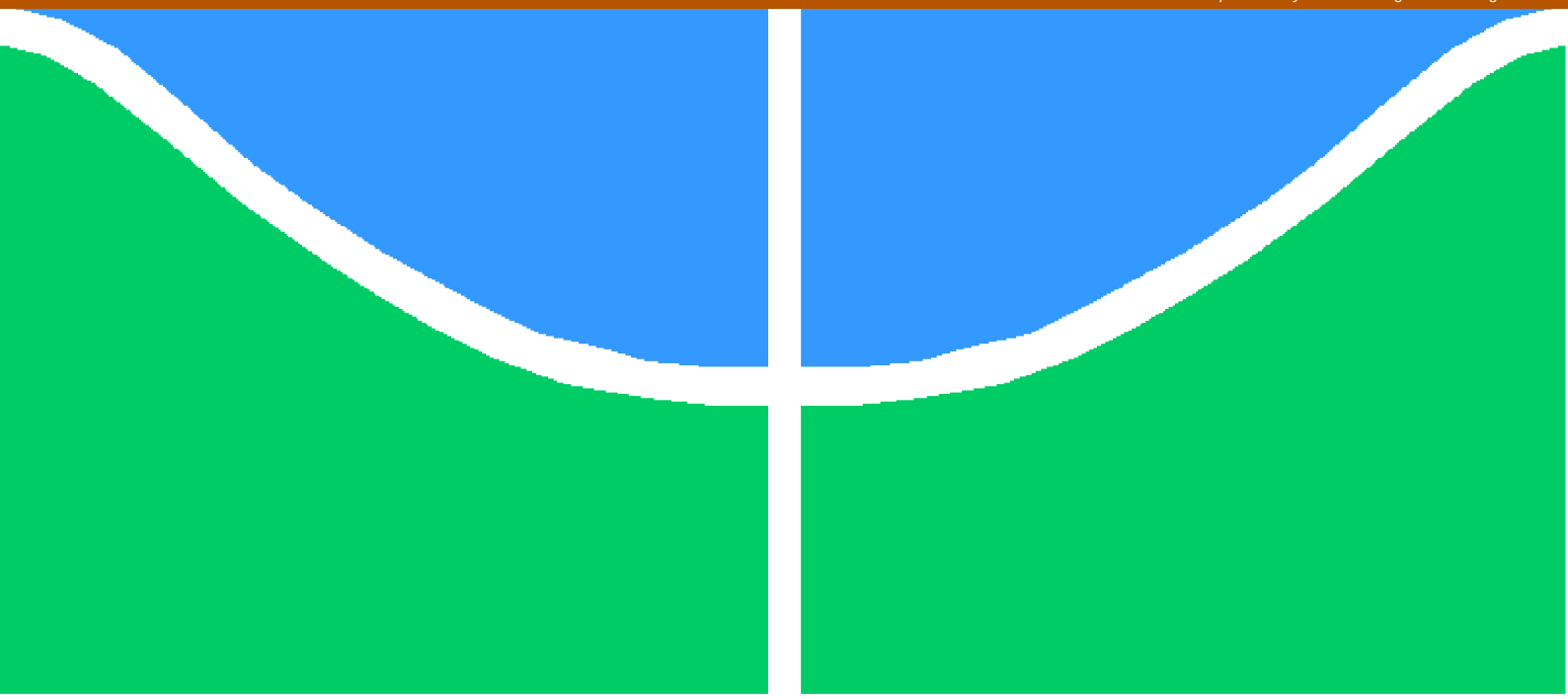


Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Energia

Métodos em *machine learning* para construção de curvas de carga a partir de medições

Autor: Tiago Simon Engelsdorff

Orientador: Dr. Alex Reis

Brasília, DF

2019

Tiago Simon Engelsdorff

Métodos em *machine learning* para construção de curvas de carga a partir de medições

Monografia submetida ao curso de graduação em Engenharia de Energia da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Energia.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Alex Reis

Brasília, DF

2019

Tiago Simon Engelsdorff
Métodos em *machine learning* para construção de curvas de carga a partir de medições/ Tiago Simon Engelsdorff. – Brasília, DF, 2019-
63 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Alex Reis

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2019.

1. curvas de carga. 2. *machine learning*. I. Dr. Alex Reis. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Métodos em *machine learning* para construção de curvas de carga a partir de medições

CDU

Tiago Simon Engelsdorff

Métodos em *machine learning* para construção de curvas de carga a partir de medições

Monografia submetida ao curso de graduação em Engenharia de Energia da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Energia.

Trabalho aprovado. Brasília, DF, 01 de junho de 2013:

Dr. Alex Reis
Orientador

Dr. Jorge Andrés Cormane Angarita
Convidado 1

Ms. Renato Coral Sampaio
Convidado 2

Brasília, DF
2019

Agradecimentos

A minha família, por todo o apoio desde sempre, e por proporcionar condições incríveis para que eu me tornasse quem sou hoje. Sem todo seu incentivo, carinho e investimento em mim, esse trabalho certamente não aconteceria.

A minha namorada, Mari, por sempre me fazer querer ser uma pessoa melhor e mais esforçada, e por ser a pessoa mais doce e carinhosa que a vida me trouxe. Além disso, pelo auxílio com as imagens utilizadas nesse trabalho.

À Amigos do Sol™, por ser minha família em Brasília ao longo de todos esses anos, e por todos os momentos especiais.

Aos amigos Gustavo e Juliana, por terem feito minha graduação muito mais divertida e empolgante.

Aos professores Alex, Loana e Renato, pela oportunidade de participar desse trabalho e pela confiança em mim depositada.

Aos professores Ronni e Shayani, pelas demais oportunidades concedidas ao longo da graduação e por serem mentores incríveis.

Á CEB Distribuidora, pelo suporte financeiro para a execução dessa pesquisa por meio do projeto de P&D "Geração distribuída no campus da Universidade de Brasília integrada à rede de distribuição da CEB"e do projeto PEE "Projeto de eficiência energética em prédios da Universidade de Brasília".

Resumo

As curvas de carga são uma ferramenta gráfica e matemática de extrema importância na operação de sistemas elétricos. O presente trabalho trata da análise de métodos para a construção de curvas de carga a partir de medições na Universidade de Brasília, através da implementação de um modelo computacional de agregação de dados. São abordados diversos métodos em *machine learning* – em especial, métodos de clusterização (k-médias, clusterização hierárquica espacial, mapas auto-organizáveis de Kohonen e maximização de expectativas) e *support vector machines* - como possíveis soluções para a agregação e tratamento dos dados colhidos. A literatura revisada nesse trabalho traz a aplicação dos métodos apresentados em problemas de elaboração de perfis típicos de carga. Foi definida a utilização do SVR (*support vector regression*) para a implementação de um algoritmo, que, alimentado com o dados dos medidores, resultou em perfis típicos de carga para os dias da semana em um dos prédios da universidade.

Palavras-chave: curvas de carga, *machine learning*, *support vector regression.*

Abstract

Load curves are a graphical and mathematical tool of extreme importance in the operation of electrical systems. The present work deals with the analysis of the methods to building load curves from measurements in the University of Brasília, through the implementation of a data-aggregation computational model. Many machine learning methods - namely, the clusterization methods (k-means, hierarquical clustering, Kohonen's self organizing maps and expectation maximazation) and support vector machines - are presented as possible solutions for the aggregation and treatment of the obtained data. The literature reviewed in this work shows the application of the presented methods for load profiling problems. The use of SVR (support vector regression) was defined for the implementation of an algorithm, which, fed with the data from the installed meter, resulted in load profiles for each day of the week in one of the university's buildings.

Key-words: load curve, machine learning, support vector regression.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

O manejo da energia em um sistema, com diversos consumidores e padrões de consumo diferentes, de um modo instantâneo e que garanta o abastecimento de todos continuamente, é um dos maiores desafios do setor elétrico. Seja para sua boa operação e atendimento ininterrupto e satisfatório, seja para determinação de tarifas, encargos contratuais ou contratação de demanda, é primordial que o operador de um sistema tenha a capacidade de prever o consumo de seus clientes.

Dessa necessidade, surge o conceito de curva de carga. As curvas de carga são elementos utilizados para antever o comportamento de um sistema elétrico ao longo do tempo. Sua construção parte da análise do comportamento anterior do mesmo sistema e dos fatores externos que o levaram a se comportar como tal, em comparação com os fatores aos quais o mesmo sistema infere-se ser inserido no momento em que desejamos analisá-lo.

No contexto no qual este trabalho está inserido, o sistema delimitado para análise é a rede elétrica da Universidade de Brasília (UnB), a partir da instalação de medidores em alguns de seus prédios e coleta dos dados de consumo nos pontos analisados, de forma a agregá-los, formando um conjunto de perfis típicos para a universidade e um algoritmo de previsibilidade.

Para realização de tal levantamento, foram desenvolvidos, ao longo dos anos, diversos métodos, sejam eles estatísticos ou baseados em inteligência artificial, bem como outras soluções não abordadas aqui. Em especial, serão enfatizados os métodos em *machine learning*.

Portanto, o escopo desse trabalho consiste basicamente da compreensão do processo de construção de uma curva de carga e de uma revisão dos diferentes métodos *machine learning*. Analisados os diferentes métodos, o objetivo é selecionar aquele que melhor se adeque ao problema a ser resolvido, e desenvolver um algoritmo capaz de conferir curvas de carga para os prédios da UnB a partir das medições coletadas.

1.1 Objetivos

1.1.1 Objetivo geral

Criação de um algoritmo para construção de curvas típicas de carga referentes a cada dia da semana, a partir de dados coletados em um dos medidores instalados nos prédios da Universidade de Brasília.

1.1.2 Objetivos específicos

- Revisar e conhecer alguns dos diferentes métodos em *machine learning*;
- Determinar o método mais apropriado para o problema;
- Elaborar o algoritmo e aplicar nos dados disonibilizados para avaliar seu comportamento.

1.2 Organização do trabalho

O trabalho se encontra subdividido conforme os seguintes itens:

- Caracteriação do problema: apresentação dos medidores e dos requisitos a serem cumpridos pelo algoritmo;
- Explicação e análise dos diferentes métodos em *machine learning* estudados ao longo do trabalho;
- Escolha do algoritmo mais adequado para o problema em questão;
- Implementação do algoritmo elaborado e resultados obtidos.

2 Fundamentação Teórica

2.1 Curvas de carga

Os perfis típicos de carga – ou curvas de carga – de um sistema elétrico são a representação gráfica da demanda de energia do referido sistema ao longo de um determinado período de tempo (usualmente, um dia). Estas ferramentas são úteis por permitirem às empresas do setor elétrico a representação dos padrões de consumo de suas unidades consumidoras.

As curvas de carga são uma das ferramentas de grande importância quando da análise e planejamento de sistemas elétricos, desde aqueles de pequeno porte, tais como unidades residenciais até grande porte, tais como o sistema de transmissão brasileiro. A utilização dos perfis típicos de carga permite a previsão da demanda, de modo a possibilitar a antecipação da geração necessária. Também têm grande utilidade para a análise de alterações na demanda (por exemplo, quando da instalação de novos equipamentos em unidades residenciais ou conexão de novas cargas), e como essas alterações afetam a viabilidade do sistema, técnica e economicamente.

As Figuras 1, 2 e 3 apresentam perfis típicos de carga para consumidores residenciais, comerciais e industriais, respectivamente:

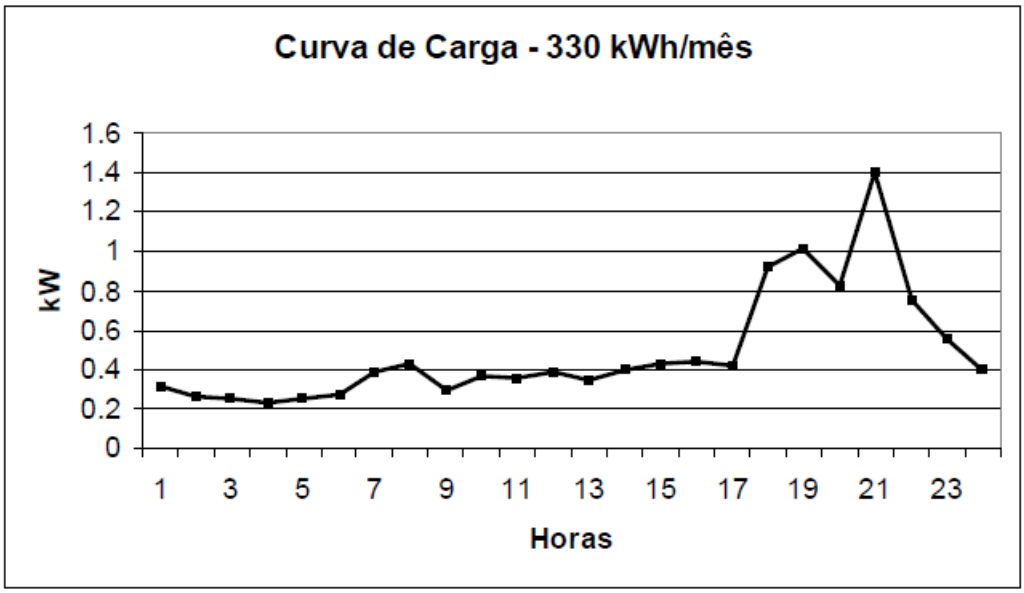


Figura 1 – Perfil típico de carga para um consumidor residencial

Fonte: Francisquini (2006)

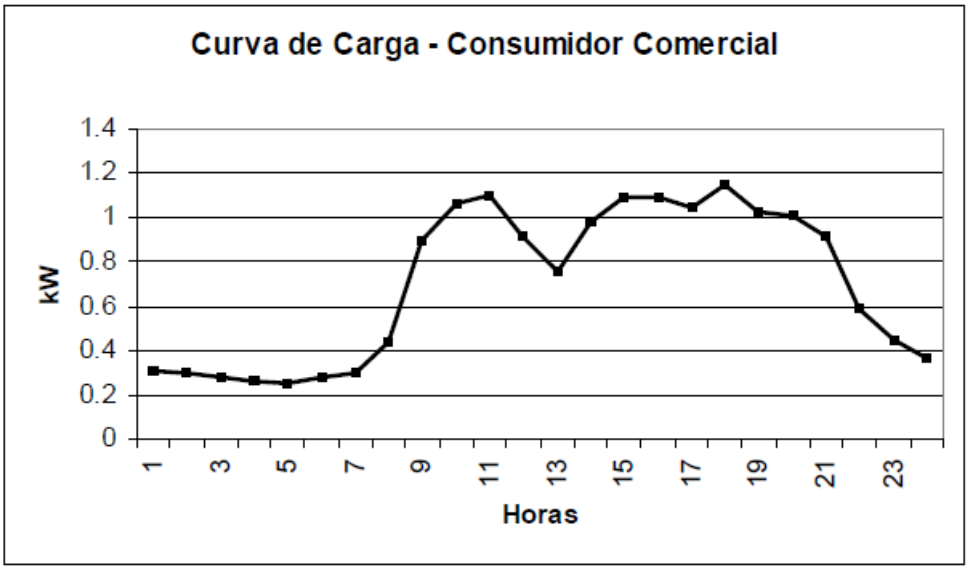


Figura 2 – Perfil típico de carga para um consumidor comercial

Fonte: Francisquini (2006)

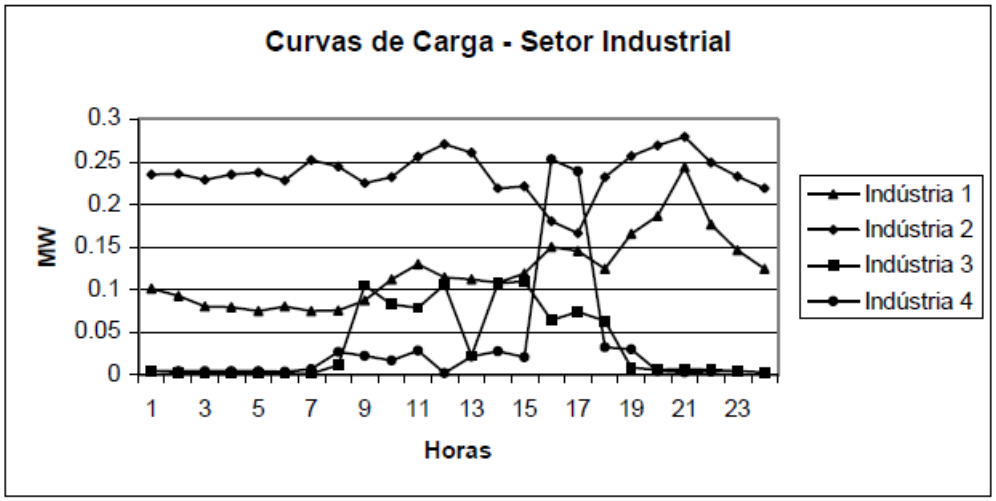


Figura 3 – Perfis típicos de carga para consumidores industriais

Fonte: Francisquini (2006)

É fácil perceber que uma série de fatores influencia no comportamento diário do perfil de carga de um sistema. Dentre os principais, pode-se citar como exemplos o dia da semana, a estação do ano e a ocorrência ou não de feriados ou outras ocasiões que tornem o dia em questão atípico (eventos esportivos, acontecimentos políticos, acidentes, etc). Todos esses fatores têm grande influência sobre o consumo de energia elétrica em um determinado dia. Portanto, é de grande utilidade que os dados resultantes sejam facilmente manipuláveis, de modo a permitir a utilização daqueles cujo fatores característicos sejam mais similares aos previstos no dia a ser analisado.

