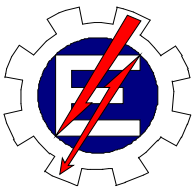


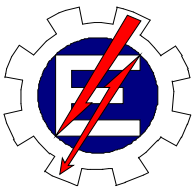
Universidade Federal de Itajubá

APLICAÇÃO DE TÉCNICAS DE BIG DATA À PREVISÃO DA CARGA ELÉTRICA

Cláudio Inácio de Almeida Costa

Itajubá, MG

2017



Universidade Federal de Itajubá

APLICAÇÃO DE TÉCNICAS DE BIG DATA À PREVISÃO DA CARGA ELÉTRICA

Cláudio Inácio de Almeida Costa

Tese submetida ao Programa de Pós-Graduação da Universidade Federal de Itajubá, como parte dos requisitos para obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Prof. Luiz Eduardo Borges da Silva

Coorientador: Prof. Cláudio Ferreira

Itajubá, MG

2017

Aos meus pais, que souberam me educar pelo exemplo; a minha esposa, pelo carinho e amor; e a meus filhos, pela alegria de viver.

Agradecimentos

Uma longa caminhada só tem valor quando suas lembranças são compartilhadas com bons amigos.

Durante toda minha vida acadêmica, tive o prazer de compartilhar da sabedoria e do conhecimento do Prof. Dr. Germano Lambert Torres. A ele, devo muito mais que agradecimentos. Seu incentivo e seu exemplo sempre me levaram a vencer desafios, a crescer e a chegar cada vez mais longe.

Agradeço ao meu orientador, Prof. Dr. Luiz Eduardo Borges da Silva, pela confiança em mim depositada. Sua certeza em minha capacidade de cumprir meus objetivos foi decisiva para que eu seguisse em frente, e vencesse mais esta etapa.

Agradeço ao meu amigo e homônimo, Prof. Dr. Cláudio Ferreira, por me despertar o gosto pela matemática, por sempre me propor problemas difíceis, e me forçar a encontrar soluções mais simples.

Agradeço ao meu amigo, Prof. Dr. Carlos Henrique Valério de Moraes, por dividir comigo todas as novidades da tecnologia e da computação, sempre me ensinando a programar e pensar de forma mais concisa e rápida.

Agradeço aos demais membros da banca examinadora, Prof. Dr. Maurílio Pereira Coutinho e ao Prof. Dr. Hector Gustavo Arango, por todas as sugestões e opiniões.

Por fim, agradeço principalmente a Deus, pela benção de viver este momento tão feliz, com saúde, na companhia de toda minha família e de tão bons amigos.

Resumo

Com o avanço das telecomunicações e com o barateamento dos dispositivos de medição, os sistemas elétricos de potência passaram a gerar um enorme volume de dados. Estes chegam aos centros de operação com diferentes frequências, desde alguns minutos para o estado de disjuntores, até alguns milissegundos para medidas de tensão e corrente durante transitórios. O desafio atual é tornar estes dados disponíveis de forma simples e eficiente aos operadores. O objetivo é transformar a avalanche de dados em informações úteis ao processo de decisão. Neste sentido, várias técnicas de mineração de dados foram desenvolvidas. Recentemente, as técnicas de Big Data possibilitaram a manipulação de grandes bases de dados e a elaboração de modelos baseados em aprendizado de máquina e inteligência artificial.

Este trabalho apresenta a aplicação das técnicas de Big Data ao problema de previsão de carga. Bons modelos de previsão são fundamentais para o planejamento, operação e manutenção dos sistemas elétricos. Diversos fatores podem influenciar o comportamento futuro da carga, não necessariamente em intervalos regulares, ou da mesma forma, para os diversos horizontes de previsão. Dentre as diversas técnicas de Big Data disponíveis, foi escolhida a modelagem por Florestas Aleatórias (*Random Forests*). Esta técnica permite tratar grandes bases de dados, formadas tanto por atributos numéricos como categóricos, além de serem bastante robustas quanto à presença de dados faltosos, inconsistentes e com ruído. Seu algoritmo de aprendizado é rápido e gera modelos precisos e de fácil aplicação.

Este trabalho propõe também alterações na técnica de Florestas Aleatórias para a previsão de carga. Estas alterações foram aplicadas com sucesso a um conjunto de dados de uma concessionária brasileira obtendo-se bons resultados sem nenhuma intervenção humana. Por exemplo, a carga média no horizonte de até um ano à frente, importante para a contratação da energia pela concessionária, foi prevista com erro da ordem de décimos de porcentagem.

Palavras-chave: Mineração de Dados, Big Data, Florestas Aleatórias, Previsão de Carga

Abstract

Today, power systems are generating a huge amount of data, due to the availability of cheap measuring devices and advances on telecommunications. Those data arrive in control centers with different frequencies, from few minutes in the case of circuit breakers status, until few milliseconds for voltage and current measurements during transients. The actual challenge is to make these data available, in a simply and efficient way to operators. The aim is to turn all this data into useful information, relevant to the decision process. In this way, several data mining techniques were developed. Recently, Big Data techniques made practicable the manipulation of large databases, allowing the use of machine learning and artificial intelligence algorithms.

This thesis presents the application of Big Data techniques for the load forecasting problem. Good predictive models became a fundamental tool for planning, operation and maintenance of electrical systems. Several factors can influence the future behavior of electrical load, on different forecast horizons, but not necessarily at the same time intervals or in the same way. Among the several Big Data techniques available today, Random Forests were chosen. This technique allows the treatment of large databases, with both numerical and categorical attributes. It is quite robust as far as missing, inconsistent and noise data is concerned, and its learning algorithm is fast, accurate and easy-to-apply.

This thesis also proposes some changes in the Random Forests technique for load forecasting. These changes were successfully applied to a set of data from a Brazilian power distribution company, providing good results without any human intervention. For example, the average load forecasting for one year ahead horizon, important for energy contracting by the concessionaire, was predicted with an error in the order of magnitude of tenths of percentage.

Keywords: Data Mining, Big Data, Random Forests, Load Forecasting

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1
Introdução

1.1 - A Previsão de Carga no Sistema Elétrico de Potência

O comportamento da carga dentro de um Sistema Elétrico de Potência depende de um conjunto de fatores. Estes fatores podem ser classificados como internos ou externos. Os fatores internos são inerentes ao próprio sistema elétrico, como por exemplo: a ocorrência de falhas no sistema, o despacho ou não de geradores, características construtivas das linhas de transmissão, entre outros. Já os fatores externos são alheios ao sistema elétrico. Por exemplo: a temperatura ambiente na região geográfica da carga, costumes sociais dos clientes, políticas públicas, crises econômicas, etc. Todos estes fatores contribuem para determinar a energia consumida a cada instante pelos clientes.

Infelizmente, a contribuição de todos os fatores que influenciam a carga não obedece ao princípio da superposição, ou seja, não há necessariamente uma combinação linear das variáveis independentes. Assim, o problema da previsão de carga é um problema não linear, com inúmeras correlações, interações e interdependências entre seus fatores. Tudo isso torna a previsão de carga um problema desafiador, que atrai um grande número de pesquisadores, técnicos e planejadores do sistema elétrico.

1.2 - Horizontes de Previsão

A previsão de carga é dividida em três classes distintas, conforme a antecedência com que se calculam seus valores. Essa distância no tempo é chamada de “horizonte de previsão”, e divide-se em: curto, médio e longo prazo, conforme mostrado na Figura 1.

Figura 1 – Horizontes de previsão de carga

1.3 - Fatores que Influenciam a Carga

Cada um dos horizontes de previsão sofre influência maior ou menor de diferentes fatores. Os fatores que influenciam a carga podem ser agrupados conforme seu impacto nos diversos horizontes. A seção seguinte lista cada um dos horizontes e seus principais fatores.

1.3.1 - Fatores de Curto Prazo

Alguns dos principais fatores que influenciam na previsão de curto prazo [1][2][3] estão representados na Figura 2.

Figura 2 – Principais fatores que influenciam na previsão de curto prazo

- **Fatores Horários:** o comportamento da carga apresenta um ritmo cíclico bem característico, conforme o passar do dia. No caso de consumidores residenciais, aumentos significativos de carga podem ocorrer logo após o amanhecer ou no início da noite. Já alguns consumidores industriais, que operam em três turnos de oito horas, podem ter uma carga estável durante todo o dia e noite. Consumidores comerciais podem também ter cargas estáveis, mas somente durante o chamado "horário comercial". Somados todos estes fatores, nota-se uma clara assinatura no ciclo horário da carga, representando uma periodicidade de vinte e quatro horas.
- **Fatores Semanais:** cada dia da semana pode possuir uma tendência de demanda diferente. De um modo geral, podem-se dividir os dias em dias úteis (segunda a sexta) e finais de semana (sábado e domingo). Em uma segunda aproximação, nota-se que os sábados são diferentes dos domingos. Nota-se também que o início de uma segunda-feira pode sofrer a influência do domingo anterior. O mesmo ocorre com o final da sexta-feira, que é influenciada pela chegada do sábado seguinte. Por esta razão, é prática comum escolher-se a quarta-feira como representativa de um dia útil típico, e a carga dos domingos como referência de um feriado.
- **Feriados:** feriados nacionais e religiosos representam uma fonte de incerteza na previsão de carga. O consumo de energia nos feriados será diferente, não só dos dias úteis, mas também pode variar conforme o dia da semana no qual ocorre o feriado. É interessante notar que feriados fixos, como o sete de setembro, ocorrem cada ano em um dia de semana diferente. Por isso, feriados fixos causam efeitos variáveis na carga. Por outro lado, feriados religiosos móveis, como a sexta-feira Santa, apesar de variarem sua data a cada ano, tem um efeito fixo na carga, pois ocorrem sempre no mesmo dia da semana.

  Além disso, a ocorrência de feriados pode impactar não só a carga do próprio dia, mas também, dos dias anteriores e posteriores. A ocorrência de feriados próximos a finais de semana cria os chamados feriados prolongados. Para compensar os chamados dias "emendados", empresas

podem aumentar a produção em semanas anteriores, ou em finais de semana seguintes, por exemplo.

- **Categoria da Carga:** diferentes tipos de clientes apresentam diferentes necessidades e comportamentos. Os principais grupos de clientes são os residenciais, comerciais, industriais, rurais, poder público e iluminação pública. Apesar de ser uma classificação usual, estes grupos podem ainda possuir membros bem distintos entre si, justificando a criação de subgrupos.
- **Fatores Climáticos:** existem diversos fatores climáticos que podem influenciar a carga, desde a temperatura e umidade, até a velocidade dos ventos, índice pluviométrico, luminosidade e ocorrências de raios. A temperatura é um dos fatores mais importantes, pois afeta diretamente a quantidade de energia elétrica necessária para aquecer ou resfriar ambientes, água, alimentos, materiais e processos industriais. A luminosidade, por outro lado, afeta diretamente o acionamento da iluminação pública.
- **Distúrbios aleatórios:** O sistema elétrico está sujeito à ocorrência de distúrbios, sejam eles programados ou não. Manutenções, ocorrências na rede, falhas do sistema, entre outros, são eventos frequentes no sistema elétrico. Além destes, alguns consumidores industriais possuem grande variação em seu consumo de energia. A entrada ou saída de grandes blocos de cargas, como fornos, fundições ou grandes motores, causa um impacto que não pode ser negligenciado na previsão.

1.3.2 - Fatores de Médio Prazo

A previsão de médio prazo inclui várias incertezas que representam novos desafios [4][5][6][7][8]. Alguns dos principais fatores para previsão de médio prazo estão representados na Figura 3.

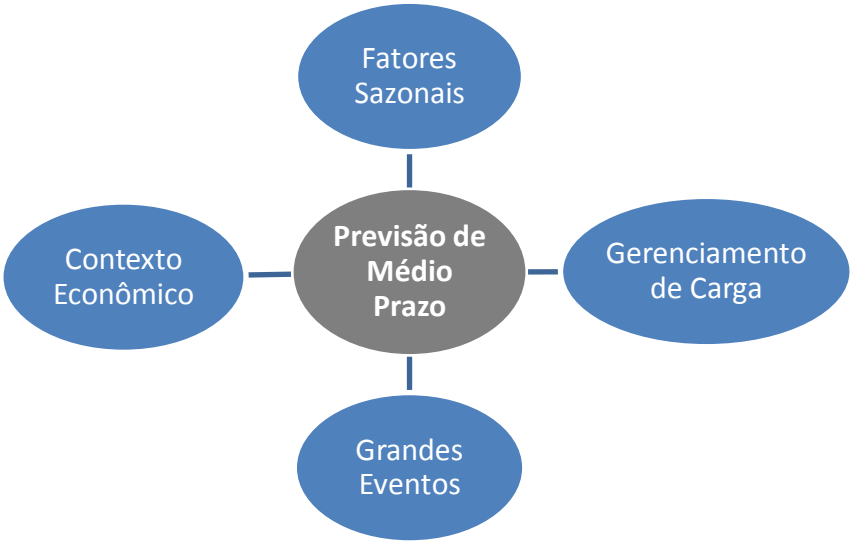


Figura 3 – Principais fatores que influenciam na previsão de médio prazo

- **Fatores Sazonais:** As mudanças sazonais durante o ano, principalmente as ligadas às quatro estações climáticas, têm influência direta sobre a curva de carga. As cargas térmicas, sejam elas de aquecimento ou de refrigeração, costumam representar uma grande fatia da demanda. Além da temperatura, o ciclo de seca e chuva costuma estar ligado à demanda, muitas vezes por razão da prática de preços diferenciados da energia em meses secos e chuvosos. Para garantir a segurança do fornecimento e a confiabilidade, muitas empresas consideram o cenário de pior caso para o seu planejamento, gerando excessos de capacidade, muitas vezes desnecessários.
- **Gerenciamento de Carga:** Várias medidas podem causar impacto direto no padrão de consumo de energia por parte dos usuários. A adoção do horário de verão é a maior delas. O desacoplamento dos picos residenciais e comerciais, causado pela modificação do horário, modifica completamente o perfil diário da carga. Medidas como esta permitem a postergação de investimentos de expansão do sistema, pois reduzem os picos de demanda, tornando a carga mais constante. Além do horário de verão, várias outras medidas de gerenciamento da carga podem ser adotadas. A promoção do uso mais eficiente da energia é outro exemplo. Campanhas para adoção de iluminação fluorescente compacta, LEDs, motores de alta eficiência ou mudanças tarifárias que incentivem o deslocamento de cargas para fora do

horário de pico podem gerar grandes impactos no perfil da carga. Outras estratégias de gerenciamento de carga incluem geração de emergência, produção de energia independente e cogeração industrial. Essas medidas são necessárias para minimizar o risco de sobrecargas ou interrupções de cargas essenciais, até que a capacidade de geração adicional possa ser disponibilizada.

- **Grandes Eventos:** Dependendo da magnitude de um evento, em termos de necessidades energéticas, a demanda associada a um evento pode influenciar significativamente a carga planejada de um sistema. Por exemplo, o início de atividades relacionadas a uma Olimpíada, ou a realização de uma Copa do Mundo, influencia muito a carga. O início de operação de grandes consumidores ou a sua interrupção, seja ela programada ou não, são eventos que apresentam grande dificuldade para serem considerados nos modelos de previsão de carga.
- **Contexto Econômico:** O contexto econômico é representado pelo conjunto de variáveis contextuais que influenciam o desempenho da atividade econômica. A evolução da taxa de inflação e da taxa de juros moldam a atividade empresarial, e consequentemente o consumo de energia. Baixas taxas de juros e inflação geram maior confiança e incentivo ao investimento, a produção, e a aquisição de bens e serviços. O contexto econômico inclui ainda outras variáveis como as taxas de câmbio, de desemprego, a balança comercial, os custos energéticos e de mão de obra. Conforme o setor de atividade de cada empresa, qualquer destes fatores pode influenciar seu padrão de consumo de energia elétrica.

1.3.3 - Fatores de Longo Prazo

A previsão de longo prazo é normalmente influenciada por fatores externos a carga [9][10][11], conforme mostra a Figura 4.

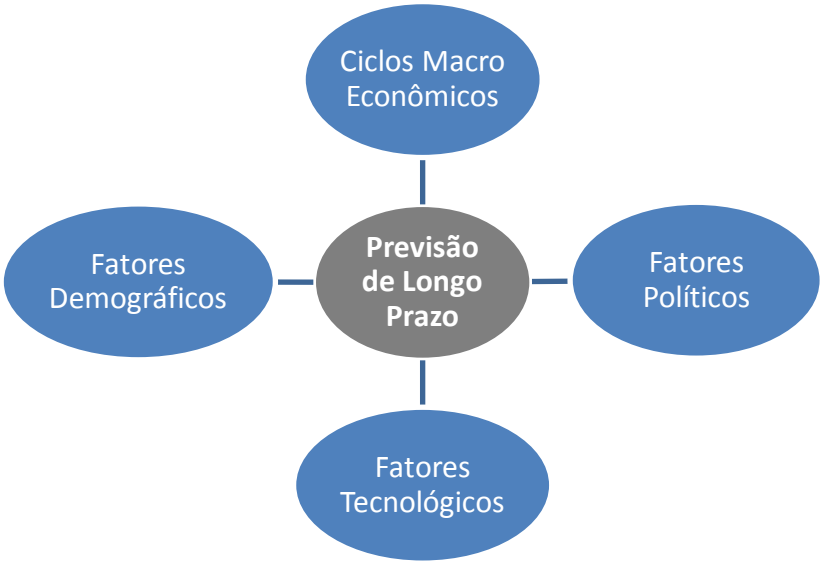


Figura 4 – Principais fatores que influenciam na previsão de longo prazo

- **Ciclos Macroeconômicos:** O comportamento de longo prazo do sistema elétrico é dependente principalmente dos fatores econômicos, entre eles a variação do PIB. Sua evolução reflete diretamente a atividade econômica em geral, e consequentemente, o consumo de energia. Quando o crescimento do PIB é sustentável, há maiores níveis de investimento e de consumo, resultando em cargas crescentes. Por outro lado, uma queda da atividade econômica conduz à redução da procura de bens e serviços e, por consequência, à redução do consumo de energia. Atualmente, a evolução do PIB de outros países também se tornou importante, na medida em que seus efeitos se propagam cada vez mais rapidamente devido a globalização econômica. Um bom exemplo é o da "Crise Imobiliária Americana" de 2008 que com maior ou menor intensidade, se alastrou pelo mundo todo.
- **Fatores Políticos:** A influência política na economia é evidente, mesmo em países ditos liberais. Decisões e resultados políticos guiam a evolução econômica dos países, impactando diretamente na atividade econômica e no consumo de energia. Idealmente, decisões políticas deveriam ser tomadas com cautela e de modo a não causar solavancos na economia. Porém, no dia a dia, mudanças políticas importantes causam grandes distúrbios nas expectativas e avaliações de risco da economia, algumas vezes resultando em grandes mudanças na carga.

- **Fatores Tecnológicos:** Mudanças tecnológicas podem influenciar positivamente ou negativamente a carga. Por exemplo, o aumento do uso de aparelhos de condicionamento de ar a gás pode trazer como resultado redução da carga elétrica do sistema. A adoção de novas tecnologias, muitas vezes por determinação de leis internacionais, envolve políticas destinadas a promover a eficiência energética. Políticas ambientais contra o efeito estufa e o aquecimento global levam a adoção de fontes alternativas de energia sustentável.
- **Fatores Demográficos:** O aumento da população, seja por crescimento vegetativo, ou devido a grandes fluxos migratórios, pode resultar em um aumento na demanda. Para uma única localidade, o impacto pode ser relativamente pequeno. Mas como o efeito é cumulativo, em toda uma região, ele pode tornar-se não negligenciável.

1.4 - Aplicações da Previsão de Carga

A previsão de carga sempre foi importante para o planejamento e a decisão operacional realizada pelas empresas. Com a desregulamentação do setor de energia elétrica, a previsão de carga ganhou ainda mais atenção [12].

Modelos precisos para previsão de carga são essenciais tanto para o planejamento como para a operação de uma empresa de energia. Coordenadas pelo ONS (Operador Nacional do Sistema Elétrico), as empresas apoiam-se na previsão para tomada de decisões importantes, como o despacho de máquinas de geração, remanejamento de cargas, programação de manutenção, entre outras [13].

Devido à flutuação da oferta e da demanda, e as mudanças nas condições climáticas, os preços da energia podem aumentar muito durante as situações de pico. Todos os participantes da CCEE (Câmara de Comercialização de Energia Elétrica), entre eles os consumidores livres e especiais, usam a previsão de carga para estimar os fluxos de carga e tomar decisões que evitem sobrecargas e minimizem os custos. Tais decisões levam à melhoria da confiabilidade da rede e redução das ocorrências de falhas de equipamentos e apagões. Além disto, a previsão permite o correto

escalonamento das manutenções nos diversos equipamentos do sistema, indicando o melhor momento para interrupção do funcionamento de equipamentos. [14]

A Figura 5 ilustra uma nuvem dos diversos agentes e órgãos que utilizam a previsão de carga no sistema elétrico brasileiro.

Figura 5 – Agentes que utilizam a previsão de cargas

As previsões de carga são extremamente importantes não só para os produtores, distribuidores e transmissores de energia, mas também para instituições financeiras, investidores e comercializadores. Ela pode ser usada nas avaliações de contratos e de vários outros produtos financeiros, que se baseiam em preços de energia oferecidos pelo mercado. Mesmo consumidores cativos utilizam uma estimativa de demanda para analisar seus custos futuros e decidir por modalidades de fornecimento mais vantajosas [15].

A agência reguladora ANEEL (Agência Nacional de Energia Elétrica), bem como outros órgãos governamentais como a EPE (Empresa de Pesquisas Energéticas) e o CMSE (Comitê de Monitoramento do Setor Elétrico) usam dados de previsão para analisar os investimentos em ampliação da infraestrutura do sistema. As decisões

sobre despesas de capital com base na previsão de longo prazo são a base para definição de taxas de retorno em diversos investimentos [16][17].

1.5 - Dificuldades para Previsão de Carga

Como já foi dito, a carga elétrica é uma grandeza que depende de uma série de fatores. De forma ideal, tentar prever seu comportamento envolveria a criação de modelos para cada um destes fatores. Infelizmente, este não é um caminho viável. Existem diversas barreiras neste caminho, começando pela disponibilidade de dados históricos de diversos fatores, bem como sua precisão, frequência, abrangência, confiabilidade e cobertura. [18]

Por exemplo, em muitas situações, o fator de maior influência na carga pode ser a temperatura, como no caso da previsão de curto-prazo. Porém, pode acontecer de os dados disponíveis serem apenas os valores máximos e mínimos diários. Estes valores não apresentam uma resolução suficiente para ajudar na previsão de carga de 15 em 15 minutos. Outras vezes, o histórico não é preciso, pois a medição foi feita com um instrumento descalibrado. Ou o dado pode não ser confiável, pois a leitura foi feita manualmente, por pessoal não capacitado, podendo até conter erros grosseiros. [19]

Mesmo quando a medição é feita de forma automática, não é raro a ocorrência de *GAPs* (buracos) nos dados, ou seja, intervalos onde a medição não foi feita, ou não foi gravada. Isso ocorre, por exemplo, por panes nos sensores, que podem travar ou ficar inoperante, até que uma equipe de manutenção seja acionada.

Mesmo quando o histórico existe, e é adequado e confiável, pode ser difícil utilizá-lo na previsão. Por exemplo, a umidade relativa do ar também pode ser um fator importante para a carga. Apesar de existirem bons históricos passados, para que este fator seja utilizado na previsão, é necessário que se conheça além de seus valores passados, também seus valores futuros. Prever o comportamento das variáveis exógenas é uma tarefa tão difícil, ou até mais, que a previsão da carga elétrica. Fato idêntico ocorre na previsão de longo prazo, quando se tenta prever como estará a economia nacional nos próximos três ou cinco anos. [20]

Assim, na maioria das vezes, assume-se que todos os fatores que influenciam a carga já estão representados no próprio valor da carga. O único histórico utilizado é o próprio histórico de carga.

Este princípio é semelhante à Teoria de Dow, o precursor da chamada Análise Técnica dos mercados financeiros. Charles H. Dow, no final do século XIX, ao refletir sobre o comportamento do mercado, chegou ao princípio conhecido como "Os índices já descontam tudo". Para ele, os índices como o Dow Jones e o Ibovespa, já refletem todo o consenso do mercado sobre o passado, o presente e o futuro. Supondo que o mercado seja eficiente, qualquer notícia será rapidamente incorporada aos movimentos dos preços, e os índices irão refletir todos estes movimentos. Embora o mercado não possa prever catástrofes naturais e outros acontecimentos exógenos, ele se ajusta rapidamente a eles, descontando estas ocorrências e incorporando seus efeitos sobre os preços dos ativos. Deste modo, não faz sentido realizar análises paralelas ao mercado, uma vez que a estimativa do mercado será sempre a melhor e mais eficiente. Essa premissa também é conhecida como hipótese do mercado eficiente. [21]

1.6 - Dimensões do *Big Data*

O *Big Data* é um novo paradigma de mineração de dados, que combina técnicas de análise e modelagem que exigem grande capacidade de armazenamento e poder de processamento. Os desafios do gerenciamento de dados com grande diversidade, sua integração, limpeza, indexação e consulta, deram origem a várias ferramentas de análise e mineração. Isso só foi possível graças aos avanços tecnológicos que permitiram uma computação cada vez mais rápida, com armazenamento em grande escala em redes mais rápidas e poderosas. Tudo isso a preços cada vez mais baixos. Por esta razão, as empresas buscam a crescente preservação e utilização de grandes volumes de dados no processo de tomada de decisão. [22]

O conceito de Big Data ganhou maior destaque em 2001, quando Doug Laney escreveu em um relatório de pesquisa [23] do MetaGroup (posteriormente adquirido pelo Gartner Group), a seguinte definição: "Big Data é um ativo de informação com

alto volume, alta velocidade e alta variedade, que exige formas inovadoras e econômicas de processamento para sua melhor compreensão e para tomada de decisões". Desta definição, surgiram os famosos 3Vs do Big Data: Volume, Velocidade e Variedade.

Posteriormente, pesquisadores da IBM [24] adicionaram novas dimensões à definição original: Valor e Veracidade. Assim, o termo Big Data passou a ser caracterizado por cinco dimensões, conhecidas como 5V's: Volume, Velocidade, Variedade, Veracidade e Valor, conforme representado na Figura 6.

Figura 6 – Dimensões do Big Data (5Vs)

A dimensão do Volume está relacionada com o crescimento exponencial da quantidade de dados disponível. O uso frequente de sensores e sua ligação em rede formam a realidade atual. No sistema elétrico, por exemplo, os *Smart Grids* (Redes Inteligentes) podem conter milhares de sensores, gerando dados on-line sobre diversas grandezas elétricas. [25]

A dimensão da Velocidade retrata diretamente o fluxo de dados dos sensores. Enquanto antes todas as análises eram feitas em lote, *a posteriori*, no Big Data, as análises são contínuas. Não há tempo ou possibilidade de acesso de todos os dados na

memória. Idealmente, os cálculos devem ser feitos *on-line*, à medida que os dados chegam pelos fluxos de dados (*Data Streams*). [26]

A dimensão da Variedade retrata as diferentes formas e fontes dos dados. Todas as fontes possíveis de dados devem colaborar para a geração de valor para a empresa. No sistema elétrico, além das grandezas elétricas, podem-se buscar dados em registros de atendimento, na posição e movimentação de viaturas de manutenção, na ocorrência de grandes eventos, e até mesmo na movimentação das redes sociais. [27]

A dimensão da Veracidade está relacionada com a incerteza ou imprecisão dos dados. É aqui que o *Big Data* mostra uma de suas forças. Ao se analisar uma grande massa de dados, é comum encontrar dados fora de faixa, dados com excesso de ruídos, erros de medição, lacunas nos históricos, ou dados inconsistentes. As técnicas de Big Data desenvolvidas para análise e processamento de fluxos de dados foram criadas para serem robustas, e conseguir lidar com todos estes obstáculos. [28]

Por fim, a dimensão do Valor é destacada como a mais importante. Isso indica que, em última instância, o objetivo de qualquer projeto de *Big Data* deve ser o de gerar valor e novas oportunidades de negócio para a empresa.

Para acomodar a rápida evolução tecnológica e a diversidade de aplicações das técnicas de Big Data, alguns autores propõem ainda outros Vs como:

- **Visualização**: formas fáceis de ler e entender os conjuntos de dados;
- **Volatilidade**: tempo de validade dos dados;
- **Viralidade**: velocidade de compartilhamento dos dados e resultados;
- **Variabilidade**: amplitude de mudanças nos fluxos de dados;
- **Viscosidade**: taxa de resistência a mudança nos fluxos de dados;
- **Vitalidade**: capacidade dos dados para continuar gerando valor no futuro;
- **Versatilidade**: aplicabilidade dos dados em diferentes formas e áreas;
- **Visão**: definição do problema específico a ser respondido pelos dados;
- **Viabilidade**: possibilidade de superar as dificuldades do projeto.

1.7 - Objetivo deste Trabalho

Este trabalho tem como objetivo aplicar e aperfeiçoar as técnicas de Big Data ao problema de previsão de carga elétrica. Estas técnicas permitem o desenvolvimento de modelos de análise preditiva para os diferentes horizontes de carga. Através da análise dos históricos, com a ajuda de estatísticas, mineração de dados, aprendizado de máquinas e modelos de florestas aleatórias, será possível gerar boas estimativas do comportamento futuro da carga, permitindo a empresa tomar diversas decisões importantes com boa antecedência, gerando valor e novas oportunidades de negócio.

1.8 - Organização deste Trabalho

Um projeto de análise e mineração de dados com Big Data obedece a uma estrutura de cinco passos, desde a definição do problema até a sua implantação. A Figura 7 resume as etapas do processo.

Figura 7 – Etapas de um projeto de mineração com Big Data

Embora seja usual seguir a ordem descrita, durante a realização do projeto, haverá iterações entre as diferentes etapas, o que pode exigir que o trabalho concluído em fases anteriores seja revisto. Por exemplo, pode ser necessário voltar a preparação de dados (passo 2) durante a etapa de análise exploratória dos dados (passo 3), a fim de efetuar modificações com base no que está sendo aprendido. Este trabalho foi elaborado obedecendo à sequência destes cinco passos.

O capítulo 1 apresenta uma introdução ao problema da previsão de carga no sistema elétrico, seus fatores de influência, os horizontes de previsão e os métodos já usados, bem como as principais dificuldades enfrentadas.

O capítulo 2 apresenta a metodologia CRISP, o padrão internacional para os processos de mineração de dados. Descreve como se planeja um projeto de mineração de dados, como se define o escopo do problema e os recursos necessários. Em

seguida, aborda a fase mais importante para o sucesso de um projeto de mineração: a preparação dos dados. Esta fase é responsável pela maioria dos fracassos, por ser longa, cara e tediosa. É uma atividade sempre subestimada, seja em sua importância, sua duração, seu custo ou sua complexidade. Por este motivo, são enumeradas todas as tarefas necessárias para correta execução desta fase, já que a maioria das dificuldades apresentadas nas etapas seguintes decorre de erros na preparação dos dados.

O capítulo 3 apresenta a base de dados utilizada neste trabalho. Esta fase corresponde a "Compreensão dos dados" da metodologia CRISP. Conhecer seus dados é imprescindível na era do *Big Data*. Aqui a correta preparação dos dados mostra seu primeiro efeito. São feitas análises de anomalias, *outliers*, histogramas e todas as transformações necessárias para a geração dos modelos de previsão. Em seguida, descreve-se a análise exploratória dos dados. Utilizando-se de visualizações gráficas, são analisadas tendências, sazonalidades e a influência de fatores externos, como horário de verão e os feriados.

O capítulo 4 inicia a fase de modelagem. Aqui são apresentados os conceitos de aprendizado de máquina, como conjuntos de treinamento, de validação, o binômio viés-variância, e o ajuste de parâmetros utilizando validação cruzada.

O capítulo 5 apresenta a teoria de funcionamento das árvores de decisão, que serão os componentes fundamentais do modelo de previsão. O conceito de *Stump* é apresentado, demonstrando-se como é feita a partição dos dados, e como é calculado o valor de corte.

O capítulo 6 introduz o conceito de fluxo de dados, e o novo paradigma de programação para métodos numéricos que lidam com grandes volumes de dados. São descritos alguns algoritmos que serão posteriormente utilizados no desenvolvimento do modelo preditivo, como por exemplo, a amostragem por reserva e com *bootstrap*.

O capítulo 7 introduz o conceito de Modelos Combinados, onde um conjunto de árvores de decisão é combinado para formar uma Floresta Aleatória. Em um breve histórico, são apresentadas as ideias que levaram à evolução dos modelos de árvore

até se chegar as florestas aleatórias. Neste capítulo, descreve-se o algoritmo de limiarização e o processo de treinamento de uma floresta aleatória. É neste ponto que fica clara a maior vantagem das florestas aleatórias: sua robustez em relação a uma base de dados incompleta, inconsistente, ou com ruído.

O capítulo 8 apresenta a aplicação das florestas aleatórias ao problema de previsão de carga. Utilizando os dados preparados e analisados nos primeiros capítulos, são feitas diversas simulações com o objetivo de investigar a viabilidade e a adequação dos modelos de floresta aleatória ao problema de previsão de carga.

Por fim, no capítulo 9 são apresentadas as conclusões e as recomendações para trabalhos futuros.

Capítulo 2
Mineração de Dados com Big Data

2.1 - Introdução

No mundo atual, uma quantidade sem precedentes de dados digitais está sendo gerada, a velocidades muito rápidas, e em diversas disciplinas. Empresas de varejo coletam dados sobre suas vendas, organizações analisam cliques feitos em seus sites, bancos gerenciam milhões de transações financeiras em todo o mundo. O setor elétrico não é uma exceção. Sistemas de monitoramento de alarmes, medidores inteligentes, centrais de atendimento ao cliente, negociações no mercado de energia, dados de reservatórios, disponibilidade de ventos, entre outros, geram grande massa de dados a ser armazenada e analisada [18].

É praticamente impossível tentar entender conjuntos de dados que contém mais do que algumas centenas de registros sem a ajuda de programas de computador. Traduzir os dados brutos coletados de diversas maneiras em informações úteis requer o uso de análise exploratória de dados, métodos de mineração de dados e aprendizagem de máquina. Todas estas técnicas tem o objetivo de transformar um grande volume de dados em informações compreensíveis, para desta forma, permitir tomar decisões oportunas e bem fundamentadas no futuro. Estas decisões podem ser utilizadas, por exemplo, para maximizar uso de recursos, cortar custos, segmentar produtos, bem como compreender as preferências dos consumidores.

O uso de sistemas automáticos, que empregam diversos tipos de sensores de monitoramento remoto, está produzindo dados em taxas cada vez mais rápidas. Essa é uma tendência que deverá continuar no futuro próximo. Os desafios da manipulação e da interpretação desses fluxos de dados são imensos, devido não só ao aumento do volume, mas também a complexidade dos diversos tipos de informações.

