plot in 4 Clusters
setwd(results_directory)
png('22-Reduced Clusters.png', width = 700, height = 500, res = 100)
autoplot(pam(k[-c(3,4)], 4)) +
  theme_bw()
dev.off()
setwd(working_directory)
```