Como exemplo, Vivian (2015) traz um histórico do consumo mensal de uma uni-

dade consumidora, de modo que é nítida a influência do mês do ano (por consequência das alterações de temperatura, feriados, e demais fatores) nos padrões de consumo da unidade analisada, conforme apresentado na Figura 4:

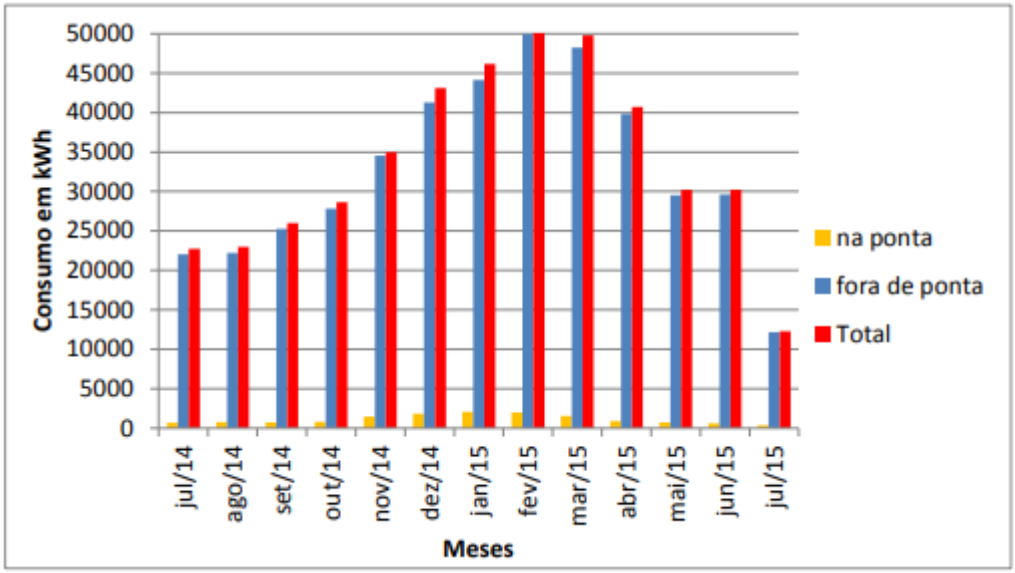


Figura 4 – Variação do consumo de energia ao longo do ano

Fonte: Vivian (2015)

Também em Francisquini (2006), podemos observar como o perfil de carga de uma determinada unidade consumidora varia conforme o dia da semana, assim como demonstrado nas Figuras 5, 6 e 7:

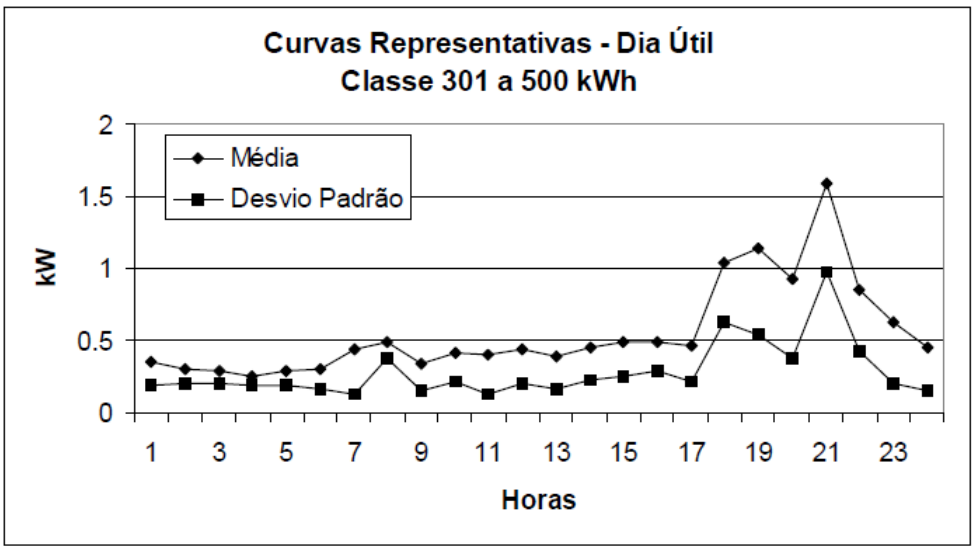


Figura 5 – Perfil típico de consumo para um dia útil

Fonte: Francisquini (2006)

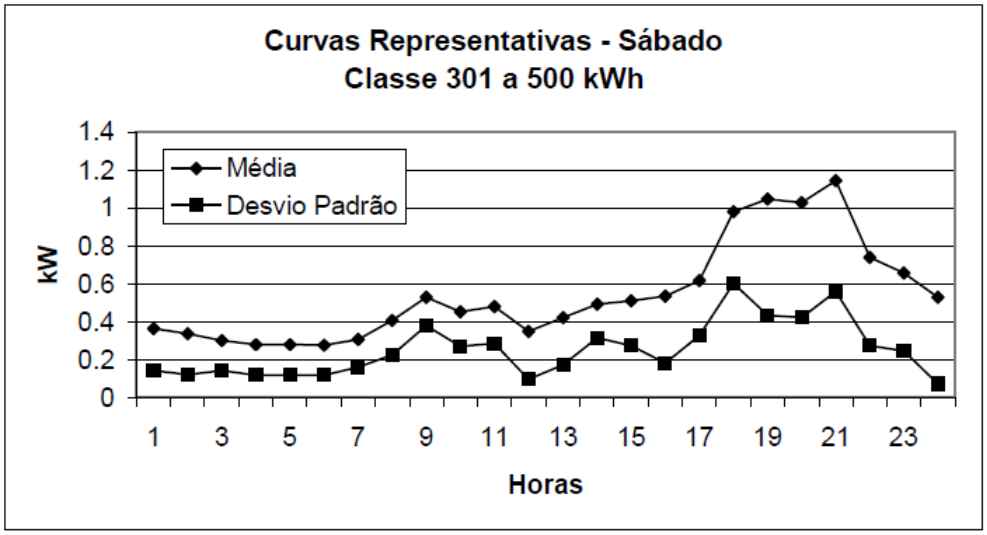


Figura 6 – Perfil típico de consumo para um sábado

Fonte: Francisquini (2006)

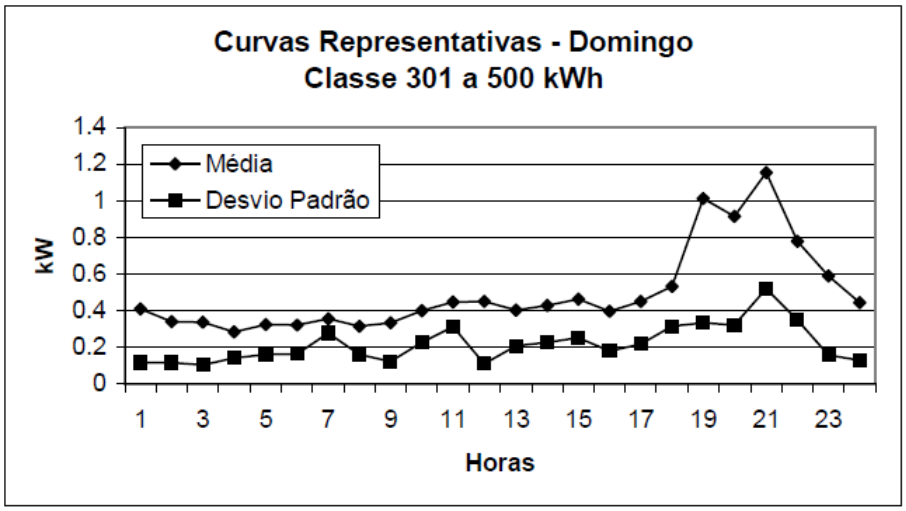


Figura 7 – Perfil típico de consumo para um domingo

Fonte: Francisquini (2006)

A Universidade de Brasília, objeto de estudo deste trabalho, possui características específicas que a diferenciam dos perfis clássicos de consumo. A presença de laboratórios, com equipamentos de alto consumo e picos específicos de funcionamento, a realização de aulas nos períodos matutino, vespertino e noturno, o número elevado de computadores e equipamentos de ar-condicionado são apenas alguns dos fatores que segregam o comportamento da UnB como consumidora, em relação aos demais integrantes do sistema.

Tendo em vista esse comportamento, é de importância que a análise de viabilidade do sistema elétrico da UnB parta, inicialmente, da construção de um perfil típico de carga para a universidade, que leve em consideração suas particularidades.

2.2 *Machine Learning*

Machine learning é o nome usado para definir o conjunto de modelos estatísticos e algoritmos que se baseiam em padrões de inferência para realizar tarefas específicas. Tais algoritmos podem ser divididos entre supervisionados e não supervisionados.

Os processos supervisionados consistem daqueles onde os padrões, ou "rótulos", são previamente definidos, e o algoritmo implementado tem a função de determinar qual padrão melhor corresponde aos valores de entrada apresentados na série de dados. Sendo assim, o algoritmo é treinado para reconhecer os padrões e identificar quando os dados de entrada se assemelham e eles. Por sua vez, os processos não supervisionados são aqueles onde os resultados esperados, ou padrões de saída, não são conhecidos. O algoritmo, então, tem como responsabilidade identificar os rótulos mais adequados, de acordo com sua entrada. Em suma, os processos não supervisionados tomam uma das variáveis como dado de entrada e tentam identificar padrões, enquanto os processos supervisionados tomam ambas as variáveis e mapeiam a função que as relacionam.

McLoughlin, Duffy e Colon (2015) oferecem uma explicação para as vantagens da adoção dos métodos em *machine learning*, em detrimento dos métodos estatísticos convencionais, os quais são amplamente empregados em sistemas comerciais. Segundo os autores, a aplicação dos métodos estatísticos para detecção de padrões – sendo o caso abordado no estudo em questão a construção de perfis de carga residenciais – esbarra na perda de informações consideradas importantes, como a relação temporal entre os valores, quando da análise de um conjunto consideravelmente grande de dados. Tal problema pode ser evitado agrupando os dados a partir de algoritmos de inteligência artificial, preservando a integridade dos valores obtidos pelos medidores e facilitando a alternância entre os diversos tipos de padrão a serem estudados.

Em outra análise semelhante, Francisquini (2006) realiza um levantamento das metodologias para a construção de curvas de carga em unidades de consumo e em transformadores de distribuição. O referencial bibliográfico levantado pelo autor abrange uma série de estudos que fazem uso tanto de métodos puramente estatísticos quanto de redes neurais não supervisionadas e processos de clusterização. Dentre os principais trabalhos resumidos pelo autor, são relevantes em nosso contexto as observações realizadas por Srinivasan, Liew e Chang (1994) e Senjyu, Higa e Uezato (1998).

Srinivasan, Liew e Chang (1994) apontam para o fato de que a utilização de algoritmos não supervisionados possui a vantagem de facilitar a atualização frequente dos dados conforme novas medições são realizadas. Em comparação, as metodologias estatísticas de cálculo normalmente fazem uso de dados aferidos anteriormente ao cálculo, podendo estes estarem defasados ou não representarem a realidade para uma determinada situação específica não abrangida pela base de dados.

Senjyu, Higa e Uezato (1998), por sua vez, ressaltam a vantagem da utilização de algoritmos de inteligência artificial, como a clusterização, através da ótica da previsibilidade dos dados. A utilização desses algoritmos facilita a criação de modelos preditivos de perfis de carga para o dia seguinte, dadas as condições determinadas.

Ponderadas as observações realizadas no referencial citado, optou-se por focar este trabalho exclusivamente nos métodos baseados em *machine learning*, quando do tratamento de dados.

Na busca pela compreensão dos diferentes métodos e pelo algoritmo mais adequado para o problema de construção das curvas de carga para a UnB, métodos não supervisionados, como os algoritmos de clusterização, e métodos supervisionados, como os algoritmos de *support vector machine*, foram considerados.

2.3 Clusterização

O processo de clusterização consiste da "divisão de um conjunto de dados em grupos de objetos similares". (ABBAS, 2008) Assim, pode-se entender a clusterização como um processo de quantização de vetores com objetivo de identificar padrões similares quando provido de um conjunto de dados e catalogar esses dados de modo a conferir significado aos mesmos, facilitando sua interpretação e utilização para outros fins quaisquer.

Os principais métodos de clusterização tendem a se encaixar dentre os não supervisionados: quando da aplicação dos mesmos, o algoritmo implementado não tem compreensão dos resultados que se espera dele, e se preocupa somente com o relacionamento dos dados de entrada.

Dentre os diversos métodos de clusterização existentes, quatro foram selecionados para a revisão neste trabalho: k-médias, clusterização hierárquica espacial, mapas auto-organizáveis de Kohonen (SOM) e maximização de expectativas. A escolha desses quatro algoritmos segue o padrão do trabalho de Abbas (2008), que propõe uma comparação entre estes diferentes métodos de clusterização. Este foi utilizado como principal referência quando da comparação dos métodos, por ser um dos únicos trabalhos encontrados a oferecer uma noção de padrão de aplicabilidade para cada um dos algoritmos. Abbas conclui como cada método tende a se portar quando alimentado por bancos de dados de características específicas, servindo como uma referência que pode ser considerada, após a análise do cenário, para a escolha do método mais adequado.

2.3.1 K-médias

O método k-médias é um algoritmo de clusterização baseado na noção de agrupamento de dados partindo de valores iniciais arbitrários. Foi introduzido inicialmente nos

anos 70 por Hartigan, e é uma das técnicas de clusterização mais difundidas atualmente, devido à sua simplicidade matemática, fácil implementação e flexibilidade. (SANGALLI et al., 2010)

O algoritmo k-médias consiste na classificação de um conjunto de dados, rotulando os mesmos como sendo pertencentes a um entre k grupos (*clusters*) que dividem o conjunto. A afiliação de cada ponto ao seu respectivo grupo é determinada pela distância entre o centro do *cluster* e o ponto, a qual deve ser mínima. (ABBAS, 2008) Em outras palavras, o ponto pertencerá ao *cluster* cujo centro for o mais próximo do ponto sendo analisado. O processo de determinação dos pontos pertencentes a cada grupo é iterativo, se repetindo até que o critério de parada pré-estabelecido seja alcançado. Usualmente, o critério de parada do algoritmo k-médias é tal que o algoritmo é interrompido quando, após a realização da última iteração, nenhum ponto muda de *cluster*.