O processo de captação de dados brutos e sua conversão em informação e conhecimento formam o caminho para tomada de decisões inteligentes. A Figura 8 ilustra este processo, conhecido como KDD (*Knowledge Discovery in Databases*). [29]

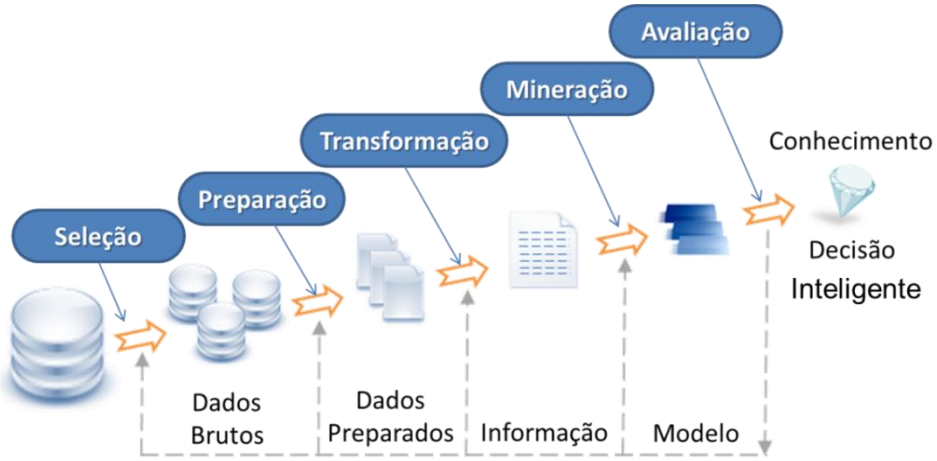


Figura 8 – Processo KDD – Transformando dados em conhecimento

2.2 - Metodologia CRISP-DM

A metodologia CRISP-DM (*Cross-Industry Process for Data Mining*) nasceu no final da década de 90, a partir da iniciativa de profissionais que trabalhavam com mineração de dados. Estes buscavam desenvolver um modelo de processo capaz de funcionar em qualquer tipo de indústria, e que descrevesse as fases e interações presentes nos projetos de mineração de dados. A Figura 9 mostra o processo CRISP-DM, que divide o processo de mineração de dados em seis fases principais. [30]

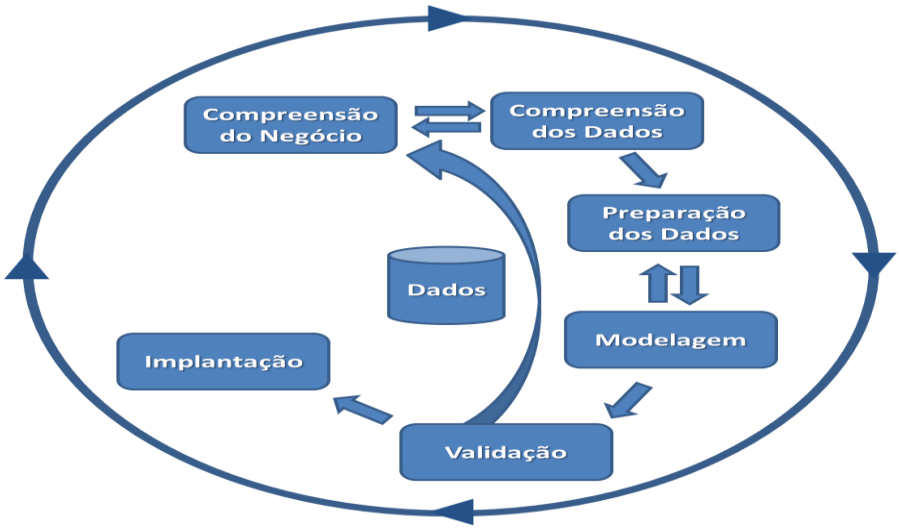


Figura 9 – Metodologia CRISP-DM

A sequência de fases não é unidirecional, ou seja, o processo pode mover-se para frente e para trás durante o projeto. As setas no diagrama de processo indicam as dependências mais importantes entre as fases. O círculo externo simboliza a natureza cíclica do processo. Um projeto de mineração de dados é composto de vários giros deste diagrama, ou seja, ele não se encerra após a implantação de uma solução. As lições aprendidas durante o projeto podem desencadear novas e muitas vezes mais focadas questões de negócios e então novos processos de mineração de dados se beneficiarão dos resultados anteriores.

Os dados, que ficam no centro do diagrama, são a peça fundamental do método. Eles servem de matéria prima para todo o processo de mineração. As seis fases da metodologia CRISP-DM são apresentadas a seguir.

2.2.1 - Compreensão do Negócio

Esta fase inicial se concentra na compreensão dos objetivos e requisitos do projeto a partir de uma perspectiva do negócio alvo da empresa. Este conhecimento será convertido em uma definição do problema em termos das tarefas de mineração de dados a serem executadas, bem como em um plano preliminar para atingir os objetivos.

2.2.2 - Compreensão dos Dados

A fase de compreensão dos dados começa com a coleta inicial dos dados e prossegue com atividades de familiarização com os dados. Uma das frases mais repetidas na área de mineração de dados é: “Conheça os seus dados!” (*Know your data*). Identificar problemas na qualidade dos dados, descobrir subconjuntos interessantes para formular hipóteses, descobrir informações ocultas e várias outras atividades são executadas nesta fase.

2.2.3 - Preparação dos Dados

A fase de preparação de dados cobre todas as atividades necessária para construir um conjunto de dados final, a partir dos dados brutos iniciais. Tarefas de preparação de dados podem ser realizadas várias vezes, conforme evolui a compreensão dos dados e do negócio. As tarefas incluem seleção de tabelas, registros e atributos, bem como transformação e limpeza de dados, que irão alimentar as

ferramentas de modelagem. A experiência mostra que esta fase é a que toma mais tempo em um projeto. Justamente por isso, é a responsável pela maioria dos fracassos dos projetos, pois tanto sua duração, seus custos e sua complexidade costumam ser subestimados.

2.2.4 - Modelagem

Nesta fase, várias técnicas de modelagem são selecionadas e aplicadas, e seus parâmetros são ajustados de acordo com algum critério de otimização. Normalmente, existem várias ferramentas possíveis para o mesmo tipo de problema de mineração de dados. Cada uma pode ter diferentes requisitos quanto à forma dos dados. Portanto, muitas vezes, após escolhida a técnica mais adequada, pode ser necessário voltar à fase de preparação de dados.

2.2.5 - Validação

Nesta fase do projeto, o modelo (ou modelos) que apresentou melhores resultados é avaliado da perspectiva do objetivo do trabalho. Antes de prosseguir para a implantação final do modelo, é importante avaliar e revisar as etapas executadas em sua construção, para ter certeza de que ele atinge adequadamente os objetivos do negócio. Um dos principais alvos é determinar se existe alguma questão comercial importante que não tenha sido considerada suficientemente. Deve-se garantir que todos os passos dados durante a criação do modelo podem ser replicados em campo, durante a sua utilização. No final desta fase, o uso dos resultados obtidos até aqui devem ser discutidos. Conforme o caso, novas dúvidas e perguntas podem ser identificadas, tornando necessária uma nova volta no diagrama, reiniciando o processo.

2.2.6 - Implantação

A obtenção do modelo geralmente não é o fim do projeto. Mesmo se o propósito do modelo for simplesmente aumentar o conhecimento sobre os dados, o conhecimento adquirido terá de ser organizado e apresentado de uma forma que seja útil para a empresa. Dependendo dos requisitos, a fase de implantação pode ser tão simples quanto gerar um relatório ou tão complexa quanto instalar um fluxo *on-line* de dados entre o negócio e o modelo, inserindo-o dentro do processo de tomada de

decisões da empresa. Para isso, a empresa deve dimensionar todas as ações necessárias para realmente fazer uso dos modelos criados.

2.3 - Planejamento do Processo

A primeira etapa de um projeto de Big Data é descrever o problema a ser abordado e gerar um plano. Uma série de questões deve ser considerada nesta primeira fase. A Figura 10 apresenta algumas dessas principais questões. [31]

Definição do Problema e Planejamento 	Identificar o problema ou necessidades
	Listar os objetivos a serem atingidos
	Gerar os fatores de sucesso
	Entender cada recurso e identificar suas limitações
	Montar uma equipe adequada
	Criar um plano de trabalho
	Fazer uma análise de custos e benefícios

Figura 10 – Planejamento de um projeto de Big Data

É importante documentar claramente o problema a ser resolvido junto com outras informações de contexto que sejam relevantes. Em determinadas situações, no entanto, pode não ser possível, ou mesmo desejável, saber precisamente o tipo de informações que serão geradas a partir do projeto. Em projetos mais abertos, muitas vezes, surgem novas perguntas ao explorar grandes bases de dados. Mas, mesmo nesses casos, uma identificação geral do problema ajuda a limitar e concentrar o esforço.

Além de compreender que tipo de informação será gerado, é útil também saber como os resultados serão entregues, quais objetivos devem ser perseguidos, e quais produtos serão gerados. Um produto gerado pelo projeto pode ser um relatório, um programa de computador a ser utilizado para fazer previsões, ou um conjunto de regras de negócio. A definição destes produtos irá definir as expectativas daqueles que trabalham no projeto e das demais partes interessadas.

Os critérios de sucesso relacionados ao objetivo do projeto devem ser definidos de maneira que possam ser medidos. Por exemplo, um critério pode ser “aumentar as

receitas em 10% nos próximos três meses". Um critério de sucesso nada mais é que uma meta a ser atingida. A definição de meta envolve uma unidade de medida (o custo, por exemplo), um valor (aumentar em 10%) e um prazo (em três meses).

Os recursos disponíveis para um projeto normalmente têm uma série de restrições. Tais restrições influenciam a forma como o projeto avança. Pode haver restrições ou limitações desde os dados disponíveis, bem como o hardware computacional ou software a serem usados. Questões relativas à privacidade ou confidencialidade dos dados devem ser identificadas e documentadas. Por exemplo, pode haver limitações relativas à quantidade de tempo disponível para um algoritmo computacional fazer uma previsão. Algumas vezes, torna-se necessário trocar ou balancear precisão e tempo de resposta. Devem-se buscar algoritmos precisos, mas que gerem respostas em tempo hábil, para que possam ser utilizados.

Equipes interdisciplinares podem resolver problemas complexos, olhando para os dados a partir de diferentes perspectivas. Devido a gama de conhecimentos necessários, as equipes são essenciais, especialmente para projetos de grande escala. É útil considerar os diferentes papéis necessários para uma equipe interdisciplinar. Um líder de projeto dirige o projeto e monitora seus resultados. Especialistas de domínio fornecem conhecimento específico do assunto. Especialistas em análise e mineração de dados são os que estão familiarizados com as estatísticas, os métodos de análise e algoritmos de mineração. Um especialista em TI deve ter experiência na integração de diferentes bases de dados.

A elaboração do plano de trabalho depende do tamanho e escopo do projeto. O progresso é feito de forma iterativa e não estritamente sequencial, movendo-se entre as fases à medida que surgem novas questões. Um calendário de eventos deve ser elaborado, incluindo as fases de preparação, análise exploratória, modelagem e validação. Tarefas com dependências e contingências devem ser documentadas usando ferramentas de apoio à gestão de projetos, como gráficos de Gantt ou diagramas PERT.

2.4 - Extração dos Dados

Antes de iniciar um projeto de análise e mineração de dados, os dados devem ser recolhidos, caracterizados, limpos, transformados e particionados em uma forma adequada para o processamento. A experiência mostra que esta fase é a fase que toma mais tempo em um projeto. Porém, ela é frequentemente subestimada, levando ao atraso ou mesmo a falha total do projeto. Em muitos projetos, os dados são geralmente integrados de várias fontes, com diferentes representações e formatos. A Figura 11 ilustra alguns dos passos necessários para a preparação de um conjunto de dados. [31]

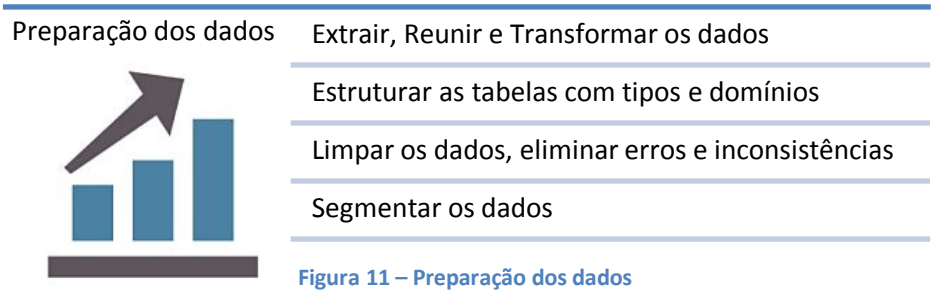


Figura 11 – Preparação dos dados

Em situações em que os dados foram recolhidos para uma finalidade diferente da finalidade do projeto, os dados terão de ser transformados em uma forma adequada para análise. Por exemplo, os dados podem estar na forma de uma série de documentos em diferentes formatos, o que requer que sejam extraídos a partir do texto de cada documento, e convertidos para uma forma tabular que é mais favorável para a análise. Os dados devem ser preparados para espelhar o mais próximo possível a população alvo, sobre as quais novas perguntas serão feitas. Como podem ser usadas várias fontes de dados diferentes, é preciso ter cuidado para não introduzir erros durante o processo de junção destas fontes. Reter informações sobre a fonte é útil tanto para a rastreabilidade dos dados como para a interpretação dos resultados.

É importante determinar e documentar o tipo de cada um dos atributos que foram recolhidos ao longo dos diferentes itens do conjunto de dados. Por exemplo, verificar quais atributos representam categorias discretas, como “classe de tensão” ou “tipo de consumidor”; e quais são valores numéricos contínuos como “temperatura” ou “corrente”. Esta categorização ajuda a identificar valores inesperados.

Além de identificar erros ou inconsistências nas bases de dados, pode ser importante transformar os dados para uma forma mais favorável à análise. Por exemplo, em alguns métodos de mineração, é necessário que todos os atributos tenham uma mesma faixa de variação (por exemplo, dados normalizados ou em p.u.). Isso é conhecido como normalização de valores, e evita que atributos em uma escala menor se sobreponham a outro representado em uma unidade de escala maior. Pode também ser necessário dividir os dados em subconjuntos, ou filtrá-los com base em algum critério específico, para torná-los susceptíveis a responder os problemas delineados no início do projeto.

A preparação dos dados envolve várias tarefas, desde a montagem das tabelas, a determinação dos tipos e domínios dos atributos, o tratamento de valores em branco, a identificação de atributos categóricos e a limpeza dos dados.

2.5 - Montagem das Tabelas de Dados

O ponto de partida para uma análise de dados é uma tabela de dados, muitas vezes referida como um conjunto de dados, que contém valores medidos ou recolhidos, representados por números e texto. Antes de serem transformados ou modificados, os dados dessas tabelas são chamados dados brutos. Uma tabela de dados lista diferentes propriedades, ou características, das entidades ou eventos, das quais os dados foram recolhidos ou medidos. Nestes quadros, a informação considerada interessante é mostrada através de diferentes atributos. [32]

Os itens individuais geralmente são mostrados como linhas da tabela, enquanto os diferentes atributos representam as colunas. Por exemplo, na indústria automotiva, uma possível entidade seriam os carros. Cada linha representa uma observação, um item, um caso, ou uma instância; e contém informações sobre o item específico representado nesta linha. Uma tabela de carros pode conter muitas informações sobre cada carro, por exemplo: o peso do carro, o número de cilindros, a capacidade do tanque, e assim por diante. Cada atributo, ou variável, é pensado como um conjunto de valores que descrevem algum aspecto presente em todas as observações. Assim,

cada linha da tabela descreve uma instância, um carro específico, e cada coluna descreve uma variável, um atributo específico de um carro. [33]

No caso da previsão de cargas, cada linha da tabela equivale a um registro de medição. Os atributos podem ser quaisquer dos fatores que influenciam a carga. Eles vão desde os atributos de data e hora (como dia da semana, horário, feriados, número de dias úteis no mês, entre outros) até atributos ligados ao próprio sistema elétrico (como patamar de carga, nível de tensão tipo de consumidor, região de fornecimento, etc). A Figura 12 mostra o formato geral de um conjunto de dados para um projeto típico de análise e mineração de dados.[34]

		Variáveis				
		x_1	x_2	x_3	...	x_p
Observações	o_1	x_{11}	x_{12}	x_{13}	...	x_{1p}
	o_2	x_{21}	x_{22}	x_{23}	...	x_{2p}
	o_3	x_{31}	x_{32}	x_{33}	...	x_{3p}
	...	...	...	...	...	...
	o_n	x_{n1}	x_{n2}	x_{n3}	...	x_{np}

Figura 12 – Formato geral de uma tabela de dados

Uma tabela de dados descreve uma série de observações, de O_1 até O_n, onde cada observação é descrita usando uma série de variáveis (de X_1 até X_P). Um valor é fornecido para cada variável de cada observação. Por exemplo, o valor da primeira variável da primeira observação é x_{11}, o valor para a primeira variável da segunda observação é x_{21}, e assim por diante.

Este tipo de visualização, em forma de tabela ou planilha, é útil para uma primeira análise e apresentação dos dados brutos. No entanto, quando esta contém mais do que algumas centenas de observações ou variáveis, sua análise e visualização tornam-se difíceis. Ordenar a tabela com base em uma ou mais variáveis é útil para organizar os dados. No entanto, é difícil identificar tendências, ou relações, simplesmente olhando para os valores brutos sozinhos.

2.6 - Tipos de Atributos

Muitas técnicas de análise de dados têm restrições quanto aos tipos de atributos que elas podem processar. Assim, conhecer os tipos dos atributos permite escolher ou eliminar certas técnicas, ou mesmo determinar se os dados precisam ser transformados em uma forma mais adequada para análise. [35]

O tipo dos atributos está relacionado à forma com que cada variável é armazenada e representada na memória do computador. Os tipos básicos mais simples são os tipos Numéricos e os tipos Texto. Os tipos numéricos podem ser subdivididos em tipos inteiros e tipos de ponto flutuante.

Devido a erros de arredondamento, e a forma com que tipos com ponto flutuante são armazenados na memória do computador, pode haver discrepância entre os valores teóricos e os valores obtidos em cálculos feitos pelo computador, principalmente em rotinas iterativas, onde tais erros tem efeito cumulativo. Para tentar minimizar estes efeitos, principalmente quando a variável envolvida representa um valor em dinheiro, alguns sistemas mais modernos usam uma representação chamada Inteiro Decimal. Nesses casos, os valores são armazenados na memória como inteiros, e são calculados com operações de inteiros. Porém, quando apresentados ao usuário, utiliza-se o ponto decimal, deslocado um determinado número de casas, transformando o inteiro em um número decimal. Este tipo de variável possui duas características: sua escala (quantas casas decimais serão deslocadas) e sua precisão (quantos bits são utilizados para seu armazenamento).

Além dos tipos numéricos básicos, alguns sistemas modernos possuem vários outros tipos de dados derivados dos tipos numéricos. Por exemplo, o tipo Data é utilizado para representar os dias de um determinado calendário. O tipo Data normalmente é tratado internamente como um número inteiro, representando a quantidade de dias passados desde uma data inicial, como por exemplo, o dia 1-janeiro-1900, ou mesmo o dia 01-janeiro-0001. Outros sistemas possuem também o tipo Hora, para representar um instante do dia (com precisão de segundos ou frações de segundos), ou o tipo Data-Hora, que combina os dois tipos anteriores. Tipos mais

sofisticados incluem também o fuso horário onde o valor foi coletado, possibilitando a análise de dados globais. Devido a sua grande utilização, os tipos Data-Hora possuem operações específicas para realização de cálculos. Não se deve subestimar a complexidade dessas operações aparentemente simples. Por exemplo, adicionar "um mês" a uma data, pode significar, conforme o mês, somar 29, 30 ou 31 dias. Determinar o dia da semana de uma data é uma tarefa que deve considerar a ocorrência de anos bissextos.

A Tabela 1 mostra os tipos mais comuns nos bancos de dados atuais. [36]

Tipos de Dados	Descrição
BOOLEAN	Valor lógico, Verdadeiro ou Falso
SMALLINT	Valor inteiro (16 bits);
INTEGER	Valor inteiro (32 bits)
BIGINT	Valor inteiro (64 bits)
DECIMAL(p,s)	Valor inteiro, com precisão (p) e escala (s). Por exemplo: decimal(5,2) é um número formado por 5 dígitos totais, sendo 2 após a vírgula
DATE	Valores de dia-mês-ano
TIME	Valores de hora:minuto:segundo
DATETIME	Valores de dia-mês-ano hora:minuto:segundo
CHARACTER(n)	Cadeia de caracteres de tamanho fixo n
VARCHAR(n)	Cadeia de caracteres de tamanho variável, com no máximo n caracteres
VARBINARY(n)	Cadeia de dados binários de tamanho variável, com no máximo n caracteres
REAL	Valor de ponto flutuante, aproximado, com mantissa 8
DOUBLE	Valor de ponto flutuante, aproximado, com mantissa 16

Tabela 1 – Tipos de dados mais comuns

2.7 - Domínio dos Atributos

O domínio de um atributo é o conjunto de todos os valores que um atributo pode possuir. É um conceito mais restritivo que o conceito de tipo. Por exemplo, uma variável "idade" pode ser representada por um tipo inteiro, mas possuir uma restrição

de domínio, impondo que seus valores estejam sempre dentro do intervalo de zero a cem. Os domínios se dividem em domínios discretos e contínuos. [35]

Os domínios contínuos são associados aos valores numéricos com ponto flutuante, onde teoricamente, existe uma infinidade de valores possíveis, como por exemplo, o valor da corrente elétrica em um cabo. Um domínio discreto possui um conjunto finito de valores possíveis. Por exemplo, uma variável "tipo de equipamento" poderia possuir um domínio com apenas algumas opções como: "transformador", "disjuntor", "chave seccionadora", "relê de proteção" e "banco de capacitores". Todos os tipos inteiros são considerados tipos com domínio discreto.

As variáveis do tipo texto, algumas vezes, são consideradas como se tivessem um domínio contínuo. Por exemplo, uma variável "nome", possui como domínio todos os nomes de pessoas possíveis. Outras vezes, uma variável do tipo texto pode ter um domínio bem restrito, como no caso de uma variável que só pode assumir um dos três valores: "Em Estoque", "Em Teste" ou "Em Operação". Existem ainda casos limite, onde é difícil estabelecer exatamente o tipo de domínio, como por exemplo, uma variável "Rua". Apesar de ser possível listar todos os nomes de "ruas" de uma certa cidade, esta lista pode ser grande demais, ou estar sujeita a futuras modificações e ocorrência de duplicidades.

É importante reconhecer, compatibilizar e documentar o domínio de todos os atributos utilizados em um projeto de mineração de dados.

2.8 - Valores em Branco ou Nulos

Em sistemas de bancos de dados modernos, os atributos de uma tabela podem assumir um valor especial para representar a ausência de valor. É conhecido como valor "nulo", "em branco", "faltante", "não disponível" (N/D) ou "não aplicável" (N/A). Essa capacidade aumenta em muito o poder de representação dos sistemas, pois, na prática, são inúmeras as razões e situações que requerem este valor. Em alguns sistemas antigos, utilizava-se o chamado "número mágico" para indicar um valor nulo. Por exemplo, quando uma variável "tempo de operação" não era conhecida, utilizava-se o número "-1" (um negativo). Porém, devido à inexistência de uma padronização na

escolha destes “números mágicos”, os algoritmos tornavam-se desnecessariamente confusos, apenas para tratar alguns poucos casos especiais.

Os valores nulos podem inclusive ser utilizados nas variáveis lógicas. Assim, além de verdadeiro ou falso, uma variável lógica pode assumir também o valor “nulo”. Isso faz com que a lógica associada às operações destas variáveis seja uma lógica de três estados, um pouco diferente da lógica booleana tradicional. Por exemplo, na lógica de três estados, a união dos registros onde o atributo “potência” é maior ou igual a 1 pu; unido com o conjunto de dados com “potência” menor que 1 pu, não resultará no conjunto total dos dados. Isso ocorre porque um registro com “potência” nula, ou em branco, não será selecionado, pois seu valor não é nem maior igual a 1pu, nem menor que 1pu.

A tolerância à presença de valores nulos no conjunto de dados reflete a robustez de um método de mineração. Por exemplo, como reagirá um modelo de previsão que usa uma função matemática para calcular o seu valor de saída, quando uma de suas variáveis de entrada estiver em branco? É aqui que muitos métodos computacionais sofisticados simplesmente param, travam, ou geram valores absurdos. Nestes casos, uma solução possível é passar os dados primeiramente por um estimador de estados, que irá atribuir valores a todos os atributos faltantes. Outra opção é simplesmente eliminar todos os registros que contenham valores nulos. Infelizmente, ambas as soluções trazem mais complexidade, custos e desvantagens ao projeto.

2.9 - Variáveis Ordenáveis e Categóricas

De um modo geral, todos os tipos de atributos podem ser ordenados em ordem crescente ou decrescente. Porém, nem sempre esta ordem representa a correta interpretação desejada dos dados. Por exemplo, um atributo “Situação” pode inicialmente possuir como domínio três valores: “Em Estoque”, “Em Teste” ou “Em Operação”. Caso este atributo seja armazenado como um tipo texto, sua ordem será a ordem alfabética dos valores. Assim, “Em Estoque” vem em primeiro lugar, seguido de “Em Operação” e depois “Em Teste”. Obviamente, esta não é a ordem natural que se

esperaria deste atributo. Para evitar este tipo de problema, escolhe-se um código para cada valor do domínio, de modo a restabelecer a relação de ordem desejada. No exemplo anterior, “Em Estoque” receberia o código “1”, “Em Testes” o código “2” e finalmente “Em Operação” o código “3”. A tabela de correspondência é chamada de Dicionário de Dados, e deve ser bem documentada em qualquer projeto de mineração de dados.

O uso do dicionário de dados contribui para diminuir radicalmente o espaço de armazenamento necessário para guardar os dados. A substituição de campos texto por campos numéricos inteiros pode reduzir o tamanho do arquivo de dados em dezenas de vezes. Além deste ganho, outra vantagem desta técnica é que, durante a substituição, muitas vezes são localizadas ocorrência de múltiplas formas textuais para um mesmo valor. Por exemplo, os valores textuais “Em Operação”, “operando” ou “Op.” podem estar presentes em um atributo indicando sempre o mesmo valor. Dependendo das configurações de um sistema, a mera troca entre letras maiúsculas e minúsculas poderia levar o computador a tratar um mesmo texto como dois valores diferentes.

Porém, algumas vezes, a técnica do dicionário de dados não é tão simples quanto se imagina. Voltando ao caso do atributo “Situação”, outros valores poderiam ser inseridos no domínio, como por exemplo: “Em Manutenção”. Agora um dicionário de dados não consegue mais resolver o problema de ordenação. A estrutura interna dos valores do domínio não pode mais ser representada por um conjunto totalmente ordenável. Agora, o conjunto é apenas “parcialmente ordenável”, representado por um diagrama de Hasse, como na Figura 13.

Figura 13 – Exemplo de diagrama de Hasse

Nestes casos, o dicionário de dados pode associar a cada valor do domínio, além de seu código, também um valor de ordem, mas agora não necessariamente único. Este valor poderá ser usado nas situações que necessitem uma comparação direta entre dois valores do domínio.

Além dos domínios com estrutura totalmente ordenável e parcialmente ordenável, existem os casos onde não há nenhuma relação de ordem entre os valores. Por exemplo, o atributo “tipo de equipamento” citado anteriormente, com os valores de domínio: “transformador”, “disjuntor”, “chave seccionadora”, “relê de proteção” e “banco de capacitores”. Apesar de ser possível atribuir um código para cada um dos valores, o código não deve ser usado como relação de ordem. Este tipo de atributo é chamado Atributo Categórico ou Atributo de Classe. Os atributos categóricos tem um tratamento especial em praticamente todos os métodos de mineração de dados. Sua distinção é tamanha que sua presença (ou não) é o fator que dá nome aos tipos ou subtipos de algoritmos, ou leva a rotinas e procedimentos distintos em sua execução.

Vários métodos limitam a quantidade de valores diferentes no domínio de um atributo categórico. Os limites mais comuns são 8, 16 ou 32 valores. Quando há uma quantidade maior que o limite, muitos sistemas passam a tratar o atributo como um tipo inteiro ordenável. Porém, algumas vezes, este comportamento pode levar a perda de qualidade dos resultados, ou mesmo a erros de interpretação.

2.10 - Chaves Primárias e Índices Únicos

Certos tipos de atributos não são utilizados diretamente na análise de dados, mas podem ser úteis durante a preparação de tabelas e para interpretação dos resultados. O mais comum deles é o atributo utilizado para identificar cada um dos casos ou linhas de uma tabela. Este atributo terá um valor diferente e único que identifica cada registro. Ele é chamado de chave primária, ou pelo apelido de “ID”.

Por exemplo, em uma tabela de dados de consumidores, cada consumidor tem um número de referência. Este atributo não é usado diretamente na elaboração de modelos preditivos, uma vez que seus valores são únicos. Porém, um produto esperado de um projeto de mineração de dados pode ser identificar um subconjunto

de consumidores que sejam responsáveis por um volume significativo de ocorrências na rede. Ao incluir este identificador exclusivo na tabela, possibilitamos resgatar posteriormente informações mais detalhadas destes consumidores.

Em alguns casos, o identificador único pode ser formado pela junção de vários atributos de interesse. Por exemplo, o identificador dos consumidores em uma distribuidora de energia, pode incluir em seus dígitos, uma letra que indique que o consumidor é uma pessoa física ou jurídica, ou indicar sua classe de tensão. Nesses casos, deve-se desmembrar esta informação, criando-se novos atributos que possam ser úteis na modelagem dos dados.

2.11 - Limpeza da Base de Dados

O processo de limpeza inicial de um banco de dados é chamado de Sanitização de Dados. Tanto para variáveis ordinais, como para variáveis categóricas, é útil inspecionar todos os valores presentes em um atributo para descobrir eventuais erros ou inconsistências [37][38]. Os principais casos são:

2.11.1 - Unicidade de Strings

Alguns campos texto em uma tabela podem apresentar vários valores diferentes para uma única instância. Por exemplo, o nome de uma rua pode ter várias grafias diferentes como "Avenida JK", "Avenida Juscelino Kubstcheck" ou "Av. J.K.”. Quando estes valores se referem à mesma rua, os vários termos devem ser consolidados em um só. Experiência sobre o domínio específico do problema pode ser necessária para localizar, corrigir e harmonizar essas variáveis.

2.11.2 - Erros de Conversão, Overflow e Perda de Precisão

Um problema comum com atributos numéricos é a inclusão de termos não numéricos ou fora de faixa. Por exemplo, em uma das bases de dados pode aparecer em um atributo supostamente numérico, o valor textual "mais de 50". Durante uma carga de dados, este registro gerará um erro de conversão, pois o valor não pode ser convertido em um número. Os métodos numéricos não podem interpretar valores não-numéricos e, portanto, estes termos devem ser convertidos em um número, deixados em branco ou mesmo removidos dos dados. Outro caso ocorre quando um

número inteiro é maior que a capacidade do tipo escolhido. É o chamado Overflow. Ou também quando um número real é carregado, mas somente com suas casas decimais mais significativas, levando a erros de perda de precisão.

2.11.3 - Dados em Branco e Ausência de Dados Relacionais

Outro problema comum na preparação de dados surge quando observações para um determinado atributo estão faltando. Em vários bancos de dados atuais, há a possibilidade de se atribuir um significado específico para um valor em branco. Algumas vezes, estes valores podem ser substituídos ou estimados com base no conhecimento de um especialista ou com base em outros atributos, ou no conhecimento de como os dados foram coletados. Durante a limpeza de dados, quando ocorrem alguns dos outros erros aqui listados, pode-se optar em manter o registro e substituir apenas o valor do atributo problemático por um campo em branco. Esta solução depende da possibilidade do banco de dados e do algoritmo de analise conseguir armazenar e tratar valores chamados “nulos”.

Em sistemas mais sofisticados, podem existir mais de um tipo de valor nulo. Alguns sistemas diferenciam N/A (não se aplica) de N/C (não consta), apesar de ambos serem considerados valores nulos. Um valor N/C significa um atributo válido, mas que tem seu valor desconhecido. Por exemplo, um atributo “potência” de um transformado pode ter seu valor como N/C, caso a placa do equipamento tenha sido removida. O valor existe, mas no momento não é conhecido. Já um valor N/A (não se aplica) representa um atributo que, por algum motivo, não pode ser atribuído aquele registro. Por exemplo, uma chave seccionadora poderia ter N/A como valor do atributo “tensão secundária”.

2.11.4 - Valores Atípicos e *Outliers*

Valores nos pontos extremos da faixa esperada são conhecidos como *Outliers*. Normalmente em pequeno número, estes valores diferem significativamente do resto dos dados. Há muitas razões para valores atípicos. Por exemplo, um *outlier* pode indicar um erro de medição, mas também pode se revelar um evento raro, ainda desconhecido, imprevisto, e portanto, um ponto de dados legítimo e bastante valioso. Uma simples ordenação dos dados, ou a elaboração de um histograma, pode

identificar alguns *outliers*. Uma técnica mais robusta é escolher um intervalo de variação válido, baseado na distribuição do atributo, usando um intervalo de confiança pré-determinado. Por exemplo, pode-se considerar valores fora do intervalo de 3 ou 5 desvios padrões como *outliers*.

Capítulo 3
Compreensão dos Dados

3.1 - Introdução

As primeiras fases de um projeto de mineração de dados com *Big Data* envolvem a definição do problema, a aquisição, compreensão, transformação e preparação da base de dados. O passo seguinte é a análise exploratória dos dados. Todas estas fases têm como objetivo conhecer e familiarizar-se com a massa de dados em estudo.