A implementação do algoritmo k-médias parte, inicialmente, da determinação do número de *clusters*, definido pela variável k. O valor de k é um parâmetro de entrada do algoritmo, sendo definido pelo usuário. A determinação do parâmetro k é uma das maiores dificuldades em algumas aplicações desse algoritmo, uma vez que, em certas ocasiões, não se possui nenhum conhecimento prévio sobre o número de conjuntos de dados que devem ser definidos como similares. (CARVALHO; NETO, 2011)

O procedimento para esse algoritmo é sintetizado por Abbas (2008) como:

1. O algoritmo k-médias toma um conjunto de dados S e um número inteiro de grupos k na sua entrada, e retorna k partições do conjunto S, quais sejam S_1, $S_2 \ldots$ S_k, conforme a Figura 8;

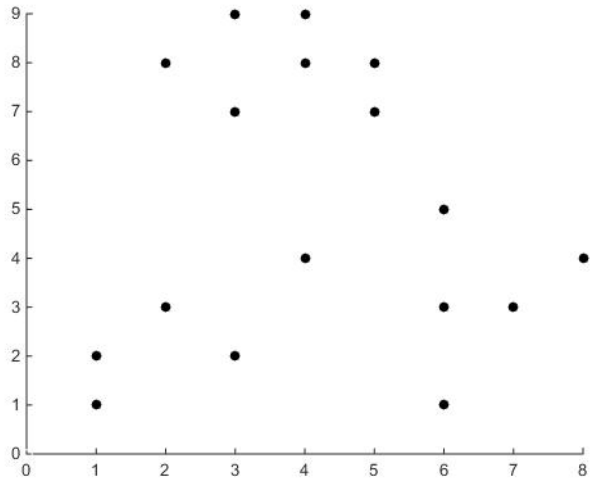


Figura 8 – K-médias - dados apresentados ao programa

2. O programa então determina, de forma aleatória, padronizada ou por entrada do usuário, o centroide inicial de cada *cluster*, denotado como x^{-1}, conforme a Figura 9;

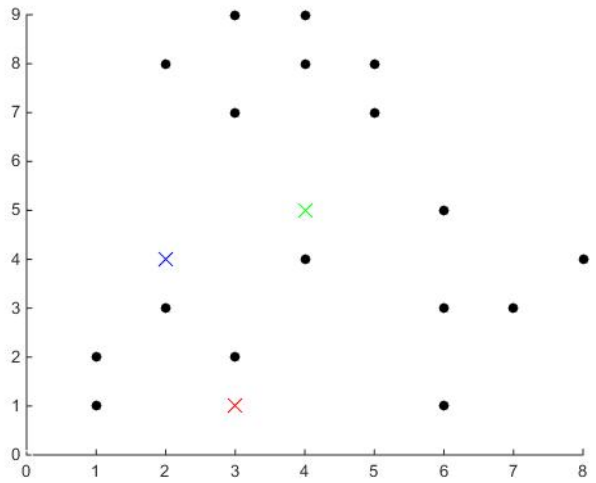


Figura 9 – K-médias - determinação dos centroides iniciais

3. Cada ponto é afiliado ao *cluster* de centroide mais próximo. A afiliação é feita com base no cálculo da distância entre cada ponto e os centroides disponíveis, $d(x_r^i, x^{-1})$, em que x_r^i é o r-ésimo elemento do cluster i. O ponto x^i pertencerá ao *cluster* S_r cujo quadrado da distância ao seu centroide, $d(x^i, x_r^{-1})^2$ seja o menor, conforme a Figura 10;

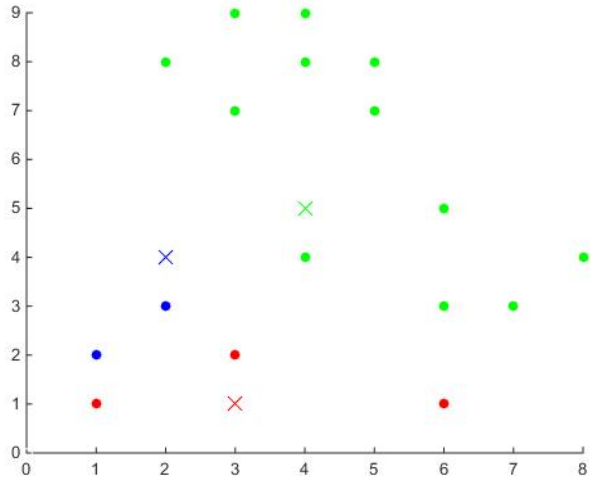


Figura 10 – K-médias - afiliação dos pontos ao *cluster* mais próximo

4. O novo centroide de cada *cluster* é determinado. Para tal, é necessária a utilização de alguma medida estatística de tendência central, nesse exemplo, a mediana, devido à

sua menor sensibilidade a anomalias e dados discrepantes (a variação do algoritmo k-médias utilizando a mediana como medida de tendência central também é conhecida por k-medianas). Assim sendo, o novo centroide do *cluster* será a mediana dos itens associados ao grupo, como indicado na Figura 11;

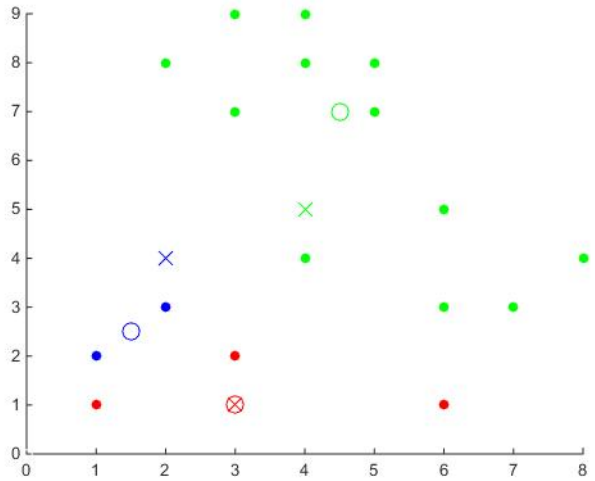


Figura 11 – K-médias - determinação dos novos centroides

5. Definidos os novos centroides, é possível que alguns itens mudem de grupo, de acordo com o critério estabelecido no passo 3. A modificação da posição dos centroides altera a distância entre o centroide e cada ponto. Na ocorrência dessa situação, é necessário repetir os passos 3 e 4;

6. Uma vez atingido o critério de parada, seja ele a não alteração da afiliação de nenhum ponto, seja um número de iterações pré-determinado, o algoritmo é encerrado, e a medida de tendência central se torna o ponto representativo de cada *cluster*.

Além da determinação do parâmetro k, outra dificuldade encontrada na aplicação dessa técnica advém da determinação dos centroides iniciais. Um mau posicionamento dos centroides pode acarretar em resultados insatisfatórios. A necessidade de conhecimento das tendências decorrentes do conjunto de dados apresentado ao programa, de modo a determinar um conjunto de centroides iniciais não prejudiciais ao tratamento estatístico, torna o algoritmo de difícil execução, quando da análise de um conjunto de dados suficientemente grande. (SHAMEEM; FERDOUS, 2009)

Sangalli et al. (2010) define uma abordagem, utilizando o método k-médias, para clusterização de séries temporais e curvas desalinhadas. Tal abordagem passa pela análise da amplitude e fase das curvas, através da utilização de um índice de similaridade, e classificação dessas conforme as mesmas grandezas.

2.3.2 Clusterização hierárquica espacial

O algoritmo k-médias, observado na seção anterior, é um exemplo de método de clusterização por particionamento. Tais métodos exigem a determinação de um número de *clusters* k previamente à segregação dos dados.

Em constraste, a clusterização hierárquica não exige a determinação do número de *clusters*, trabalhando com a agregação ou divisão de grupos conforme a distância entre seus pontos. (ABBAS, 2008)

A clusterização hierárquica parte, inicialmente, da definição de uma "função-custo". Essa função representa a distância entre dois *clusters*, podendo ser estabelecida de diversas maneiras, seja através da distância entre o centro de dois diferentes grupos, da maior ou menor distância entre dois pontos de cada grupo, ou qualquer outra medida representativa de sua separação.

Em um primeiro momento, a clusterização hierárquica trata cada ponto (ou dado de entrada) como sendo seu próprio *cluster*. A partir de então, utiliza-se a função-custo para determinar qual agregação entre dois *clusters* diferentes é mais "barata" - ou seja, qual par de *clusters* se encontra mais próximo, conforme o critério estabelecido pela função-custo. Reconhecidos, os dois grupos identificados pela função são aglomerados.

O algoritmo segue, então, com o mesmo procedimento, até que todos os pontos estejam abrangidos por um só *cluster*. Sempre que identificados os dois grupos mais próximos, estes são unificados em um só, independente de ser este um grupo de um só ponto ou mais de um.

As Figuras 12 a 17 representam o funcionamento do algoritmo de clusterização hierárquica. Para esse exemplo, foi escolhida como função-custo a menor distância entre um ponto de determinado grupo para qualquer ponto em outro grupo. Assim sendo, os grupos são sempre comparados através de seus pontos mais próximos.

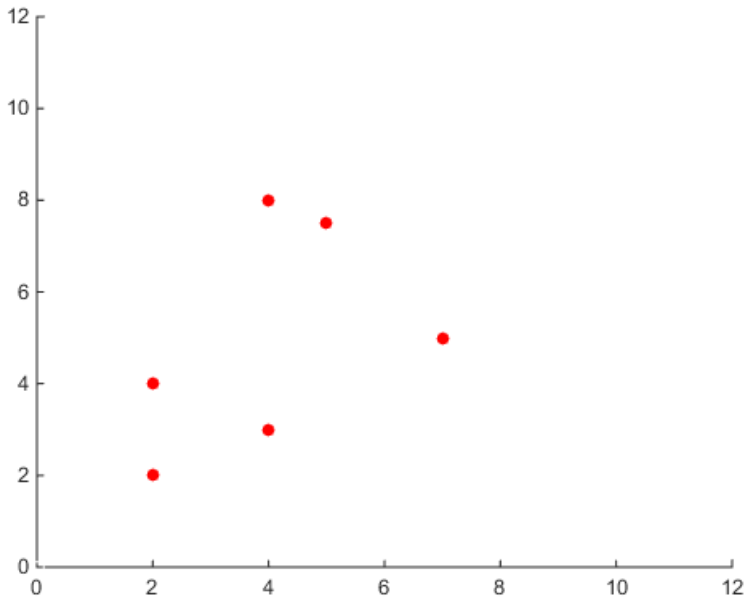


Figura 12 – Clusterização hierárquica - configuração inicial dos dados

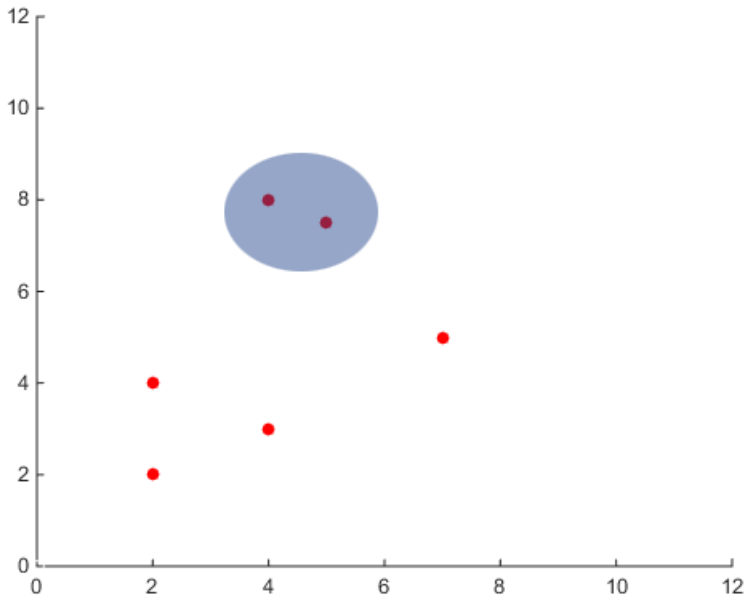


Figura 13 – Clusterização hierárquica - primeira iteração

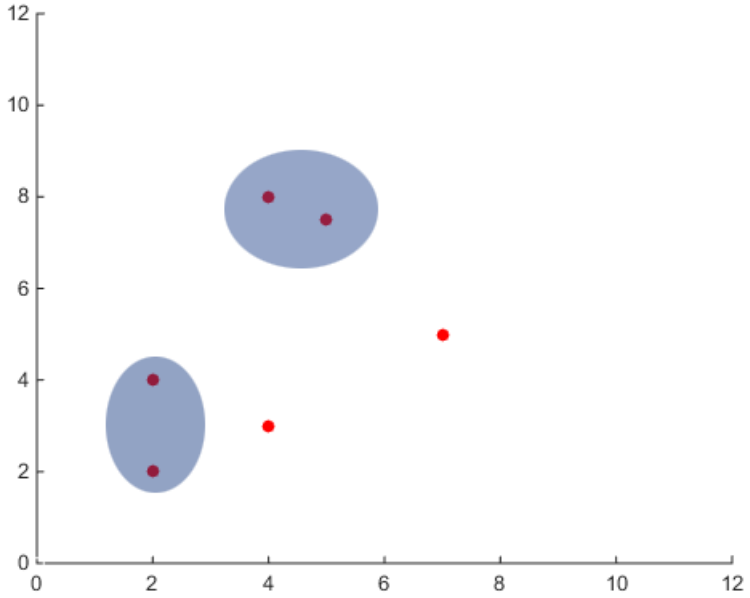


Figura 14 – Clusterização hierárquica - segunda iteração

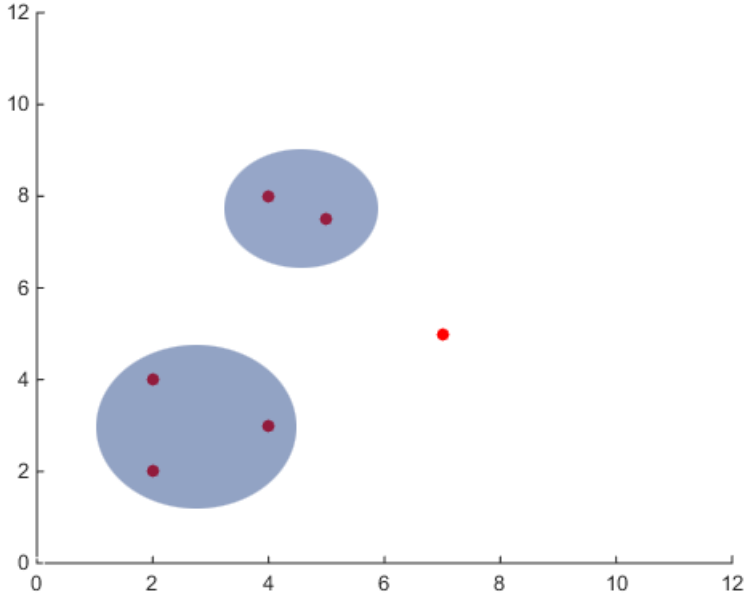


Figura 15 – Clusterização hierárquica - terceira iteração

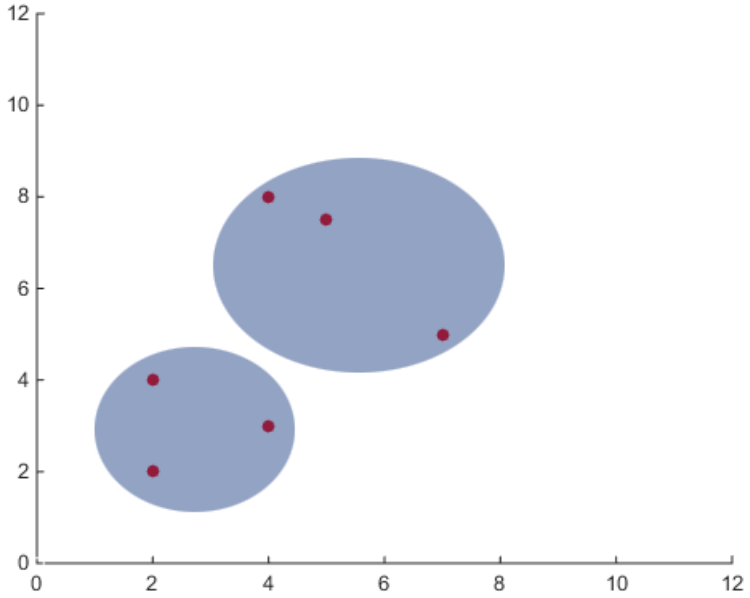


Figura 16 – Clusterização hierárquica - quarta iteração

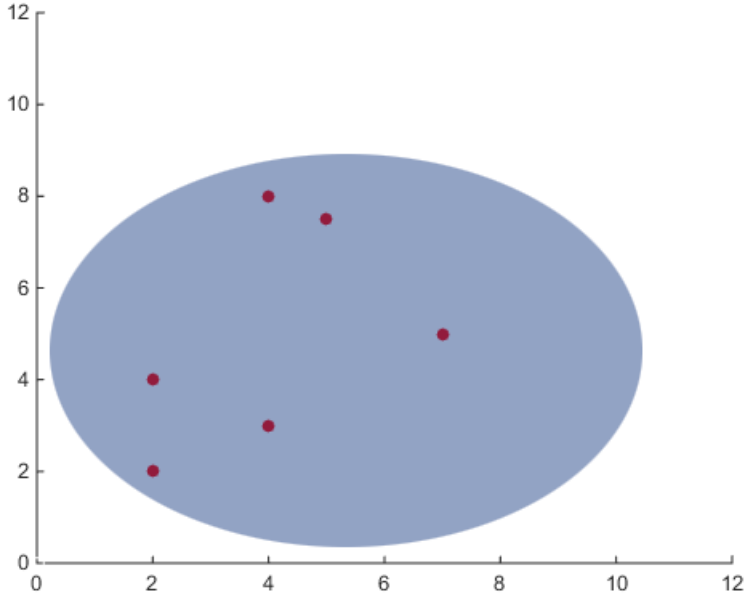


Figura 17 – Clusterização hierárquica - quinta iteração

Após a agregação de todos os pontos trabalhados, o resultado do algoritmo de clusterização hierárquica é um dendograma - uma forma de diagrama representativo da relação entre os pontos. O dendograma referente ao exemplo apresentado acima está demonstrado na Figura 18:

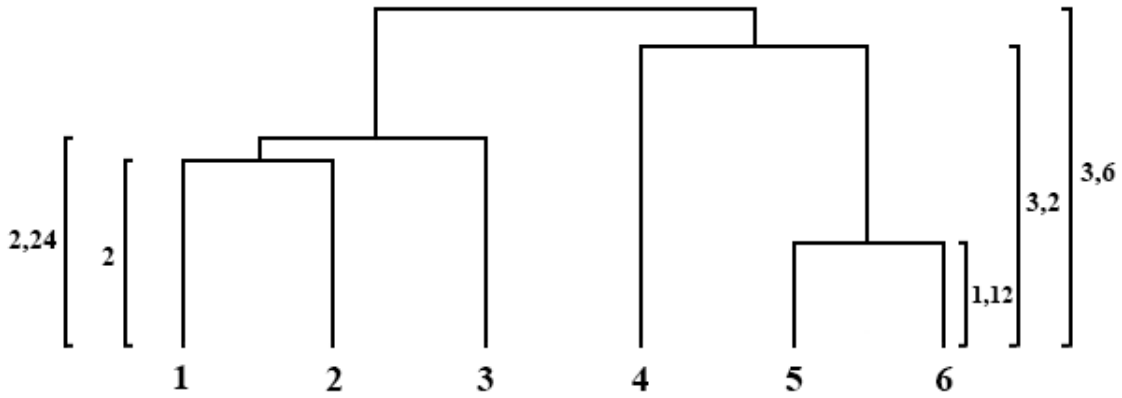


Figura 18 – Clusterização hierárquica - dendograma

A altura de cada linha que une dois objetos é diretamente proporcional ao valor da função-custo determinada na junção dos dois *clusters* em questão.

O dendograma é um instrumento gráfico que permite auxiliar na determinação do número de *clusters* que melhor representa o grupo de dados de entrada do programa. Entretanto, é importante ressaltar que, por si só, o dendograma não indica um número exato de *clusters* ideal. É necessária a análise dos resultados obtidos de modo a garantir que o processo de clusterização ocorra conforme desejado pelo usuário.

A escolha do número de *clusters* pode se dar através da análise do dendograma, de maneira arbitrária, ou, em uma abordagem mais técnica, através da determinação de uma distância máxima aceitável entre dois pontos para que estes sejam agrupados conjuntamente. Neste exemplo, foi determinado que a distância máxima aceita entre dois pontos pertencentes ao mesmo *cluster* seria de 3 unidades. Analisando o dendograma, é possível concluir que haverão, então, 3 *clusters*, conforme apresentado na figura 15.

Ao final de execução desse procedimento, cada *cluster* pode ser representado através da medida estatística considerada mais apropriada para a aplicação em questão, seja ela a média dos pontos, mediana ou qualquer outra escolhida previamente.

2.3.3 Mapas auto-organizáveis de Kohonen

O mapa auto-organizável de Kohonen - mais conhecidos por seu acrônimo em inglês, SOM (*self-organizing maps*) - é algoritmo em redes neurais que faz uso do aprendizado de máquina não supervisionado.

Os algoritmos SOM fazem uso do princípio de competição entre neurônios de modo a encontrar o padrão mais próximo dos dados de entrada e atualizar os valores de suas componentes, de forma a tornar a rede neural mais competitiva.

As redes neurais artificiais consistem de um conjunto de técnicas de programação cujo intuito é replicar o funcionamento de um neurônio biológico de modo a realizar operações lógicas complexas. Os neurônios artificiais são modelados em função do neurônio biológico, conforme apresentado nas Figuras 19 e 20:

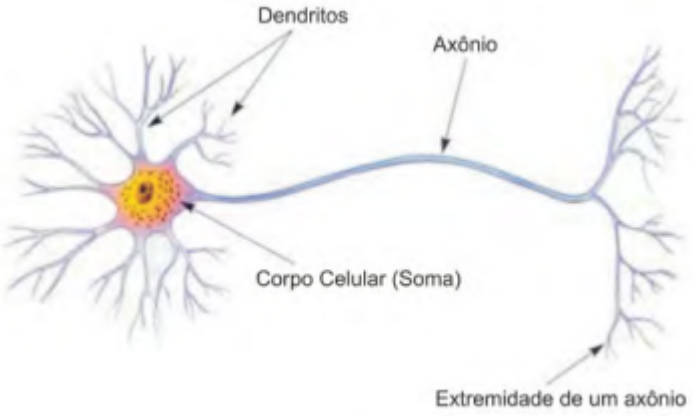


Figura 19 – Estrutura de um neurônio biológico

Fonte: Campos (2010)

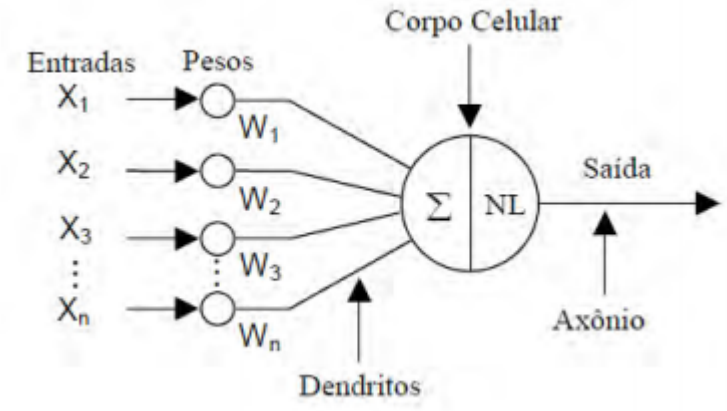


Figura 20 – Estrutura de um neurônio artificial

Fonte: Campos (2010)

Na representação anterior, o conjunto de valores denotados por x se referem aos dados de entrada do neurônio, enquanto w representa os pesos sinápticos aplicados. O produto entre x e w deve ser maior do que um valor pré-determinado w_0. (CAMPOS, 2010)

O método SOM consiste em uma rede de neurônios que competem entre si, e tem seus pesos atualizados constantemente, de modo a tornar seus fatores de saída cada vez mais adequados aos padrões de entrada. A seguinte explicação para o funcionamento da rede SOM toma por base o estudo realizado por Natita, Wiboonsak e Dusadee (2016).

Sejam x_1, x_i... x_n as entradas da rede SOM, e y_1, y_i... y_n as saídas. Sejam w_i os vetores de peso associados a cada neurônio, com seus valores iniciais determinados aleatoriamente, tem-se que cada valor de entrada é conectado a todos os neurônios, de modo a identificar aquele mais semelhante a si. Assim, a rede SOM pode ser representada conforme o padrão da Figura 21:

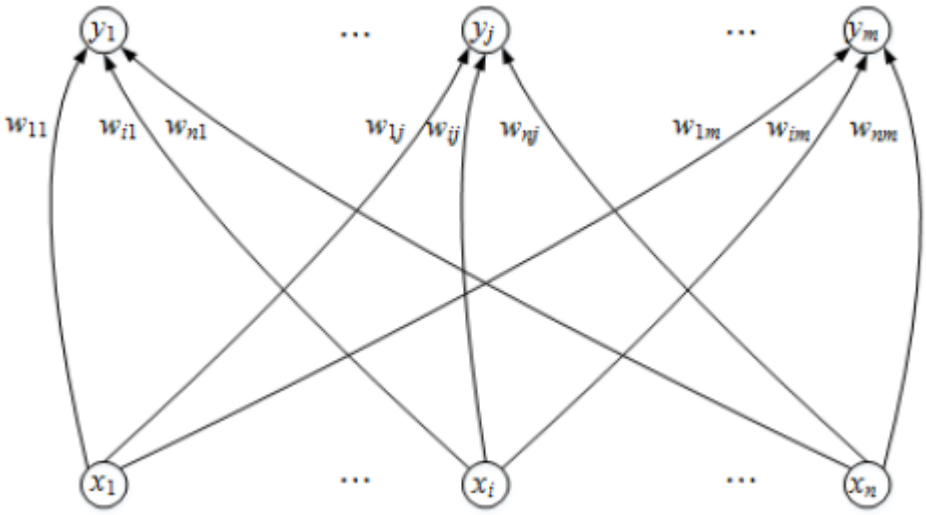


Figura 21 – Arquitetura padrão de uma rede SOM

Fonte: Natita, Wiboonsak e Dusadee (2016)

Para uma entrada x, o neurônio "vencedor" será aquele cujo peso w seja mais próximo do valor de x. Assim, o melhor neurônio - denotado por BMU, *best matching unit* - será:

$$BMU = min||x(t) - w_i(t)|| \tag{2.1}$$

Em que t é o número da iteração sendo realizada.

Após a detecção do neurônio vencedor, cada neurônio tem seu peso atualizado de modo a tornar a rede mais competitiva, conforme a função:

$$w_i(t+1) = w_i(t) + \alpha(t) * h_{ij}^c(t) * (x(t) - w_i(t)) \tag{2.2}$$

Em que $\alpha(t)$ é a taxa de aprendizado da rede neural e $h_{ij}^c(t)$ é a função vizinhança, ambas funções pré-determinadas pelo programador.

Durante a fase de aprendizado - etapa onde os parâmetros dos neurônios são ajustados e os resultados servem apenas como indicativo da qualidade parcial da rede - os neurônios se especializam em detectar conjuntos de padrões na entrada do programa, além de se posicionarem dentro da rede de modo a permitir que padrões semelhantes sejam inseridos em neurônios mais próximos. (ALMEIDA; PESSANHA; CALOBA, S.D.)

A rede SOM também pode possuir mais de uma dimensão, apesentando arquitetura semelhante à da Figura 22:

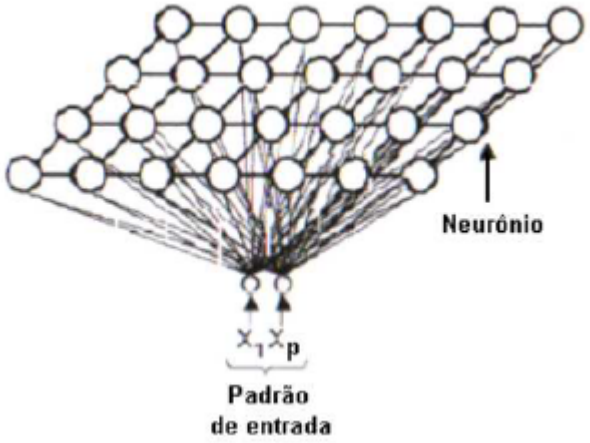


Figura 22 – Arquitetura padrão de uma rede SOM bidimensional

Fonte: Almeida, Pessanha e Caloba (S.D.)

Stefanovic e Kurasova (2014) trazem uma definição do princípio de funcionamento de uma rede neural bidimensional: nela, cada entrada consiste de um vetor $X_p = x_{p1}, x_{p2}...x_{pn}$, e o conjunto de dados pode ser representado por $X = X_1, X_2...X_N$.

Os pesos dos neurônios são dados por w_{ij}, e todos eles são comparados com cada vetor de entrada X_p, sendo o vetor (neurônio) W_c mais próximo do vetor de entrada o vencedor. Tal modelo é mais tangível para curvas de carga, devido à melhor representação dos dados de entrada como forma vetorial, mantendo a correlação e sequência dos mesmos.

A grande dificuldade na aplicação da rede SOM está na determinação da taxa de aprendizado e da função vizinhança mais adequadas para o problema em questão. Inúmeros estudos para determinação de funções que entregam bons resultados existem nas mais variadas áreas de aplicação dessa metodologia, como em Natita, Wiboonsak e Dusadee (2016), que buscam determinar fatores adequados para a análise de padrões de umidade no sul da Tailândia.

Como alguns exemplos de funções para a taxa de aprendizado, podemos utilizar:

$$\alpha(t) = \frac{1}{t} \tag{2.3}$$

$$\alpha(t) = (1 - \frac{t}{T}) \tag{2.4}$$

Em que T representa o número total de iterações a serem realizadas pelo algoritmo no seu processo de aprendizagem, quando este é definido como critério de parada.

Para a função de vizinhança, é possível utilizar, dentre outras, as expressões:

$$h_{ij}^c = \begin{cases} \alpha(t), (i,j) \in N_c \\ 0, (i,j) \notin N_c \end{cases} \tag{2.5}$$

$$h_{ij}^c = \alpha(t) * e^{(\frac{-||R_c - R_{ij}||^2}{2(\eta_{ij}^c(t))^2})} \tag{2.6}$$

Onde $\eta_{ij}^c(t)$ é uma função da proximidade entre dois neurônios.

McLoughlin, Duffy e Colon (2015) apresentam, em seu trabalho, o algoritmo SOM como sendo o mais adequado para segmentação de dados de curvas de carga, quando comparado a outros dois (k-médias e k-medoides).

2.3.4 Maximização de expectativas

O último algoritmo de clusterização abordado por Abbas (2008) é o de maximização de expectativas. Esse é um algoritmo baseado em distâncias que possui grande aceitação no campo estatístico por sua capacidade de lidar bem com dados de entrada "ruidosos" e convergir rapidamente quando de uma boa inicialização, dentre outros fatores.

Abbas (2008) define o processo do algoritmo de maximização de expectativas em duas etapas: na etapa inicial, conhecida como expectativa, o programa assume que o *dataset* pode ser modelado através da combinação linear de múltiplas distribuições normais, e define uma variável que indica a qualidade dessas distribuições em classificar dados semelhantes no mesmo grupo, conhecida como verosimilhança logarítmica (*log-likelihood*). Na segunda etapa, entitulada maximização, o algoritmo ajusta os parâmetros da distribuição de modo a maximizar o valor da função de verosimilhança logarítmica.

Em outras palavras, o algoritmo de maximização de expectativas infere valores iniciais para a distribuição de probabilidades de um determinado ponto estar classificado em cada *cluster*, e atribui tal ponto ao *cluster* cuja probabilidade seja maior. Feito isso, a qualidade da distribuição aplicada é avaliada através das distâncias, considerando alguma

medida estatística como referência (como o desvio padrão ou a variância dos pontos de cada *cluster*), e os parâmetros da distribuição de probabilidade são rearranjados de modo a minimizar essa medida - ou maximizar a verosimilhança logarítmica.

2.3.5 Aplicabilidade dos algoritmos de clusterização

Abbas (2008) apresenta uma comparação dos quatro algoritmos de clusterização aqui explicados, avaliando sua performance quando da alteração de diferentes fatores. Tal análise é interessante por fornecer indicações das vantagens da aplicação de cada tipo de algoritmo. É fato que a resposta do algoritmo utilizado será dependente da situação em que este se aplica, com inúmeras variáveis envolvidas no processo que não permitem a determinação de um método que seja superior. Entretanto, a observação de padrões no trabalho de Abbas torna a seleção das metodologias a serem testadas uma tarefa menos laborosa.