Um exame inicial dos dados é importante para entender o tipo de informação que foi recolhida e o significado dos dados. A Figura 14 representa o processo chamado *ETL - Extract Transform Load* (extração, transformação e carga dos dados).[39]

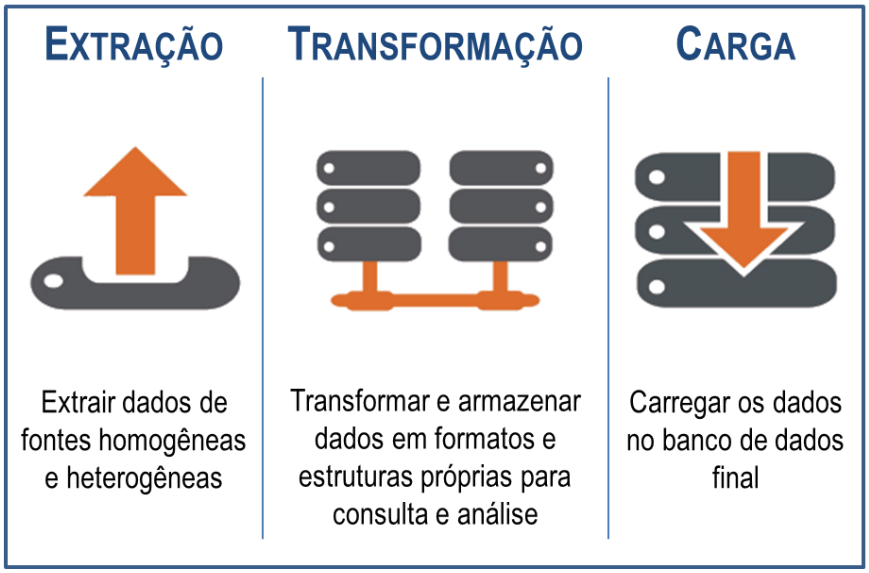


Figura 14 – Extração, transformação e carga dos dados

Esta etapa envolve desde a identificação da estrutura original dos dados, os tipos dos campos, transformações, análise de índices, distribuição de valores até a limpeza e identificação de valores extremos. Tudo isso é feito através de um conjunto de comandos na linguagem SQL - *Structured Query Language* (Linguagem Estruturada

de Consulta), a linguagem padrão para Sistemas Gerenciadores de Bancos de Dados (SGBD). [40]

A análise exploratória dos dados visa criar intimidade com os dados disponíveis, reconhecendo características marcantes dos dados, tendências, sazonalidades, médias e dispersões. As principais ferramentas para análise exploratória são técnicas estatísticas e a visualização de gráficos.

Ao final desta etapa, os dados estarão prontos para geração dos modelos de previsão. Para modelagem, serão escolhidos algoritmos e técnicas que se adequem a esta estrutura final dos dados.

3.2 - Estrutura Inicial da Base de Dados

A base de dados inicial utilizada neste trabalho pertence à Companhia Energética de Minas Gerais S.A. – CEMIG - uma das principais concessionárias de energia elétrica do Brasil. Os dados apresentam-se inicialmente em uma única tabela, chamada *CurvaCEMIG*, contendo 292.061 registros, com apenas duas colunas: *DataHoraCurva* e *ValCurva*

O campo *DataHoraCurva* é do tipo Data-Hora, contendo valores de 15 em 15 minutos, começando às 00:00hs de cada dia, totalizando 96 registros por dia. Os valores vão de 01 de janeiro de 1998 até 02 de maio de 2006 23:45. Não existem valores nulos neste campo.

O campo *ValCurva* é do tipo numérico, com dupla precisão, contendo o valor da carga global da empresa, em MW. Os valores vão de 1916 até 7226. Neste atributo existiam 4 valores nulos e 936 valores zerados. A Tabela 2 resume a estrutura inicial dos dados.

Tabela: *CurvaCEMIG*					**292.061 registros**
Campos	**Tipo**	**Mínimo**	**Máximo**	**Zeros**	**Nulos**
DataHoraCurva	Data-Hora	01/01/1998 00:00	02/05/2006 23:45	Nenhum	Nenhum
ValCurva	Double	1916	7226	936	4

Tabela 2 – Estrutura inicial da base de dados

3.2.1 - Transformações de Tipo de Variáveis

O campo *ValCurva* inicialmente era do tipo numérico, de dupla precisão. Porém, os valores armazenados não possuíam casas decimais. Assim, optou-se por convertê-los para valores inteiros, com o objetivo de acelerar rotinas de cálculos, facilitar a exibição de resultados e principalmente reduzir o espaço de armazenamento necessário. Isso pode ser feito com o seguinte comando na linguagem SQL:

```
ALTER TABLE CurvaCEMIG ALTER COLUMN ValCurva INT;
```

3.2.2 - Análise de Índices e Chaves Primárias

A tabela original não apresentava nenhum campo de chave primária e nenhum índice. Foi então aplicada uma transformação para criação de um índice único (sem repetição) no campo *DataHoraCurva*, com o objetivo de verificar a existência de valores duplicados. A criação do índice obteve sucesso, não havendo nenhum valor duplicado na base de dados.

```
CREATE UNIQUE INDEX idxDataHoraCurva ON CurvaCEMIG (DataHoraCurva);
```

Optou-se também por criar um campo ID para ser o identificador dos registros e a chave primária da tabela. Como a periodicidade de 15 minutos é fixa nos dados, escolheu-se usar um campo de número inteiro, que marcasse a quantidade de períodos de 15 minutos, desde o instante inicial de 01 de janeiro de 1998.

```
ALTER TABLE CurvaCEMIG ADD ID INT;
UPDATE CurvaCEMIG SET ID = DateDiff("n",#1/1/1998#,[DatHoraCurva])/15;
ALTER TABLE CurvaCEMIG ADD PRIMARY KEY (ID);
```

3.2.3 - Análise de Anomalias

A análise de anomalias visa identificar e corrigir registros que apresentem algum tipo de inconsistências. Os casos mais comuns são registros com valores nulos, zerados ou faltosos.

O campo *DataHoraCurva* não apresentava dados nulos ou zerados. Porém, como a tabela possuía 292.061 registros, em intervalos de 15 em 15 minutos,

iniciando-se em 01 de janeiro de 1998, seu registro final deveria ser 01/05/2006 06:15:00. Como o maior valor encontrado foi de 02/05/2006 23:45, concluiu-se que haviam 163 dados faltosos. Optou-se por encontrar estes instantes faltosos e inseri-los na base de dados, com valores de carga nulos.

O campo *ValCurva* apresentava 4 valores nulos e 936 zerados. Após a inserção dos registros faltosos, ele ficou com a distribuição da Tabela 3.

Anomalias Campo *ValCurva*	
Valores Faltosos	163
Valores Nulos	4
Valores Zerados	936
Total de Anomalias:	**1103**

Tabela 3 – Análise de anomalias do campo *ValCurva*

Optou-se por transformar e padronizar todos estes casos simplesmente como valores nulos, usando o seguinte comando:

```
UPDATE CurvaCEMIG SET ValCurva = Null WHERE ValCurva = 0;
```

3.2.4 - Análise Descritiva

A análise descritiva dos dados visa determinar a distribuição de valores, calcular tendências centrais e dispersão, com o objetivo de localizar e eliminar valores extremos (*outliers*). A técnica mais comum é chamada de Seis Sigma (6σ). Ela se baseia na suposição que os dados apresentem uma distribuição normal. Nestas condições, aproximadamente 99,7% dos dados devem estar localizados no intervalo de três desvios padrões antes e depois da média, totalizando uma faixa de seis desvios padrões.

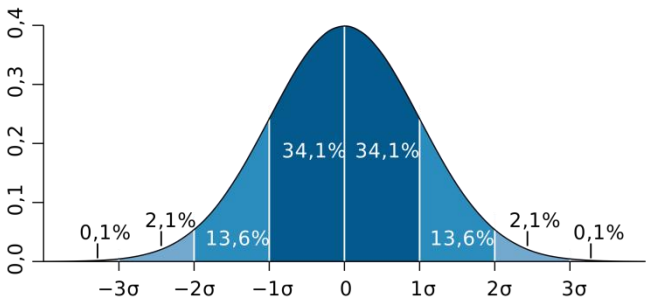


Figura 15 – Detecção de valores extremos em uma distribuição normal

O valor da média e do desvio padrão do campo *ValCurva* são apresentados na Tabela 4.

Média	Desvio Padrão	Intervalo Mais Provável (6σ)		Qtd Registros	
μ	σ	μ-3 σ	μ+3 σ	< μ-3 σ	> μ+3 σ
4949	642	3023	6875	18	270

Tabela 4 – Média e desvio padrão do campo *ValCurva*

Outro método simples de detecção de valores extremos utiliza a mediana e o intervalo inter-quartil (IQR). Consideram-se como valores extremos os valores fora da faixa de três intervalos interquartil (3L), ou seja, 1.5L antes e depois da mediana. Estes valores são apresentados na Tabela 5.

Mediana	Q1 25%	Q3 75%	IQR Q3-Q1	Intervalo Mais Provável (3L)		Qtd Registros	
M	Q1	Q3	L	M-1,5L	M+1,5L	< M-1,5L	M+1,5L
4932	4464	5406	942	3051	6819	27	173

Tabela 5 – Mediana e quartis do campo *ValCurva*

3.2.5 - Análise de Histograma

O histograma do campo *ValCurva* (Figura 16) apresenta a quantidade de registros em cada intervalo de 100 MW, já excluídos os dados zerados e nulos.

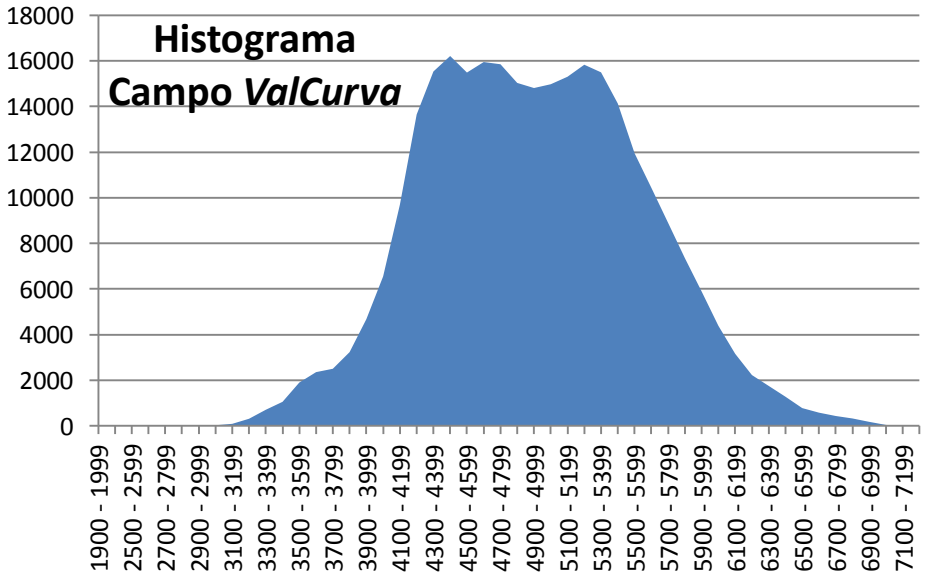


Figura 16 – Histograma do campo *ValCurva*

Como se vê, o histograma é ligeiramente assimétrico, possuindo um alongamento para o lado direito. Do lado esquerdo, sua inclinação é ligeiramente maior. Os valores mais baixos de carga podem indicar alguma eventual ocorrência na rede, como desligamentos, saídas de carga, ou mesmo erros ou falta de medição em algum ponto da rede. Já os valores mais altos indicam picos de consumo, havendo menor probabilidade de representarem falhas do sistema.

Uma análise mais detalhada da quantidade de registros nos intervalos extremos do histograma resultou nos dados da Tabela 6.

Extremo Inferior	
Carga [MW]	Qtd
1900 - 1999	1
2100 - 2199	1
2500 - 2599	1
2600 - 2699	1
2700 - 2799	2
2800 - 2899	2
2900 - 2999	7
3000 - 3099	28
3100 - 3199	90
3200 - 3299	313

Extremo Superior	
Carga [MW]	Qtd
6300 - 6399	1754
6400 - 6499	1280
6500 - 6599	780
6600 - 6699	581
6700 - 6799	434
6800 - 6899	325
6900 - 6999	177
7000 - 7099	43
7100 - 7199	1
7200 - 7299	1

Tabela 6 – Análise de faixas extremas dos valores de carga

3.2.6 - Análise de *Outliers*

Com base na análise descritiva e do histograma, optou-se por descartar uma quantidade aproximadamente igual de registros de ambos os lados do histograma. O intervalo mínimo e máximo de valores válidos do campo *ValCurva*, bem como a quantidade de *outliers* são apresentados na Tabela 7.

Valores Válidos de *ValCurva*		**Qtd Outliers**		
Min	Máx	< Min	> Máx	Total
3100	7000	43	45	88

Tabela 7 – Valores de carga descartados (*outliers*)

O comando a seguir fez com que fossem anulados 88 valores do campo *ValCurva*, sendo considerado como intervalo válido, somente os valores entre 3.100 e 7000 MW.

```
UPDATE CurvaCEMIG SET ValCurva = Null
WHERE ValCurva < 3000 OR ValCurva > 7000;
```

Apesar do campo *DataHoraCarga* não apresentar valores extremos, optou-se por descartar alguns registros de modo a arredondar o intervalo de tempo em estudo. O comando seguinte excluiu 192 registros, após 01 de maio de 2006.

```
DELETE * FROM CurvaCEMIG WHERE DataHoraCarga > #5/1/2006#;
```

Com isso, o último valor válido deste campo ficou sendo 30 de abril de 2006, 23:45h.

3.2.7 - Base de Dados Preparada para Carga

A estrutura final da massa de dados, após as transformações e análises anteriores, resultou na tabela *CurvaCEMIG*, contendo 292.032 registros, com três colunas. São 8,3 anos (100 meses) de dados de carga, em valores integralizados de 15 em 15 minutos. A Tabela 8 resume estes valores:

Tabela: *CurvaCEMIG*				**292.032 registros**	
Campos	**Tipo**	**Mínimo**	**Máximo**	**Zeros**	**Nulos**
ID	Inteiro	0	292.031	-	-
DataHoraCurva	Data-Hora	01/01/1998 00:00	31/04/2006 23:45	-	-
ValCurva	Inteiro	3.100	7.000	-	1.191

Tabela 8 – Estrutura final da base de dados para carga

3.3 - Análise Exploratória dos Dados

A Figura 17 ilustra as principais tarefas da análise exploratória de dados, para um projeto de mineração com *Big Data*. [38]

Análise Exploratória dos Dados	
	Resumir cada atributo
	Explorar relações entre os atributos
	Agrupar dados e recombinar atributos
	Identificar padrões, tendências e fatos não triviais

Figura 17 – Tarefas para análise exploratória dos dados

A primeira tarefa da Análise Exploratória é chamada de Resumo de Atributos. Resumir é um processo pelo qual os dados são reduzidos para a interpretação, sem sacrificar informações importantes. Resumos podem ser desenvolvidos para os dados como um todo ou em partes. Por exemplo, uma empresa de geração de energia que recolheu dados sobre suas operações poderia desenvolver resumos do total de energia vendida, nos vários períodos possíveis: anual, mensal, trimestral, mensal, etc. Além disso, a empresa também poderia gerar resumos de vendas por contrato, por região, ou por tipo de consumidor. Pode ser importante fazer declarações com medidas de confiança sobre todo o conjunto de dados ou grupos dentro dos dados.

A segunda tarefa desta etapa envolve buscar correlações entre atributos. Muitas vezes, o conjunto de atributos não é independente entre si. Isso pode levar a erros, principalmente quando se atua no sentido de maximizar ou minimizar um atributo isoladamente. Por exemplo, para minimizar a ocorrência de falhas em uma máquina, um modelo “cego” poderia sugerir simplesmente diminuir sua utilização. Isso ocorre quando o responsável pela análise não insere no modelo a correlação entre o tempo de funcionamento da máquina e o volume de produção desejado. Ou seja, com a máquina para não haverá defeitos, mas também não haverá produção.

Algumas vezes, padrões de comportamento podem estar escondidos entre vários atributos. Estes padrões mais complexos só aparecem ao combinarmos um ou mais atributos, ou ao agruparmos os dados. Por exemplo, os dados de consumo de energia só exibem certos padrões quando analisados conforme os dias de semana e a época do ano. O consumo típico de uma terça-feira pode ser radicalmente diferente em um dia de inverno e outro de verão, graças à mudança na duração do dia com as estações, ou a vigência ou não do horário de verão. Em outros casos, pode ser necessário particionar os dados para que se possam extrair alguns padrões. Por exemplo, a influência econômica é bem mais evidente na curva de carga industrial do que na comercial e residencial. Conforme o ciclo econômico, o consumo da indústria pode variar muito, enquanto um comércio ou residência levam mais tempo para sentir efeitos de uma crise por exemplo. Assim, os efeitos da correlação do consumo de energia e da atividade econômica só apresentam padrões claros quando são segregados os vários tipos de consumidores.

Nas próximas seções são feitas uma série de análises pertinentes ao comportamento da carga.

3.4 - Carga Média por Ano

O gráfico da Figura 18 apresenta a evolução da carga média por ano, no intervalo de 1998 a 2006. Nota-se uma tendência de crescimento aproximadamente linear da carga entre 1998 e 2000, bem como de 2001 a 2006. No ano de 2001 nota-se uma queda abrupta da carga.

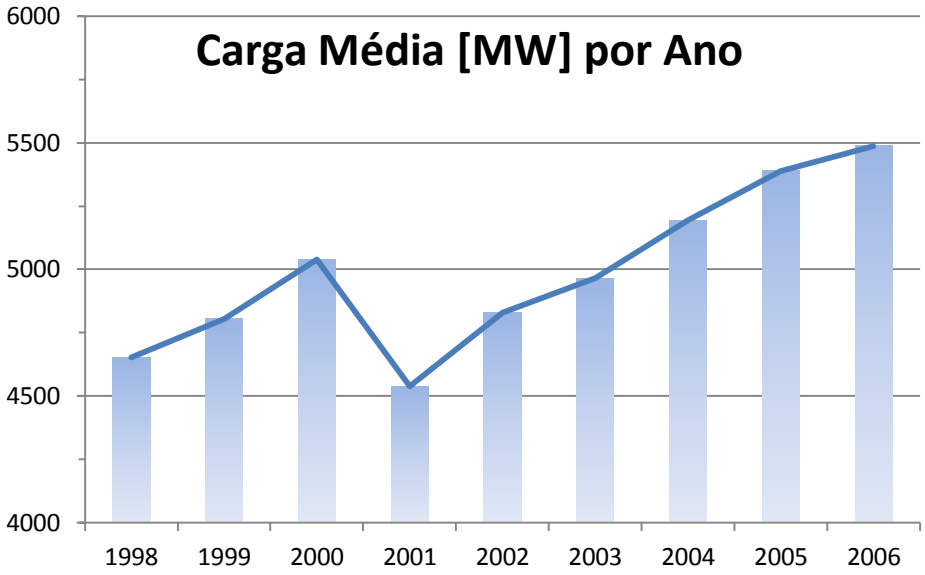


Figura 18 – Gráfico da carga média por ano

Durante o ano de 2001, houve no Brasil a chamada "Crise do Apagão". Esta foi uma crise nacional, que afetou o fornecimento e distribuição de energia elétrica no país. Ocorreu entre julho de 2001 e fevereiro de 2002, durante o segundo mandato do presidente Fernando Henrique Cardoso. A crise ocorreu por uma soma de fatores como as poucas chuvas, a falta de planejamento e ausência de investimentos em geração e transmissão de energia. Com a escassez de chuva, o nível de água dos reservatórios das hidroelétricas baixou e os brasileiros foram obrigados a racionar energia. Estipularam-se benefícios aos consumidores que cumprissem metas de redução de consumo e punições para quem aumentasse o consumo de energia. Como forma de prevenção para o futuro, o governo federal iniciou um imenso programa de investimentos em uma rede de usinas termoelétricas, movidas a gás, carvão e óleo combustível, que não dependem do ciclo das águas. Essa rede de usinas traria

flexibilidade ao sistema e serviria de back-up em épocas de secas, complementando o sistema hidroelétrico.

3.5 - Evolução Anual da Carga

Os coeficientes de crescimento anual da carga estão representados no gráfico da Figura 19. Com exceção de 2001, nota-se um crescimento médio da carga em torno de 4% ao ano. Devido à crise de 2001, a carga sofreu uma queda de 10%.

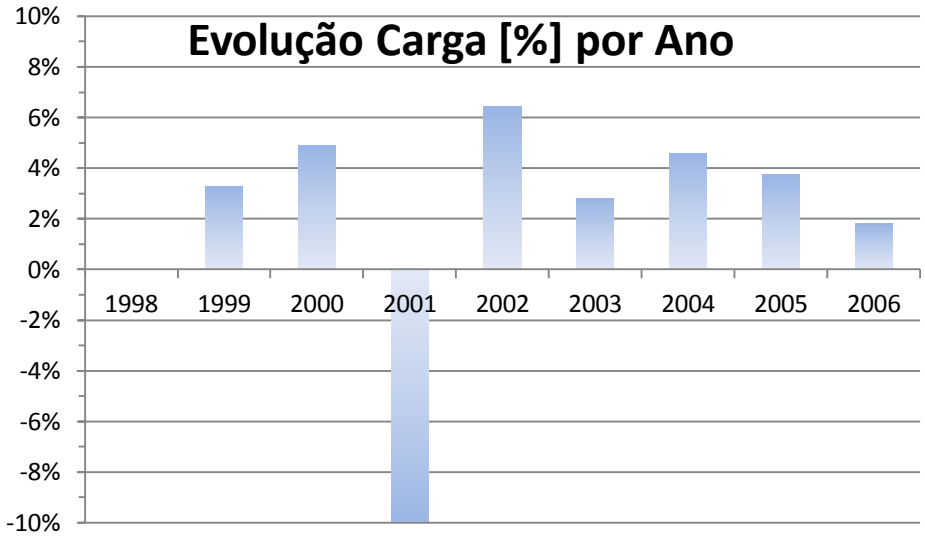


Figura 19 – Gráfico da evolução anual da carga

3.6 - Evolução Mensal da Carga

Os dados disponíveis englobam 100 meses de informação. O gráfico da Figura 20 apresenta o valor médio da carga por mês.

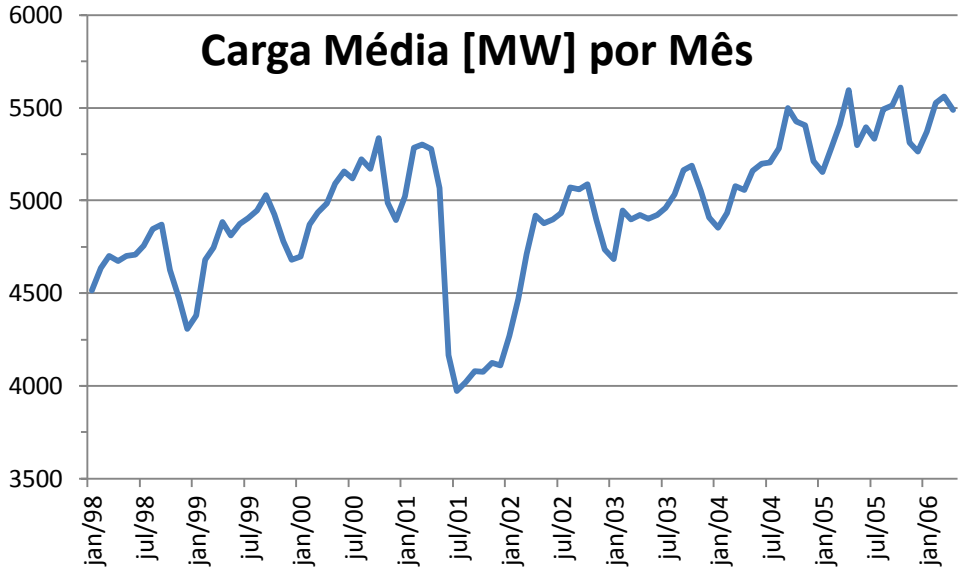


Figura 20 – Gráfico da carga média por mês

Novamente nota-se o reflexo da crise de 2001. Pode-se considerar que os meses entre junho 2001 e fevereiro de 2002 foram bastante afetados, apresentando um comportamento totalmente diferente dos demais meses.

3.7 - Análise de Tendência

A análise de tendência visa identificar a tendência de longo prazo da carga. Devido à crise de 2001, optou-se por dividir os dados antes e depois da crise. Os dados de julho de 2001 a fevereiro de 2002 foram desconsiderados no cálculo da tendência.

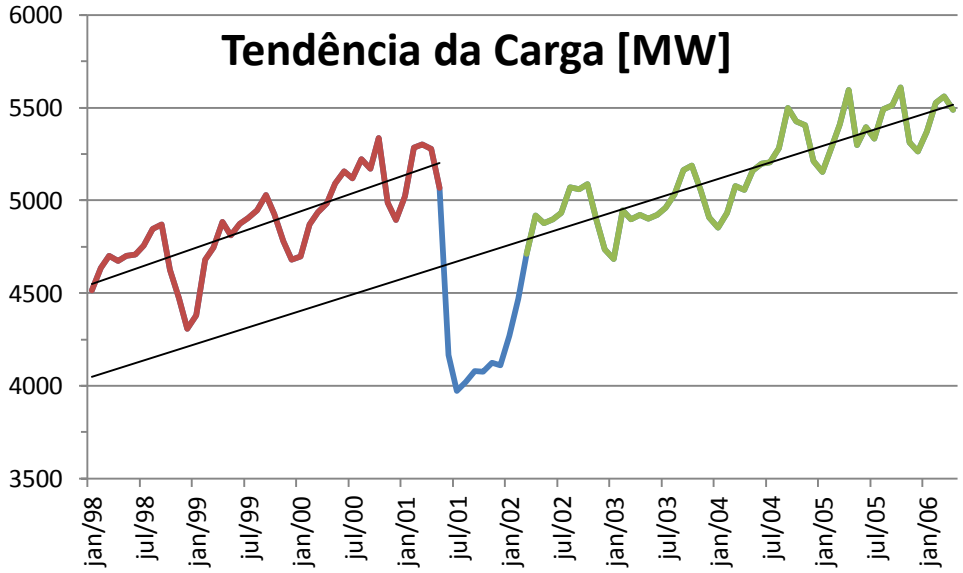


Figura 21 – Gráfico da tendência de crescimento da carga

Do gráfico da Figura 21, nota-se que, após a crise, a carga retomou seu ritmo de crescimento, praticamente com a mesma inclinação. A taxa de crescimento é de aproximadamente 15 MW/mês, ou 180 MW/ano. O impacto total da crise de 2001 foi um deslocamento de aproximadamente 500 MW, o que equivale a um retrocesso de 2,7 anos na evolução natural da carga.

3.8 - Sazonalidade Mensal

O gráfico da Figura 22 mostra como a carga varia durante o ano. O ciclo anual visa detectar aspectos de sazonalidade da carga, com o objetivo de detectar como as estações do ano influenciam o comportamento da carga.

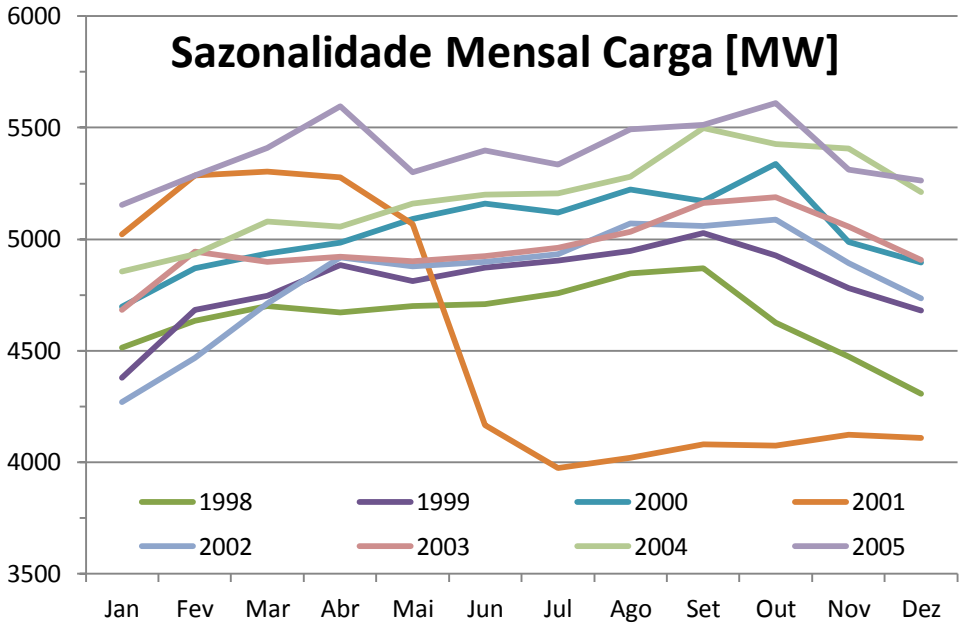


Figura 22 – Gráfico da sazonalidade mensal da carga

3.9 - Sazonalidade Percentual

No Brasil, os economistas costumam detectar um fenômeno apelidado de JASON, formado pelas iniciais dos meses de julho, agosto, setembro, outubro e novembro. De um modo geral, acredita-se que estes são os meses de maior atividade econômica, principalmente na indústria.

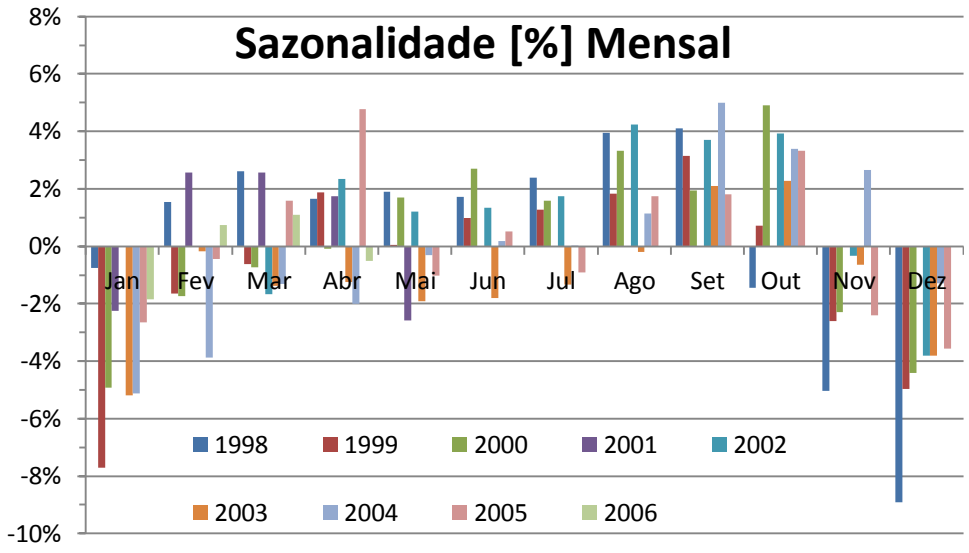


Figura 23 – Gráfico da sazonalidade percentual da carga

No gráfico da Figura 23, nota-se que a carga nos meses de novembro, dezembro, janeiro e fevereiro são relativamente mais baixas. Os meses de junho, julho, agosto, setembro e outubro são costumeiramente meses de carga mais alta. De um modo geral, as sazonalidades variam na faixa de -5% até 4%. Apesar dos padrões gerais observados, não se pode dizer que a sazonalidade da carga possua uma característica constante ao longo dos anos analisados.

3.10 - Influência do Horário de Verão

Com base na análise da sazonalidade da carga, pode-se afirmar que a vigência do horário de verão parece ser um fator importante no comportamento da carga. Com o objetivo de explorar melhor este fato, optou-se por acrescentar a base de dados uma coluna indicando a vigência ou não de horário de verão.

```
ALTER TABLE CurvaCEMIG ADD COLUMN HorárioVerão Text(1);
```

Nos anos em questão, o horário de verão teve sempre seu início e término na madrugada de sábado para domingo. Uma exceção ocorreu em 06 de outubro de 1997, quando ocorreu de domingo para segunda, e também em 02 de novembro de 2004, quando o horário começou de segunda para terça, talvez devido ao feriado prolongado de finados. A Tabela 9 lista os dias de início e término do horário de verão durante os anos em estudo.

Anos	Data Início	Data Término
1997-1998	06/10/1997 00:00	01/03/1998 00:00
1998-1999	11/10/1998 00:00	21/02/1999 00:00
1999-2000	03/10/1999 00:00	27/02/2000 00:00
2000-2001	08/10/2000 00:00	18/02/2001 00:00
2001-2002	14/10/2001 00:00	17/02/2002 00:00
2002-2003	03/11/2002 00:00	16/02/2003 00:00
2003-2004	19/10/2003 00:00	15/02/2004 00:00
2004-2005	02/11/2004 00:00	20/02/2005 00:00
2005-2006	16/10/2005 00:00	19/02/2006 00:00

Tabela 9 – Vigência do horário de verão

Após a atualização do novo campo, este passou a ter a seguinte distribuição da Tabela 10.

Campo	Tipo	Distribuição de Valores			Zeros	Nulos
		Valor	Qtd	%		
HorárioVerão	Text(1)	"S"	101.568	34,8%	-	-
		"N"	190.464	65,2%		

Tabela 10 – Distribuição de valores do campo horário de verão

O gráfico da Figura 24 destaca os períodos de vigência do horário de verão durante os meses em estudo.

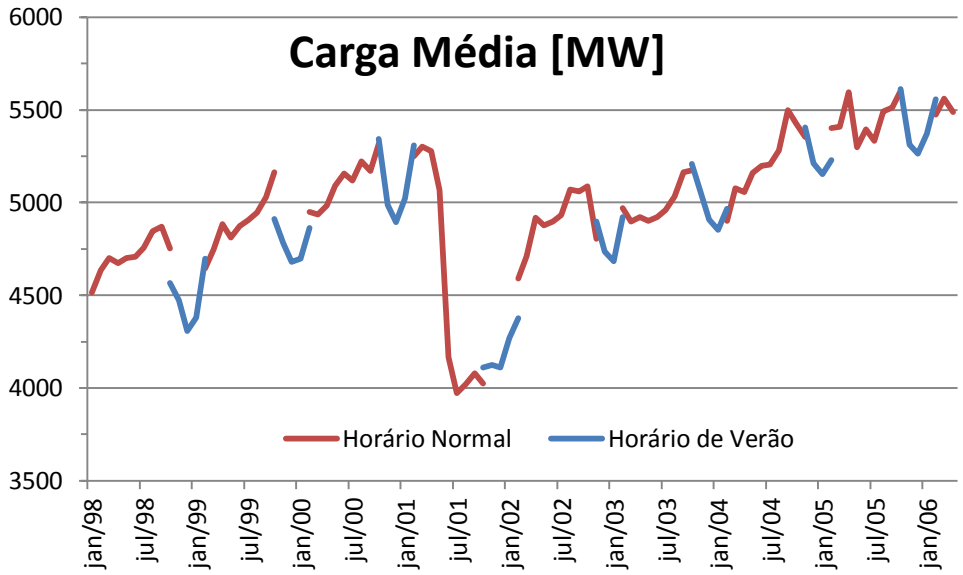


Figura 24 – Carga média dentro e fora do horário de verão

Do gráfico anterior, pode-se concluir que quase todas as grandes quedas do valor da carga ocorreram após o início do horário de verão. Após o fim da vigência do horário de verão, há uma retomada do crescimento da carga. Somente no período da crise de 2001, que teve seus reflexos até o início de 2002, é que o início do horário de

verão representou um aumento da carga. Outro período que merece destaque é o significativo crescimento em abril de 2005, seguido de uma queda nos três meses seguintes.