Os critérios variados por Abbas (2008) para comparação do desempenho dos algoritmos foram:

- Tamanho do *dataset*;
- Número de *clusters*;
- Tipo de *dataset*;
- Tipo de software utilizado para implementação dos algoritmos.

Variando esses parâmetros, Abbas (2008) atingiu uma série de conclusões sobre o comportamento dos quatro métodos de clusterização analisados, dentre as quais pode-se destacar:

- Os métodos k-médias e maximização de expectativas possuem maior qualidade quando tratando de *datasets* grandes, enquanto os algoritmos de clusterização hierárquica e SOM são usualmente melhores para *datasets* pequenos;
- Os algoritmos de clusterização hierárquica e SOM apresentam melhores resultados quando trabalhando com dados aleatórios, enquanto k-médias e maximização de expectativas funcionam melhor com dados ideais;
- Os algoritmos k-médias e maximização de expectativas são muito sensíveis a ruidos e anomalias no conjunto de dados, tendo sua performance altamente comprometida pela ausência de um tratamento de dados qualificado na fase prévia à implementação. Já o método SOM é o mais resiliente a anomalias e ruídos.

2.4 *Support Vector Machines*

O modelo de aprendizado conhecido como *Support vector machine* - ou SVM - é um algoritmo que parte da definição de vetores de suporte para a classificação ou regressão de dados. A primeira versão do algoritmo SVM foi elaborada por Vapnik e Lerner (1963), A publicação mais completa do seu modelo de funcionamento, incluindo o uso de kernels, se encontra em Cortes e Vapnik (1995).

O termo SVM abrange ambas as aplicações da técnica de utilização dos vetores de suporte: problemas de classificação e regressão. Entretanto, como o algoritmo original foi elaborado para problemas de classificação, e essa permanece sendo a sua aplicabilidade mais usual, é comum o uso do termo SVM para denotar o algoritmo utilizado para classificação.

O problema que o algoritmo SVM para classificação busca solucionar é o seguinte: é tomado um conjunto de dados como entrada do algoritmo. Tais dados são rotulados, indicando sua afiliação a uma de duas classes, conforme indicado na Figura 23.

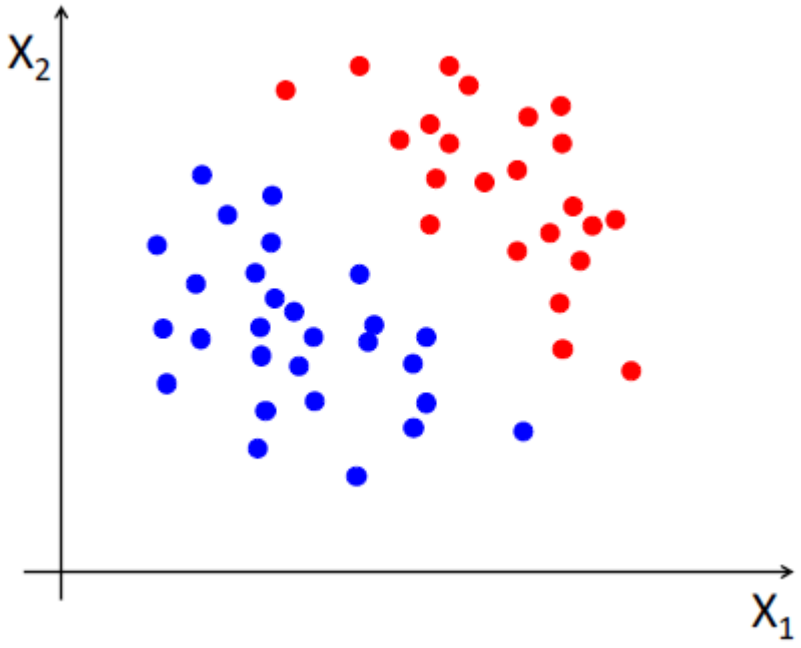


Figura 23 – Dados de entrada do SVM

Fonte: Filgueiras (2014)

O algoritmo SVM busca a criação de um hiperplano que separe os dados da melhor maneira possível, de modo que, quando da apresentação de novos dados não rotulados ao algoritmo, esse seja capaz de indicar a qual classe tal dado deve pertencer, apenas observando em que lado do plano se encontra. Como pode-se observar na Figura 24, podem existir diversos planos capazes de separar os dados de maneira adequada conforme o padrão de entrada. A questão, então, está em como otimizar esse plano de separação.

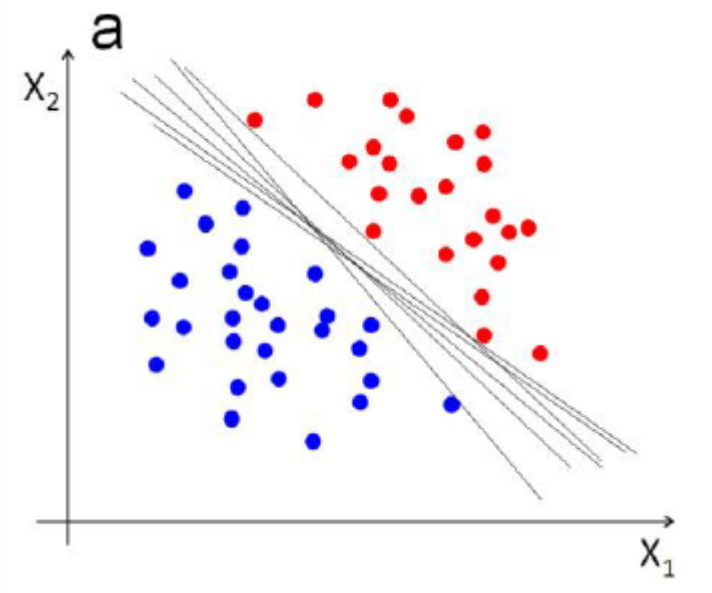


Figura 24 – Diferentes planos de separação dos dados

Fonte: Filgueiras (2014)

Podemos representar cada um dos dados de entrada como um vetor $(\mathbf{x_i}, y_i)$, em que $\mathbf{x_i}$ são as coordenadas do ponto e y_i é um indicativo da classe à qual ele pertence: +1, -1.

O hiperplano de separação entre as classes pode, então, ser escrito através da equação $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, em que $\mathbf{w}$ é um vetor de pesos.

O próximo passo da solução desse problema passa por duas definições importantes. Chama-se "margem" a distância entre o hiperplano de separação e os pontos da amostra que se encontram mais próximos e "vetores de suporte" os pontos que estão sobre a margem, conforme observado na Figura 25.

O objetivo do SVM é definir o hiperplano que garanta a maior separação entre as amostras limitantes, ou vetores de suportes: ou seja, a maior margem.

A solução do SVM, então, pode ser apresentada como:

minimizar: $\frac{1}{2}||w||^2$

sujeito a: $\begin{cases} \mathbf{w} \cdot \mathbf{x} + b \geq +1 & \forall \quad y \in \{+1\} \\ \mathbf{w} \cdot \mathbf{x} + b \geq -1 & \forall \quad y \in \{-1\} \end{cases}$

O SVM leva em consideração somente a separação correta dos vetores de suporte, partindo da inferência de que, uma vez separados os vetores mais próximos - e, portanto, mais difíceis -, os demais estarão também separados corretamente. Entretanto, ao trabalhar com *datasets* reais, esse nem sempre é o caso.

Para contornar a situação em que não é possível separar linearmente todos os

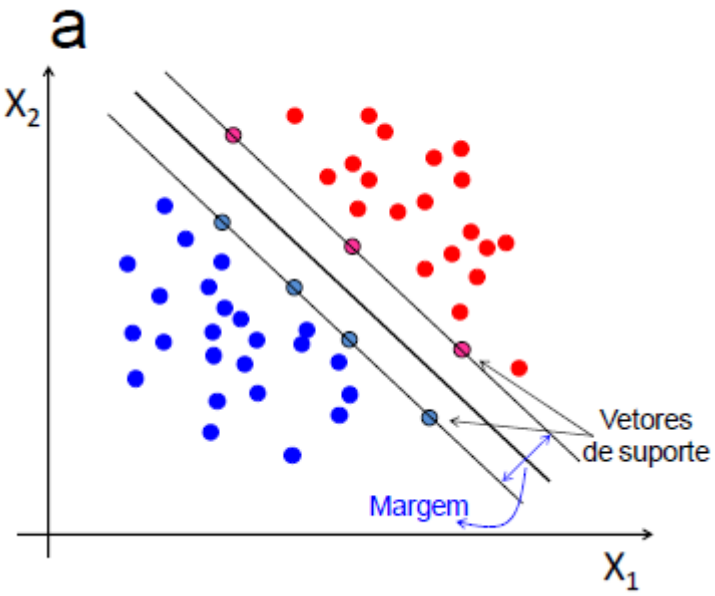


Figura 25 – Hiperplano ótimo de separação

Fonte: Filgueiras (2014)

dados de maneira correta usando um hiperplano, Cortes e Vapnik (1995) introduziram o conceito de margem suave. Nessa consideração, é inserido nas equações um fator ξ a fim de computar o erro resultante da classificação incorreta de alguns pontos. Esse erro é diretamente proporcional à distância da amostra para a margem referente à sua classe correta, conforme indicado na Figura 26. (FILGUEIRAS, 2014)

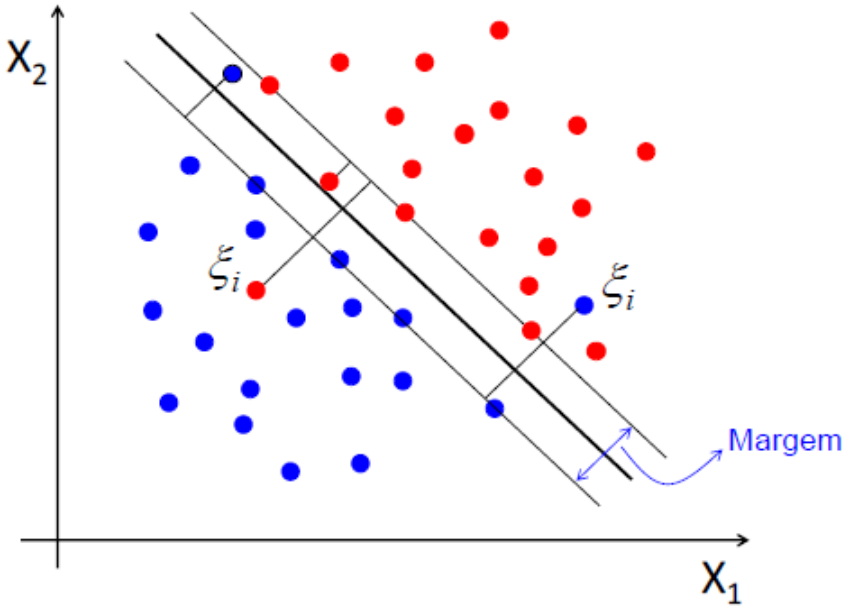


Figura 26 – Aplicação de margens suaves

Fonte: Filgueiras (2014)

Além disso, uma constante C é inserida para ponderar o erro, de modo a garantir que a presença de anomalias na amostra não distorça os resultados do algoritmo. A solução para o problema de classificação com margem suave, então, pode ser escrita como:

$$\text{minimizar: } \tfrac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{sujeito a: } \begin{cases} \mathbf{w}\cdot\mathbf{x} + b \geq +1 - \xi_i \\ \mathbf{w}\cdot\mathbf{x} + b \geq -1 + \xi_i \\ \xi_i \geq 0 \end{cases}$$

Alguns *datasets*, entretanto, não são linearmente separáveis. O uso de um hiperplano linear, mesmo com a consideração das margens suaves, não permite que o comportamento da amostra de dados seja devidamente representado. Para resolver esse problema, sao introduzidas funções kernel.

A aplicação de uma função kernel não linear $\phi(\mathbf{x}_1, \mathbf{x_2})$ resulta no mapeamento dos dados em uma dimensão superior à original, conhecida como espaço de características, em que a separação por um plano é possível. Feita essa separação, o hiperplano é transportado de volta ao espaço original de dimensão inferior.

A Figura 27 demonstra a utilização de um kernel RBF (*radial basis function*) na separação de um conjunto de dados. Já a Figura 28 apresenta os tipos de kernel mais utilizados em SVM, e suas descrições matemáticas.

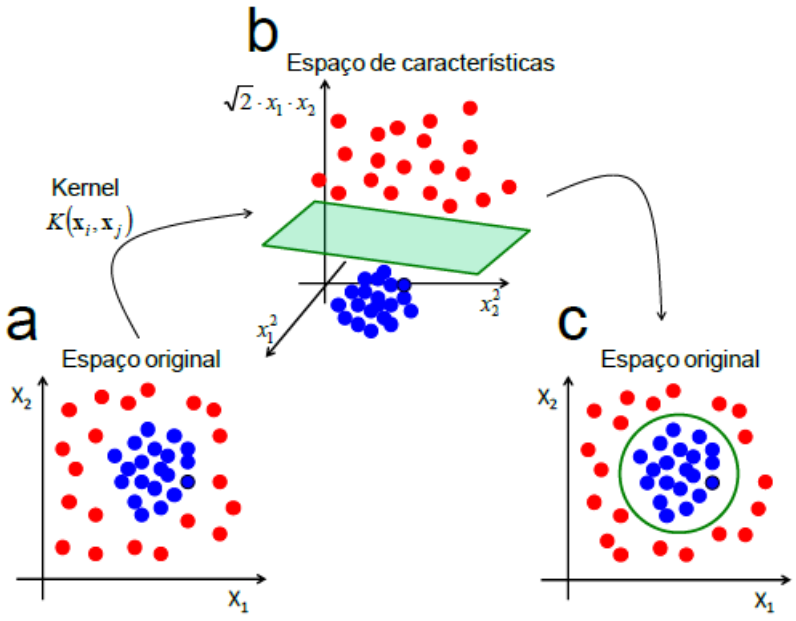


Figura 27 – Aplicação de kernel RBF para separação de dados

Fonte: Filgueiras (2014)

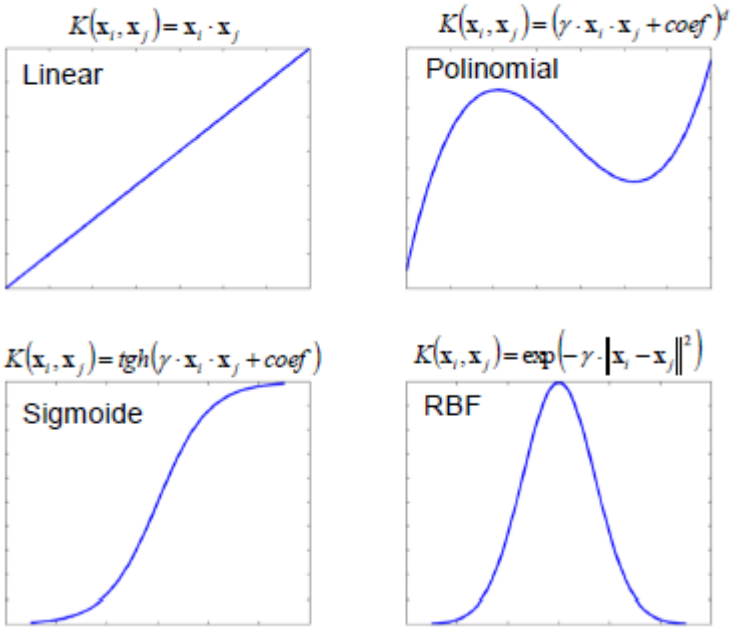


Figura 28 – Exemplos de diferentes funções kernel

Fonte: Filgueiras (2014)

2.4.1 Support Vector Regression

O modelo SVM foi adaptado por Drucker et al. (1997) para resolver problemas de regressão. Tal algoritmo ficou conhecido como *support vector regression*, ou SVR.

Na aplicação do algoritmo SVR, um número d é adicionado e subtraído, simultaneamente, da valor de y_i, que é o valor de interesse em nossa análise, formando, assim, duas classes distintas, conforme apresenta a Figura 29. Assim, o problema de regressão se torna, essencialmente, um problema de classificação binária, permitindo a aplicação do SVM.