3.11 - Sazonalidade Semanal - Dias Típicos

O comportamento da carga ao longo de um dia apresenta várias características interessantes. Os dias típicos seguem uma periodicidade semanal, e variando conforme a vigência ou não do horário de verão. O gráfico da Figura 25 mostra as curvas de carga média sem horário de verão.

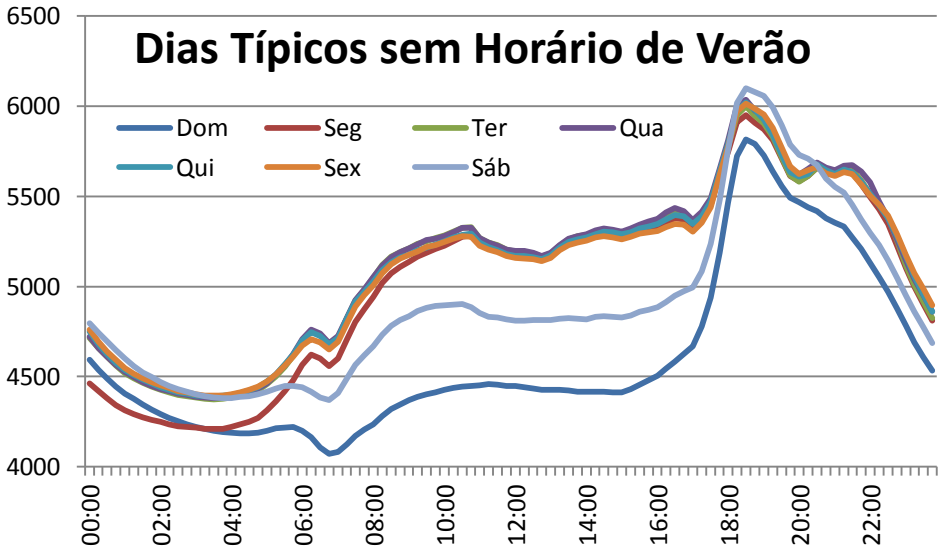


Figura 25 – Carga média por dia típico sem horário de verão

Nota-se que os dias do final de semana têm um comportamento bem diferente dos dias do meio da semana. A carga durante o dia é maior durante a semana, pouco mais baixa no sábado, e mínima no domingo. Essas diferenças diminuem conforme chega o anoitecer. Esta diferença pode ser mais bem representada pela carga média por dia da semana, conforme a Tabela 11.

Dia da Semana	Carga Média [MW]	[%]
Domingo	4603	89,9%
Segunda	5031	98,3%
Terça	5105	99,8%
Quarta	5117	100,0%
Quinta	5105	99,8%
Sexta	5098	99,6%
Sábado	4904	95,8%

Tabela 11 – Carga média diária sem horário de verão

Interessante notar que o pico máximo de carga ocorre no início da noite de sábado, com carga máxima levemente superior aos dias de semana. Por outro lado, durante a madrugada, nota-se que a carga da segunda feira é até menor que a carga do domingo. A carga da segunda-feira inicia muito baixa, mas cresce rapidamente durante o alvorecer. Antes das 8 horas da manhã, a carga da segunda-feira já alcançou valores semelhantes aos demais dias de semana.

Todos os dias, em torno das 7 horas da manhã, há um ligeiro dente na curva de carga, representando uma queda. Tal figura representa a saída da carga de iluminação pública. Durante toda a manhã, a carga cresce, até atingir um pico em torno das 11 horas da manhã. Durante o almoço a carga cai levemente e inicia uma recuperação gradual durante a tarde. Após as 17 horas, durante os dias de semana, a carga apresenta um novo dente, como se fosse cair, mas muda rapidamente sua tendência para um crescimento expressivo, atingindo seu pico em torno das 19 horas.

Após atingir seu pico, a carga vai caindo gradualmente pela noite e madrugada, atingindo seu mínimo em torno das 4 horas. Nos dias de semana, durante sua trajetória de queda no início da noite, a carga apresenta um patamar oscilante em torno das 19:30 e 21:30 horas.

Com o início do horário de verão, os perfis de carga sofrem uma alteração drástica, principalmente no final da tarde. O gráfico da Figura 26 representa os dias típicos durante a vigência do horário de verão.

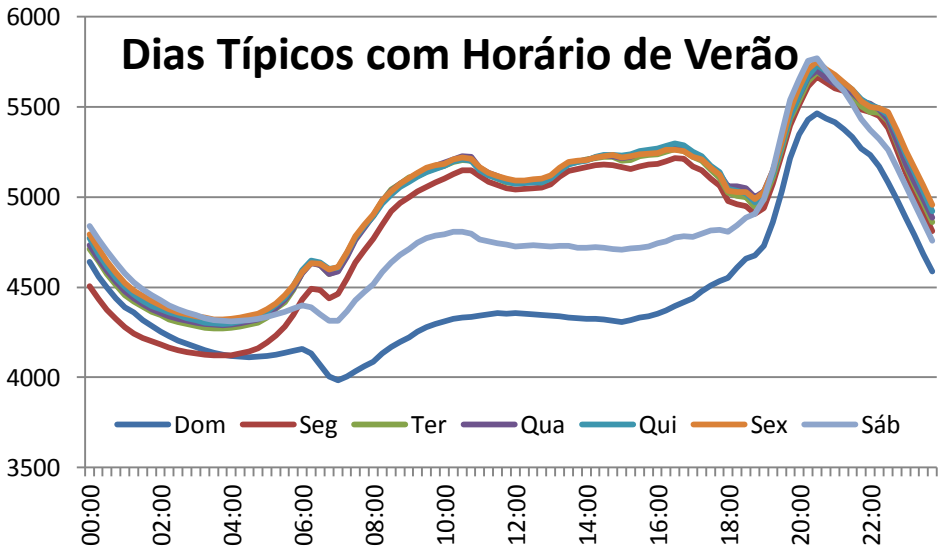


Figura 26 – Carga média por dia típico com horário de verão

Durante a vigência do horário de verão, o comportamento da carga praticamente não sofre alteração na madrugada e durante a manhã e início da tarde. Porém, no final da tarde e início da noite, inicia-se uma grande diferenciação do comportamento típico da carga. Durante o horário de verão, devido à duração maior da insolação, a carga inicia uma trajetória de queda. Este fenômeno está associado à saída de cargas comerciais e industriais que encerram seu expediente ou turno, sem a necessidade de acionamento da iluminação. A carga só volta a crescer mais tarde, quando se inicia o pico residencial da noite.

No gráfico da Figura 27, esta diferença fica representada de maneira mais clara. Nele é feita uma comparação entre um dia típico (quarta-feira) com e sem horário de verão.

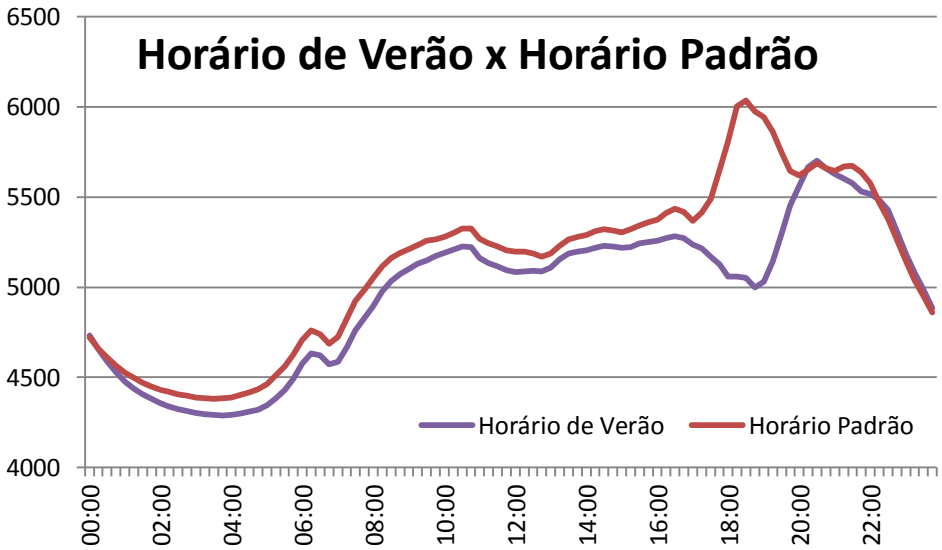


Figura 27 – Gráfico de uma quarta-feira típica, com e sem horário de verão.

Nota-se que o pico de carga sem horário de verão é completamente evitado durante a vigência do horário de verão. No final da tarde, ao invés da carga crescer, ela diminui expressivamente. O pico de carga é deslocado para a noite. Interessante notar que o patamar da noite, em torno de 20:30 horas, que, sem horário de verão, representava uma retomada da carga, no horário de verão passa a ser o novo pico diário da carga.

3.12 - Identificação de Feriados

No Brasil, existem feriados nacionais, estatuais e municipais. O calendário brasileiro é composto de oito feriados nacionais fixos e outras quatro datas religiosas móveis, cujas datas dependem da data da Páscoa. No estado de Minas Gerais, o único feriado estadual é coincidente com o feriado nacional de 21 de abril, dia de Tiradentes. Os feriados municipais ocorrem normalmente no aniversário das cidades e em algumas festas religiosas locais.

Data Fixa	Feriados Nacionais
21 de abril	Tiradentes
01 de maio	Dia do Trabalho
07 de setembro	Independência
12 de outubro	Nossa Senhora Aparecida
02 de novembro	Finados
15 de novembro	Proclamação da República
25 de dezembro	Natal
01 de janeiro	Confraternização Universal

Tabela 12 – Tabela de feriados nacionais com data fixa

Os feriados com data fixa (Tabela 12) são fáceis de calcular e prever. Porém, sua influência na carga é variável, conforme o dia da semana em que ocorrem. Aqui é interessante notar que, dentro dos feriados fixos existem dois grupos de três cada, que ocorrem sempre no mesmo dia da semana. Serão chamados aqui de Trio Dominante e Trio Menor.

O Trio Dominante é formado pelos feriados de Independência (07/set), Nossa Senhora Aparecida (12/out) e Finados (02/nov). Este trio é chamado dominante porque ele determina outros dois feriados: o dia 15 de novembro, que cai sempre no dia de semana anterior, e o 21 de abril, que ocorre no dia de semana seguinte. Por exemplo, quando os feriados do trio dominante caem em uma quinta-feira, o dia 21 de abril cairá em uma quarta-feira, e o dia 15 de novembro em uma sexta-feira. Estes eventos estão ilustrados na Figura 28.

	Dom	Seg	Ter	Qua	Qui	Sex	Sáb
21 abr						D+1	
01 mai		M					
07 set					D		
12 out					D		
02 nov					D		
15 nov				D-1			
25 dez		M					
01 jan		M					

Figura 28 – Exemplo do Trio (D)ominante e (M)enor dos feriados fixos

O segundo trio de feriados, o Trio Menor, é formado pelo dia do Trabalho (01/mai), Natal (25/dez) e Confraternização Universal (01/jan). O dia da semana do trio menor é sempre três dias antes do trio maior. Assim, se o trio dominante cai em uma quinta-feira, o trio menor cai em uma segunda-feira.

Como os feriados fixos ocorrem cada ano em um dia da semana, sua influência na carga pode variar a cada ano. Por exemplo, um feriado que cai em uma quinta-feira pode influenciar a carga da sexta-feira, formando um Feriado Prolongado. O mesmo acontece quando um feriado acontece em uma terça-feira, criando um feriado prolongado que afeta a segunda-feira anterior, e talvez até todo o final de semana.

Além dos feriados fixos, existem datas religiosas móveis, que frequentemente são tratados como feriados. A Tabela 13 apresenta a lista de festas religiosas móveis.

Dia da Páscoa (D)	Dias Atípicos
D – 47	Terça-feira de Carnaval
D – 2	Sexta-feira da Paixão
D	Domingo de Páscoa
D + 60	Quinta-feira de Corpus Christi

Tabela 13 – Festas religiosas móveis

Todas estas datas são calculadas conforme o dia da Páscoa. Apesar de serem datas móveis, são dias de semana fixos. Assim, sua influência na carga costuma ser sempre a mesma.

O domingo de Páscoa, por definição, é o primeiro domingo após a primeira lua cheia que ocorrer depois do equinócio vernal, ou seja, o equinócio de início do outono no Brasil. Assim, a Páscoa pode cair desde o dia 22 de março até o dia 25 de abril.

O cálculo do dia da Páscoa, conhecido como *Computus* em latim, pode ser feito de várias maneiras. Um dos algoritmos computacionais mais simples é o de Gauss, que utiliza duas constantes (X e Y), 5 variáveis (A,B,C,D,E) e o operador módulo (% ou mod), além de alguns testes para correções. Para os anos (pAno) entre 1900 e 2099, as constantes são X=24 e Y=5. As 5 variáveis são calculadas pelas fórmulas da Listagem 1.

```
1  //Constantes X e Y válidos entre 1900 e 2099
2  X=24; Y=5;
3  // Fórmulas das 5 Variáveis A,B,C,D,E
4  A = pAno mod 19;
5  B = pAno mod 4;
6  C = pAno mod 7;
7  D = (19 * A + X) mod 30;
8  E = (2 * B + 4 * C + 6 * D + Y) mod 7;
```

Listagem 1 – Constantes X,Y e variáveis A,B,C,D,E para cálculo da Páscoa

Com estes valores pode-se calcular a data da Páscoa, e então aplicar algumas correções nos casos em que a Páscoa cai no mês de abril. O dia (pDia) e mês (pMês) da Páscoa serão dados pelo pseudo-código da Listagem 2.

```
1    pDia = D + E + 22;
2    pMes = 3;
3
4    SE (pDia > 31) {
5        pDia = pDia - 31;
6        pMes = 4;
7        //Exceção do ano de 2049
8        SE ((pDia == 25) && (D == 28) && (A > 10)) pDia = 18;
9        //Exceção do ano de 2076
10       If (pDia == 26) pDia = 19;
11   }
```

Listagem 2 – Cálculo da data da Páscoa

Os feriados municipais do estado de Minas Gerais que tem maior chance de influenciar a carga são os feriados da capital Belo Horizonte, listados na Tabela 14.

Data Fixa	Feriado
15 de agosto	Assunção de Nossa Senhora
08 de dezembro	Imaculada Conceição (até 2006)
12 de dezembro	Aniversário de BH (2006 em diante)

Tabela 14 – Feriados municipais da capital do estado

Com base nas listas de feriados nacionais, estaduais e municipais, e no algoritmo de cálculo do dia da Páscoa, foram gerados todos os feriados desde 1998 até 2006. Criou-se então a tabela Feriados, com a estrutura exibida na Tabela 15.

Tabela: Feriados	108 registros
Campos	Tipo
IDFerido	Inteiro (PK)
DataFeriado	Data-Hora
Feriado	Texto(64)

Tabela 15 – Estrutura da tabela Feriados

Para adicionar a informação de feriado na massa de dados, foi criada uma coluna para indicar a ocorrência de feriados.

```
ALTER TABLE CurvaCEMIG ADD COLUMN Feriado Text(1);
```

Os dias de feriado presentes na tabela Feriados foram preenchidos conforme os códigos da Tabela 16.

Código de Feriado	Descrição
“U”	Dia Útil normal
“F”	Dia de (F)eriado
“A”	Dia (A)nterior a um feriado
“P”	Dia (P)osterior a um feriado

Tabela 16 – Códigos do campo Feriado

Assim, o campo feriado ficou com a distribuição apresentada na Tabela 17.

Campo	Tipo	Distribuição de Valores		Zeros	Nulos
		Valor	Qtd %		
Feriado	Text(1)	“N”	91,6%	-	-
		“F”	2,8%		
		“A”	2,8%		
		“P”	2,8%		

Tabela 17 – Estrutura do campo Feriado

3.13 - Identificação de Dias Tipo

Conforme foi visto na análise das curvas de carga para cada dia da semana (Figura 25), os dias úteis de terça a sexta são praticamente iguais. Após a inserção dos feriados, verificou-se a influência dos feriados prolongados. As segundas-feiras são afetadas por feriados nas terças, enquanto as sextas e sábados recebem a influência de um feriado na quinta-feira. Os domingos são únicos, ou seja, não são afetados por feriados. Assim, foram identificados 13 tipos de curva de carga, resumidos na Tabela 18.

Descrição	Código de Dia Tipo	Qtd Registros
Domingo Típico	DT	41.760
Segunda-feira Típica	“2T”	39.360
Dia Útil Típico (terças, quartas, quintas ou sextas	“UT”	159.648
Sábado Típico	ST	39.360
Dias de Feriados (segunda a sábado)	2F	864
	3F	1920
	4F	864
	5F	1632
	6F	1632
	SF	1056
Feriado Prolongado segunda, sexta e sábado	2P	1440
	6P	1152
	SP	1344

Tabela 18 – Códigos de dia tipo

Estes códigos foram atribuídos ao campo DiaTipo criado na tabela de dados.

```
ALTER TABLE CurvaCEMIG ADD COLUMN DiaTipo Text(2);
```

3.14 - Análise de Variabilidade

Com o objetivo de conhecer como as cargas de um mesmo dia da semana se comportam durante o período em estudo, foi feito um estudo de variabilidade. Foram escolhidas para análise e plotagem as curvas de carga de todas as quartas-feiras típicas, com e sem horário de verão. Destes dados, foi possível verificar que, dentro de um intervalo de dois desvios padrões (2σ), para mais e para menos, existe um envelope que engloba quase a totalidade dos dados. A Figura 29 apresenta este envelope para as quartas-feiras sem horário de verão, enquanto a Figura 30 mostra este mesmo dia com horário de verão.

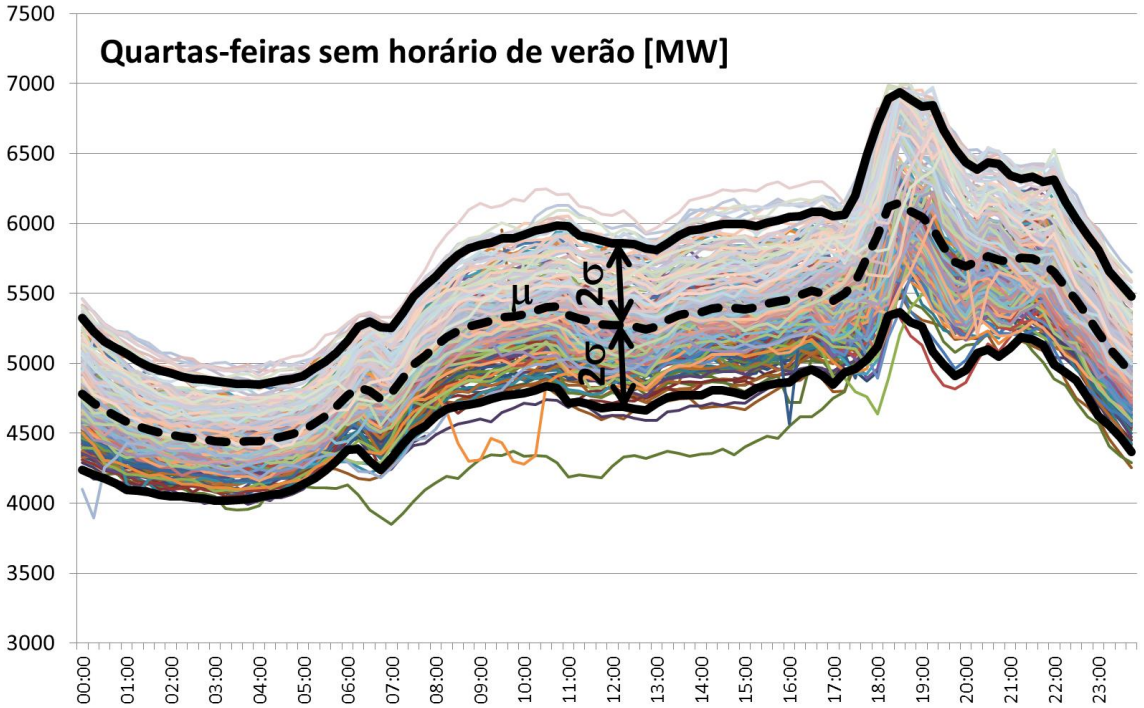


Figura 29 – Variância das quartas-feiras típicas, sem horário de verão.

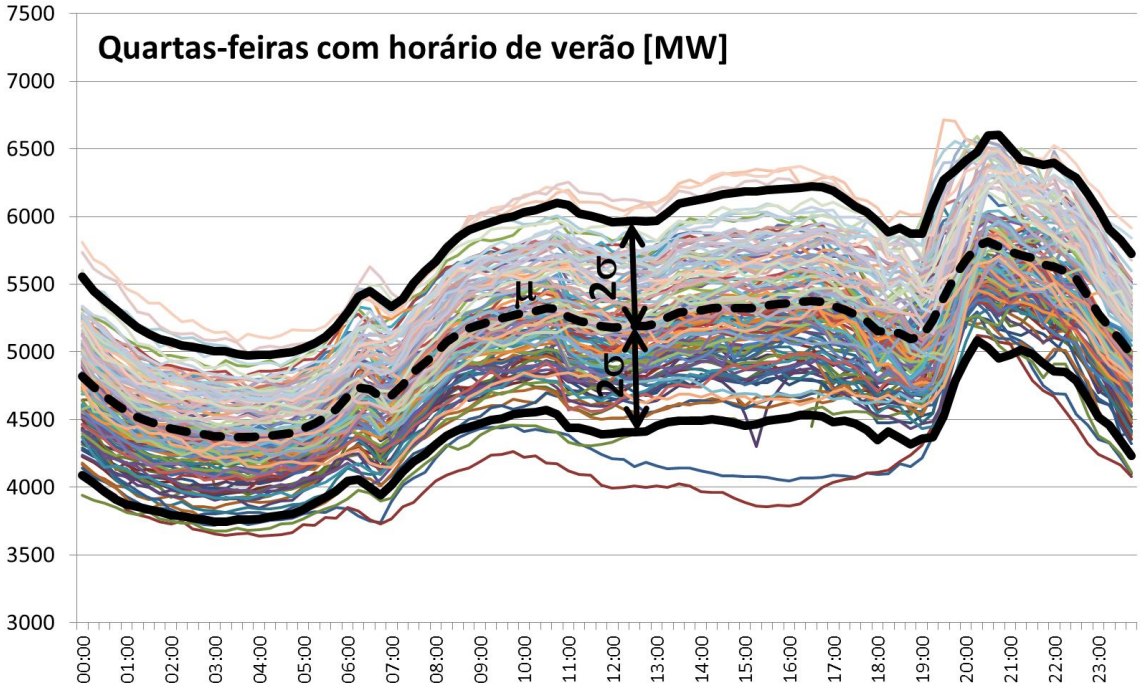


Figura 30 – Variância das quartas-feiras típicas, com horário de verão.

Dos gráficos anteriores, pode-se concluir que, dentro do período em estudo, para um mesmo dia da semana, os valores de carga variam na ordem de 10% a 12% em relação à média, para mais ou para menos. Isso significa uma amplitude de variação superior a 20% do valor da carga.

Capítulo 4
Aprendizado de Máquina

Os métodos de mineração e geração de modelos preditivos são também chamados de métodos de Aprendizado de Máquina (*Machine Learning*). Eles foram desenvolvidos simultaneamente em diversas áreas, tais como Estatística Aplicada e Reconhecimento de Padrões. [41]

Aprendizado de Máquina, como um campo de estudo, surgiu como uma subárea da Inteligência Artificial, que estava interessada em métodos para melhorar o desempenho de agentes inteligentes ao longo do tempo, em resposta à experiência do agente com o mundo. Tal melhora muitas vezes envolve analisar dados do ambiente e fazer previsões. Uma subdivisão dos métodos de Aprendizado de Máquina é chamada Descoberta de Conhecimento em Bancos de Dados (KDD - *Knowledge Discovery in Databases*). Este campo de pesquisa é mais focado em problemas práticos ligados a aplicações comerciais e industriais.

Com a explosão do uso de microprocessadores, e sua interligação em rede, chegou-se a chamada Internet das Coisas (IoT – *Internet of Things*). Nesta era, a quantidade de dados armazenados cresceu exponencialmente, aumentando a necessidade, e criando a oportunidade para aplicação e adaptação dos métodos de Aprendizado de Máquina. Esta nova realidade foi batizada de Era do Big Data.

A Figura 31 lista as tarefas envolvidas na criação de um modelo preditivo através do aprendizado de máquinas. [42]

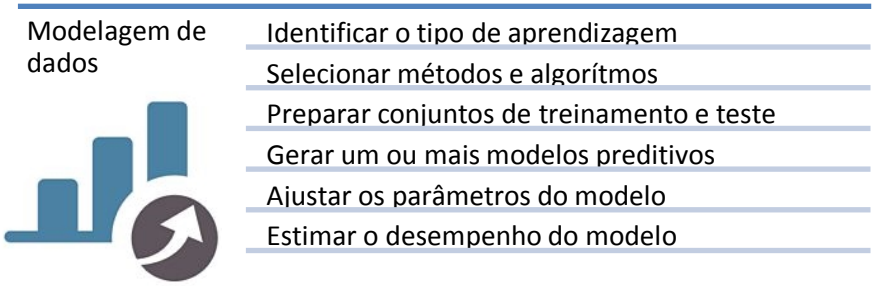


Figura 31 – Tarefas de modelagem de dados

4.1 - Aprendizado Supervisionado e Não Supervisionado

A área de aprendizado de máquina se divide em dois grandes grupos: aprendizado supervisionado e aprendizado não supervisionado. Metaforicamente, um professor “supervisiona” seus alunos fornecendo a eles, as respostas de um conjunto de problemas exemplo, cuidadosamente escolhidos. Já no aprendizado não supervisionado, apesar de poder usar o mesmo conjunto de exemplos, as respostas não são fornecidas. O aprendiz deve tirar suas próprias conclusões, encontrando sozinho o que os exemplos têm em comum ou não.

Em problemas de aprendizado supervisionado, um programa prevê uma saída, para certa entrada, aprendendo com pares ordenados de entradas e saídas fornecidos, ou seja, o programa aprende com exemplos de respostas certas. Na aprendizagem não supervisionada, um programa não aprende a partir de dados rotulados. Em vez disso, ele tenta descobrir padrões nos dados. Aprendizagem supervisionada e aprendizagem não supervisionada podem ser consideradas como extremos opostos de um mesmo espectro. [43]

Há muitos nomes diferentes para as entradas e saídas de um programa de aprendizagem de máquina. Como várias disciplinas utilizam estes métodos, cada área possui sua própria terminologia. As variáveis de entrada podem ser chamadas de: preditores, regressores, variáveis controladas, variáveis manipuladas, variáveis independentes, ou variáveis de exposição. Já a variável de resposta pode ser chamada: variável dependente, regressante, variável de critério, variável medida, variável explicativa, variável experimental, rótulo, previsão, atributo de saída, entre outros.

4.2 - Métodos e Modelos de Aprendizado

Uma habilidade crítica para um cientista de dados é sua capacidade de decompor um problema em partes, de tal modo que, a cada uma corresponda uma tarefa para os quais estão disponíveis ferramentas e métodos conhecidos e eficientes. Reconhecer problemas familiares e suas soluções evita desperdício de tempo e recursos. Permite também que as pessoas se concentrem nas partes mais críticas do

processo, que requerem envolvimento humano, criatividade e inteligência, habilidades que não são facilmente automatizáveis.

Apesar do grande número de algoritmos de mineração de dados desenvolvidos ao longo dos anos, existem apenas alguns tipos fundamentais. É interessante identificar suas características para que se se possam determinar as oportunidades de uso de cada um. Os principais tipos de modelos são apresentados a seguir. [43]

4.2.1 - Modelos de Classificação

Um modelo classificador é um procedimento que, dado um novo indivíduo, determinará a qual classe ele pertence. Normalmente, as classes são mutuamente exclusivas. Ou seja, o modelo tem como variável de saída um tipo categórico. O atributo de saída pode assumir um dos valores de seu domínio, formado por um pequeno conjunto discreto de classes. Por exemplo, um problema de classificação poderia ser: “Criar um modelo preditivo para prever os tipos de falhas que um equipamento irá apresentar nos próximos noventa dias”. Outra saída possível para um problema de classificação é calcular a probabilidade do indivíduo pertencer a cada uma das classes. Nestes casos, pode-se aplicar uma pontuação a cada indivíduo, produzindo, em vez de uma só classe, uma contagem representando a probabilidade (ou alguma outra quantificação) de que o indivíduo pertença a cada classe. No exemplo anterior, um modelo de pontuação seria capaz de avaliar cada equipamento e produzir uma pontuação para cada tipo de falha. Classificação e pontuação estão intimamente relacionadas. Um modelo que faz um, pode geralmente ser modificado para fazer o outro.

4.2.2 - Modelos de Regressão

Modelos de regressão têm como objetivo estimar um valor numérico para cada novo indivíduo. Ou seja, sua variável de saída é do tipo numérico. Por exemplo, um modelo preditivo para previsão do consumo de energia de um cliente. O atributo, a ser previsto aqui, é a quantidade de utilização de um serviço. Um modelo assim pode ser gerado a partir da observação de outros indivíduos semelhantes na população. A regressão tem algumas semelhanças com a classificação, mas devem-se ressaltar suas

principais diferenças. Na classificação, o objetivo é prever se algo vai ou não acontecer, ao passo que na regressão, espera-se conhecer o quanto algo vai acontecer.

4.2.3 - Modelos de Semelhança

Modelos de semelhança tentam identificar, na base de dados, casos que se assemelham ao caso em estudo. Por exemplo, uma empresa pode estar interessada em pesquisar, no universo de potenciais novos clientes, aqueles que mais se assemelham aos seus clientes mais rentáveis. Isso possibilitaria concentrar sua força de vendas sobre as melhores oportunidades. Um modelo de semelhança emprega algum tipo de medida de distância que indique o quanto duas instâncias de dados são próximas. Quanto mais próximas, mais semelhantes. Por exemplo, pode-se definir que dois indivíduos serão próximos se houverem comprado produtos semelhantes em um site. Esta informação pode ser usada para sugerir novos produtos para os clientes, na esperança que haja uma maior probabilidade de compra. O conceito de máxima semelhança é usado implicitamente em vários outros tipos de modelos, mas com abordagens diferentes.

4.2.4 - Modelos de Agrupamentos

O objetivo de um modelo de agrupamento, utilizado em aprendizado não supervisionado, é descobrir, dentro dos dados de treinamento, grupos de observações relacionadas, chamados de *Clusters*. Esta tarefa é conhecida como análise de clusters. Ela cria grupos de observações semelhantes entre si, com base em alguma medida de similaridade. A técnica de agrupamento é frequentemente utilizada para explorar um conjunto de dados. Por exemplo, dado um conjunto de resenhas de filmes, um modelo de agrupamento pode descobrir conjuntos de filmes semelhantes, recomendando novos filmes para um espectador. Outra aplicação comum é criar grupos de clientes dentro do mercado de um produto. Ao entender quais atributos são comuns a um grupo específico de clientes, os fornecedores podem decidir quais aspectos de suas campanhas precisam ser enfatizadas.

4.2.5 - Modelos de Coocorrência

Conhecidos também como modelos de análise de cestas de compras, ou descoberta de regras de associação, estes modelos tentam encontrar associações

entre entidades com base em listas de transações. Uma pergunta típica para este modelo seria: Quais itens de um supermercado são comumente comprados juntos? Enquanto um modelo de agrupamento procura semelhanças entre os objetos, com base nos atributos dos objetos, os modelos de coocorrência analisam a frequência com que os objetos aparecem em uma lista de transações destes objetos. Depois de identificadas associações, as vendas podem ser maximizadas mantendo-se os estoques dos produtos sincronizados, evitando-se que um deles esteja em falta, quanto o outro estiver disponível. A saída destes modelos é a frequência de ocorrência simultânea entre dois ou mais produtos em uma transação, e também uma estimativa de quão surpreendente isto é.

4.2.6 - Modelos de Perfil

Também conhecidos como modelos comportamentais, estes tentam caracterizar o comportamento típico de um indivíduo, grupo ou população. Um exemplo de problema de perfil é: "Qual é o consumo típico de energia em um segmento de clientes?" Um comportamento pode não ter uma descrição simples. Determinar um perfil de uso pode exigir uma descrição complexa em várias dimensões, como hora do dia, finais de semana, tipos de equipamentos utilizados, dependência de fatores exógenos como temperatura e economia, etc. Um comportamento pode ser descrito tanto para uma população inteira, ou para níveis de pequenos grupos ou mesmo indivíduos. O termo *Profiling* é muitas vezes usado para estabelecer padrões de comportamento em aplicações de detecção de fraudes e monitoramento de invasões em sistemas de computador. Por exemplo, se a curva de consumo típica de um cliente é conhecida, pode-se desconfiar, apenas com base na análise de modificações em seu consumo, a probabilidade de instalação de um dispositivo para fraudar um medidor de energia. Pode-se usar o grau de incompatibilidade como uma pontuação suspeita e emitir um alarme se o valor for demasiado elevado.

4.2.7 - Modelos de Conexão

Modelos de conexão tentam prever ligações entre dois itens de dados, sugerindo possíveis novas ligações, bem como estimar a força desta ligação. Previsão de ligação é comum em sistemas de redes sociais: "Uma vez que você conhece X, e X é

amigo de Y, você gostaria de ser amigo de Y?". A previsão de conexão também pode estimar a força de uma conexão. Por exemplo, para recomendar filmes para clientes, pode-se pensar em um grafo entre os clientes, seus amigos e os filmes que você assistiu ou classificou. Dentro do grafo, podemos procurar conexões que ainda não existam, entre clientes e filmes, onde haja grande chance da nova ligação ser uma ligação forte.

4.2.8 - Modelos de Redução de Dimensionalidade

Modelos para Redução de Dimensionalidade são comuns em grandes bases de dados. Alguns problemas podem conter milhares ou até milhões de variáveis explicativas, o que pode ser computacionalmente caro para trabalhar. Além disso, a capacidade do programa de generalização pode ser reduzida devido à presença de algum ruído na captura das variáveis explicativas.

Algumas variáveis podem ser irrelevantes para a relação subjacente. A redução de dimensionalidade é o processo de descobrir quais variáveis explicativas representam as maiores mudanças na variável resposta. Ela pode ser usada também na visualização de dados. É fácil visualizar, em um problema de regressão, como o preço previsto de uma casa varia conforme seu tamanho. O tamanho da casa pode ser traçado no eixo x, e o preço da casa no eixo y. Do mesmo modo, quando uma segunda variável explicativa é adicionada, por exemplo, o número de banheiros, ela poderia ser plotados no eixo z. Já para um problema com milhares de variáveis explicativas, esta representação visual torna-se impossível.

Um modelo de redução de dimensionalidade tenta eliminar uma grande quantidade de atributos e substituí-los por um conjunto menor, que ainda mantenha muita da informação importante do conjunto original. Um conjunto de dados menor é mais fácil de ser tratado e processado. Algumas vezes, um conjunto menor de atributos pode caracterizar os dados de maneira até mais precisa. Outras vezes, pode acontecer alguma perda de informação. O importante é analisar o custo benefício envolvido.

4.2.9 - Modelos de Causalidade

Uma modelagem causal tenta entender como eventos ou ações realmente influenciam uns nos outros. Por exemplo, imagine que uma consultoria sugeriu segmentar em várias mídias diferentes os anúncios de uma campanha de aumento da eficiência energética. Após as campanhas, observa-se que os clientes de certo segmento apresentaram uma queda no consumo de energia. Então surge a dúvida: o resultado se deu devido à campanha publicitária segmentada, ou eram clientes que reduziriam o desperdício de qualquer maneira? Técnicas de modelagem causal envolvem um investimento substancial na coleta de dados. Experimentos randomizados controlados (por exemplo, os chamados "testes A / B"), ou métodos sofisticados para tirar conclusões causais a partir de dados observacionais levam tempo e custam caro. Ambos os métodos, experimentais e observacionais, geralmente podem ser vistos como análise de "contra-prova", quando se tenta compreender a diferença entre situações mutuamente exclusivas. Um cientista de dados cuidadoso deve sempre incluir na conclusão do trabalho, os pressupostos exatos que foram feitos. Uma empresa precisa pesar o custo/benefício em aumentar ou não o investimento para reduzir as margens de erro, ou decidir, dadas algumas premissas, se as conclusões são boas o suficiente.

Mesmo em um experimento randomizado cuidadosamente controlado, suposições são feitas. A descoberta do "efeito placebo", na medicina, ilustra uma situação notória, onde um pressuposto importante foi esquecido.

4.3 - Conjuntos de Treinamento e Teste

A coleção de exemplos utilizados no treinamento supervisionado de um modelo é chamada Conjunto de Treinamento. Após o aprendizado, o modelo deve ser avaliado quanto a sua capacidade de generalização, ou seja, sua capacidade de acertar novas perguntas, não pertencentes ao conjunto de treinamento. O conjunto de exemplos utilizado para avaliar o desempenho de um modelo é chamado Conjunto de Teste. Modelos de aprendizagem supervisionada aprendem com um conjunto de treinamento, tendo como objetivo seu bom desempenho em novos casos diferentes. O sucesso neste objetivo é estimado através do Conjunto de Testes. [44]

O conjunto de treinamento compreende a experiência que o algoritmo usa para aprender. Em problemas de aprendizado supervisionado, cada observação consiste em uma variável resposta observada, conforme o valor de uma ou mais variáveis explicativas.

O conjunto de teste é uma coleção de observações semelhantes, utilizado para avaliar o desempenho do modelo, usando algum parâmetro de desempenho. É importante que nenhuma das observações do conjunto de treinamento seja incluída no conjunto de testes. Se o conjunto de teste contém exemplos do conjunto de treinamento, será difícil avaliar se o algoritmo realmente aprendeu a generalizar, ou simplesmente memorizou o conjunto de treinamento. Um modelo capaz de generalizar é capaz de executar eficazmente sua tarefa com novos dados. Em contraste, um programa que memoriza os dados de treinamento, cria um modelo excessivamente complexo, que pode prever o valor da variável resposta no conjunto de treino com precisão extrema, mas deixará a desejar quando for submetido a novos casos diferentes. [44]

Memorizar o conjunto de treinamento é chamado de sobre ajuste (*overfitting*). Um programa que memoriza suas observações acaba por absorver as relações e estruturas de ruídos ou simples coincidências do conjunto de treinamento. Equilibrar a precisão no treinamento e a capacidade de generalização nos testes é um problema comum a muitos algoritmos de aprendizado de máquina.

4.4 - Ajuste de Parâmetros e Desempenho

Além dos conjuntos de treinamento e teste, pode-se utilizar um terceiro conjunto de observações, chamado Conjunto de Validação (*holdout*). O Conjunto de Validação é usado para o ajuste de parâmetros que controlam a forma como o modelo é aprendido. O objetivo é melhorar seu desempenho no mundo real. É comum particionar o conjunto de observações em conjuntos de treinamento, validação e teste. Não há requisitos para os tamanhos das partições. Os tamanhos podem variar de acordo com a quantidade de dados disponíveis. É comum alocar 50 por cento ou

mais dos dados para o conjunto de treinamento, 25 por cento para o conjunto de testes, e o restante para o conjunto de validação. [40]

Alguns conjuntos de treinamento podem conter apenas algumas centenas de observações. Outros podem incluir milhões. O baixo custo de armazenamento, o aumento da conectividade de rede e a ubiquidade dos smartphones, repletos de sensores, contribuíram para a situação contemporânea de grandes bases de dados (*Big Data*). Tais conjuntos de dados tiram vantagem do processamento paralelo, e de máquinas multiprocessadas, ou com processadores de vários núcleos, para amplificar o poder preditivo de muitos algoritmos. Na medida em que a quantidade de dados aumenta, melhor fica o desempenho dos modelos. Porém, os algoritmos de aprendizado de máquina também seguem a máxima americana: "*garbage in*, *garbage out*" ("lixo que entra, lixo que sai"). Um algoritmo treinado com uma grande coleção de dados ruidosos, irrelevantes, ou rotulados incorretamente, não terá um desempenho melhor do que um treinado com um conjunto menor, que seja mais representativo dos problemas no mundo real.

4.5 - Validação Cruzada

Muitos conjuntos de treinamento supervisionado são preparados manualmente, ou por processos semiautomáticos. Criar uma grande coleção de dados supervisionados pode ser caro em alguns domínios. Durante o desenvolvimento, uma prática chamada de Validação Cruzada pode ser usada para treinar e ajustar os parâmetros do modelo, validando um algoritmo sobre os mesmos dados. Na validação cruzada, os dados são divididos em grupos (*bins*). O algoritmo é treinado usando todos, menos um dos grupos. Então eles são testados sobre a partição restante. Os grupos são, então, rodados várias vezes, de modo que o algoritmo é treinado e avaliado com todos os dados. O diagrama da Figura 32 descreve a validação cruzada com cinco partições. [44]

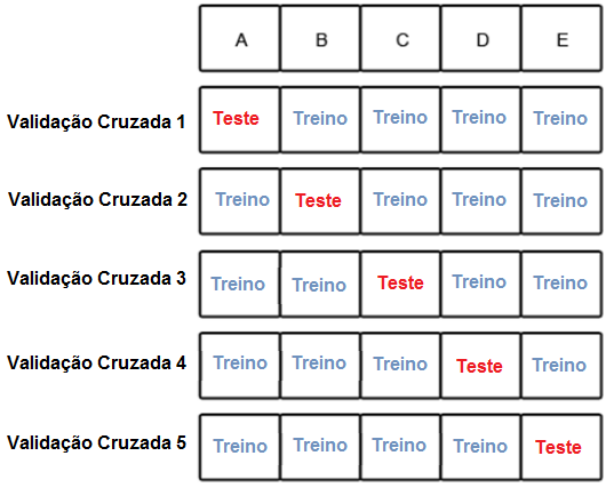


Figura 32 – Exemplo de Validação Cruzada com 5 partições

O conjunto de dados original foi dividido em cinco partições de tamanho igual, identificadas de A até E. Inicialmente, o modelo é treinado com as partições B até E, e então validado com a partição A. Na próxima iteração, a partição B é separada (*holdout*) para validação, e o modelo é treinado com as partições A, C, D e E. As partições são giradas até que o modelo seja treinado e validade em todas as partições. A validação cruzada fornece uma estimativa mais precisa do desempenho do modelo, do que se fosse treinado com uma única partição dos dados.

4.6 - Binômio Viés-Variância

Muitas métricas podem ser usadas para medir se um programa aprendeu a executar sua tarefa da forma mais eficaz. Para problemas de aprendizado supervisionado, muitas métricas usam o número de erros de previsão. Existem duas causas fundamentais para o erro de predição: o viés de um modelo e sua variância.

Um modelo com um alto viés (e baixa variância) irá produzir erros semelhantes, independente do conjunto de treinamento utilizado. O modelo interfere com suas próprias suposições sobre o comportamento dos dados de treinamento. Um modelo com alta variância (e baixo viés), pelo contrário, vai produzir erros diferentes, dependendo do conjunto de treinamento com que é treinado. Um modelo com alto viés é inflexível, enquanto um modelo com alta variância pode ser tão flexível que absorva todo o ruído do conjunto de treinamento. Isto é, um modelo com alta

variância sub-ajusta os dados, enquanto que um modelo com alto viés sobre-ajusta os dados. [45]

Pode ser útil visualizar viés e variância como dardos jogados em um alvo. Cada dardo é análogo a uma previsão a partir de um conjunto de dados diferente. Um modelo com alto viés e baixa variância vai lançar dardos sempre bem agrupados, consistentes, mas com alguma distância do centro do alvo. Um modelo com alta variância e pequeno viés vai lançar dardos distribuídos em todas as direções, mas com um centro de distribuição bem próximo do alvo. [46]

Finalmente, um modelo com baixo viés e baixa variância vai lançar dardos que são fortemente agrupados em torno do alvo. A Figura 33 ilustra todas estas situações. [47]

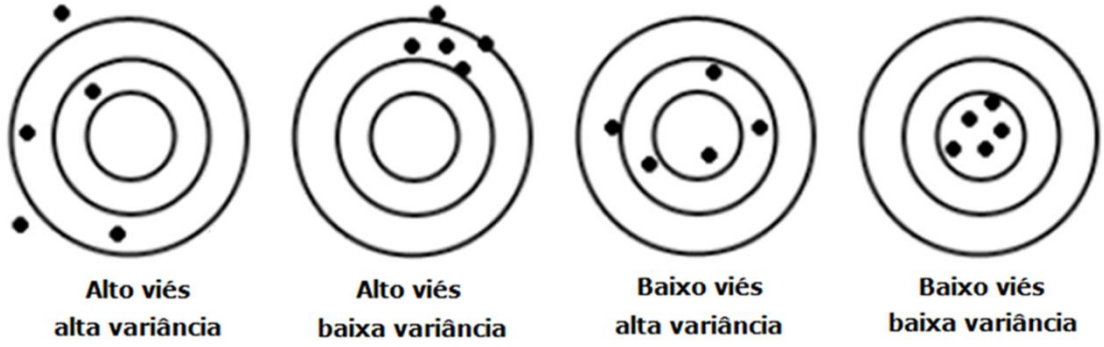


Figura 33 – Diagrama de viés e variância

Um modelo ideal deve ter baixo viés e baixa variância. Porém, os esforços para diminuir um acabam aumentando o outro. Isto é conhecido como balanço viés-variância. Os modelos combinados, abordados nos próximos capítulos, apresentam uma solução para este problema.

4.7 - Modelos de Previsão

A maioria dos métodos de previsão de carga utiliza técnicas estatísticas e algoritmos de inteligência artificial, como regressões, redes neurais, conjuntos difusos e sistemas especialistas. Uma variedade de métodos, como o de Dias Semelhantes, Modelos de Regressão e Séries Temporais, tem sido desenvolvida e aplicada nos diversos horizontes de previsão.

O desenvolvimento e a melhoria das ferramentas matemáticas levaram a criação de técnicas de previsão cada vez mais precisas. Porém, a precisão nas previsões de carga esbarra na dependência das técnicas de previsão meteorológica, previsão do tempo e do clima. As previsões meteorológicas vêm sofrendo grandes avanços, principalmente no curto e longo prazo. Previsões para até três ou mesmo cinco dias à frente, tornaram-se bastante confiáveis. No outro extremo, a evolução do clima em cenários de longo prazo também avançou, graças à demanda por informações e dados relativos ao aquecimento global. Infelizmente, no cenário de médio prazo, as notícias não são tão boas. Este horizonte é o que apresenta as maiores dificuldades e desafios, não só para a previsão do tempo, mas também para a previsão da carga.

Os principais métodos de previsão em uso atualmente são:

- **Dias Semelhantes:** este método é baseado na pesquisa de dados históricos a procura de datas com características semelhantes ao dia a ser previsto. Utiliza-se com mais frequência os históricos de um a três anos passados. São pesquisados atributos semelhantes quanto ao dia da semana, época do ano, variações de temperatura, ocorrência de feriados, vigência ou não de horário de verão, entre outros. A carga prevista é considerada uma extrapolação dos dados históricos. O conjunto de dias semelhantes pode ser combinado via pesos, regressões, coeficientes de tendência e diversos outros esquemas. [48][2]
- **Métodos de Regressão:** O cálculo de regressão é um dos métodos estatísticos mais utilizados. Empregam-se modelos de correlação entre a carga e os diversos fatores como temperatura, dia da semana, classe de consumidor, etc. A precisão dos métodos de regressão é melhor no curto-prazo, normalmente um dia à frente, pois diversos fatores que influenciam a carga são constantes neste horizonte. [49][50][51]
- **Séries Temporais:** a abordagem estatística de séries temporais é a ferramenta mais utilizada na previsão de cargas médias no horizonte de longo-prazo. Parte-se do pressuposto que, no longo prazo, os dados podem ser decompostos em componentes como: tendência linear, sazonalidade e auto correlação. Modelos ARMA (média móvel auto-regressiva) e ARIMA

(média móvel integrada auto-regressiva) e seus respectivos pares, ARMAX e ARIMAX (que incluem a influência de variáveis exógenas), vem sendo usados por anos em áreas como economia e processamento de sinais. Os modelos ARMA supõem uma série de dados estacionária, ou seja, da qual foi extraída seu componente de tendência. Já os modelos ARIMA introduzem os componentes periódicos sazonais. Ambos os métodos dependem somente dos dados de carga para realização da previsão. Já os modelos ARMAX e ARIMAX, tentam incluir a correlação entre a carga e outras variáveis. Porém, esbarram na disponibilidade destes dados, seja no histórico, mas principalmente no horizonte de previsão. A utilização de valores previstos para as variáveis exógenas insere o erro de previsão destas variáveis dentro do erro de previsão da carga [52][53][54].

- **Sistemas Especialistas:** O uso de sistemas especialistas surgiu simultaneamente ao desenvolvimento das primeiras linguagens de programação. Inicialmente, seu uso era limitado a algumas poucas áreas. No final dos anos 80, espalhou-se por todos os campos do conhecimento. Um sistema especialista usa como matéria prima o conhecimento de um especialista humano. Este é traduzido para um software, em forma de regras de produção. Para isso, o conhecimento do especialista deve ser apropriado para ser codificado, ou seja, o especialista deve ser capaz de explicar seu processo de decisão. Um sistema especialista pode conter de centenas até milhares de regras de produção. A previsão baseada em regras utiliza regras de natureza heurística para melhorar sua precisão. Assim, os sistemas especialistas são capazes de fazer previsões automaticamente, sem assistência humana. Os algoritmos desenvolvidos apresentaram melhoras em comparação aos métodos clássicos como Box & Jenkins. [55][56][57]
- **Redes Neurais:** As Redes Neurais Artificiais (ANN) ganharam grande aceitação na previsão de carga elétrica a partir da década de 90 [58] [59][60][61]. As redes neurais funcionam como modelos de regressão não linear. Cada elemento de uma rede é chamado de Neurônio. Estes são dispostos em camadas, conectando as entradas e saídas do sistema. O

número de conexões entre as camadas e a topologia da rede pode variar muito, conforme o design e o método de treinamento escolhido. Uma rede neural utiliza funções com parâmetros que são continuamente modificados durante o aprendizado supervisionado. Ou seja, os pesos atribuídos às entradas de cada neurônio são ajustados de modo que a saída da rede corresponda aos dados históricos do conjunto de treinamento. Atualmente, muitas aplicações vêm usando redes neurais artificiais em conjunto com outras técnicas de inteligência artificial, como algoritmos genéticos, inteligência de enxame ou lógica fuzzy [62][63][64][65].

- **Lógica Fuzzy:** A lógica fuzzy, ou lógica difusa, é uma generalização da lógica clássica usual, onde só se admitem dois estados lógicos distintos e excludentes: "0" ou "1"; “verdadeiro” ou “falso”. Em sua tentativa de modelar de forma mais realista o raciocínio humano, a lógica difusa permite a atribuição de variáveis linguísticas, como por exemplo, “muito alta” ou “muito baixa” a variável da carga de um transformador. Na lógica difusa, há uma transição suave entre os estados lógicos, permitindo que estes assumam qualquer valor real dentro do intervalo de zero a um. Uma das principais vantagens de sua aplicação é a robustez ao ruído do modelo final. Por não exigir entradas exatas, esta lógica pode aplicar um conjunto maior de regras durante seu processo de raciocínio, ponderando os diversos ângulos de um mesmo problema, gerando como saída um resultado mais próximo do esperado [66][67][68][69].
- **Support Vector Machines:** Conhecidas pela sigla SVM, esta é uma técnica estatística mais recente, para solução de problemas de classificação e regressão. Enquanto uma rede neural usa vários neurônios para tentar mapear uma função não linear entre a entrada e a saída, uma SVM usa um conjunto de funções núcleo (*kernel functions*) para delimitar fronteiras de entrada e saída em um espaço de mais dimensões. As funções núcleo são aproximadas por funções lineares simples, criando limites ou fronteiras de decisão lineares no novo espaço. O problema de escolher uma arquitetura para uma rede neural é substituído aqui pelo problema de escolher um conjunto adequado de funções núcleo para a SVM. Sua aplicação em

problemas de previsão de carga elétrica de curto prazo apresenta desempenho compatível com os modelos auto-regressivos. [70][71][72][73]

Capítulo 5
Árvores de Decisão

O método das árvores de decisão é uma das ferramentas mais populares e confiáveis da mineração de dados, sendo capaz de desenvolver modelos preditivos com Big Data. As árvores de decisão são relativamente fácies de construir, pois, foram desenvolvidas para imitar o modo como as pessoas pensam e resolvem os problemas. Além disso, elas podem ser adaptadas para lidar com grandes quantidades de dados, apresentando poucas restrições e necessitando de poucos pressupostos da base de dados. As árvores de decisão podem criar modelos precisos mesmo em bases de dados com ruídos, com dados inconsistentes ou incompletos. A previsão final é transparente, de fácil compreensão e consequentemente de fácil implementação. [74]

O primeiro passo no desenvolvimento de uma árvore de decisão é a determinação de um critério de partição para o nó raiz, e recursivamente, para cada um dos nós seguintes. O conjunto de dados de treinamento é primeiramente particionado na raiz. À medida que a árvore vai crescendo, a divisão dos dados continua nos ramos inferiores da árvore, utilizando-se novos atributos de partição e novos valores limite, ou valores de corte.

A partição do nó raiz é a mais eficiente, pois todos os padrões típicos dos dados estão disponíveis. Conforme o processo avança para os ramos inferiores, a possibilidade de partições distintas diminui. Nos nós terminais, o algoritmo acaba sobre ajustando os dados (o*verfitting*). Por exemplo, nos ramos mais finos de uma árvore, depois de filtrados por inúmeros critérios, os dados já estarão tão homogêneos, que poucos atributos candidatos poderão gerar algum ganho. Infelizmente, uma consequência disso, é a instabilidade do modelo, bem como a queda em seu poder de generalização.

Duas linhas de solução foram desenvolvidas para evitar os problemas de sobre ajuste nas árvores de decisão. A primeira linha foi chamada de Métodos de Poda. A

maioria dos métodos deste tipo apoia-se na minimização matemática das taxas de erro, calculada para um conjunto de dados de validação. À medida que as árvores crescem, o poder de generalização dos nós terminais diminui. A eliminação de alguns destes nós terminais diminui o erro, diminuindo o sobre ajuste e aumentando o poder de generalização da árvore.

A segunda solução proposta foi treinar não só uma árvore, mas um conjunto delas, construindo uma floresta de árvores de decisão. A saída final do modelo composto é uma combinação da previsão de cada árvore. Verificou-se que a combinação de vários modelos fracos (*weak predictors*) gera um modelo combinado bastante robusto e preciso. Ele é bem estável e não sofre sobre ajuste.

As árvores de decisão são um método de mineração de dados usado tanto para classificação como para regressão, que combina tanto a exploração de dados como a modelagem. Uma árvore de decisão consiste em uma sequência de critérios de partição para dividir dados heterogêneos em conjuntos cada vez menores e mais homogêneos. Os critérios de partição são representados por expressões lógicas simples, de fácil compreensão, e que são facilmente traduzidas em qualquer linguagem de programação. Isso torna sua implementação uma tarefa imediata.

5.1 - Árvores Binárias

O conceito de árvore tem sua origem na teoria de Grafos, um ramo da matemática discreta. Um grafo é uma estrutura *G(V,E)*, onde *V* é um conjunto não vazio de objetos denominados Vértices e *E (edges)* é um conjunto de pares de vértices, chamado Arestas. Os grafos são ordinariamente representados através de pontos, ou nós (vértices) ligados por linhas de conexão (arestas), como representado na Figura 34.

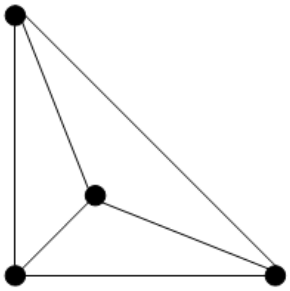


Figura 34 – Representação geral de um grafo

Dependendo da aplicação, as arestas podem ser direcionadas, formando os chamados grafos direcionais. Toda árvore é um grafo, mas nem todo grafo é uma árvore. Um grafo G é dito uma Árvore se satisfaz as seguintes condições:

- G é conexo, ou seja, há sempre um caminho (conjunto de arestas) ligando dois vértices quaisquer;
- G é acíclico, ou seja, não há um caminho fechado ligando um vértice a ele mesmo;
- G deixará de ser conexo se qualquer aresta for removida de G;
- G deixará de ser acíclico se for adicionada mais uma aresta.

Assim, uma árvore é sempre um grafo direcional conexo, acíclico, com N vértices e (N – 1) arestas.

Em um grafo direcional, as arestas formam um par ordenado de vértices denominados vértices de origem e destino. Uma árvore é dita enraizada se possuir um nó identificado de forma especial, chamado Nó Raiz. Ele é considerado o nó inicial ou nó de entrada de uma árvore. Os nós seguintes são chamados de Nós Intermediários, Ramos ou Nós de Divisão. Cada nó intermediário de um grafo direcional tem um conjunto de arestas de entrada e outro de saída. Os Nós Terminais, ou Folhas, não possuem arestas de saída.

Quando todos os vértices intermediários possuem exatamente um vértice de entrada e dois vértices de saída, a árvore é dita uma “Árvore Binária”. As Árvores Binárias são a forma mais comum de Árvores de Decisão.

A Figura 35 exemplifica o desenho de uma pequena árvore de decisão binária.

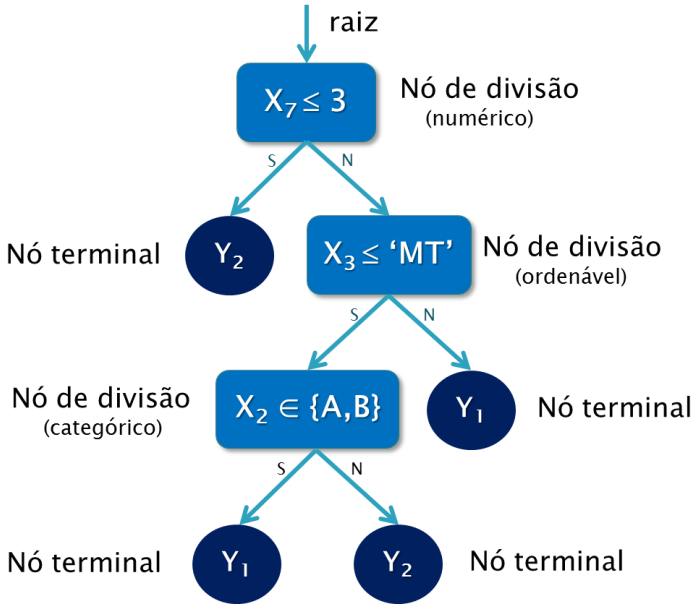


Figura 35 – Exemplo de árvore de decisão binária

5.2 - Os *Stumps*

O princípio básico de indução de uma Árvore de Decisão é o famoso lema: "Dividir para Conquistar". O processo de percorrer uma árvore é feito de cima para baixo, desde o nó raiz até um nó terminal.

A unidade fundamental da árvore é chamada de *Stump (*toco ou bifurcação*)* [75]. Este é representado por um Nó Pai (*Parent Node*) e por dois nós filhos (*Child Nodes*), identificados como Nó Esquerdo (*Left Node*) e Nó Direito (*Right Node*). A Figura 36 ilustra estes conceitos.

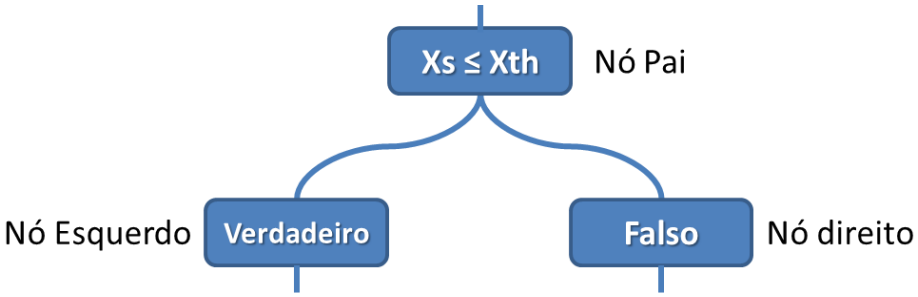


Figura 36 –*Stump*, a unidade básica de uma árvore de decisão

A cada nó pai associa-se uma expressão lógica que pode ser avaliada como verdadeira ou falsa. Quando a expressão é verdadeira, considera-se o nó esquerdo como próximo nó do caminho a ser percorrido. Quando a expressão é falsa, o caminho a percorrer continua pelo nó direito.

A expressão lógica associada ao nó pai é chamada de Critério de Partição (*Split Criterion*), e pode envolver uma ou mais variáveis. No caso mais comum, com apenas uma variável, diz-se que foi realizada uma divisão perpendicular ao eixo da variável em análise. Quando a expressão envolve mais de uma variável, a partição é dita oblíqua.

Quando o critério de partição usa somente uma variável, este pode assumir duas formas comuns, dependendo do tipo de variável em questão. Quando o tipo é ordenável, utiliza-se uma expressão como "Xs ≤ Xth", onde Xs representa a variável de partição (*Split Variable*) e Xth representa o valor limite, ou valor de corte (*threshold value*). Assim, os registros que possuem o atributo de partição Xs até o valor limite serão direcionados para a esquerda. Os registros com valor de Xs superior ao valor de corte serão direcionados para a direita.

Quando a variável de partição Xs é do tipo categórico, o critério de divisão assume a forma: "Xs $\in$ X_{th}List", onde X_{th}List = { X_{th1}, X_{th2}, ... X_{thj} } representa um conjunto de valores limite discretos. Desta maneira, quando o valor de Xs pertencer à lista de valores limites, os dados serão encaminhados à esquerda. Quando o valor de Xs não pertencer a lista, os registros serão encaminhados para direita.

5.3 - Processo de Partição Recursiva

O algoritmo geral para construção de uma Árvore de Decisão, a partir de um conjunto de dados, teve sua estrutura descrita de forma completa por Leo Breiman e outros, em seu livro de 1984 - CART Classification and Regression Trees [74]. O algoritmo CART inicial foi melhorado e estendido de várias formas desde então, mas os princípios básicos permanecem os mesmos.

Inicia-se com um conjunto de dados de treinamento. Este conjunto é composto de N registros, cada um contendo M atributos de entrada e um atributo de saída. A nomenclatura mais comum é chamar as variáveis de entrada de X_1, X_2, X_3...X_M, e a variável de saída Y. Normalmente, um dos atributos é um identificador do registro, chamado ID. Este conjunto de dados de treinamento pode ser representado em um formato de tabela ou planilha como na Figura 37.

ID	X_1	X_2	...	X_M	Y
1					
⋮					
N					

Figura 37 – Representação de uma tabela de dados

Quando a variável de saída Y é do tipo categórico, a árvore gerada é chamada Árvore de Classificação. Quando a variável Y é numérica, com um tipo ordenável, a árvore gerada é chamada Árvore de Regressão.

A construção de uma árvore de decisão é um processo recursivo, feito de cima para baixo (*top-down*). Seu objetivo é, a cada nível e em cada *Stump*, dividir o conjunto de treinamento, conforme um critério de partição. Os dois subconjuntos gerados, o conjunto direito e o esquerdo, devem ser mais homogêneos que o conjunto pai inicial.

Surge assim, a necessidade da definição do conceito de homogeneidade de um conjunto de dados. A esse conceito, dá-se o nome de índice de Impureza. O processo de divisões sucessivas faz com que esta grandeza seja minimizada, conforme se percorre a árvore de decisão. A diminuição do índice de impureza é chamada de Ganho de Informação. Todos estes conceitos possuem definições matemáticas precisas, apresentadas a seguir.

5.4 - Índice de Impureza e Ganho de Informação

Para determinar o critério de partição de cada nó intermediário em uma árvore de decisão, deve-se determinar qual das variáveis de entrada será utilizada como variável de partição (Xs), e qual será o seu valor limite (Xth). A escolha do critério de partição será tal que, crie dois subconjuntos com valores de Y mais homogêneos do que o conjunto inicial de dados. A diminuição da impureza (ou aumento da homogeneidade) é chamada de Ganho de Informação. [76]

O objetivo em cada *Stump* é determinar qual das variáveis de entrada propiciará o maior ganho de informação. Assim, para cada possível variável de partição Xs, encontra-se seu valor limite Xth tal que maximize o ganho de informação. A variável que obtiver o maior ganho de informação será escolhida como variável de

partição do *Stump,* e juntamente com seu valor limite Xth, formará o critério de partição do nó.

Para medir a impureza dos dados, nos problemas de classificação, a medida mais utilizada é a Impureza de Gini (*Gini Impurity*). Nos problemas de regressão, utiliza-se a Variância como medida de dispersão dos dados.

A Impureza de Gini é uma medida de quanto um elemento de um conjunto será incorretamente classificado, caso a classificação seja feita de forma totalmente aleatória, usando como distribuição a própria variável alvo. Seu cálculo é feito pela somatória do produto das frequências de ocorrência relativa com a frequência de não ocorrência. Sua fórmula é:

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) \tag{1}$$

Os valores de impureza de um conjunto atingem um mínimo quando todos os valores do conjunto são iguais (máxima homogeneidade). Quando há mais de um valor diferente no conjunto, o mínimo será quando todos os valores possuírem a mesma frequência no conjunto. Na expressão anterior, f_i representa a frequência relativa do iésimo elemento do atributo Y, no conjunto de dados. Por exemplo, em um conjunto Y com 13 valores temos:

$$Y = \{ a, b, a, a, b, c, a, a, c, b, c, b, d \} \tag{2}$$

As frequências relativas de cada elemento serão:

$$f_a=5/13 \quad , \quad f_b=4/13 \quad , \quad f_c=3/13 \quad e \quad f_d=1/13 \tag{3}$$

Portanto, a Impureza de Gini (I_G) é dada por:

$$I_G(f)= f_a.(1-f_a) + f_b.(1-f_b) + f_c.(1-f_c) + f_d.(1-f_d) \tag{4}$$

$$I_G(f)= 5/13.8/13 + 4/13.9/13 + 3/13.10/13 + 1/13.12/13 = 118/13^2 = 0,6982 \tag{5}$$

Outras fórmulas equivalentes para o cálculo da Impureza de Gini são:

$$I_G(f) = \sum_{i=1}^{m}(f_i - f_i^2) = \sum_{i=1}^{m} f_i - \sum_{i=1}^{m} f_i^2 = 1 - \sum_{i=1}^{m} f_i^2 = \sum_{i \neq k} f_i f_k \quad (6)$$

Por outro lado, nas Árvores de Regressão, quando a variável de saída Y é ordenável, a impureza é dada pela variância da variável alvo (σ^2 ou Var(Y)). A variância mede o quanto um conjunto de dados se distancia da média (μ). A fórmula geral da variância é:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)^2 \quad , \quad \mu = \frac{1}{N}\sum_{i=1}^{N} y_i \quad (7)$$

O valor da variância fica menor à medida que os valores de Y ficam mais homogêneos, apresentando-se mais próximos da média. Uma fórmula alternativa e equivalente para cálculo da variância, sem a necessidade do cálculo prévio da média é:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N} y_i^2 - \frac{1}{N^2}(\sum_{i=1}^{N} y_i)^2 \quad (8)$$

Utilizando-se a função Sum(), deixando implícito os limites do somatório, temos:

$$\mathrm{Var(Y)} = \mathrm{Sum(Y^2)/N} - \mathrm{Sum^2(Y)/N^2} \quad (9)$$

5.5 - Valor Limite

O objetivo em cada *Stump* é encontrar qual a variável de partição Xs que, juntamente com seu valor limite Xth (threshold), formem um critério de partição $Xs \leq Xth$ que maximize o ganho de informação no nó. A Figura 38 representa como será feita a partição dos registros em cada *Stump* da árvore de decisão.

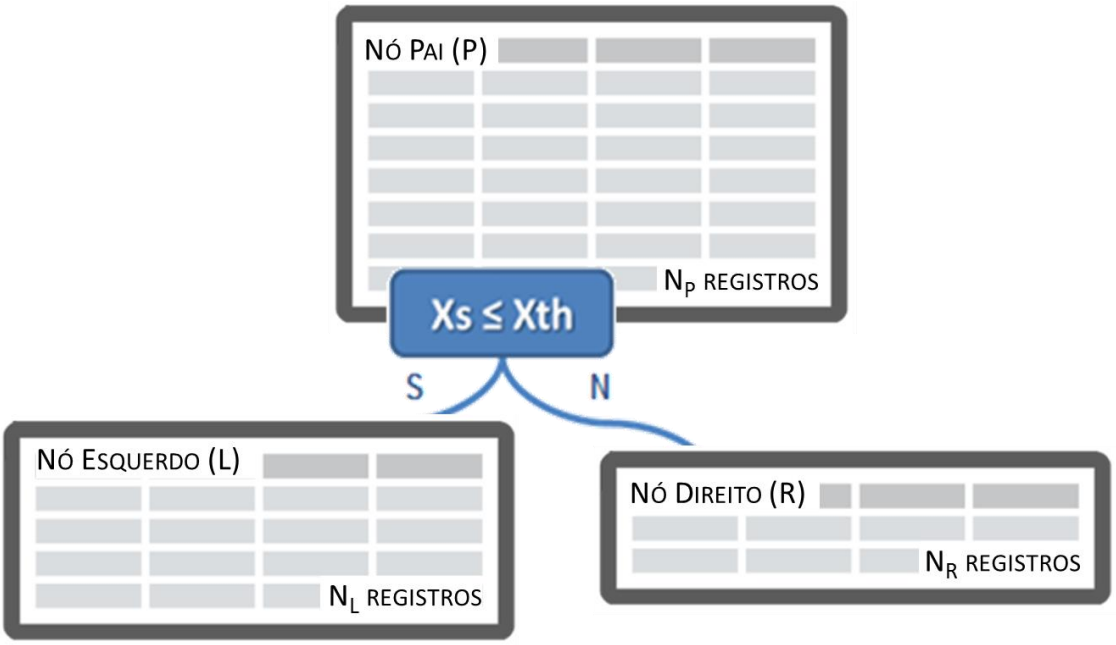


Figura 38 – Partição de dados em uma árvore de decisão

O conjunto de dados do nó pai, contendo N_P registros, será dividido em dois subconjuntos filhos, um esquerdo (*Left*) e outro direito (*Right*), contendo N_L e N_R registros respectivamente.