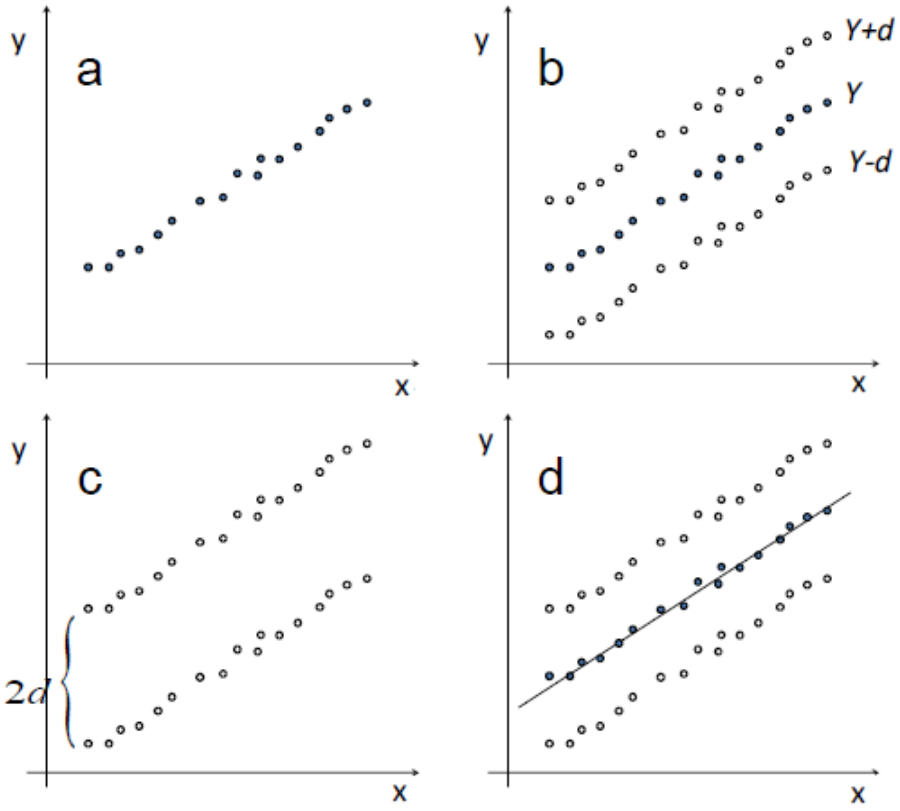


Figura 29 – Transformação do problema de regressão em classificação binária

Fonte: Filgueiras (2014)

Os erros associados, são denotados por ξ_i e ξ_i^*, simbolizando os erros das classes com a adição e subtração de d, respectivamente. Além disso, é definida uma tolerância ε, que delimita a região em volta do hiperplano de separação onde os erros de classificação são considerados aceitáveis. A Figura 30 apresenta a configuração do problema.

A solução do algoritmo SVR pode ser escrita como:

minimizar: $\frac{1}{2}||w||^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$

$$\text{sujeito a: } \begin{cases} y_i - \mathbf{w} \cdot \phi(\mathbf{x}_i) - b \leq \varepsilon + \xi_i \\ \mathbf{w} \cdot \phi(\mathbf{x}_i) + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$

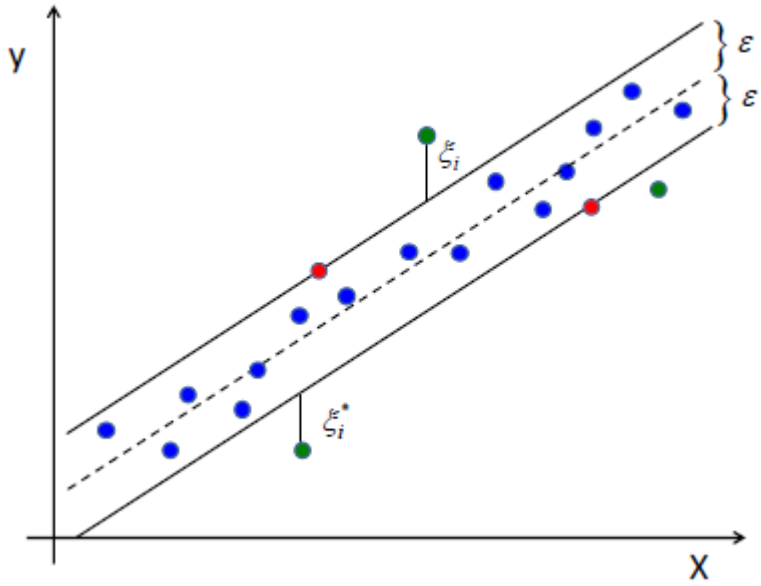


Figura 30 – Representação de uma solução SVR

Fonte: Filgueiras (2014)

3 Discussão

3.1 Metodologia

A metodologia aplicada nesse trabalho foi, majoritariamente, explicativa, tendo em vista que o desenvolvimento se ocupará da descrição de funcionamento de diversos métodos aplicáveis. A abordagem foi qualitativa, avaliando os diferentes algoritmos com base em suas características. Entretanto, a execução do algoritmo final proposto resultarou em uma curva, que pode ser avaliada através de dados quantitativos sobre a qualidade do algoritmo. O procedimento aplicado consiste, basicamente, da revisão da literatura sobre curvas de carga e algoritmos de previsão em *machine learning*.

Os medidores instalados na execução do projeto são do modelo MD30, da Embrasul. Ao todo, foram instalados 18 medidores em diferentes localidades da Universidade de Brasília. O *dataset* em utilização para a confecção do algoritmo de construção das curvas de carga foi originado a partir de medições do equipamento instalado no prédio UED da Faculdade do Gama (FGA). Os dados foram coletados pela equipe do projeto e disponiblizados em uma planilha Excel.

O banco de dados disponibilizado para o exercício era composto por 62137 medições, encompassando o período de 12/03/2019 a 03/04/2019. As medições foram realizadas por dois transdutores diferentes, de minuto em minuto.

A Tabela 1 apresenta um trecho de alguns dos dados aferidos pelo medidor analisado para a fase A, no transdutor 2.

Tabela 1 – Dados coletados dos medidores

U (V)	**I (A)**	**P (W)**	**Q (VAr)**	**S (VA)**	**Data**
218,35	5,38	-1175,49	17,09	1175,62	2019-03-12 13:59:03
218,01	6,36	-1385,66	-30,43	1385,99	2019-03-12 14:00:02
218,15	5,99	-1306,56	-26,24	1306,83	2019-03-12 14:01:02
217,93	6,96	-1516,43	-23,26	1516,61	2019-03-12 14:02:03
217,92	6,01	-1310,49	-20,83	1310,65	2019-03-12 14:03:03

Onde U representa a tensão, I, a corrente, P, a potência ativa, Q, a potência reativa, e S, a potência aparente na fase analisada.

Coletados os dados, o trabalho consiste em tratá-los da maneira adequada e traçar uma curva que represente o comportamento da carga do prédio sendo analisado.

3.2 Construção do algoritmo

Tendo em vista as seguintes características do problema que esse trabalho se propõe a resolver:

- É um problema de regressão: o que se procura analisar é um comportamento das variáveis ao longo do tempo, e não sua classificação;
- Permite a utilização de modelos de aprendizado supervisionados, uma vez que ambas as variáveis, x e y, são conhecidas, restando, então, definir a função que descreva o comportamento $y = f(x)$;

O algoritmo que melhor responde às necessidades citadas, dentre aqueles estudados nas etapas anteriores desse projeto, é o SVR, o qual foi definido como aquele a ser implementado e testado, na busca pela construção de um modelo preditivo.

A linguagem de programação escolhida para elaboração do algoritmo foi o Python, e a plataforma para implementação, o JupyterLab. Os dados foram importados de uma tabela Excel contendo as medições citadas anteriormente, e para teste e determinação dos parâmetros da SVR, foi definido, de maneira arbitrária, que o dia da semana delimitado para a construção do algoritmo e seus conjuntos de treinamento, validação e teste seria a quinta-feira. Assim, o conjunto de dados utilizado para essa função possuiu 4279 medições, referentes às três quintas-feiras englobadas pelo período de aferição. Um trecho desse conjunto para a fase A e o transdutor 2 pode ser observado na Tabela 2.

Tabela 2 – Conjunto de dados para quinta-feira após tratamento

P (W)	**Q (VAr)**	Hora	Mês	Ano	Dia	Dia da semana	Hora fracionada
0,30	-36,71	00:00:03	3	2019	14	quinta-feira	0,0000
0,33	-135,28	00:01:03	3	2019	14	quinta-feira	0,0167
0,35	-117,28	00:02:02	3	2019	14	quinta-feira	0,0333
0,34	-126,22	00:03:02	3	2019	14	quinta-feira	0,0500
0,35	-111,43	00:04:02	3	2019	14	quinta-feira	0,0667

No exemplo extraído para a construção da Tabela 2, a potência ativa da fase A se encontra normalizada. Essa etapa foi feita pela simples divisão de cada valor da potência ativa da fase A pelo valor máximo encontrado para essa variável ao longo do *dataframe*.

A criação de colunas referentes ao mês, ano, dia do mês e dia da semana foi realizada a fim de permitir a filtragem dos dados pelo usuário conforme desejado. Assim, caso se deseje analisar apenas os dados de um ano específico, por exemplo, é mais simples fazer tal separação.

O *dataframe* representado pela Tabela 2 pode ser observado, em sua forma original e mais completa, no Apêndice A, sob a alcunha de "df2thurs".

Para determinação dos parâmetros do SVR e validação do algoritmo, foi necessário dividir os dados em conjuntos de treino, teste e validação. Essa divisão foi feita através da função *train_test_split* da biblioteca *scikit learn*, aplicada duas vezes conforme o código em Apêndice. Foi determinada uma escala de 70%/15%/15% para cada um desses conjuntos, respectivamente, totalizando em um conjunto de treino com 2995 medições, e teste e validação com 642 medições cada. Os conjuntos de treino, teste e validação podem ser vistos na Figura 31.

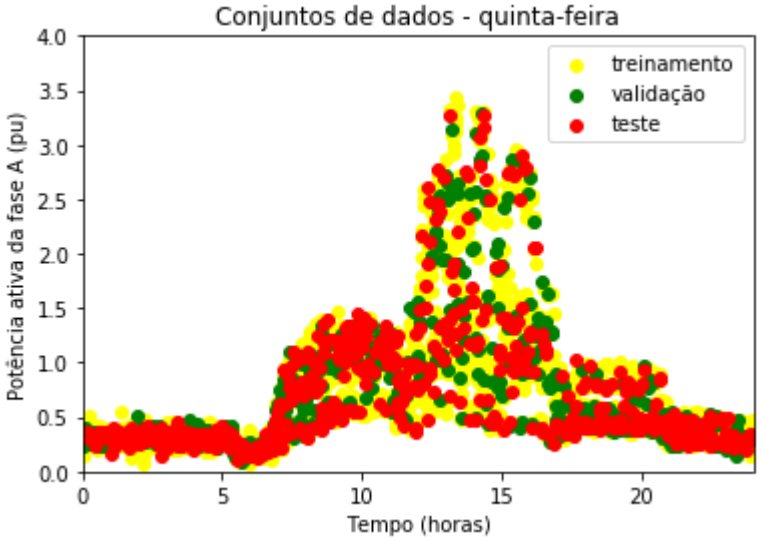


Figura 31 – Conjuntos de treino, teste e verificação da SVR

O conjunto de treinamento é utilizado para calibração dos dados, gerando diferentes valores para os parâmetros do SVR. O conjunto de validação, então, serve para determinar quais valores desses parâmetros são os mais adequados, enquanto o conjunto de teste serve para obter os resultados finais do algoritmo.

Realizada a divisão dos conjuntos, foi implementada no algoritmo a função SVR, da biblioteca *scikit learn*. Tal função toma os parâmetros de entrada do SVR e resolve as equações supracitadas. Em primeiro momento, foi observado que a uma função kernel RBF era mais condizente com o comportamento dos dados, vistos graficamente, e, portanto, era a mais aconselhada a ser usada para testes. Segundo Santos et al. (2017), a função kernel RBF é de mais fácil treinamento, por possuir apenas um parâmetro a mais a ser ajustado (γ). Além disso, foi delimitada a potência ativa da fase A como variável de interesse, e uma base de 1 MVA para normalização dos dados.

Os parâmetros a serem otimizados são os valoes de C e ε, conforme apresentados no item 2.4. Além disso, a função SVR aplicada também possui o parâmetro γ, que representa o grau de influência dos pontos alimentando o algoritmo: quanto maior o valor de γ, mais relevantes os pontos mais distantes serão para o algoritmo, enquanto valores menores de γ resultarão em uma menor importância aos pontos distantes da curva de

separação.

A otimização dos parâmetros do SVR foi feita utilizando a ferramenta NIOTS - *Nature-Inspired Optimization Tool for SVM* -, desenvolvida por Santos et al. (2017).

A ferramenta NIOTS realiza, basicamente, a sugestão de valores para os parâmetros da SVR a partir de conjuntos de treinamento e validação como dados de entrada. Em sua execução, foram realizadas 150 iterações, resultando em diferentes valores para os parâmetros C, γ e ε. Destes 150, 10 foram separados para a análise, sendo destacados na Tabela 3.

Tabela 3 – Iterações realizadas para calibragem dos parâmetros do SVR

Iteração	**C**	γ	ε	**MSE**	SV
1	9,225478	3,811758	0,122693	0,127912	1154
2	5,481887	1,369536	1,478192	0,217827	3
3	3,107481	2,173555	0,370127	0,135927	534
4	7,356346	3,055351	0,286302	0,133997	677
5	0,147883	1,369209	0,884919	0,173143	197
6	0,742931	1,000000	1,107234	0,191065	86
7	8,846984	3,737792	0,662520	0,169999	303
8	9,216248	3,052636	0,209304	0,130087	856
9	1,927158	2,104547	0,294549	0,132280	711
10	2,871742	1,296545	1,369537	0,196991	16

Em que MSE é o erro quadrático médio e SV é o número de vetores de suporte de cada iteração. É válido notar que, quanto menor o erro quadrático médio, melhor a função descreve, matematicamente, o comportamento dos pontos. Ao mesmo tempo, quanto mais o número de vetores de suporte, maior a complexidade da rede e a possibilidade de *overfit*: quando a função se torna demasiado voltada para a descrição daquele conjunto específico de pontos, e não de um comportamento generalizado.

Os erros quadráticos médios apresentados se referem ao conjunto de validação: ou seja, os valores dos parâmetros são calculados pela ferramenta a partir do conjunto de treinamento, e então aplicados ao conjunto de validação, resultando nos erros indicados.

Tendo em vista que a primeira iteração foi a que resultou num menor erro quadrático médio (MSE), e, portanto, melhor aproximou a curva dos dados sendo apresentados ao programa, ficou definido que essa seria a utilizada para a determinação dos parâmetros da função SVR.

Entretanto, é válido observar que a primeira iteração também foi a que resultou no maior número de vetores de suporte quando da implementação no conjunto de treinamento, aumentando, assim, a possibilidade de *overfit* do modelo. Santos et al. (2017), para contornar tal problema, define um coeficiente de correlação, que avalia a solução de acordo com o erro quadrático médio e o número de vetores de suporte. A dificuldade de

implementação dessa função, e a correlação, via de regra, proporcional entre erro quadrático médio e coeficiente de correlação, levou à opção por considerar somente o erro quando da escolha da solução para o algoritmo.

3.3 Resultados

Definidos os coeficientes, a função SVR foi aplicadano conjunto de teste dos diferentes dias da semana, resultando nas Figuras 32 a 38.