O Ganho de Informação (GI) é dado pela diferença (ganho) entre a impuridade inicial do nó pai, e a impuridade final dos nós filhos. Quanto maior o ganho de informação, melhor será o critério de partição escolhido. A fórmula para o ganho de informação é:

$$\text{Ganho Informação (GI)} = \text{Impuridade(pai)} - \text{Impuridade(filhos)} \quad (10)$$

No caso de uma árvore de regressão, a impuridade é medida pela variância da variável alvo (Y). A impuridade dos nós filhos é dada pela média ponderada das variâncias de cada um. Então:

$$GI = Var(Y_P) - Var(Y_f) \quad (11)$$

$$Var(Y_f) = N_L/N_P \cdot Var(Y_L) + N_R/N_P \cdot Var(Y_R) \quad (12)$$

$$GI = Var(Y_P) - (N_L/N_P \cdot Var(Y_L) + N_R/N_P \cdot Var(Y_R)) \quad (13)$$

O melhor ponto de corte será aquele que maximize o ganho de informação, ou seja:

$$Xth = ArgMax(GI) \quad (14)$$

Como a variância do nó pai não depende do valor de corte Xth, maximizar o ganho de informação é equivalente a minimizar a variância dos nós filhos $Var(Y_f)$.

$$Xth = ArgMin(Var(Y_f)) \tag{15}$$

Utilizando-se a fórmula alternativa da variância, e a função Sum() para o somatório, pode-se escrever que:

$$\begin{aligned} Var(Y_f) = {} & N_L/N_P \cdot (Sum(Y_L^2)/N_L - Sum^2(Y_L)/N_L^2) + \ldots \\ & N_R/N_P \cdot (Sum(Y_R^2)/N_R - Sum^2(Y_R)/N_R^2) \end{aligned} \tag{16}$$

$$\begin{aligned} Var(Y_f) = {} & 1/N_P \cdot (Sum(Y_L^2) - Sum^2(Y_L)/N_L + \ldots \\ & Sum(Y_R^2) - Sum^2(Y_R)/N_R) \end{aligned} \tag{17}$$

Como Y_p é a união dos conjuntos Y_L e Y_R, é fácil enxergar que:

$$Sum(Y_L^2) + Sum(Y_R^2) = Sum(Y_p^2) \tag{18}$$

Substituindo na equação de $Var(Y_f)$ anterior, tem-se:

$$Var(Y_f) = 1/N_P \cdot (Sum(Y_P^2) - Sum^2(Y_L)/N_L - Sum^2(Y_R)/N_R) \tag{19}$$

Como todas as grandezas do nó pai, N_P e $Sum(Y_P^2)$, não dependem do ponto de corte escolhido, minimizar $Var(Y_f)$ passa a ser equivalente ao problema de maximizar a expressão SyYm definida abaixo:

$$SyYm = Sum^2(Y_L)/N_L + Sum^2(Y_R)/N_R \tag{20}$$

O valor de SyYm (somatório de Y vezes Y médio) pode ser reescrito em termos das médias (Ym) de cada nó filho, resultando no seguinte critério de partição:

$$Xth = ArgMax(SyYm) \tag{21}$$

$$SyYm = Sum(Y_L) \cdot Ym_L + Sum(Y_R) \cdot Ym_R \tag{22}$$

$$\therefore \quad Xth = ArgMax(Sy_L Ym_L + Sy_R Ym_R) \tag{23}$$

Este critério de partição equivalente tem a principal vantagem de permitir encontrar o valor de Xth de forma rápida e sequencial. O processo de busca pelo atributo e ponto de corte que maximize o ganho de informação do *Stump*, pode ser ilustrado pela Figura 39.

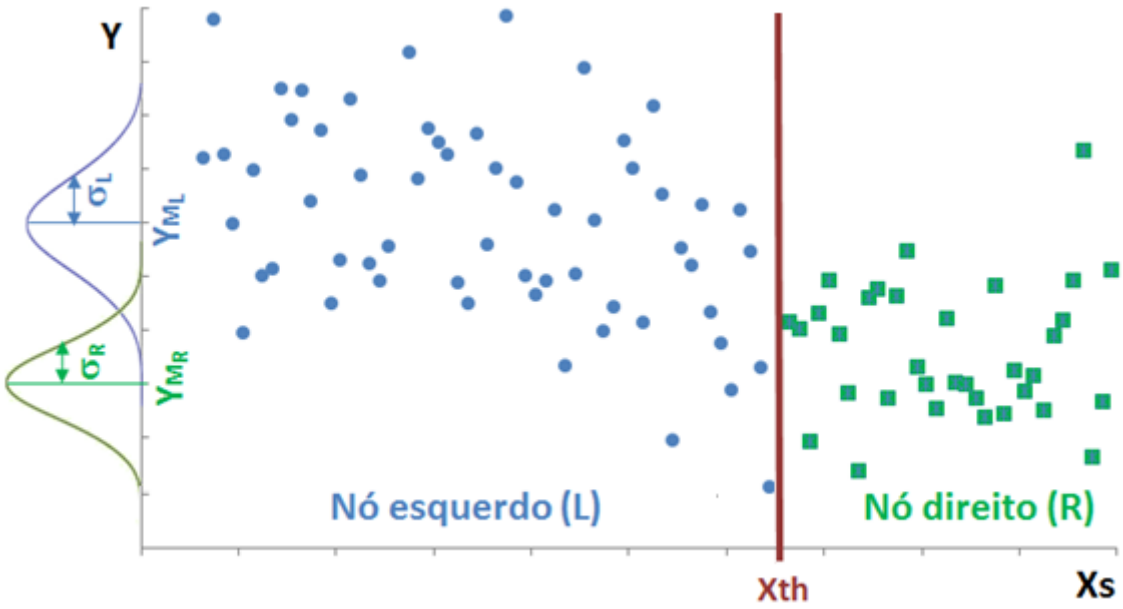


Figura 39 – Busca do valor limite para um atributo de corte

O nó esquerdo (L) contém os registros onde Xs≤Xth, e o nó direito (R) os demais registros. A busca é feita sequencialmente por todos os valores de Xs. Inicialmente, Xth é posicionado no primeiro valor de Xs, ou seja, (L) começa vazio e (R) com todos os registros do *Stump*. O ponto Xth é então movido da esquerda para direita, movendo cada registro de (R) para (L), na ordem de Xs. O valor de Xth que maximize a expressão SyYm será armazenado como valor de corte, juntamente com o atributo Xs. Este algoritmo é apresentado na Listagem 3.

```
1   FUNÇÃO SplitStub_CART(DS) {
2
3   //Entrada: DS=Conjunto de dados no Stump
4   //Saída: Xs= atributo de partição; Xth= valor de corte
5
6       VAR max;
7
8       PARA_CADA atributo candidato Xs em DS{
9           Selecionar os registros em DS, ordenados por Xs
10          Criar dois subconjuntos de dados: (L)eft e (R)ight
11          Inicializar (L)=vazio
12          Inicializar (R) com todos os dados do Stump
13          PARA_CADA elemento Xi em (R) {
14              Removê-lo de (R)
15              Adicioná-lo em (L)
16              Calcular SyYm = SyLYmL + SyRYmR
17              SE SyYm > max ENTÃO Armazenar Xth=Xi e Xs
18          }
19      }
20      RETORNAR (Xs,Xth)
21  }
```

Listagem 3 – Algoritmo de partição de um Stump

5.6 - Critérios de Parada

O processo de partição é recursivo, ou seja, depois de encontrado o critério de partição do nó raiz, o processo continua nos dois nós filhos, e assim, sucessivamente. Isso continua até que um critério de parada seja desencadeado. As seguintes condições são utilizadas como critérios de parada:

- O número de registros do nó chegou a um valor mínimo escolhido;
- O grau de impureza do nó atingiu um mínimo estabelecido;
- A profundidade máxima escolhida para a árvore foi alcançada;
- O ganho de informação não é superior a um limite estabelecido.

5.7 - Árvores de Decisão para Previsão

Quando um conjunto de dados percorre os nós de uma árvore de decisão, eles são avaliados conforme a expressão de partição e encaminhados para esquerda ou para direita, até que atinjam um dos nós terminais. Os dados que atingem um mesmo nó terminal são ditos semelhantes, ou próximos. Isso permite a utilização de uma árvore de decisão com o propósito de previsão.

Um conjunto de casos conhecidos é segmentado e têm seus registros agrupados nos diversos nós terminais de uma árvore de decisão. Quando se obtém um novo registro, pode-se percorrer a árvore de decisão e identificar em qual nó terminal este novo caso cairá. Os casos localizados neste nó terminal podem ser utilizados para previsão do comportamento desta nova instância em análise. [77]

Uma das vantagens intrínsecas das árvores de decisão são sua robustez em relação à presença de valores extremos (*outliers*) ou até mesmo valores com erro, fora de escala ou inconsistentes. O seu processo de segmentação automaticamente posiciona dados atípicos em nós terminais distintos dos dados típicos. Isso representa um ganho enorme, pois na prática, por mais cuidado que se tenha na coleta de dados, sempre haverá casos atípicos, que podem facilmente estragar métodos matemáticos sofisticados, caso não sejam devidamente filtrados.

Outra vantagem das árvores de decisão está na sua capacidade de rastreio e seu bom desempenho. Uma árvore de decisão é capaz de analisar inúmeros casos em pouco tempo, de forma automática e com bons resultados. Os resultados da análise são claros e compreensíveis, tanto para usuários especialistas como para leigos. O caminho percorrido pelos dados na árvore é facilmente traduzido para qualquer linguagem de programação, e podem ser aplicados imediatamente, evitando-se conversões ou adaptações complexas. As regras podem ser expressas como sequências de instruções SQL para recuperar dados de uma determinada categoria em bancos de dados.

Capítulo 6
Técnicas de Big Data

Este capítulo apresenta algumas das técnicas de *Big Data* utilizadas na indução e treinamento de um modelo de floresta aleatória para previsão de carga.

6.1 - Introdução

A primeira geração de algoritmos de aprendizado de máquina foi dominada por algoritmos de Aprendizado em Lote (*Batch Learning*). Estas técnicas processam normalmente pequenos conjuntos de dados, sendo possível acessá-los várias vezes, em qualquer ordem, pois os dados são armazenados na memória. Os métodos consistem basicamente em selecionar uma amostra de dados e gerar um modelo estático de previsão. A premissa básica é que os dados são originados de uma distribuição de probabilidades estacionária. Esta estratégia tem exibido bons resultados para horizontes de previsão mais curtos. Porém, após algumas iterações, conforme o horizonte de previsão aumenta, o desempenho dos modelos começa a se deteriorar, sendo necessário um novo treinamento. Esse problema é consequência do caráter estático dos modelos. [78]

Nos últimos tempos, a atenção dos pesquisadores tem se voltado para a modelagem de ambientes dinâmicos, onde os dados são coletados continuamente, gerando grandes conjuntos de dados (*Big Data*). Nesta nova realidade, os algoritmos são valorizados pela sua habilidade em receber e responder a novos dados. É o chamado Aprendizado Incremental (*Incremental Learning*). Além disso, o processo gerador dos dados pode não ser mais estritamente estacionário, ou seja, o conceito que está sendo modelado pode mudar gradualmente com o passar do tempo. Esse fenômeno é chamado de Deriva Conceitual (*Concept Drifting*). [79]

Sistemas tradicionais, com modelos monolíticos, treinados com um conjunto fixo de dados armazenados em memória, não são aptos a processar a evolução de um grande volume de dados. Eles não são desenhados para reagir a mudanças rápidas, ou

para manter um modelo contínuo de previsão, que seja permanentemente consistente como o estado real do sistema.

6.2 - Fluxos de Dados

Analisar e modelar um Fluxo de Dados (*Data Stream*) é diferente de consultar um banco de dados convencional. As principais características da mineração com fluxos de dados são: [26]

• Os fluxos de dados são virtualmente ilimitados em seu tamanho;

• Os valores dos dados em um fluxo chegam on-line, normalmente em alta velocidade;

• O sistema não tem controle sobre a ordem que os dados chegam, seja dentro de um mesmo fluxo, ou em fluxos diferentes;

• Uma vez que um dado do um fluxo é processado, ele é descartado ou arquivado. Ele não pode ser resgatado de maneira fácil ou rápida, a não ser que seja armazenado temporariamente na memória, que tem um tamanho reduzido, se comparado ao fluxo de dados.

A Tabela 19 ilustra as principais diferenças entre a mineração de dados tradicional e a mineração com fluxos de dados. [28]

	Técnicas de Mineração	
	Tradicionais	**Com Fluxos de Dados**
Volume de dados	Pequeno	Grande
Tempo de processamento	Ilimitado	Restrito
Uso de memória	Ilimitado	Restrito
Acesso a memória	Aleatório	Sequencial
Número de passagens	Várias	Uma
Objetivo da busca	Ponto ótimo	Um dos melhores
Tipo de resultado	Exato	Aproximado
Modelo de execução	Monolítico	Distribuído
Conceito	Estático	Evolutivo

Tabela 19 – Comparação entre técnicas de mineração de dados

Exemplos de aplicações para mineração com fluxos de dados incluem o monitoramento de redes de energia, a mineração de dados na internet, o gerenciamento da infraestrutura de telecomunicações, o sensoriamento remoto de sistemas entre outros. Nestas aplicações, torna-se inviável carregar todos os dados adquiridos em um sistema de banco de dados tradicional, que não seja projetado para realizar consultas contínuas ou exibir valores e resultados instantâneos.

Outro desafio recente é o crescimento exponencial do número de sensores de aquisição de dados. Este aumento pode gerar problemas de escalabilidade nos algoritmos de aprendizado. Além disso, os sensores muitas vezes trabalham em condições adversas, com a presença de ruídos, condições climáticas desfavoráveis, falhas de comunicação, descarga de baterias entre outras. Assim, um fluxo de dados representa um ambiente dinâmico, variável com o tempo e cheio de incertezas e imperfeições. A aquisição dos dados pode ocorrer com diferentes períodos, em diferentes escalas de tempo e com resolução, precisão e velocidades variadas. [27]

A mineração de dados, neste contexto, requer um processamento contínuo dos dados de entrada, monitorando tendências e detectando mudanças. As tarefas de mineração mais comuns nestes novos cenários são: a identificação de perfis de consumidores, a previsão de valores em diferentes horizontes, a previsão de valores de pico, detecção de mudanças no comportamento dos clientes, detecção de falhas ou atividades anormais, bem como a identificação de valores extremos e anormais.

6.3 - Média e Variância com Fluxos de Dados

Para ilustrar a mudança de paradigma entre os métodos tradicionais e os novos métodos de aprendizado, esta seção utiliza o cálculo da média e da variância para apresentar e comparar os algoritmos tradicionais com os algoritmos de mineração em fluxos de dados.

Na abordagem tradicional, partindo-se de um conjunto com N valores de Y, armazenados na memória, o cálculo da média e da variância pode ser feito com um algoritmo semelhante a Listagem 4.

```
1   PARA i = 1 ATÉ N
2       s = s + Y(i)          //soma dos valores Y
3   PRÓXIMO
4   Média = s / N
5
6   PARA i = 1 ATÉ N
7       m2 = m2 + (Y(i)-Média)^2    //soma dos desvios quadráticos
8   PRÓXIMO                         //ou segundo momento central
9   Variância = m2 / N
```

Listagem 4 – Cálculo tradicional da média e variância

Na mineração com fluxos de dados, duas premissas usadas acima não são mais válidas. A primeira é a de que os dados tem um tamanho definido N. Isso significa que não se pode mais usar um laço do tipo "**PARA** i = 1 **ATÉ** N". Em um fluxo de dados, não se sabe antecipadamente quantos valores serão processados. A estrutura de repetição utilizada deve ser um laço "**ENQUANTO**" como na Listagem 5.

```
1   ENQUANTO Fluxo.Ativo {          //Enquanto houver dados no fluxo
2       Y = Fluxo.LerValor()        //Lê um valor do fluxo
3       N = N + 1                   //Conta os valores lidos
4       ...                         //Processa o valor
5       resultado = ...             //Atualiza o resultado
6   }
```

Listagem 5 – Laço de repetição com fluxos de dados

Este trecho de código cria um laço que lê os valores de Y um a um. Dentro dele, o algoritmo deve ser capaz de emitir seus resultados a qualquer momento, pois qualquer um dos dados pode ser o último.

Outra premissa que não é mais válida é que os dados podem ser examinados em várias passagens, ou em qualquer ordem. No algoritmo tradicional, na primeira passagem, é feito o cálculo da média. Em uma segunda passagem, todos os valores são acessados novamente para o cálculo da variância. Na mineração com fluxos de dados, cada valor do fluxo deve ser consumido totalmente, antes que o próximo seja acessado. Enquanto o fluxo está ativo, os valores são lidos sequencialmente, e não ficam armazenados.

Sob estas novas premissas, é necessário um novo método de cálculo para a média e a variância de um fluxo de dados. Uma possível solução é o algoritmo apresentado na Listagem 6.

```
ENQUANTO Fluxo.Ativo {
    Y = Fluxo.LerValor()       //Lê um valor do fluxo
    N = N + 1                  //Conta os valores lidos
    delta1 = Y - Média         //Variável auxiliar delta1
    Média = Média + delta1/N   //Atualiza a média
    delta2 = Y - Média         //Variável auxiliar delta2
    M2 = M2 + delta1 * delta2 //Segundo momento central
    Variância = M2 / N         //Atualiza a variância
}
```

Listagem 6 – Cálculo da média e variância com fluxos de dados

Nesse algoritmo, os valores da média e variância são calculados e atualizados a cada iteração. Após esse cálculo, o próximo valor é lido, e o valor anterior é descartado, pois não é mais necessário. Algumas variáveis auxiliares também são usadas no exemplo. A variável M2 é chamada de segundo momento central, e é acumulada para o cálculo da variância. As variáveis delta1 e delta2 são apenas auxiliares que medem a diferença (delta) entre o valor Y e a média. Deve-se notar que, delta1 é calculado antes da atualização da média, enquanto delta2 é calculado com o novo valor da média.

6.4 - Amostragem sem Reposição

Outro exemplo interessante é a obtenção de amostras, sem reposição, de um conjunto de dados. Neste caso, deseja-se obter uma amostra de k elementos, de um conjunto com um total de Nt elementos, através de k sorteios, sem reposição. No método tradicional, onde todo o conjunto de dados está disponível na memória, podendo ser acessado de forma aleatória, o problema pode ser facilmente resolvido com a utilização de uma lista. Ela irá armazenar k índices, sorteados entre 1 e Nt, sem repetição. A Listagem 7 mostra esta ideia.

```
PARA i = 1 ATÉ k              //Repete k vezes
    FAÇA
       r = randomEntre(1,Nt) //r=Sorteio entre [1,Nt](inclusive)
    ENQUANTO Lista.Contém(r) //Verifica se r já foi sorteado
    Lista.Inserir(r)          //Insere r na lista
    Amostra(i)=Y(r)           //Copia o item sorteado na amostra
PRÓXIMO
```

Listagem 7 – Amostragem sem repetição

Um problema com o algoritmo anterior é que os valores de Y são acessados de maneira aleatória, sendo necessário mantê-los todos na memória, até que o processo de amostragem termine. Outro problema é a necessidade do conhecimento prévio do

tamanho total (Nt) da população. Novamente, essas condições não estão presentes na mineração com fluxos de dados.

Um algoritmo que resolve estes problemas é conhecido como Amostragem por Reserva (Reservoir Sampling). Nele, a amostra é chamada de reserva, e somente ela é mantida na memória. Conforme cada item é lido do fluxo, um sorteio determina se ele será armazenado na reserva ou se será descartado. Este algoritmo é apresentado na Listagem 8. [80]

```
1   //Inicia a reserva (amostra) com k primeiros valores do fluxo
2   PARA i = 1 ATÉ k
3       Y = Fluxo.LerValor()       //Lê um valor do fluxo
4       Amostra(i)= Y              //Armazena o valor na reserva
5   PRÓXIMO
6   N = k
7
8   //Continua processando o fluxo e atualizando a reserva
9   ENQUANTO Fluxo.Ativo {
10      Y = Fluxo.LerValor()    //Lê um valor do fluxo
11      N = N + 1               //Conta os valores
12      r = randomEntre(1,N)    //r=sorteio entre [1,N] (inclusive)
13      SE r <= k               //Se r está dentro da reserva
14          Amostra[r] = Y     //armazena Y na reserva
15  PRÓXIMO
```

Listagem 8 – Amostragem por reserva (*Reservoir Sampling*)

Este algoritmo é composto pelos seguintes passos. Inicialmente, a amostra é inicializada com os primeiros k elementos do fluxo. Os valores seguintes, após serem lidos, passam por um sorteio. Ele determinará se o valor será armazenado ou descartado. Para isso, um número r é sorteado entre 1 e N, inclusive, sendo N a quantidade de valores lidos do fluxo até aquele instante. Se o valor sorteado for menor igual que k, ou seja, se r estiver dentro do intervalo da reserva, o item será armazenado, exatamente na posição sorteada. Se o número for maior que k, o item será descartado.

É interessante notar que, uma vez colocado na reserva, o item poderá ser substituído por um novo valor lido do fluxo, conforme os próximos sorteios forem realizados. A primeira vista, pode parecer que a probabilidade de um item terminar dentro da amostra não é a mesma para todos. Porém, uma análise mais detalhada mostra que, todos os itens lidos do fluxo têm exatamente a mesma probabilidade de

aparecer na amostra. A probabilidade de um valor qualquer Y_i pertencer à amostra é sempre:

$$P(Y_i \in amostra) = \frac{k}{Nt} \tag{24}$$

onde *Nt* é o tamanho total do fluxo, conhecido somente no final do processo.

Para demonstrar isso, pode-se dividir o cálculo da probabilidade P em duas etapas:

$$P(Y_i \in Amostra) = P1 \cdot P2 \tag{25}$$

onde *P1* é a probabilidade do j-ésimo elemento do fluxo ser sorteado para ser guardado na reserva; e *P2* é a probabilidade deste elemento não ser substituído por nenhum outro elemento seguinte, até o final do processo.

A probabilidade *P1* é facilmente calculada, bastando verificar que o j-ésimo sorteio de *r* será no intervalo [1, j]. Como o item somente será armazenado quando *r≤k*, temos:

$$P1 = \frac{k}{j} \tag{26}$$

A probabilidade *P2* será o produto das chances do j-ésimo elemento não ser substituído por nenhum dos próximos elementos, de *(j+1)* até *Nt*. A probabilidade do n-ésimo elemento do fluxo substituir *j* na reserva é de 1/n, pois a substituição só ocorre se o número sorteado, entre 1 e *n,* for igual a posição do elemento na reserva. Portanto, a probabilidade complementar, ou seja, da substituição não ocorrer, é de *(1-1/n) = (n-1)/n*. Assim, *P2* é dado por:

$$P2 = \prod_{j+1}^{N_t} \frac{(n-1)}{n} = \frac{j}{(j+1)} \cdot \frac{(j+1)}{(j+2)} \cdot \frac{(j+2)}{(j+\cdots)} \cdot \ldots \cdot \frac{j+\cdots}{N_t} = \frac{j}{N_t} \tag{27}$$

Portanto:

$$P(Y_i \in amostra) = P1 \cdot P2 = \frac{k}{j} \cdot \frac{j}{Nt} = \frac{k}{Nt} \qquad \texttt{c.q.d} \tag{28}$$

Esta técnica de amostragem pode ser utilizada na indução da floresta aleatória, para sortear quais atributos serão analisados em cada nível das árvores de decisão.

6.5 - Amostragem com Reposição

Outra técnica utilizada na indução de uma floresta aleatória é a de amostras *bootstrap*. Em estatística, uma Amostra *Bootstrap* (*Bootstrap Sample*) é definida como: uma amostra de *N* casos, sorteados aleatoriamente, com reposição, de um conjunto total de *N* elementos. Os dois pontos importantes desta definição são: (a) o tamanho final da amostra é igual ao tamanho da população; (b) o sorteio é feito com reposição. Isso significa que um elemento pode aparecer mais de uma vez na amostra, enquanto outros podem não aparecer. [81]

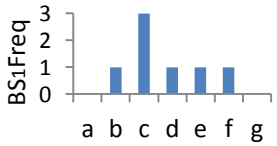
Para ilustrar este conceito, seja um conjunto exemplo X formado por 7 elementos:

$$X = \{ a, b, c, d, e, f, g \} \quad (29)$$

Duas amostras *bootstrap* (BS) do conjunto X poderiam ser:

$$BS_1(X) = \{ f, e, b, c, c, d, c \}$$
$$BS_2(X) = \{ a, a, c, c, f, g, a \} \quad (30)$$

Como são sorteios com reposição, cada elemento pode aparecer de zero até *N* vezes nas amostras. Um conjunto BS(X) pode ser representado em um histograma, associando-se uma frequência (*BS Freq*) a cada elemento:



Y	A	b	c	d	e	f	g
$BS_1Freq(Y)$	0	1	3	1	1	1	0
$BS_2Freq(Y)$	3	0	2	0	0	1	1

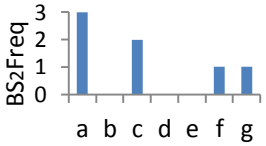


Figura 40 – Exemplo de amostra *bootstrap*

Um algoritmo simples para obter o sorteio com reposição é utilizar um vetor *BSFreq* para armazenar os valores da frequência de repetição de cada elemento do conjunto. Realizam-se N sorteios, e conforme os valores são sorteados, incrementa-se o valor de *BSFreq* do elemento sorteado. O algoritmo da Listagem 9 ilustra este procedimento:

```
1   PARA i = 1 ATÉ N
2       r = randEntre(1,n)      //r=Sorteio entre [1,n] (inclusive)
3       BSFreq(r) = BSFreq(r) + 1
4   PRÓXIMO
```

Listagem 9 – Amostra *bootstrap* com acesso aleatório à memória

O problema com o algoritmo anterior é que todo o vetor de frequências (BSFreq) deve estar carregado na memória, pois ele é acessado em uma ordem imprevisível. Em projetos de *Big Data*, o melhor seria o uso de um algoritmo sequencial. O ideal seria obter um algoritmo semelhante ao da Listagem 10, onde os valores do vetor são atribuídos sequencialmente, em uma única pasagem.

```
1   PARA i = 1 ATÉ N
2       BSFreq(i) = foo(i) //como calcular BSFreq ???
3   PRÓXIMO
```

Listagem 10 – Esquema para *bootstrap* sequencial

Para determinar a fórmula para ser usada no algoritmo anterior, inicia-se com o cálculo da probabilidade de um elemento x_0 ser escolhido em um sorteio com *N* elementos. Essa probabilidade é *1/N*. A probabilidade deste elemento não ser sorteado é *1-(1/N)*. Então, em *N* sorteios sequenciais, com reposição, a probabilidade do elemento ser sorteado *k* vezes repetidas, e não sorteado *(N-k)* vezes, será dada pela fórmula binomial:

$$P(x = x_0, k) = \binom{N}{k} (\frac{1}{N})^k (1 - \frac{1}{N})^{N-k} \tag{31}$$

Quando o conjunto de dados é um fluxo de dados, pode-se assumir que ***N*** tende ao infinito. Assim, a probabilidade de um elemento ser sorteado *k* vezes, de um conjunto virtualmente infinito será:

$$\lim_{N \to \infty} P(x = x_0, k) = \frac{1}{e \cdot k!} \tag{32}$$

Esta formula pode ser usada para elaborar uma tabela de probabilidades P(k), e também da probabilidade acumulada P(≤k), conforme a Tabela 20.

	P(k)	P(≤k)
K	$\frac{1}{e \cdot k!}$	$\sum_{0}^{k} P(i)$
0	0,36788	36,788%
1	0,36788	73,576%
2	0,18394	91,970%
3	0,06131	98,101%
4	0,01533	99,634%
5	0,00307	99,941%
6	0,00051	99,992%

Tabela 20 – Probabilidade de repetição no *bootstrap*

A coluna P(≤k) da tabela anterior mostra a probabilidade acumulada de um elemento ser sorteado até k vezes. Seus valores são chamados de "Os 7 números mágicos das amostras *bootstrap*". Eles permitem atribuir, de maneira sequencial, a frequência com que cada elemento aparece em uma amostra. Basta gerar números aleatórios, uniformemente distribuídos, no intervalo [0,100) e verificar em qual intervalo da probabilidade acumulada ele se localiza. A função que faltava para o algoritmo de *bootstrap* sequencial é apresentada na Listagem 11.

```
1   FUNÇÃO BSFreq(){
2
3         r = randEntre(0,100); //r=Sorteio entre [0,100)
4
5         //Pequisa na coluna P(≤k)
6            SE     r < 36.788 ENTÃO k = 0
7         SENÃO_SE  r < 73.576 ENTÃO k = 1
8         SENÃO_SE  r < 91.970 ENTÃO k = 2
9         SENÃO_SE  r < 98.101 ENTÃO k = 3
10        SENÃO_SE  r < 99.634 ENTÃO k = 4
11        SENÃO_SE  r < 99.941 ENTÃO k = 5
12        SENÃO_SE  r < 99.992 ENTÃO k = 6
13         SENÃO                     k = 7
14
15        RETORNE k;
16  }
```

Listagem 11 – Função frequência de bootstrap

Usando a linguagem SQL, os comandos necessários para criar uma amostra *bootstrap* de uma tabela de dados de treinamento são:

Passo 1) Criar um campo BSFreq na tabela de treinamento:

```
ALTER TABLE YourDataTablea ADD COLUMN BSFreq INT
```

Passo 2) Preencher o campo BSFreq usando os 7 números mágicos:

```
UPDATE YourDataTable
  SET BSFreq = CASE( RAND()*100 )
           WHEN <36.788 THEN 0
           WHEN <73.576 THEN 1
           WHEN <91.970 THEN 2
           WHEN <98.101 THEN 3
           WHEN <99.634 THEN 4
           WHEN <99.941 THEN 5
           WHEN <99.992 THEN 6
           ELSE 7 END
```

6.6 - Partição com Fluxo de Dados

Conforme apresentado no capítulo anterior, o processo de indução de uma árvore de regressão envolve encontrar, para cada nó da árvore, o melhor critério de partição para certo atributo X. O algoritmo apresentado na Listagem 3 realiza uma busca exaustiva por todos os valores de X presentes no conjunto de dados. Mas um dos primeiros passos do algoritmo é a ordenação. Porém, quando se trabalha com um fluxo de dados, a ordenação é um procedimento que deve ser evitado a todo custo.

Existem diversos métodos de ordenação, cada um desenhado para diferentes cenários. Porém, no cenário de *Big Data*, eles tendem a apresentar uma série de inconvenientes como: tempo de processamento muito longo, uso excessivo de memória, necessidade de múltiplas passagens sobre os dados, e acesso aos elementos do fluxo de forma aleatória. [82]

Um dos únicos métodos de ordenação que se adequa parcialmente aos fluxos de dados é o chamado *Bucket Sort, ou Bin Sort* (Ordenação com Baldes, ou Cestos). Neste método, um conjunto de recipientes (*Buckets* ou *Bins*) é criado para armazenar os valores que serão ordenados. Cada recipiente armazenará um subconjunto de dados dentro de um intervalo [min,max) (mínimo incluído e máximo excluído). Um objeto *Bin,* apenas com estas duas propriedades, pode ser inicializado como na Listagem 12.

```
1   PROCEDIMENTO Bin.Init (prmMin, prmMax) {
2       .Min = prmMin
3       .Max = prmMax
4   }
```

Listagem 12 – Inicialização de um objeto *Bin*

Uma coleção sequencial de N *bins*, igualmente espaçados, é usada para armazenar todos os valores do fluxo de dados, dentro de um intervalo total [min,max). Esta coleção é inicializada com o procedimento da Listagem 13.

```
1   PROCEDIMENTO Bins.Init (prmN,prmMin,prmMax) {
2   //Parâmetros: N=qtd objetos, (Min e Max]=intervalo dos dados
3
4       width=(prmMax-prmMin)/prmN    //Largura de cada bin
5       iMin = prmMin                 //variável auxiliar
6
7       //Cria os bins, e insere na coleção
8       PARA i = 1 ATÉ prmN
9           Bins.Insert new Bin.Init(iMin, iMin+width)
10          iMin = iMin+width         //Min do próximo Bin
11      PRÓXIMO
12  }
```

Listagem 13 – Método *Init* para a coleção *Bins*

Conforme cada item for lido do fluxo, ele será inserido em um dos recipientes da coleção. Para encontrar qual o recipiente correto, utiliza-se a função da Listagem 14.

```
1   FUNÇÃO Bins.getMyBinIdx(prmX) {
2   //Parâmetros: prmX=valor de X lido do fluxo
3
4       PARA i = 1 ATÉ Bins.Count
5           SE (Bins(i).Min<=prmX) AND (prmX<Bins(i).Max)
6               RETORNE i
7       PRÓXIMO
8   }
```

Listagem 14 – Método *getMyBinIdx* para a coleção *Bins*

Finalmente, após definidas todas as rotinas acima, a primeira parte de um algoritmo de Bucket Sort para fluxos de dados pode ser escrito como na Listagem 15.

```
1   PROCEDIMENTO BucketSort_DataStream {
2
3       //Cria uma coleção de N Bins, no intervalo [Min, Max)
4       Bins = new Bins.Init(N, Min, Max)
5
6       ENQUANTO Fluxo.Ativo {
7           (X,Y) = Fluxo.LerXY()     //Lê o par ordenado do fluxo
8           i=Bins.getMyBinIdx(X)     //Encontra o Bin que contém X
9           Bins(i).AddPoint(X,Y)     //Insere o ponto (X,Y) no Bin
10      }
11  ...                               // ... continua o processo
```

Listagem 15 – Trecho de *Bucket Sort* com fluxo de dados

Ao final deste trecho, cada recipiente conterá um subconjunto dos dados do fluxo, com seus valores dentro de um intervalo determinado. Tradicionalmente, a ordenação continuaria de forma recursiva, analisando agora o conteúdo de cada recipiente, através de uma nova chamada ao mesmo procedimento. Porém, se isso for feito para um fluxo de dados, incorre-se nos mesmos problemas já citados, como por exemplo, várias passagens pelos dados.