Figura 32 – Curva de carga do prédio UED - segunda-feira

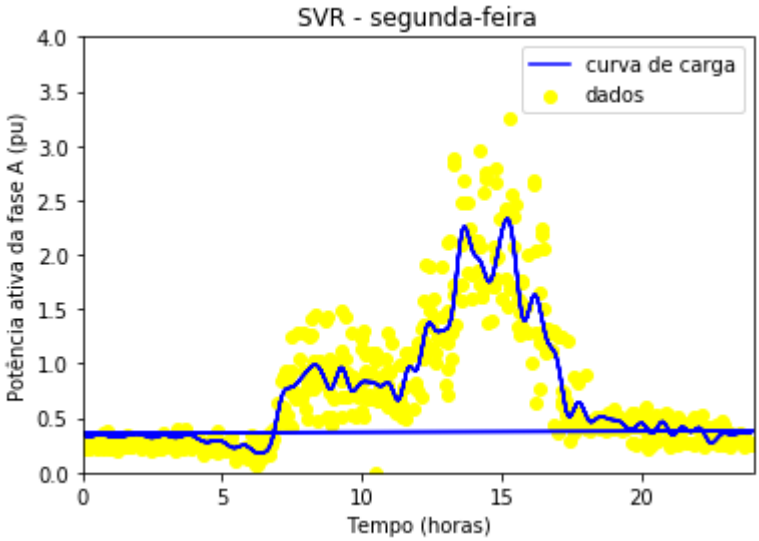


Figura 33 – Curva de carga do prédio UED - terça-feira

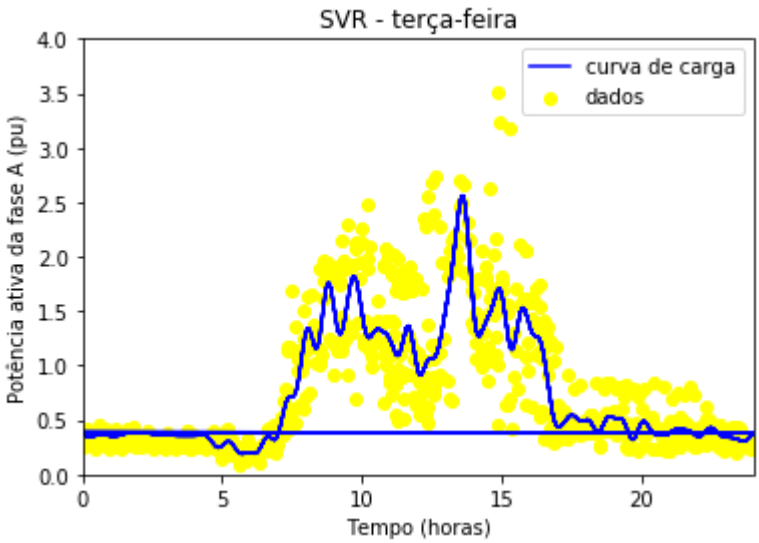


Figura 34 – Curva de carga do prédio UED - quarta-feira

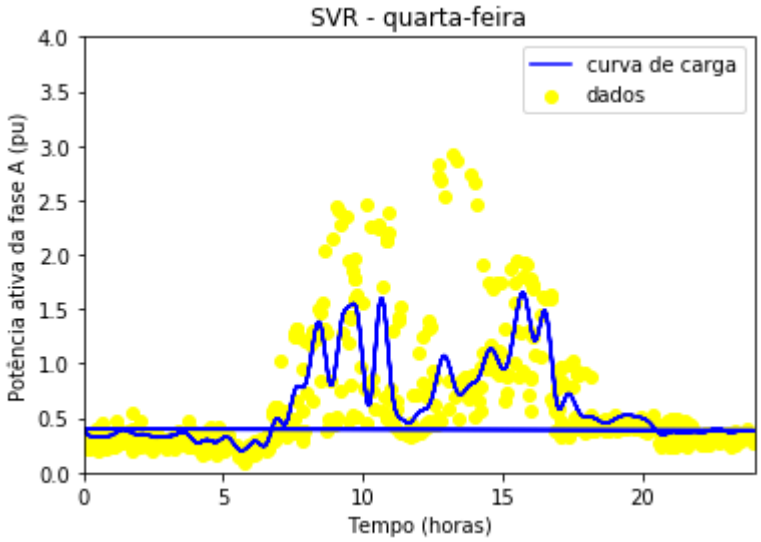


Figura 35 – Curva de carga do prédio UED - quinta-feira

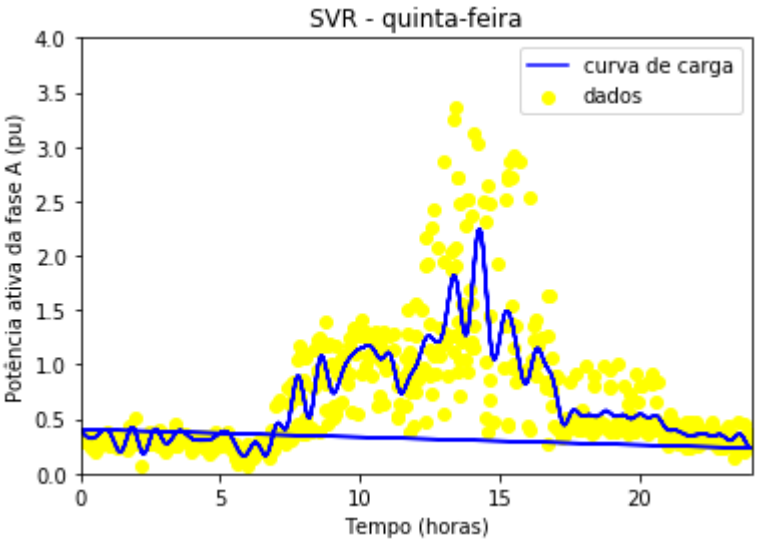


Figura 36 – Curva de carga do prédio UED - sexta-feira

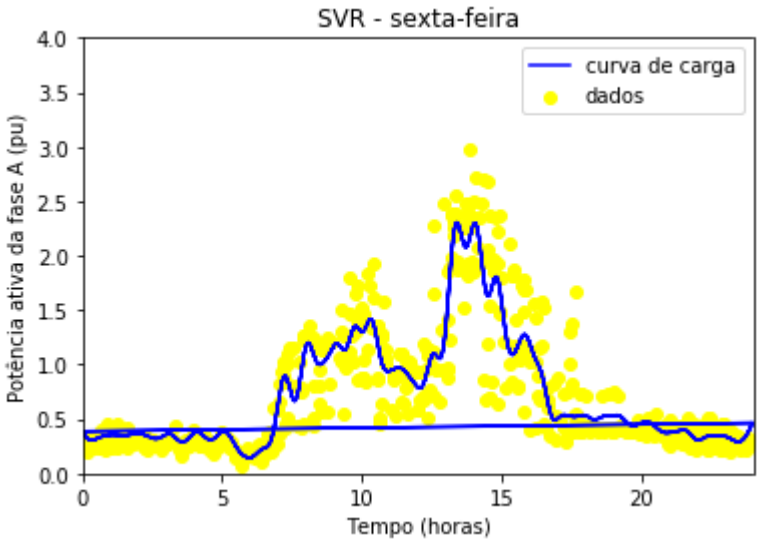


Figura 37 – Curva de carga do prédio UED - sábado

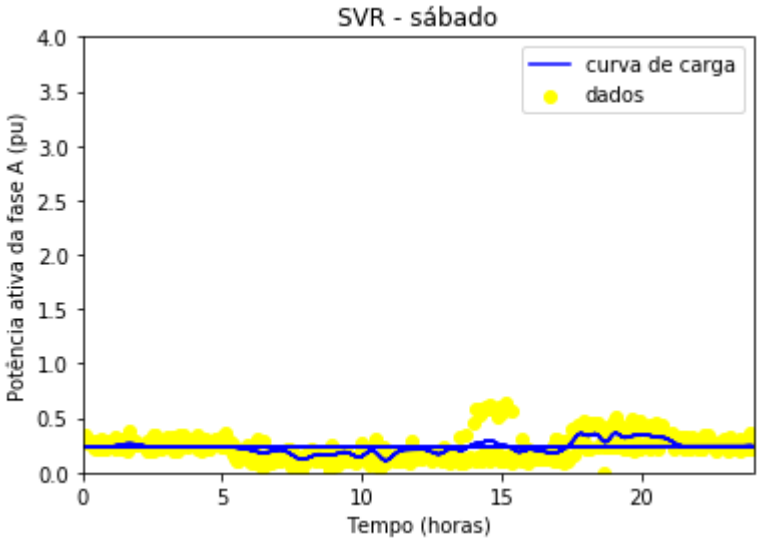
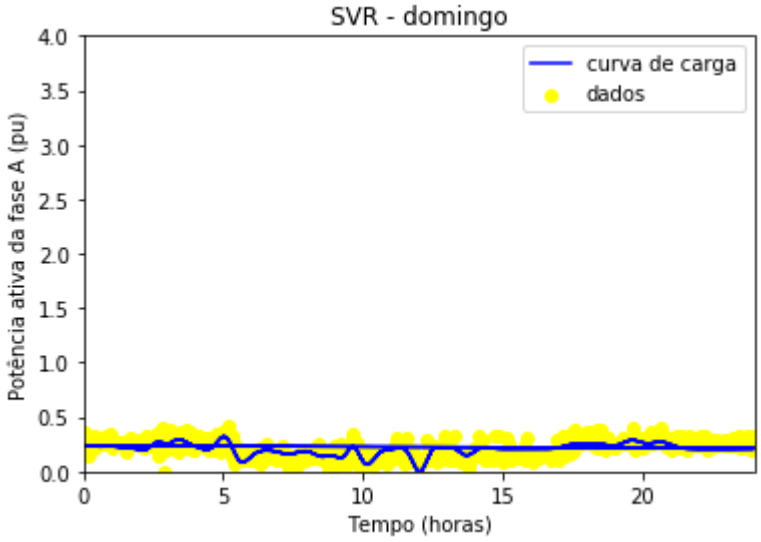


Figura 38 – Curva de carga do prédio UED - domingo



A Tabela 4 apresenta os erros quadráticos médios observados para cada um dos dias em uma das iterações do programa.

Tabela 4 – Erros quadráticos médios do conjunto de testes

Dia da semana	MSE
Segunda-feira	0,07
Terça-feira	0,13
Quarta-feira	0,19
Quinta-feira	0,15
Sexta-feira	0,08
Sábado	0,01
Domingo	0,01

Alguns problemas foram encontrados ao longo da execução do trabalho. Dentre eles, os principais se encontram na disposição dos dados e no grande número de vetores de suporte da solução apresentada.

Primeiramente, os dados utilizados apresentaram uma variação muito grande entre si. É necessário avaliar se tal variação é explicada por imprecisão dos medidores, se retrata um comportamento específico do período sendo analisado, ou se revela questões de instabilidade na rede elétrica do prédio em questão. Essa dúvida pode ser respondida pela coleta de novos dados e utilização dos mesmos para alimentação do algoritmo desenvolvido, observando a alteração do comportamento das curvas.

Além disso, o critério de seleção dos parâmetros, escolhendo aqueles que resultam no menor erro quadrático médio, não garante a resiliência do algoritmo em relação a *overfits*.

4 Conclusão

A construção de perfis típicos de carga é uma ferramenta de extrema importância na análise de sistemas elétricos. O presente trabalho se ocupa em abordar o caso da Universidade de Brasília, traçando perfis para os diferentes dias da semana a partir de medições adquiridas.

Foram analisados diversos métodos diferentes em *machine learning*, possibilitando o aprendizado sobre as mais variadas abordagens para tratamento de dados em problemas diversos. Também foi determinado como o mais adequado dentre eles, para a aplicação em questão, o SVR, que foi utilizado na construção de um algoritmo.

O algoritmo desenvolvido foi alimentado com dados colhidos de um dos medidores instalados no programa, permitindo a construção de curvas típicas de carga para um dos prédios da UnB. Tal algoritmo pode servir de molde para aplicações em outros prédios e adaptações.

Trabalhos futuros correlatos podem aprimorar o algoritmo criando, em cima da ferramenta, um modelo preditivo. Para tal, será necessário levar em considerações demais fatores relevantes para a curva de cada dia, como os dias anteriores, época do ano, períodos de férias ou aula, entre outros. Também é necessário avaliar a credibilidade dos dados utilizados para a criação da ferramenta e a necessidade de adaptação dos parâmetros, de modo a evitar um *overfit* das curvas.

Referências

ABBAS, O. A. Comparisons between data clustering algorithms. Computer Science Department, Yarmouk University, Irbid, Jordânia, p. 6, 2008. Citado 5 vezes nas páginas 18, 19, 22, 30 e 31.

ALMEIDA, V. A.; PESSANHA, J. F. M.; CALOBA, L. P. Uma metodologia para o tratamento de dados de carga baseada em técnicas de inteligência artificial. Centro de Pesquisas de Energia Elétrica (CEPEL) e Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia, Rio de Janeiro, Brasil, p. 10, S.D. Citado na página 29.

CAMPOS, J. R. *"Desenvolvimento de um Sistema Dinâmico para Predição de Cargas Elétricas por Redes Neurais Através do Paradigma de Programação Orientada a Objeto sob a Linguagem JAVA*. Dissertação (Mestrado em Engenharia Elétrica) — Faculdade de Engenharia de Ilha Solteira, Universidade Estadual Paulista "Júlio de Mesquita Filho", Ilha Solteira, Brasil, 2010. Citado 2 vezes nas páginas 27 e 28.

CARVALHO, J. P. J.; NETO, M. K. *Agrupamento de padrões de curva de carga utilizando algoritmos e técnicas de agrupamento como alternativa tarifária*. Graduação em Engenharia Industrial Elétrica-Eletrotécnica — Departamento Acadêmico de Eletrotécnica, Universidade Tecnológica Federal do Paraná, Curitiba, Brasil, 2011. Citado na página 19.

CORTES, C.; VAPNIK, V. N. Support-vector networks. ATT Bell Labs, p. 25, 1995. Citado 2 vezes nas páginas 32 e 34.

DRUCKER, H. et al. Support vector regression machines. Bell Labs and Monmouth University, Department of Electronic Engineering, p. 7, 1997. Citado na página 37.

FILGUEIRAS, P. H. *Regressão por vetores de suporte aplicado na determinação de propriedades fisico-químicas de petróleo e biocombustíveis*. Doutorado em Ciências — Instituto de Química, Universidade Estadual de Campinas, Campinas, Brasil, 2014. Citado 7 vezes nas páginas 32, 33, 34, 35, 36, 37 e 38.

FRANCISQUINI, A. A. *Estimação de Curvas de Carga em Pontos de Consumo e em Transformadores de Distribuição*. Dissertação (Mestrado em Engenharia Elétrica) — Faculdade de Engenharia de Ilha Solteira, Universidade Estadual Paulista "Júlio de Mesquita Filho", Ilha Solteira, Brasil, 2006. Citado 5 vezes nas páginas 13, 14, 15, 16 e 17.

MCLOUGHLIN, F.; DUFFY, A.; COLON, M. A clustering approach to domestic electricity load profile characterisation using smart metering data. Dublin Institute of Technology, Dublin, Irlanda, p. 12, 2015. Citado 2 vezes nas páginas 17 e 30.

NATITA, W.; WIBOONSAK, W.; DUSADEE, S. Appropriate learning rate and neighborhood function of self-organizing map (som) for specific humidity pattern classification of southern thailand. In: *International Journal of Modeling and Optimization*. [S.l.: s.n.], 2016. v. 6, No. 1. Citado 2 vezes nas páginas 28 e 29.

SANGALLI, L. M. et al. Functional clustering and alignment methods with applications. Dipartamento di Matematica, Politecnico di Milano, Milão, Itália, p. 20, 2010. Citado 2 vezes nas páginas 19 e 21.

SANTOS, C. E. et al. A svm optimization tool and fpga system architecture applied to nmpc. Proceedings of SBCCI '17, Fortaleza - CE , Brazil,, p. 7, 2017. Citado 2 vezes nas páginas 41 e 42.

SENJYU, T.; HIGA, S.; UEZATO, K. Future load curve shaping based on similarity using fuzzy logic approach. In: *Generation, Transmission and Distribution, IEE Proceedings*. [S.l.: s.n.], 1998. v. 145, Issue 4. Citado 2 vezes nas páginas 17 e 18.

SHAMEEM, M. U. S.; FERDOUS, R. An efficient k-means algorith integrated with jaccard distance mesure for document clustering. Dept. of Computer Science and Engineering, University of Development Alternative, Dhaka, Bangladesh and University of Trento, Trento, Itália, p. 6, 2009. Citado na página 21.

SRINIVASAN, D.; LIEW, A. C.; CHANG, C. S. Forecasting daily load curves using a hybrid fuzzy-neural approach. In: *Generation, Transmission and Distribution, IEE Proceedings*. [S.l.: s.n.], 1994. v. 141, Issue 6. Citado na página 17.

STEFANOVIC, P.; KURASOVA, O. Investigation on learning patterns of self-organizing maps. In: *Baltic Journal of Modern Computing*. Intitute of Mathematics and Informatics, Vilnius University, Vilnius, Lituânia: [s.n.], 2014. v. 2, No. 2, p. 44–55. Citado na página 29.

VAPNIK, V. N.; LERNER, A. Y. Pattern recognition using generalised portraits. Automation and Remote Control, p. 24, 1963. Citado na página 32.

VIVIAN, C. H. V. *Estudo de Caso da Curva de Carga de Consumidor Residencial*. Graduação em Engenharia Elétrica — Departamento de Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil, 2015. Citado 2 vezes nas páginas 14 e 15.

Apêndices

APÊNDICE A – Relatório de execução da ferramenta NIOTS

General Parameters
Optimizer: MODE
Iterations: 150
Cross-set: 4
Samples: 1
Machine: Regression
MODE parameters

Population: 20
Scale factor: 0.7000
Crossover rate: 0.4000
Lower limit: -10.0000
Upper limit: 10.0000
Diversity factor: classic
Kernel: RBF

Sample 1

Index	C	Gamma	Epsilon	MSE	SV
1.	9.22547778	3.81175799	0.12269304	0.12791151	1154.00000000
2.	3.67834105	-1.93071591	1.62523162	1.42108830	2.00000000
3.	5.48188693	1.36953562	1.47819215	0.21782666	3.00000000
4.	3.10748079	2.17355479	0.37012734	0.13592712	534.00000000
5.	7.35634631	3.05535078	0.28630169	0.13399693	677.00000000
6.	0.14788342	1.36920894	0.88491949	0.17314261	197.00000000
7.	0.74293053	1.00000000	1.10723405	0.19106479	86.00000000
8.	8.84698431	3.73779166	0.66252043	0.16999869	303.00000000
9.	9.21624809	3.05263649	0.20930431	0.13008662	856.00000000
10.	1.92715807	2.10454689	0.29454855	0.13228008	711.00000000
11.	2.87174213	1.29654542	1.36953692	0.19699082	16.00000000
12.	4.19388522	2.10454689	0.46798412	0.14277672	413.00000000
13.	6.25969310	3.94366853	0.13317455	0.12819839	1108.00000000
14.	8.37147921	3.68028398	0.40000000	0.14235215	478.00000000
15.	4.19388522	3.41689942	0.19058815	0.12985376	924.00000000
16.	3.70193134	3.06680690	0.25369281	0.13205184	769.00000000
17.	8.61126440	3.94366853	0.95911558	0.18245053	135.00000000

18.	3.47050042	3.82213433	0.14784906	0.12886841	1054.00000000
19.	3.26568941	3.01396982	0.39330234	0.14050226	499.00000000
20.	-0.21475830	1.00000000	0.69445638	0.16304009	319.00000000
21.	3.52356598	2.94033967	0.18437099	0.12947003	954.00000000
22.	8.59534644	3.12282236	0.22157902	0.13044262	821.00000000
23.	5.35576358	2.53262281	0.58322961	0.15887524	353.00000000
24.	-0.30174228	1.00000000	0.92979218	0.17616413	173.00000000
25.	6.78194014	3.96274886	0.12572511	0.12805186	1142.00000000
26.	5.00139746	4.30827112	0.17037802	0.12938818	993.00000000
27.	2.58461113	3.67737579	0.16005520	0.12901534	1023.00000000
28.	4.19388522	2.10454689	0.50000000	0.14703838	391.00000000
29.	4.19388522	2.10454689	0.50874165	0.15164724	380.00000000
30.	4.19388522	3.12282236	0.24257288	0.13174330	790.00000000
31.	4.75758499	2.70517597	0.53348046	0.15770577	364.00000000
32.	2.87174213	1.29654542	0.96195115	0.17622162	151.00000000
33.	3.38135076	2.71928169	0.24788504	0.13138907	804.00000000
34.	4.08695609	1.00000000	1.40000000	0.21326303	14.00000000
35.	1.81834396	4.34959037	0.15209445	0.12894140	1042.00000000
36.	4.33697143	3.33854897	0.18562889	0.12960840	938.00000000
37.	7.38908196	3.04235701	0.17037802	0.12910187	999.00000000
38.	3.16722864	1.22275223	0.98379229	0.18216136	145.00000000

APÊNDICE B – Algoritmo elaborado

In [1]:

```
from datetime import datetime
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.svm import SVR
%matplotlib inline
```

In [2]:

```
df = pd.read_excel (r'C:\Users\Tiago\Desktop\energy-measurements.xlsx')
```

In [3]:

```
df.head()
```

Out[3]:

	measurements_ptr_id	voltage_a	voltage_b	voltage_c	current_a	current_b	current_c	a
0	2	218.466461	218.401352	218.214279	0.197309	0.065270	0.136819	
1	1	218.348907	219.600113	216.990646	5.384120	1.463921	4.357771	
2	3	218.008759	219.212936	217.174164	6.357503	2.047830	4.458454	
3	4	218.294418	218.180557	218.259857	0.180909	0.063610	0.134145	
4	5	218.730743	217.945007	217.938721	0.061388	0.066549	0.142513	

In [4]:

```
df.dropna(subset=["voltage_a"], axis=0, inplace = True)
```

In [5]:

```
df.drop(['measurements_ptr_id', 'voltage_a', 'voltage_b', 'voltage_c', 'current_a', 'currer
```

In [6]:

```
df['collection_date'] = pd.to_datetime(df['collection_date'])
```

In [7]:

```
df.head()
```

Out[7]:

	active_power_a	active_power_b	active_power_c	reactive_power_a	reactive_power_b	reactiv
0	-43.066216	-14.217773	-29.836258	1.838711	-1.031398	
1	-1175.492432	-311.543274	-944.647095	17.091665	79.299324	
2	-1385.657349	-441.491974	-966.369812	-30.432302	81.276054	
3	-39.458797	-13.843461	-29.278353	1.603372	-0.985405	
4	-13.384355	-14.501029	-31.039928	-1.074474	0.294941	

In [8]:

```
from datetime import timedelta
df['collection_date'] = df['collection_date']-timedelta(hours=4)
```

In [9]:

```
df.head()
```

Out[9]:

	active_power_a	active_power_b	active_power_c	reactive_power_a	reactive_power_b	reactiv
0	-43.066216	-14.217773	-29.836258	1.838711	-1.031398	
1	-1175.492432	-311.543274	-944.647095	17.091665	79.299324	
2	-1385.657349	-441.491974	-966.369812	-30.432302	81.276054	
3	-39.458797	-13.843461	-29.278353	1.603372	-0.985405	
4	-13.384355	-14.501029	-31.039928	-1.074474	0.294941	

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 53974 entries, 0 to 62135
Data columns (total 8 columns):
active_power_a      53974 non-null float64
active_power_b      53974 non-null float64
active_power_c      53974 non-null float64
reactive_power_a    53974 non-null float64
reactive_power_b    53974 non-null float64
reactive_power_c    53974 non-null float64
collection_date     53974 non-null datetime64[ns]
transductor_id      53974 non-null int64
dtypes: datetime64[ns](1), float64(6), int64(1)
memory usage: 3.7 MB
```

In [11]:

```
df['month'] = df.collection_date.dt.month
df['year'] = df.collection_date.dt.year
df['day'] = df.collection_date.dt.day
df['weekday'] = df.collection_date.dt.strftime('%A')
df['hour'] = df.collection_date.dt.hour
df['full_hour'] = df.collection_date.dt.hour + (df.collection_date.dt.minute/60)
```

In [12]:

```
df.head()
```

Out[12]:

	active_power_a	active_power_b	active_power_c	reactive_power_a	reactive_power_b	reactiv
0	-43.066216	-14.217773	-29.836258	1.838711	-1.031398	
1	-1175.492432	-311.543274	-944.647095	17.091665	79.299324	
2	-1385.657349	-441.491974	-966.369812	-30.432302	81.276054	
3	-39.458797	-13.843461	-29.278353	1.603372	-0.985405	
4	-13.384355	-14.501029	-31.039928	-1.074474	0.294941	

In [13]:

```
df.active_power_a = -df.active_power_a
```

In [14]:

```
df.tail()
```

Out[14]:

	active_power_a	active_power_b	active_power_c	reactive_power_a	reactive_power_b	re
62128	294.483521	-467.660797	-485.497437	-129.404770	-22.564837	
62130	331.791718	-475.034149	-444.193878	-124.117004	-34.624096	
62131	285.433746	-361.627594	-440.487244	-154.786316	-46.666035	
62134	316.716949	-343.325592	-443.965790	-136.730316	-28.734371	
62135	333.383484	-467.175110	-432.656158	-129.252380	-27.565830	

In [15]:

```
df['active_power_a'] = df['active_power_a']/1000
```

In [16]:

```
df.tail()
```

Out[16]:

	active_power_a	active_power_b	active_power_c	reactive_power_a	reactive_power_b	re
62128	0.294484	-467.660797	-485.497437	-129.404770	-22.564837	
62130	0.331792	-475.034149	-444.193878	-124.117004	-34.624096	
62131	0.285434	-361.627594	-440.487244	-154.786316	-46.666035	
62134	0.316717	-343.325592	-443.965790	-136.730316	-28.734371	
62135	0.333383	-467.175110	-432.656158	-129.252380	-27.565830	

In [17]:

```
thursday = df["weekday"]=="Thursday"
```

In [18]:

```
df_thursday = df[thursday]
```

In [19]:

```
trans2 = df_thursday["transductor_id"] == 2
```

In [20]:

```
df2thurs = df_thursday[trans2]
df2thurs.head()
```

Out[20]:

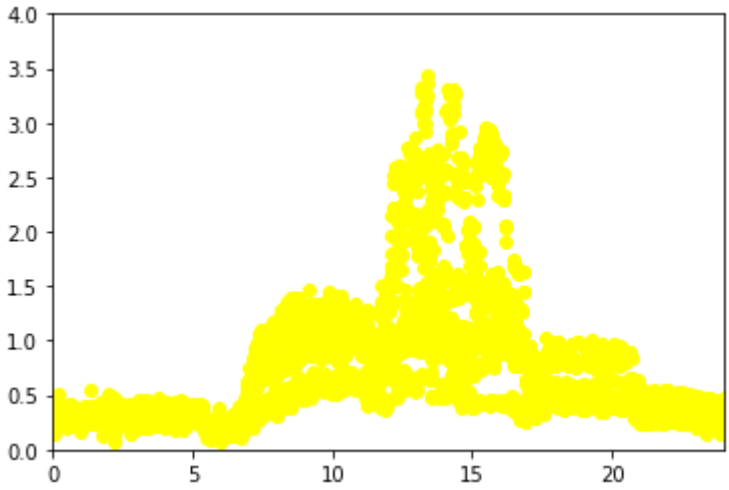
	active_power_a	active_power_b	active_power_c	reactive_power_a	reactive_power_b	rea
4564	0.304451	-320.739624	-693.780090	-112.419380	-54.265694	
4566	0.325967	-330.845367	-706.374878	-135.279968	-49.093582	
4569	0.347280	-327.841797	-702.110474	-117.283577	-38.812450	
4571	0.335536	-328.654602	-737.848083	-126.222153	-38.987278	
4572	0.347466	-332.925598	-731.392456	-111.425110	-41.631100	

In [21]:

```
plt.scatter(df2thurs["full_hour"], df2thurs["active_power_a"], c='yellow', label='data')
plt.xlim(0, 24)
plt.ylim(0, 4)
```

Out[21]:

```
(0, 4)
```



In [22]:

```
X = df2thurs.iloc[:,-1].values
X = np.array(X).reshape(-1,1)
```

In [23]:

```
y = df2thurs.iloc[:,0].values
```

In [24]:

```
y
```

Out[24]:

```
array([0.30445108, 0.32596661, 0.34728006, ..., 0.28188748, 0.35679559,
       0.25327531])
```

In [25]:

```
X
```

Out[25]:

```
array([[0.00000000e+00],
       [1.66666667e-02],
       [3.33333333e-02],
       ...,
       [2.39500000e+01],
       [2.39666667e+01],
       [2.39833333e+01]])
```

In [26]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [27]:

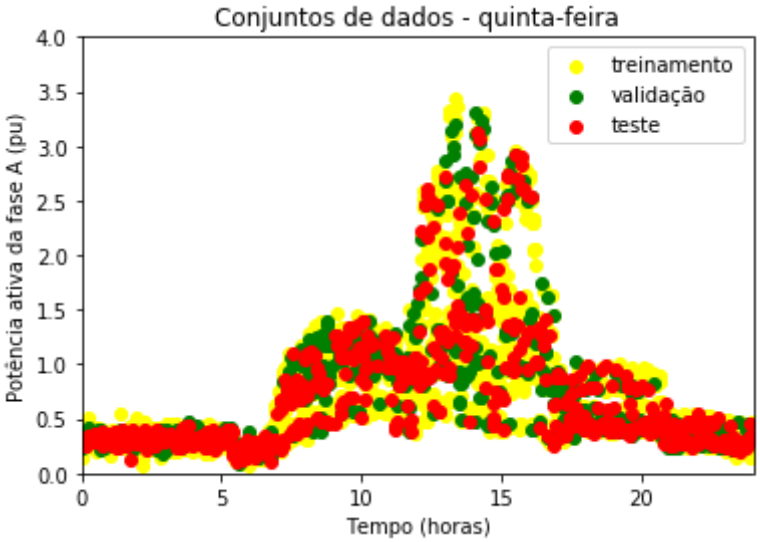
```
X_val, X_test, y_val, y_test = train_test_split(X_test, y_test, test_size=0.5)
```

In [28]:

```
plt.scatter(X_train, y_train, color='yellow', label='treinamento')
plt.scatter(X_val, y_val, color='green', label='validação')
plt.scatter(X_test, y_test, color='red', label='teste')
plt.xlabel('Tempo (horas)')
plt.ylabel('Potência ativa da fase A (pu)')
plt.title('Conjuntos de dados - quinta-feira')
plt.legend()
plt.xlim(0, 24)
plt.ylim(0, 4)
```

Out[28]:

```
(0, 4)
```



In [29]:

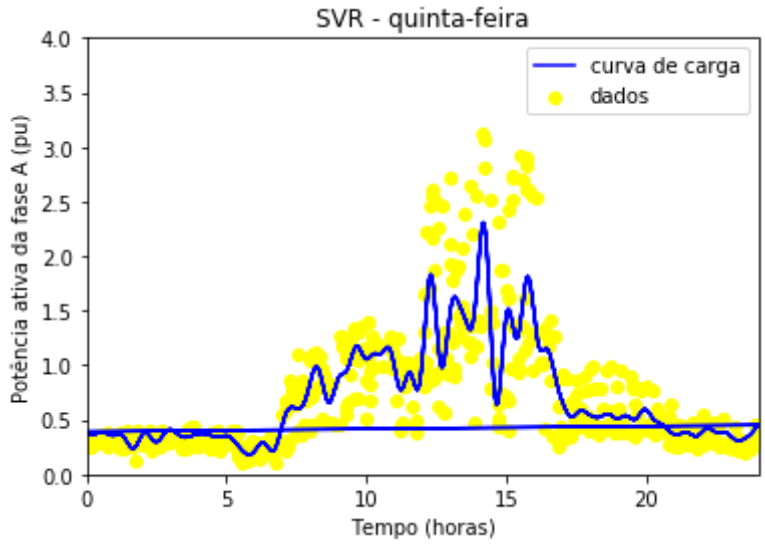
```
svr_rbf = SVR(kernel='rbf', C=9.225478, gamma=3.811758, epsilon= 0.122693)
```

In [30]:

```
y_rbf = svr_rbf.fit(X_test, y_test).predict(X)
```

In [31]:

```
plt.scatter(X_test, y_test, c='yellow', label='dados')
plt.plot(X, y_rbf, c='blue', label='curva de carga')
plt.xlim(0, 24)
plt.ylim(0, 4)
plt.xlabel('Tempo (horas)')
plt.ylabel('Potência ativa da fase A (pu)')
plt.title('SVR - quinta-feira')
plt.legend()
plt.show()
```



In [32]:

```
from sklearn.metrics import mean_squared_error
```

In [33]:

```
mean_squared_error(y, y_rbf)
```

Out[33]:

```
0.14946262111573233
```

In [ ]:

```

```