6.6.1 - Método proposto

A solução proposta neste trabalho, para o problema de partição com fluxos de dados, consiste na utilização de uma coleção dinâmica de *bins* e *sub bins*. Esta estrutura é representada na Figura 41.

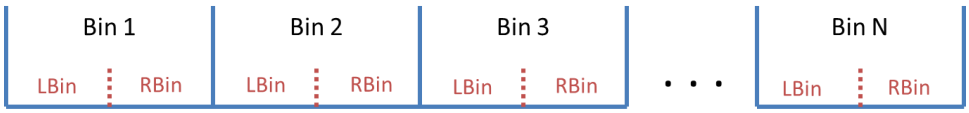


Figura 41 – Estrutura de *bins* e sub *bins* para partição de dados

Cada um dos *bins* principais possui dois sub *bins*, um esquerdo (LBin) e outro direito (RBin), que dividem o *bin* principal em duas partes. Além dos valores máximos e mínimos, cada objeto *bin* passa agora a ter algumas propriedades estatísticas, conforme a declaração da Figura 16.

```
1   Classe Bin {
2
3       var Min          //valor Mínimo do intervalo (inclusive)
4       var Max          //valor Máximo do intervalo (exclusive)
5       var N            //Número de itens no bin
6       var Média        //Média dos valores no bin
7       var M2           //Segundo Momento central dos valores
8       var Variância    //Variância dos valores no bin
9
10      var LBin         //sub bin esquerdo
11      var RBin         //sub bin direito
12  }
```

Listagem 16 – Propriedades de um objeto *Bin*

Sempre que um valor for adicionado ao *bin*, suas propriedades serão atualizadas, e este valor será adicionado também a um de seus sub *bins*. O processo de atualização da média e da variância é o mesmo descrito anteriormente neste capítulo. A Figura 17 mostra este procedimento.

```
1   PROCEDIMENTO Bin.AddPoint (prmX,prmY) {
2
3       .N = .N + 1                      //Atualiza N
4       delta1 = prmY - .Média
5       .Média = .Média + delta1 / .N    //Atualiza a Média
6       delta2 = prmY - .Média
7       .M2 = .M2 + delta1 * delta2      //Atualiza M2
8       .Variância = .M2 / .N            //Atualiza a Variância
9
10      SE (.LBin<>Null) {                //Se possui sub bins,
11          SE (prmX<.LBin.Max)           //adiciona ao sub bin
12              .LBin.AddPoint(prmX,prmY)
13          SENÃO
14              .RBin.AddPoint(prmX,prmY)
15      }
16  }
```

Listagem 17 – Método *AddPoint* para um objeto *Bin*

Durante o processamento do fluxo de dados, haverá uma tentativa de otimizar os intervalos (min,max] dos *bins* da coleção. Para isso, serão utilizadas duas operações na coleção: a divisão (s*plit*) de um *bin*; e a fusão (*merge*) de dois *bins* consecutivos. Em uma divisão, um *bin* principal será substituído pelos seus dois sub *bins*. Em uma operação de fusão, dois *bins* principais serão fundidos, formando um único novo *bin*.

O procedimento de divisão de um *bin* é apresentado na Listagem 18.

```
1   PROCEDIMENTO Bins.SplitBin(prmBinIdx) {
2
3       b=Bins(prmBinIdx)      // b é o bin a ser dividido
4
5       Bins.Insert b.LBin     //Insere b.LBin e b.RBin na coleção
6       Bins.Insert b.RBin
7
8       b.LBin.InitSubBins()   //Cria e inicializa 2 novos sub bins
9       b.RBin.InitSubBins()   //para cada um dos novos bins
10
11      Bins.Remove b          //Remove b da coleção
12  }
```

Listagem 18 – Método *SplitBin* para a coleção *Bins*

A Figura 42 ilustra a coleção após a divisão do segundo *bin*.

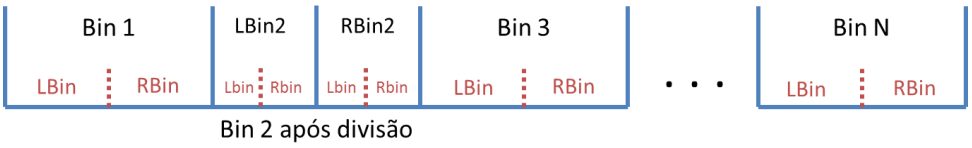


Figura 42 – Operação de divisão de um objeto *Bin*

Note-se que, ao passar os sub *bins* para o nível superior da estrutura, cada um deles deve criar e inicializar dois novos sub *bins*. Para isso, supõe-se uma distribuição

homogênea dentro do *bin*, ou seja, cada novo sub *bin* receberá metade dos valores do *bin* original, a mesma média, a mesma variância, e metade do momento. Isso é feito pelo procedimento da Listagem 19.

```
1   PROCEDIMENTO Bin.InitSubBins() {
2
3       .LBin = new Bin.Init( .Min , (.Max+.Min)/2 )
4       .RBin = new Bin.init((.Max+.Min)/2 , .Max )
5
6       .LBin.N = N/2
7       .LBin.Média = .Média
8       .LBin.M2 = .M2/2
9       .LBin.Variância = .Variância
10
11      .RBin.N = N/2
12      .RBin.Média = .Média
13      .RBin.M2 = .M2/2
14      .RBin.Variância = .Variância
15  }
```

Listagem 19 – Método *InitSubBins* para um objeto *Bin*

A segunda operação de otimização é a fusão (merge) de dois *bins* consecutivos. Ela é descrita no procedimento da Listagem 20.

```
1   PROCEDIMENTO Bins.MergeBin(prmBinIdx) {
2
3       b1=Bins(prmBinIdx)        //b1 e b2 são dois bins
4       b2=Bins(prmBinIdx+1)      //consecutivos para a fusão
5
6       b = new Bin.Init(b1.Min, b2.Max)  //Cria um novo bin e
7       b.InitByMerging(b1,b2)            //inicia suas estatísticas
8
9       b.LBin = b1   //transforma b1 e b2
10      b.RBin = b2   //em sub bins de b
11
12      Bins.Insert b             //Insere o novo bin na coleção
13      Bins.Remove b1            //remove b1 e b2 da coleção
14      Bins.Remove b2
15  }
```

Listagem 20 – Método *MergeBin* para a coleção *Bins*

A Figura 43 ilustra uma fusão do primeiro e segundo *bin*:

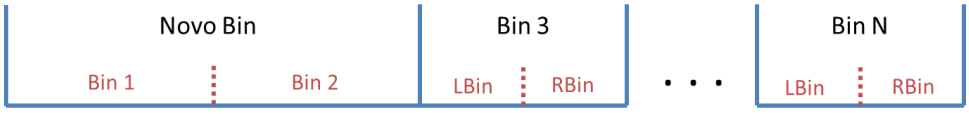


Figura 43 – Operação de fusão de dois objetos *Bin*

Durante a operação de fusão, um novo *bin* é criado, e irá agregar as propriedades dos dois *bins* antigos, que por sua vez se transformam em seus sub *bins*.

Para calcular os valores das propriedades do novo elemento, utiliza-se o procedimento da Listagem 21.

```
1   PROCEDIMENTO Bin.InitByMerging(b1,b2) {
2
3       .N = b1.N + b2.N
4       .Média = (b1.N * b1.Média + b2.N * b2.Média) / .N
5       .M2 = b1.N*(b.Média - b1.Média)^2 + b1.M2 + _
6             b2.N*(b.Média - b2.Média)^2 + b2.M2 )
7       .Variância = .M2 / .N
8   }
```

Listagem 21 – Método *InitByMerging* para um objeto *Bin*

No procedimento anterior, nota-se que a quantidade de pontos do novo *bin* será a soma dos *bins* originais. A média do novo *bin* será a média ponderada dos *bins* originais. O mesmo para o momento, que é a combinação dos momentos originais, e por fim a variância.

Durante o processamento do fluxo de dados, para manter a quantidade de *bins* da coleção constante, toda operação de divisão será seguida por uma operação de fusão. O gatilho para execução destas operações será o ganho de informação (GI). Uma operação de divisão normalmente gera um ganho de informação positivo, enquanto uma operação de fusão gera um ganho negativo. Assim, a otimização (divisão seguida de fusão) só irá ocorrer se o ganho de informação total for positivo. Este procedimento é apresentado na Listagem 22.

```
1   PROCEDIMENTO Bins.Optimize {
2
3       idxToMerge = getBestBinToMerge(ganhoMerge)
4       idxToSplit = getBestBinToSplit(ganhoSplit)
5
6       SE (ganhoSplit+ganhoMerge)>0 {
7           SplitBin idxToSplit
8           MergeBin idxToMerge
9       }
10  }
```

Listagem 22 – Método *Optimize* para a coleção Bins

O bin escolhido para sofrer divisão é o que trará o maior ganho de informação. No capítulo anterior, o ganho de informação foi definido como:

$$GI = Var(Y_P) - (N_L/N_P \cdot Var(Y_L) + N_R/N_P \cdot Var(Y_R)) \qquad (33)$$

A mesma expressão pode ser escrita utilizando-se o segundo momento central da seguinte forma:

$$GI = (M2(Y_P) - M2(Y_L) - M2(Y_R)) / N_P \quad (34)$$

O algoritmo da Listagem 23 mostra como encontrar o *bin* que, quando dividido, gerará o maior ganho de informação:

```
1   FUNÇÃO Bins.getBestBinToSplit(outMaxGanho) {
2
3       PARA i = 1 ATÉ Bins.Count
4           b=Bins(i)        //b é o bin para divisão
5
6           SE (b.LBin.N=0) OR (b.RBin.N=0)
7               iGanho=0     //ganho=zero para bin vazio
8           SENÃO
9               iGanho = (b.M2- b.LBin.M2 - b.RBin.M2)/b.N
10
11          SE (i=1) OR (iGanho>=outMaxGanho) {
12              outMaxGanho = iGanho
13              idxBinToSplit = i
14          }
15      PRÓXIMO
16
17      RETORNE idxBinToSplit
18  }
```

Listagem 23 – Método *getBestBinToSplit* para a coleção *Bins*

Os *bins* consecutivos escolhidos para sofrer fusão também são os que apresentarem o maior ganho de informação. Vale lembrar que, no caso da fusão, o ganho de informação tem seu sinal invertido em relação à divisão. O algoritmo da Listagem 24 mostra como calculá-lo:

```
1   FUNÇÃO Bins.getBestBinToMerge(outMaxGanho) {
2
3       PARA i = 1 ATÉ Bins.Count-1
4           b1=Bins(i)        //b1 e b2 são dois bins
5           b2=Bins(i+1)      //consecutivos para a fusão
6
7           SE (b1.N=0) OR (b2.N=0)
8               iGanho=0      //ganho=zero para bin vazio
9           SENÃO {
10              b = new Bin.InitByMerging(b1,b2)
11              iGanho = ( b1.M2 + b2.M2 - b.M2)/b.N
12          }
13
14          SE (i=1) OR (iGanho>=outMaxGanho) {
15              outMaxGanho = iGanho
16              idxBinToMerge = i
17          }
18      PRÓXIMO
19
20                          RETORNE idxBinToMerge
21  }
```

Listagem 24 – Método *getBestBinToMerge* para a coleção *Bins*

Com o objetivo de diminuir o tempo de processamento do fluxo de dados, o procedimento de otimização da coleção de *bins* somente será tentado após a leitura de um bloco (*chunk*) de valores. Isso também ajuda a aumentar a precisão das estatísticas acumuladas em cada *bin*, evitando-se assim operações de divisão e fusão prematuras, causadas por pequenas diferenças nos ganhos de informação. A Listagem 25 apresenta um exemplo de laço de processamento de um do fluxo de dados, utilizando essa ideia.

```
1   PROCEDIMENTO Bins.LoadDataStream {
2
3       Bins.Init(5,0,6000)          //Inicia uma coleção de 5 bins
4                                    //no intervalo (0,6000]
5       ENQUANTO Fluxo.Ativo {
6           (X,Y) = Fluxo.LerXY()    //Lê o par ordenado do fluxo
7           i=getMyBinIdx(X)         //Encontra o Bin que contém X
8           Bins(i).AddPoint(X,Y)    //Insere o ponto (X,Y) no Bin
9           n = n + 1                //Conta os valores do fluxo
10          SE (n Mod 100 = 0)       //Otimiza a cada 100 valores
11              Bins.Optimize
12      }
13  }
```

Listagem 25 – Método *LoadDataStream* para a coleção *Bins*

O exemplo acima cria uma coleção inicial de 5 *bins*, para processar valores do fluxo no intervalo de zero a 6000. Em seguida, os valores são lidos e processados em blocos de 100 pontos. Após cada bloco, uma tentativa de otimizar a coleção de *bins* é feita. Uma operação de divisão irá separar um *bin* com grande variância em dois menores, que espera-se, tornem-se mais homogêneos. Em seguida, uma operação de fusão irá agrupar dois *bins* consecutivos, que possuem características semelhantes.

Após o processamento de todos os pontos do fluxo, a etapa final é determinar o melhor ponto de corte Xth. Isso pode ser feito analisando-se todos os sub *bins* restantes. Se o procedimento de fusão for aplicado a todos os sub *bins*, até que só restem dois, o ponto de corte será exatamente a fronteira entre esses dois últimos *bins*. Esse procedimento é descrito na Listagem 26.

```
1   FUNÇÃO Bins.getXth() {
2
3       SubBins = new Bins          //Nova coleção de sub bins
4
5       PARA_CADA b EM Bins         //loopd pelos bins principais
6           SubBins.Insert b.LBin   //insere o sub bin esquerdo e
7           SubBins.Insert b.RBin   //direiro na nova coleção
8       PRÓXIMO
9
10      ENQUANTO (SubBins.Count>2)  //Funde bins até restar só 2
11          SubBins.MergeBin(getBestBinToMerge())
12
13      RETORNE SubBins(1).Max      //Xth= ponto entre os 2 bins
14  }
```

Listagem 26 – Método *getXth* para a coleção *Bins*

Concluindo, o método apresentado é capaz de determinar o valor de corte Xth de um conjunto de dados lido de um fluxo, fazendo apenas uma passagem pelos dados, dispensando a ordenação dos dados, com baixo uso de memória, e em um tempo de processamento que não sofre degradação de desempenho com o aumento da quantidade de dados.

6.7 - Algoritmo de Parada Ótima

Os problemas de Parada Ótima (*Optimal Stopping*) são estudados em diversas áreas como estatística, economia e finanças. Seu objetivo é determinar quando parar antecipadamente um processo, com base na observação sequencial de uma variável aleatória, de modo a maximizar um ganho esperado. [83]

O problema clássico desta área é chamado de Problema da Secretária (*Secretary Problem*). Seu enunciado é:

> "Deseja-se contratar uma secretária. Há N candidatas, que serão entrevistadas uma a uma. Durante a entrevista, atribui-se uma nota a cada uma, e deve-se decidir imediatamente se ela será contratada ou não. Caso não seja escolhida, ela não poderá ser contatada novamente. Qual a estratégia que maximiza a probabilidade de contratar a melhor secretária?" [84]

Outra versão equivalente deste problema é o Problema do Turista. Ele diz:

> "Um turista tem dinheiro para tirar apenas uma foto. Seu grupo irá visitar N pontos turísticos em sequência, sendo impossível voltar aos locais já

visitados. Em cada local, o turista deve escolher se tira a foto, ou se aguarda até o próximo ponto. Qual a estratégia que maximiza sua chance de levar para casa a foto do local mais bonito da viagem?" [87]

A estratégia que resolve estes problemas é chamada de Parada Prematura (*early stopping*). Ela consiste em dividir os N candidatos em dois grupos: o grupo R de referência (*benchmarking*), e o grupo D de decisão. Inicia-se analisando o grupo R, armazenando a nota M do melhor candidato neste grupo. Em seguida, é feita a análise dos elementos do grupo D. Neste grupo, assim que um candidato superar o valor de referência M, ele será escolhido, e o processo será encerrado. [85]

O grupo R é formado por uma fração p do número de candidatos, ou seja, possui p*N elementos. O desafio é determinar o valor ótimo p* que maximize a probabilidade de terminar o processo tendo escolhido o melhor candidato.

A ordem com que os candidatos são testados não é previamente conhecida. Mas ela pode ser representada por uma combinação aleatória dos números 1,2,3...N. Existem N fatorial combinações possíveis. Um modo de determinar o valor de p* é somar todas as probabilidades das combinações em que a estratégia resulta na escolha do melhor candidato. Esses casos são listados a seguir, sendo a probabilidade de cada caso igual à interseção (produto) de suas condições.

Caso 1) Quando o segundo melhor candidato aparece no grupo de referência, e o primeiro melhor candidato no grupo de decisão.

Condição	Probabilidade	Valor
2 pertence a R;	$P(2 \in R)$	p
1 pertence a D	$P(1 \in D)$	(1-p)
Probabilidade total do Caso 1:		$P_1(p) = p.(1-p)$

Caso 2) Quando o terceiro melhor candidato está no grupo de referência; o primeiro e o segundo melhores candidatos estão no grupo de decisão, e o primeiro candidato vem antes do segundo.

Condição	Probabilidade	Valor
3 pertencer a R	P(3∈R)	p
2 pertencer a D	P(2∈D)	(1-p)
1 pertencer a D	P(1∈D)	(1-p)
1 vem antes de 2	P(1 antes de 2)	1/2
Probabilidade total do Caso 2:		$P_2(p) = p.(1-p)^2(1/2)$

Caso 3) quando o quarto melhor candidato está no grupo de referência; o primeiro, segundo e terceiro melhores candidatos no grupo de decisão; e o primeiro candidato vem antes dos demais.

Condição	Probabilidade	Valor
4 pertence a R:	P(4∈R)	p
3 pertence a D:	P(3∈D)	(1-p)
2 pertence a D:	P(2∈D)	(1-p)
1 pertence a D:	P(1∈D)	(1-p)
1 vem antes de 2,3	P(1 antes de 2,3)	1/3
Probabilidade total do Caso 3:		$P_3(p) = p.(1-p)^3(1/3)$

Dos casos anteriores, fica fácil escrever a fórmula da probabilidade total do i-ésimo caso, $P_i(p)$, como sendo:

$$P_i(p) = p \cdot (1-p)^i \left(\frac{1}{i}\right) \tag{35}$$

Assim, a probabilidade total *Pt* da estratégia resultar na escolha do melhor caso será igual à soma da probabilidade de todos os N-1 casos:

$$Pt(p) = \sum_{1}^{N-1} P_i(p) \tag{36}$$

$$\therefore Pt(p) = \sum_{1}^{N-1} p \cdot (1-p)^i \left(\frac{1}{i}\right) \tag{37}$$

Para o caso onde N é muito grande, pode-se escrever:

$$Pt(p) = \sum_{1}^{\infty} p \cdot (1-p)^i \left(\frac{1}{i}\right) = -p \cdot \ln(p) \tag{38}$$

O valor de p* que maximize *Pt*(*p*) é, tal que, sua derivada seja zero:

$$Pt'(p^*) = 0 \tag{39}$$

Resolvendo para p* tem-se:

$$Pt'(p^*) = -\ln(p^*) - 1 = 0 \tag{40}$$

$$-\ln(p^*) = 1 \quad , \quad \ln(\frac{1}{p^*}) = 1 \quad , \quad \frac{1}{p^*} = e \tag{41}$$

$$p^* = \frac{1}{e} \approx 0,3679 \tag{42}$$

Esta técnica ficou conhecida como “Estratégia 1/e”, ou estratégia dos 36,8%. Ela pode ser enunciada como: observe aproximadamente um terço (36,8%) dos candidatos, anotando o melhor caso M. Continue a busca até encontrar o próximo candidato melhor que M. Este será o candidato escolhido, e o processo será encerrado. [86][88]

6.7.1 - O novo problema da secretária

Em 2006, J. Neil Bearden [89] propôs e resolveu uma nova variação do problema da secretária. Nesta versão, o objetivo não é contratar somente o melhor candidato. Ele notou que, nos problemas reais, a contratação não é um caso de tudo ou nada. A escolha de um dos melhores candidatos também gera valor a empresa, sendo este valor apenas ligeiramente menor que a contratação do melhor candidato. Assim, nesta nova versão, cada candidato tem um “valor real”, inversamente proporcional a sua posição na classificação dos candidatos. O objetivo, então, passa a ser maximizar o valor gerado à empresa, após uma série de contratações. [90]

Ao resolver este novo problema, Bearden descobriu que o tamanho do grupo de referência cai para $\sqrt{N}$. Essa estratégia ficou conhecida com “pule os $\sqrt{N}$ primeiros e contrate o próximo melhor”. No longo prazo, esta nova estratégia é a que gera o maior valor para a empresa, e diminui consideravelmente o número de entrevistas.

Esta estratégia pode ser usada em vários problemas [91] onde uma coleção de valores V(n) é percorrida, com o objetivo de encontrar o maior valor, mas sem ler toda a sequência. Usa-se os primeiros $\sqrt{N}$ elementos para fixar uma referência M, e então, interrompe-se a busca quando um valor maior que M for encontrado.

A Listagem 27 ilustra este algoritmo.

```
1   FUNÇÃO ParadaÓtima_Vmax() {
2
3       var M = V(1)             //inicia M com o primeiro valor
4
5       PARA i = 2 ATÉ √N        //percorre o conjunto de referência
6           SE (V(i)>M)          //testa o i=ésimo valor, se é >M
7               M=V(i)           //armazena em M o maior valor
8       PRÓXIMO
9
10
11      PARA i = √N+1 ATÉ N      //percorre o conjunto de decisão
12          SE (V(i)>M)          //testa o i=ésimo valor, se é >M
13              SAIR FOR         //encerra o processo (early stop)
14      PRÓXIMO
15
16      RETORNE i
17  }
```

Listagem 27 – Algoritmo de parada ótima

Esta estratégia de parada ótima será aplicada no processo de busca dentro do subespaço aleatório de atributos; parte do algoritmo para indução das árvores da Floresta Aleatória, descrita no próximo capítulo.

Capítulo 7
Florestas Aleatórias

O método das Florestas Aleatórias (*Random Forests*) atingiu grande popularidade entre as técnicas de aprendizagem de máquina. Com o tempo, ficou demonstrado que elas têm um ótimo desempenho em termos de precisão. Além disso, apresentam uma boa robustez em relação a dados com ruídos, e principalmente, são imunes a sobre ajuste (*overfitting)*.

Outras características das florestas aleatórias são:

- Estão entre os métodos combinados mais rápidos de se treinar e avaliar;
- São fáceis de implementar e usar, exigindo poucas premissas da base de dados;
- Conseguem lidar bem com valores em branco e dados inconsistentes;
- São relativamente robustas com relação à presença de aberrações (*outliers);*
- Seu poder de previsão é equivalente, ou mesmo superior, aos métodos de *boosting* [92][93][94], em especial em conjuntos de dados com muitos ruídos;
- Possuem uma medida interna que fornece uma ótima estimativa do futuro erro de previsão [95];
- Podem fornecer, além do valor previsto, um intervalo de confiança;
- Permitem calcular um grau de importância e correlação entre os atributos [96].

As florestas aleatórias pertencem à família dos chamados Métodos Combinados, ou Métodos Compostos. Nestes casos, ao invés de apenas um modelo, emprega-se um conjunto deles, com o objetivo de conseguir resultados melhores. Uma floresta aleatória é formada por um conjunto (ou floresta) de árvores de decisão. Para obter árvores diferentes, são inseridas perturbações aleatórias durante o processo de criação de cada árvore. Cada árvore é treinada a partir de uma amostra aleatória do conjunto de treinamento.

Apesar de uma única árvore de decisão representar um modelo de baixo viés, normalmente sua variância é alta. Mas graças à combinação das saídas de todas as árvores da floresta, consegue-se baixar a variância final, preservando-se o viés baixo.

7.1 - Modelos Combinados

Um modelo de árvore de decisão é fácil de entender e fácil de gerar. Mas ele pode ser sensível a pequenas variações do conjunto de treinamento. Se isso ocorrer, a árvore de decisão pode se tornar um modelo instável.

Por exemplo, seja um conjunto de dados com dois atributos, que só se diferenciam em um a cada quinhentos registros. A qualidade da partição feita por um ou por outro atributo será praticamente idêntica. Ambas as variáveis podem fazer um bom trabalho em separar os dados. Porém, dependendo de como os dados forem analisados, uma das variáveis será escolhida como variável de partição. Nos níveis seguintes, possivelmente, a outra variável será ignorada, uma vez que, provavelmente, não ajudará muito mais na partição dos dados. Isto é uma vantagem do método, pois significa que a presença de variáveis correlacionadas é compensada automaticamente. No entanto, a seleção de uma ou de outra variável de partição foi feita com base em pequenas diferenças na base de dados. Além disso, como a correlação não era perfeita, as partições posteriores podem ser completamente diferentes, dependendo de qual variável foi escolhida. Isso pode resultar em árvores consideravelmente diferentes, e em um modelo instável.

Para aumentar a estabilidade dos resultados, ao invés da construção de um único modelo, vários métodos de mineração de dados se baseiam na utilização de Métodos Combinados (*Ensemble Methods*). Métodos de previsão que combinam um conjunto de modelos estão entre as mais importantes linhas de pesquisa na área de aprendizado de máquina. O poder de previsão demonstrado pelos modelos combinados, em muitas aplicações, é comparável, ou mesmo superior, a um modelo singular muito mais complexo e sofisticado. [97]

O método das florestas aleatórias usa várias amostras diferentes do conjunto de treinamento (*Bootstrap Samples*) para crescer árvores de decisão diferentes, uma

para cada amostra. Além disso, as variáveis candidatas, que serão usadas no processo de partição em cada nível, são sorteadas aleatoriamente, com o objetivo de descorrelacionar a estrutura de cada modelo (*Random Feature Subspace*). Uma vez criada a floresta, cada árvore irá votar para determinar qual o valor final da previsão (*Aggregating*). A Figura 44 representa este esquema.

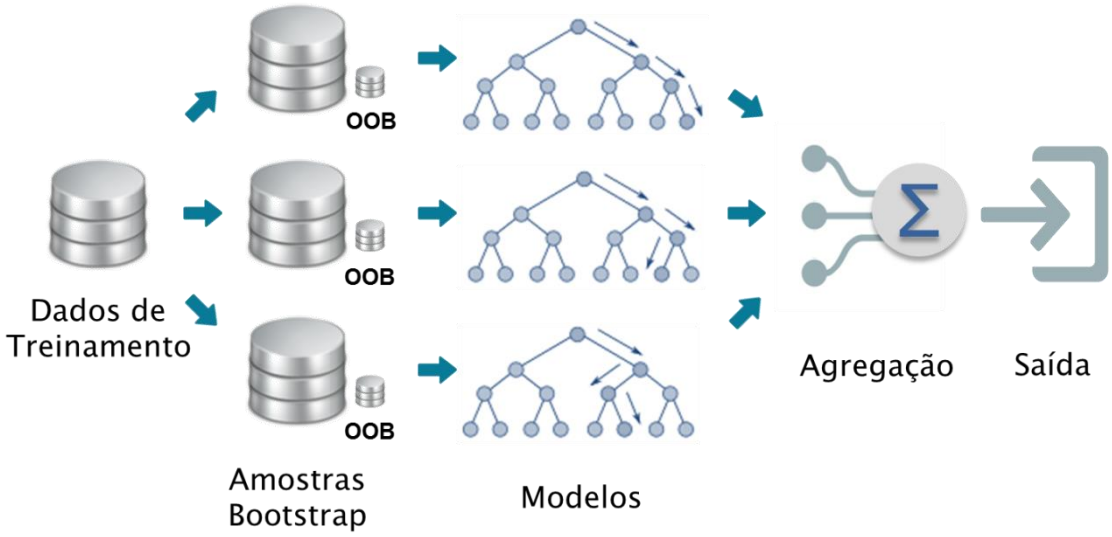


Figura 44 – Representação de uma floresta aleatória

Em problemas de classificação, cada árvore de decisão vota para escolher a classe mais popular, que representa a resposta final do modelo. Em problemas de regressão, a agregação final será dada, por exemplo, pela média dos valores de saída de cada árvore.

Obviamente, cada árvore terá um poder maior ou menor de previsão, para cada tipo de entrada ou para cada subconjunto de dados. Devido às duas fontes de aleatoriedade usadas na construção das árvores, cada árvore isolada é um previsor fraco (*Weak Learner*). Porém, a combinação de vários previsores fracos, pode resultar em um previsor de ótima qualidade (*Strong Learner*).

Leo Breiman [98] demonstrou que o desempenho ótimo de um modelo combinado pode ser obtido através da inserção de aleatoriedade, com o objetivo de diminuir a correlação entre os componentes, mas mantendo sua precisão. Em termos do binômio viés/variância, isso significa que cada modelo deve ter máxima variância, mas mantendo um viés baixo. Usando a lei dos grandes números, Breiman demonstrou

que estes modelos sempre convergem, à medida que se se adicionam mais elementos ao conjunto. Isso significa que o modelo combinado nunca irá sofrer sobre ajuste.

7.2 - Histórico das Florestas Aleatórias

Historicamente, a primeira menção a conjuntos de árvores de decisão foi feita por Kwok e Carter [99]. Estes autores observaram empiricamente que uma média de várias árvores de decisão, com estrutura diferente, consistentemente produzia melhores resultados do que qualquer um dos membros do conjunto. Esta abordagem, no entanto, não se baseava em randomização, nem foi inteiramente automática. As árvores de decisão foram geradas manualmente, selecionando-se variáveis de corte que haviam sido descartadas durante o processo de indução chamado ID3.

Mais tarde, Breiman [100] foi um dos primeiros a mostrar, tanto teoricamente como empiricamente, que agregar múltiplas versões de um mesmo modelo pode dar ganhos substanciais de precisão. Ele observou e mostrou que, o "Modelo Médio" possui um erro esperado de generalização menor do que o erro de cada um de seus componentes. Breiman criou então o algoritmo *Bagging: Bootstrap Aggregation*. Este algoritmo consiste em criar vários modelos a partir de diferentes amostras do conjunto de aprendizagem, e posteriormente agregar seus vários resultados.

O processo de amostragem utilizado, batizado de *Bootstrap Sample*, foi desenvolvido por Efron [101]. Partindo de um conjunto de N registros, o método é capaz de extrair infinitas amostras diferentes, também de tamanho N. Para isso, ele realiza N sorteios com reposição. Apesar de possuir o mesmo tamanho do conjunto original, uma amostra *Bootstrap* poderá ter elementos repetidos causados pelo sorteio com reposição. Por outro lado, haverá também elementos do conjunto de dados que nunca serão sorteados em uma amostra. Em média, 37% dos elementos originais estarão ausentes em cada amostra. Este subconjunto de dados não utilizados na criação da árvore foi batizado de dados OOB (*Out of Bag*). Como estes dados não foram utilizados na criação do modelo, eles podem ser utilizados para validação, semelhante ao processo de validação cruzada. Isso permite calcular uma boa estimativa do erro e do poder de generalização da árvore.

O pseudocódigo para criação de uma floresta com *Bagging* é resumido na Listagem 28.

```
1   FUNÇÃO CriarÁrvoresBagging(DS,F) {
2
3   //Entrada: DS=Conjunto de dados de treinamento
4              F=número de árvores da floresta
5   //Saída: Trees=array de árvores de decisão
6            MSE = array dos erros médios quadráticos
7
8       PARA i = 1 TO F {
9           BS = Criar uma amostra Bootstrap de DS;
10          OOB = Dados de DS não selecionados em BS;
11          Trees(i) = induzir uma árvore de decisão usando BS;
12          MSE(i) = calcular o erro de Trees(i) nos dados OOB;
13      }
14
15      RETORNE Trees, MSE
16  }
```

Listagem 28 – Geração de uma floresta com *Bagging*

Estendendo o trabalho de Kwok e Carter [99], Dietterich e Kong [45] propuseram uma ideia mais radical: randomizar a escolha do valor limite de corte em cada nó, selecionando, ao acaso, uma das 20 melhores divisões do nó. Dietterich [92] mostrou empiricamente que esta instabilidade gera resultados ligeiramente melhores do que o *Bagging*, em configurações de baixo ruído. Do ponto de vista da tendência, este método praticamente não altera o viés, mas aumenta a variância, devido à randomização. As experiências mostraram, no entanto, que quando o ruído é importante, *Bagging* geralmente produz resultados melhores.

Amit et al.[102], no contexto do reconhecimento de caracteres manuscritos, onde o número de variáveis de entrada "*M*" é tipicamente muito grande, propôs uma outra variante randomizada do algoritmo de indução de árvores. Ela consiste na indução de árvores utilizando-se apenas uma amostra aleatória das variáveis de entrada. Denotando como "*K*", o número de variáveis efetivamente consideradas, esta variante substitui o algoritmo de *Bagging* pelo algoritmo da Listagem 29.

```
FUNÇÃO  CriarÁrvoresParciais(DS,F,M,K) {

//Entrada: DS=Conjunto de dados de treinamento
            F=número de árvores da floresta
            M=número total de atributos dos dados
            K=número de atributos candidados, entre 1 e M
//Saída: Trees=array de árvores de decisão

    PARA i = 1 TO F {

        //Sorteia k atributos candidatos Xc
        PARA i = 1 TO K {
            Xc(i) = rand(1,M)  //sorteio entre [1,M]
        }

        P = Subconjunto dos dados somente com os atributos Xc
        Trees(i) = induzir uma árvore de decisão usando P;
    }

    RETORNE Trees
}
```

Listagem 29 – Partição com subconjunto de atributos

Quando a saída Y pode ser explicada de várias maneiras, esse algoritmo gera árvores estruturalmente diferentes, embora individualmente boas. O viés é apenas ligeiramente aumentado, mas a variância final do modelo é minimizada, graças ao cancelamento da média final (*Aggregation*).

O balanço viés/variância pode ser calibrado, ajustando-se o valor de "*K*", entre 1 e "*M*". Quando "*K*" tende a 1, o viés aumenta. Mas como também aumenta a variância dos modelos individuais, o processo de media é mais efetivo. Por outro lado, quando "*K*" tende para "*M*", o viés diminui. Mas como também diminui a variância dos modelos individuais, o benefício final da média é menor.

Inspirado nessas ideias, Ho [103] propôs o Subespaço Aleatório de Atributos (*Random Feature Subspace*) para construção de uma floresta de decisão. Aqui, o sorteio das "*K*" variáveis de entrada é feito em cada partição da árvore, em contraste com a utilização de um único sorteio para cada árvore. Como avaliado empiricamente em várias ocasiões [103][104][105], este método atingiu um desempenho próximo ao estado da arte em muitos problemas. Novamente, o balanço entre viés/variância pode ser controlado ajustando-se o tamanho "*K*" do subconjunto aleatório de atributos. O algoritmo de partição de um *Stub* utilizando o espaço aleatório de atributos é representado na Listagem 30.

```
FUNÇÃO SplitStub_RandomFeatureSubspace(DS,M,K) {

//Entrada: DS=Conjunto de dados no Stump
           M=número total de atributos dos dados
           K=número de atributos candidados, entre 1 e M
//Saída: atributo de partição Xs e valor de corte Xth

    //Sorteia k atributos candidatos Xc
    PARA i = 1 TO K {
        Xc(i) = rand(1,M)  //sorteio entre [1,M]
    }

    VAR max;

    PARA i = 1 TO K {
        Selecionar os registros de DS, ordenados por Xc(i)
        Criar dois subconjuntos de dados: (L)eft e (R)ight
        Inicializar (L)=vazio
        Inicializar (R) com todos os dados de DS
        PARA_CADA elemento Xi em (R) {
            Removê-lo de (R)
            Adicioná-lo em (L)
            Calcular SyYm = SyLYmL + SyRYmR
            SE SyYm > max ENTÃO Armazenar Xth=Xi e Xs=Xc(i)
        }
    }
    RETORNAR (Xs,Xth)
}
```

Listagem 30 – Partição com subespaço aleatório

Em seu trabalho seminal, Breiman [106] descreveu as Florestas Aleatórias (*Random Forests*), combinando seu antigo *Bagging* com as ideias de Ho [103] para seleção aleatória de variáveis em cada nó. Ao injetar aleatoriedade, simultaneamente, tanto nas amostras *Bootstrap*, como nos atributos de partição, Breiman produziu um dos métodos mais eficazes para o aprendizado de máquinas. Surpreendentemente, este método funcionou bem para quase qualquer tipo de problema, nas mais variadas áreas do conhecimento. O autor mostrou, empiricamente e teoricamente, que, enquanto cada árvore de decisão é projetada para reduzir o viés, a floresta como um todo se concentra na redução de variância. O resultado é um modelo final com ótimos resultados: baixo viés e baixa variância.

Embora os princípios originais tenham sido desenvolvidos por vários autores, Leo Breiman é frequentemente citado como o pai das *Random Forests*. Certa parte deste reconhecimento é devido à análise teórica pioneira que sempre complementou sua análise empírica de algoritmos. Outra razão, em contraste com outros autores, foi sua implementação eficiente em software disponibilizado gratuitamente [107]. Isso

permitiu que usuários fora da comunidade de aprendizagem de máquinas aplicassem rápida e facilmente as florestas aleatórias em seus problemas.

Tentando acelerar o processo de cálculo dos valores limites em cada nó, Cutler e Zhao [108] propuseram o método das Árvores de Decisão Perfeitamente Aleatórias (*PERT - Perfect Random Tree Ensembles*). Nele o valor limite de corte para uma dada variável é escolhido aleatoriamente. Mais especificamente, com K=1, após sorteada a variável de corte, o ponto de corte Xth é definido assim: sorteiam-se dois registros com X_{s1} e X_{s2}, e valores de saída diferentes ($Y_1 <> Y_2$). O valor de corte será definido como um ponto intermediário, sorteado com probabilidade uniformemente entre os pontos X_{s1} e X_{s2}. O algoritmo de corte então fica como na Listagem 31.

```
1   FUNÇÃO SplitStub_PERT(DS.M,K) {
2
3   //Entrada: DS=Conjunto de dados no Stump
4              M=número total de atributos dos dados
5   //Saída: atributo de partição Xs e valor de corte Xth
6
7       //Sorteia um atributo de corte Xs
8       Xs = rand(1,M)
9       Sortear dos registros com Xs1 e Xs2 e Y1≠Y2
10      α = rand(0,1)
11      Xth= (1- α). Xs1 + α. Xs2
12
13      RETORNAR (Xs,Xth)
14  }
```

Listagem 31 – Algoritmo de corte para árvores PERT

A indução da árvore prossegue usando tais divisões aleatórias até que todos os nós se tornem puros, ou que não seja mais possível extrair amostras com diferentes valores de saída. Do ponto de vista prático, o PERT é um modelo combinado muito fácil de codificar, e muito eficiente, uma vez que não há critério de impureza para avaliar a cada nó. Em relação à precisão, as comparações experimentais de Cutler e Zhao [108] mostram que o *PERT* é, muitas vezes, tão bom quanto as Florestas Aleatórias de Breiman [106], apesar de gerar árvores maiores do que árvores de decisão cultivadas com menos randomização. Sua principal vantagem é que, devido a sua simplicidade, o método permite uma análise teórica adequada, conforme demonstrado por Zhao [109].

Conforme investigado por Wehenkel [110] e Geurts & Wehenkel [111], a variância alta das árvores de decisão explica-se devido à alta dependência dos cortes

com relação à natureza aleatória do conjunto de treinamento. Os autores mostram empiricamente que, para o caso de variáveis de entrada ordenáveis, a variância do ponto de corte Xth pode, de fato, ser muito alta, mesmo para grandes amostras. Em particular, Geurts [47] mostrou que a variação do ponto de corte parece ser responsável por uma parte significativa do erro de generalização das árvores de decisão.

Como forma de suavizar o limite de corte, Geurts et al. [112], propôs então as Árvores Extremamente Aleatórias *(Extremely Randomized Trees*). Neste método, o ponto de corte Xth é sorteado entre o mínimo e máximo da variável de corte Xs, presentes no subconjunto de dados. Os autores propõem então um procedimento muito eficaz, descrito na Listagem 32.

```
1   FUNÇÃO SplitStub_ERT(DS, M) {
2
3   //Entrada: DS=Conjunto de dados no Stump
4              M=número total de atributos dos dados
5   //Saída: atributo de partição Xs e valor de corte Xth
6
7       //Sorteia o atributo de corte Xs
8       Xs = rand(1,M)
9       Xs1 = min(Xi ∈ Xs)
10      Xs2 = max(Xi ∈ Xs)
11      α = rand(0,1)
12      Xth= (1- α). Xs1 + α. Xs2
13
14      RETORNAR (Xs,Xth)
15  }
```

Listagem 32 – Algoritmo de corte para árvores ERT

No caso especial em que K=1, Árvores Extremamente Aleatórias reduzem-se para Árvores Totalmente Aleatórias, nas quais uma única variável Xs, e um limiar de corte Xth, são aleatoriamente escolhidos em cada nó. O ponto mais interessante aqui é que, a estrutura das árvores pode ser aprendida de forma não supervisionada, ou seja, independentemente da variável de saída Y. Nesta configuração, as árvores extremamente aleatórias estão muito próximas das PERT de Cutler e Zhao [108], uma vez que ambos sorteiam Xs e Xth ao acaso. Porém, estes métodos não são estritamente equivalentes, uma vez que sorteiam Xth de distribuições de probabilidade diferentes.

Em uma nova direção, Rodriguez et al. [113] propôs o método das Florestas de Rotação (*Rotation Forests)* para gerar árvores de decisão com base na eliminação de atributos. Como em *Bagging*, as árvores de decisão individuais são construídas com amostras *Bootstrap* do conjunto de treinamento. Mas para aumentar a diversidade, em cada amostra *Bootstrap*, as variáveis de entrada são divididas aleatoriamente em "*Q*" subconjuntos. Uma Análise de Componentes Principais (PCA) é executada separadamente em cada subconjunto, computando todos os componentes principais das projeções de "*Q*". Desta forma, os dados da amostra são transformados independentemente, de forma linear, em um novo espaço de entrada, usando rotações nos eixos de "*Q*". Conforme relatado por Rodriguez et al. [113] e Kuncheva & Rodríguez [114], as Florestas de Rotação se comparam favoravelmente com outros métodos combinados, baseados em árvores de decisão, produzindo resultados que são as vezes tão bons quanto as *Random Forests* de Breiman [106]. Porém, em termos de complexidade, a sobrecarga computacional devido às rotações de eixo "*Q*" não deve ser ignorada, principalmente quando a velocidade e os recursos computacionais são limitados.

7.3 - Subespaço Aleatório de Atributos

Em uma árvore de decisão, o objetivo em cada nó é escolher qual o critério de partição que trará o maior ganho de informação. Quando a quantidade de atributos de entrada é pequena, pode-se fazer uma pesquisa em todo o conjunto de atributos. Para cada atributo X_i, calcula-se seu valor limite Xth_i. O critério de partição $X_i \leq Xth_i$ será avaliado conforme o ganho de informação que ele gera no conjunto de dados. O atributo que apresentar o maior ganho de informação será o escolhido para aquele nó.

Porém, em problemas de Big Data, é comum encontrarmos conjuntos de dados representados por tabelas não só com muitos registros, mas também com muitas colunas. Nestes casos, analisar todos os atributos candidatos pode ser computacionalmente caro. Em aprendizado de máquina e estatística, chama-se Seleção de Variáveis, ou Seleção de Atributos (*Feature Selection*), o processo de escolha de um subconjunto de atributos relevantes para uso na construção de um modelo. Técnicas de seleção de atributos são usadas por várias razões. Uma delas é a

simplificação dos modelos, para torná-los mais fáceis de interpretar pelos usuários. Outra é o menor tempo de treinamento, bem como a melhor capacidade de generalização.

A premissa central, quando se utiliza uma técnica de seleção de atributos, é que os dados normalmente contêm muitas características que são, ou redundantes, ou irrelevantes. Assim, a remoção de alguns atributos pode ser feita sem incorrer em muita perda de informação.

Características redundantes ou irrelevantes são duas noções distintas. Um atributo é dito redundante somente na presença de outra característica relevante com a qual é fortemente correlacionada. Já um atributo é dito irrelevante quando seu poder de previsão é desprezível, ou seja, tem pouca ou nenhuma correlação com a variável de saída.

Existem várias técnicas de seleção de atributos. Uma das primeiras técnicas foi chamada de Redução da Dimensionalidade (*Dimensionality Reduction*). Neste método, buscam-se descobrir quais são os atributos mais importantes, ou seja, quais representam melhor a variação do atributo alvo. Outra técnica é combinar grupos de atributos, substituindo estes por um único. Uma extensa área de pesquisa chamada *Rough Sets* tem como objetivo encontrar um núcleo de atributos que podem determinar as variações do atributo de saída.

Em 1995, Tin Kam Ho [103], no AT&T Bell Labs., propôs uma abordagem diferente, envolvendo o chamado Subespaço Aleatório de Atributos (*Random Feature Subspace*). A ideia original foi utilizar uma seleção de atributos candidatos feita de maneira aleatória. Ao invés de uma buscar exaustivamente em todos os atributos, a pesquisa ocorreria apenas em um subconjunto aleatório de atributos, sorteados para cada *Stump*.

O algoritmo é uma opção atraente do ponto de vista computacional, especialmente em problemas de modelagem onde o número de atributos é muito maior do que o número de registros. Dois exemplos clássicos são mineração em corpos de texto (pesquisas na web) e também na análise de DNA por *micro-array*.

A implementação da ideia é bem simples. Para uma tabela de dados com *M* colunas, onde *M* é um número grande, escolhe-se um parâmetro *K*, muito menor que *M*. Os valores típicos de *K são*: $\log_2 M$ e $\sqrt{M}$, bem como a metade ou o dobro destes valores. Em cada nó de uma árvore, realiza-se um sorteio, sem reposição, de *K* atributos, a partir do conjunto total de *M* atributos. Somente estes poucos atributos sorteados serão analisados para determinar qual deles apresenta o maior ganho de informação do nó. O sorteio é refeito em cada nó. Assim, um atributo pode até ser sorteado em vários nós, ou mesmo ficar de fora de toda a análise também.

A geração de Modelos Combinados utilizando o Subespaço Aleatório de Atributos gera modelos com pouca correlação entre si. A baixa correlação entre os modelos individuais mostrou-se essencial para o aumento da precisão dos resultados na faze final de agregação (*Aggregation*). [103]

7.4 - Busca com Parada Ótima

Apesar de apresentar bons resultados práticos, a utilização do parâmetro *K* igual a $\sqrt{M}$, onde M é a quantidade de atributos de entrada, não possui nenhuma fundamentação teórica forte.

Neste trabalho, optou-se pela troca deste método empírico pelo critério de parada ótimo, descrito no capítulo anterior. Deste modo, explorar o subespaço aleatório de atributos passa a obedecer ao algoritmo da Listagem 27. Separa-se o conjunto de atributos aleatoriamente em dois grupos. O primeiro, de tamanho $\sqrt{M}$, é usado como referência. Dele, extrai-se o valor do maior ganho de informação de seus atributos. Em seguida, é feita a pesquisa dentro do segundo grupo, com os atributos restantes, até que seja encontrado um atributo que forneça um ganho maior que o valor encontrado no grupo de referência. Este será o atributo escolhido para o critério de partição.

Com esta modificação, o método das florestas aleatórias passa a ter uma boa base teórica para a escolha do atributo de corte, usado em cada partição das árvores de decisão. Além disso, a quantidade $\sqrt{M}$ deixa de ser um valor empírico, e passa a ser

uma grandeza fruto de um cálculo de maximização da probabilidade de êxito do processo de busca no subespaço aleatório.

7.5 - Algoritmo de Limiarização (*Thresholding*)

Uma árvore de decisão é criada dividindo-se recursivamente os registros de uma tabela. Em cada *Stump* de uma árvore, será selecionado um atributo de partição Xs e seu respectivo valor limite Xth, de modo a maximizar a expressão SyYm.

O algoritmo de partição pode ser mais facilmente explicado e implementado utilizando-se a notação de programação orientada a objetos (OOP) e a linguagem SQL (*Structured Query Language*).

Para cada Stump, são necessários três objetos da classe *NodeInfo*: um para o nó pai (P), outro para o nó direito (R) e outro para o esquerdo (L). Para armazenar todos os valores necessários, cada objeto da classe *NodeInfo* deve ter as propriedades declaradas na Listagem 33.

```
1   CLASSE NodeInfo {
2       ID     // Identificador do Nó
3       N      // Número de regitros neste nó
4       Sy     // Soma dos valores Y neste nó
5       Ym     // Média dos valores Y, calculada como Sy/N
6       SyYm   // Produto Sy*Ym
7   }
```

Listagem 33 – Classe NodeInfo

Usando objetos *NodeInfo*, torna-se fácil traduzir a notação matemática de subscritos para uma notação de objetos e propriedades. A Tabela 21 mostra esta conversão de notação.

NodeInfo Pai	*NodeInfo Left*	*NodeInfo Right*
N_P => P.N	N_L => L.N	N_R => R.N
$Sum(Y_P)$ => P.Sy	$Sum(Y_L)$ => L.Sy	$Sum(Y_R)$ => R.Sy
Ym_P => P.Ym	Ym_L => L.Ym	Ym_R => R.Ym
$Sum(Y_P)\ Ym_P$ => P.SyYm	$Sum(Y_L)\ Ym_L$ => L.SyYm	$Sum(Y_R)\ Ym_R$ => R.SyYm

Tabela 21 – Conversão de notação matemática para orientada a objetos

A expressão de partição (Xs ≤ Xth) pode ser armazenada em um objeto da classe *SplitInfo*, contendo seus respectivos nós esquerdo e direito. A declaração desta classe é apresentada na Listagem 34.

```
CLASSE SplitInfo {
    Xs          // Atributo de partição
    Xth         // Valor de corte de Xs
    L           // NodeInfo esquerdo
    R           // NodeInfo direito
    SyYm        // = L.SyYm + R.SyYm
}
```

Listagem 34 – Classe Splitinfo

O objetivo do algoritmo é encontrar a melhor partição (*maxSplit*) que maximize o ganho de informação, ou seja, que maximize a propriedade SyYm do objeto *SplitInfo*.

Para manter o controle da localização de um registro dentro da árvore, é necessário adicionar uma coluna chamada NodeID na tabela de dados. Todos os registros são inicializados com o identificador do nó raiz (NodeID=1). Isso pode ser feito em SQL com o comando:

```
ALTER TABLE YourDataTable ADD COLUMN NodeID INT
UPDATE TABLE YourDataTable SET NodeID = 1
```

Após encontrar a melhor partição (*maxSplit*), deve-se executar a partição dos registros, modificando a posição do registro (NodeID), para o respectivo identificador (ID) do nó filho, direito ou esquerdo, conforme a expressão Xs ≤ Xth. Isso pode ser feito com o procedimento da Listagem 35.

```
PROCEDIMENTO ExecuteSplit(prmNodeID, prmSplit) {

    UPDATE YourDataTable
    SET NodeID = IIF( prmSplit.Xs <= prmSplit.Xth ,
                      prmSplit.L.ID  ,  prmSplit.R.ID  )
    WHERE NodeID = prmNodeID;
}
```

Listagem 35 – Função para executar a partição

Os nós filhos são identificados conforme uma numeração binária. Dado o identificador do nó pai (P.ID), os nós filhos esquerdos e direitos serão identificados assim:

$$L.ID = P.ID*2 \quad , \quad R.ID = P.ID*2 + 1 \tag{43}$$

As florestas aleatórias utilizam o mesmo processo de partição do método do espaço aleatório de atributos. A Listagem 36 apresenta o algoritmo. Os cinco passos principais indicados serão detalhados a seguir.

```
FUNÇÃO SplitStub_RandomForest(DS,M,K) {

//Entrada: DS=Conjunto de dados no Stump
            M=número total de atributos dos dados
            K=número de atributos candidados
//Saída: atributo de partição Xs e valor de corte Xth

    //Sorteia k atributos candidatos Xc
    PARA i = 1 TO K {
        Xc(i) = rand(1,M)
    }

    VAR max;

    PARA i = 1 TO K {
❶       Selecionar os registros de DS, ordenados por Xc(i)
❷       Criar dois subconjuntos de dados: (L)eft e (R)ight
❸       Inicializar (L)=vazio
❹       Inicializar (R) com todos os dados de DS
❺       PARA_CADA elemento Xi em (R) {
            Removê-lo de (R)
            Adicioná-lo em (L)
            Calcular SyYm = SyLYmL + SyRYmR
            SE SyYm > max ENTÃO Armazenar Xth=Xi e Xs=Xc(i)
        }
    }
    RETORNAR (Xs,Xth)
}
```

Listagem 36 – Algoritmo de corte para uma árvore da floresta aleatória

No passo 1 do algoritmo, deve-se selecionar todos os registros presentes no nó pai, em ordem ascendente de Xs. Isso pode ser feito com a função da Listagem 37.

```
FUNÇÃO GetNodeData(prmNode,prmXs) {
    SELECT prmXs AS Xs,
           SUM(BSFreq) AS N,
           SUM(BSFreq*Y) AS Sy
    FROM YourDataTable
    GROUP BY Xs
    WHERE BSFreq>0 AND NodeID = prmNode.ID
    ORDER BY Xs;
}
```

Listagem 37 – Seleção dos registros para partição

Nos passos 2, 3 e 4 do algoritmo, são criados e inicializados os nós direito e esquerdo. O nó direito recebe todos os dados do nó pai, deixando o nó esquerdo inicialmente vazio. Isso pode ser feito em um método da classe SplitInfo, descrito na Listagem 38.

```
1   MÉTODO SplitInfo.InitChildNodes(prmParentNode) {
2       //Inicialização - Nó esquerdo
3       L.ID = prmParentNode.ID * 2;
4       L.N = 0;
5       L.Sy = 0;
6
7       //Inicialização - Nó direito
8       R.ID = prmParentNode.ID * 2 +1;
9       R.N = prmParentNode.N;
10      R.Sy = prmParentNode.Sy;
11  }
```

Listagem 38 – Inicialização dos nós esquerdo e direito

No passo 5, cada registro será movido da direita para esquerda. Isso pode ser feito com um procedimento de atualização que recebe valores Y e os adiciona (ou remove) de cada nó filho. O pseudocódigo deste método está na Listagem 39.

```
1   MÉTODO NodeInfo.AddRecord(prmN , prmSy) {
2       .N = .N + prmN;
3       .Sy = .Sy + prmSy;
4       .Ym = .Sy / .N;
5       .SyYm = .Sy * .Ym;
6   }
```

Listagem 39 – Método para atualizar as estatísticas de um nó

Assim, com o auxílio dos objetos *SplitInfo* e *NodeInfo*, a função recursiva de partição para as árvores da floresta aleatória pode ser escrita como na Listagem 40.

```
1   PROCEDIMENTO SplitStub_RandomForest(prmParentNode,M,K) {
2
3       VAR iSplit; //Objeto SplitInfo
4
5       //Inicializa iSplit.L e iSplit.R
6       iSplit.InitChildNodes(prmParentNode);
7
8       VAR maxSplit; //Objeto SplitInfo
9
10      PARA_CADA (Xs,N,Sy) EM GetNodeData(prmParentNode,prmXs) {
11
12          iSplit.R.AddRecord(-N,-Sy) //Remove do nó direito
13          iSplit.L. AddRecord(+N,+Sy) //Adiciona ao nó esquerdo
14
15          //Salva o valor de Xs como Xth
16          iSplit.Xth = Xs;
17
18          iSplit.SyYm = iSplit.L.SyYm + iSplit.R.SyYm
19
20          SE ( iSplit.SyYm > MaxSplit.SyYm )
21          ENTÃO maxSplit = iSplit;
22      }
23
24      //Divide os registros, atualizando NodeID
25      ExecuteSplit(prmParentNode.ID, maxSplit);
26
27      //Particiona recursivamente os registros dos nós filhos
28      SE ( maxSplit.L.N > ctMinN ) ENTÃO
29              SplitStub_RandomForest (maxSplit.L);
30      SE ( maxSplit.R.N > ctMinN ) ENTÃO
31              SplitData(maxSplit.R);
32  }
```

Listagem 40 – Partição recursiva para florestas aleatórias

O critério de parada desta função é checar o número de registros em cada nó. Se a quantidade for menor que uma constante ctMinN, o nó será considerado um nó terminal. Um valor típico para ctMinN, usado em problemas de regressão é 5. Assim, os registros de uma tabela são divididos e distribuídos pelos nós da árvore até que o conjunto tenha menos de cinco registros.

7.6 - Limiarização com Variáveis Categóricas

Uma variável categórica tem um domínio formado por alguns poucos valores discretos, sem uma relação de ordem específica. Quando o atributo de partição Xs é do tipo categórico, a expressão de partição passa a ter a forma Xs IN { Xth_1 , Xth_2 , …Xth_j }. Para encontrar a lista de valores limites que maximizam o ganho de informação, deve-se escolher algum modo de ordenar os valores de Xs. Em uma árvore de regressão, a ordenação pode ser feita obedecendo à respectiva média dos valores

de Y. Para fazer isso, basta mudar a cláusula de ordenação na função GetNodeData() conforme mostrado na Listagem 41.

```
1     FUNÇÃO GetNodeData(prmNodeID,prmXs) {
…     ...
8         Order By Avg(BSFreq*Y);   //Nova ordenação para Xs Categórico
9     }
```

Listagem 41 – Modificação para seleção com variável categórica

A função ExecuteSplit() também deve ser ajustada para o novo critério de partição. Esta modificação está na Listagem 42.

```
1     PROCEDIMENTO ExecuteSplit(prmID, prmSplit) {
…     ...
4         SET NodeID = IIF( prmSplit.Xs IN prmSplit.XthList,
…     ...
```

Listagem 42 – Modificação para partição com variável categórica

A lista XthList, que substitui o valor Xth, deve ser montada inserindo-se cada valor de Xs. Isso é feito na função SplitStub_RandomForest(), a medida que são movidos valores de Xs da direita para a esquerda. Ao invés de salvar o valor de Xs em Xth, deve-se inseri-lo na lista XthList, como na Listagem 43.

```
1     PROCEDIMENTO SplitStub_RandomForest(prmParentNodeID,M,K) {
…     ...
15        //Salva o valor de Xs em Xth List
16        iSplit.XthList.Insert(Xs);
…     ...
```

Listagem 43 – Modificação para limiarização com variável categórica

Para indicar que a variável é categórica, é uma prática comum usar um prefixo nos nomes das colunas. Por exemplo, pode-se utilizar o prefixo “Tipo_” como em “Tipo_Equipamento” ou “Tipo_Feriado”. Se for usada alguma convenção de nomenclatura como esta, basta fazer um teste no nome do atributo para decidir se ele é do tipo categórico ou não, e conforme o caso, executar ou não as modificações apresentadas.

7.7 - Limiarização com Valores Nulos

Em várias situações práticas, deve-se considerar o caso em que o atributo Xs de alguns registros esteja em branco. Em SQL, essa situação é representada pelo valor NULL. O uso de NULL implica em uma lógica não booleana, de três estados. Isso porque NULL não é nem maior, nem menor do que o valor limite. Isso significa que um

registro contendo um valor de Xs em branco não será mandado nem para o nó direito nem para o nó esquerdo. Quando a função GetNodeData() selecionar os registros, e ordená-los em ordem crescente, o valor nulo, se estiver presente, será o primeiro registro. Então, deve-se descartá-lo, antes de começar a movimentação dos registros. Para fazer isso, ele deve ser removido do conjunto do nó direito, sem ser inserido no nó esquerdo. Isso é feito com uma modificação na função SplitStub_RandomForest(), mostrada na Listagem 44.

```
1   PROCEDIMENTO SplitStub_RandomForest(prmParentNode,M,K) {
…   ...
10      Para_Cada (Xs,N,Sy) em GetNodeData(prmParentNode.ID) {
11
+           //Remove Nulos da direita, sem colocar na esquerda
+           ENQUANTO IsNull(Xs) {
+               iSplit.R.AddRecord(-N,-Sy);
+               MoveTo NextRecord;
+           }
+
12          iSplit.R.AddRecord(-N,-Sy) //Remove do nó direito
13          iSplit.L.AddRecord(+N,+Sy) //Adiciona ao nó esquerdo
…   ...
```

Listagem 44 – Remoção de valores nulos na partição

A função ExecuteSplit também deve ser ajustada, para deixar claro que registros contendo Xs nulos não serão enviados para nenhum dos nós filhos. A modificação é apresentada na Listagem 45.

```
1   PROCEDIMENTO ExecuteSplit(prmID, prmSplit) {
…   ...
6       WHERE NodeID = prmID AND (Xs IS NOT NULL);
…   ...
```

Listagem 45 – Execução da partição com valores nulos

7.8 - Vantagens das Florestas Aleatórias

Devido a sua versatilidade, velocidade e desempenho, as Florestas Aleatórias estão entre as principais ferramentas de análise de dados na era do Big Data. A capacidade e a facilidade que as Florestas Aleatórias têm para lidar com conjuntos de dados incompletos, inconsistentes, com ruído, com atributos tanto numéricos como categóricos, faz com que seja um dos métodos mais poderosos de aprendizado de máquina. Normalmente, é a primeira escolha em qualquer projeto de Big Data, pois é capaz de lidar com grandes conjuntos de dados, tanto do ponto de vista vertical (muitos registros) como horizontal (muitos atributos). [115]

O desempenho dos modelos de Floresta Aleatória quando não é o melhor, é comparável ao desempenho de modelos bem mais complicados, desenvolvidos e cuidadosamente ajustados para problemas particulares. Gerar uma floresta aleatória não envolve ajuste de parâmetros de modelos ou análises manuais de resultados. O número de árvores do conjunto é limitado apenas pelo tempo e recursos computacionais disponíveis. Caso mais recursos se tornem disponíveis, automaticamente o desempenho será melhorado. E não haverá problemas com sobre ajuste.

O método também é facilmente adaptado para rodar de forma paralela, possibilitando distribuir a carga computacional entre vários computadores ou núcleos em uma rede.

Por todas essas razões, as Florestas Aleatórias representam o estado da arte na área de aprendizado de máquinas com Big Data.

Capítulo 8
Estudo de Caso

Este capítulo apresenta o resultado da aplicação das técnicas de Big Data na criação de um modelo de floresta aleatória (*Random Forest*), através do estudo de casos de previsão de carga média e máxima, com resolução de 15 em 15 minutos, para horizontes de até 12 meses à frente.

8.1 - Medidas de Erro

As medidas de erro mais utilizadas para treinamento, validação e teste de um modelo de previsão são apresentadas na Tabela 22:

Medida	Sigla	Fórmula	Nome comum
Erro Percentual Médio	MPE	$\frac{100\%}{N}\sum_{i=1}^{n}\frac{(Yp_i - Yr_i)}{Yr_i}$	**Viés**
Erro Percentual Médio Absoluto	MAPE	$\frac{100\%}{N}\sum_{i=1}^{n}\left\|\frac{(Yp_i - Yr_i)}{Yr_i}\right\|$	**Erro**

Tabela 22 – Medidas de erro para florestas aleatórias

- **Erro Percentual Médio (MPE):** mede o quanto o valor previsto (*Yp*) difere do valor real (*Yr*). Ele representa o "viés" de um modelo, pois, os desvios positivos anulam os desvios negativos. Por exemplo, quando um modelo é utilizado para previsão de demanda e compra antecipada de energia, o viés representa a diferença total entre a energia comprada e a efetivamente utilizada. Idealmente, o viés deve ser próximo de zero.
- **Erro Percentual Médio Absoluto (MAPE):** mede a acurácia do valor previsto, ou seja, mede a diferença média absoluta entre o valor previsto (*Yp*) e o valor real (*Yr*). Nesta medida, não há cancelamento de desvios positivos e negativos. Portanto, ela fornece uma medida mais fiel da capacidade de previsão do modelo, ou seja, um valor esperado para os erros de previsão.

Cada árvore da floresta é treinada com uma amostra bootstrap dos dados originais. Nesta fase, os dados são separados em dois conjuntos: “In Bag” (ou conjunto de treinamento) e “Out of Bag” (ou conjunto de validação). Os erros calculados sobre os dados “In Bag” são chamados erros de ajuste, pois refletem o erro de treinamento de cada árvore. Já os erros calculados sobre os dados “Out of Bag”, representam uma estimativa do poder de generalização de cada árvore. Essas quatro medidas de erro devem ser aproximadamente iguais para cada uma das árvores, indicando que todas elas possuem aproximadamente o mesmo poder de previsão.

Após a fase de crescimento da floresta, ela pode ser testada com um novo conjunto de dados, totalmente desconhecido do modelo. Esta fase de teste é conhecida também como *Scoring*.

8.2 - Parâmetros do Modelo

Uma das principais vantagens dos modelos de árvore aleatória é a pequena quantidade de parâmetros, e sua facilidade de ajuste, o qual permite a previsão quase sem nenhuma interferência humana. Os parâmetros são:

- **Número de árvores na floresta (F):** Como o processo de crescimento da floresta é sequencial, adicionando uma árvore de cada vez, a escolha de F pode ser feita observando-se a evolução do erro OOB da floresta. Quando o erro se estabiliza, o processo pode ser interrompido.
- **Número de atributos aleatórios (K):** representa quantos atributos serão sorteados e analisados para partição de cada nó. Os valores típicos para um conjunto de treinamento com M atributos são $\log_2 M$, $\sqrt{M}$, ou a metade ou o dobro destes valores. Apesar de influenciar na precisão do modelo, variações do valor de K geram pequenos ganhos. Assim, a escolha de k, apesar de empírica, é bastante simples.
- **Número mínimo de registros para partição (MinN):** representa o critério de parada no processo de indução de uma árvore. Os nós serão particionados enquanto possuírem pelo menos *MinN* registros. Abaixo deste valor, o nó é marcado como nó terminal. Para problemas de

regressão, os valores sugeridos são de 3, 5 ou 10 registros. Novamente, a escolha deste parâmetro é empírica, mas direta e simples.

A Tabela 23 mostra como a escolha de cada um destes parâmetros influencia no tempo de treinamento da floresta e no seu erro de previsão:

Parâmetro	Variação do parâmetro	Tempo de Treinamento	Erro
F	Aumenta	Aumenta	Diminui
	Diminui	Diminui	Aumenta
K	Aumenta	Aumenta	Diminui
	Diminui	Diminui	Aumenta
MinN	Aumenta	Diminui	Aumenta
	Diminui	Aumenta	Diminui

Tabela 23 – Parâmetros do modelo de floresta aleatória

8.3 - Geração da Floresta Aleatória

A Figura 45 mostra o relatório com os parâmetros usados para criação do modelo de previsão. Foi utilizado um histórico de 40 meses, com dados de 15 em 15 minutos, para previsão até 1 ano à frente.

O algoritmo foi interrompido após a criação de 40 árvores na floresta, pois notou-se a convergência do erro OOB, em 1,37%. No conjunto de testes, obteve-se um viés de 0,21% e erro de 1,57%. Estes dados mostram que a estimativa de desempenho do modelo, dado pelo erro OOB ficou bastante próxima do desempenho real do modelo.

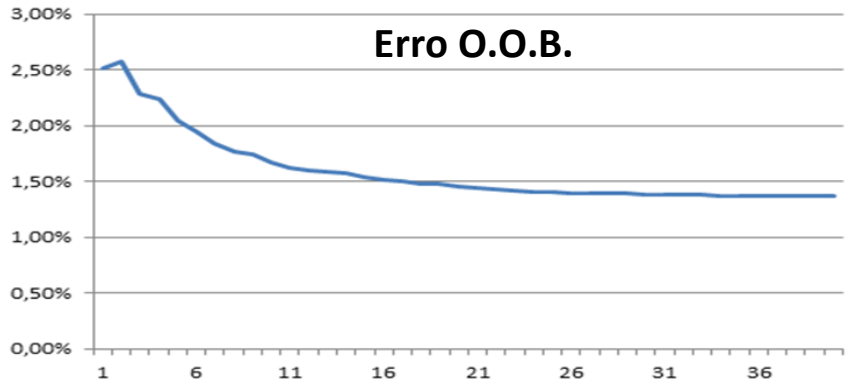
Relatório de Geração da Floresta Aleatória		
Conjunto de Treinamento		
Histórico: 40 meses	Janeiro 2002 a Abril 2005	
Intervalo: 15 em 15 minutos	116.736 registros	
Modelo de Floresta Aleatória		
Parâmetros: K=3, MinN=10	F=40 árvores	
Tempo Treinamento: 312s/árvore	129.363 nós terminais	
Validação por bootstrap		
		
	Viés OOB: 0,18%	**Erro OOB: 1,37%**
Conjunto de Teste		
Histórico: 12 meses à frente	Maio 2005 a Abril 2006	
Intervalo: 15 em 15 minutos	73.033 registros	
Score:	**Viés: 0,21%**	**Erro: 1,57%**

Figura 45 – Relatório de criação da floresta aleatória

8.4 - Previsão 24 horas à Frente, de 15 em 15 Minutos

A previsão de 24 horas à frente representa um importante parâmetro de comparação para as demais previsões. De um modo geral, à medida que o horizonte de previsão aumenta, os erros tendem a aumentar. No caso em estudo, o primeiro dia de previsão foi um domingo, em especial, um feriado nacional.

Conforme se vê no gráfico da Figura 46, o modelo foi capaz de acompanhar a variação da carga durante todo o dia, com erro médio e viés na faixa de 1% a 2%. Comparado a média geral do modelo, o viés foi alto. Isso se justifica por ser um dia de feriado, com carga ainda mais baixa que os domingos típicos.

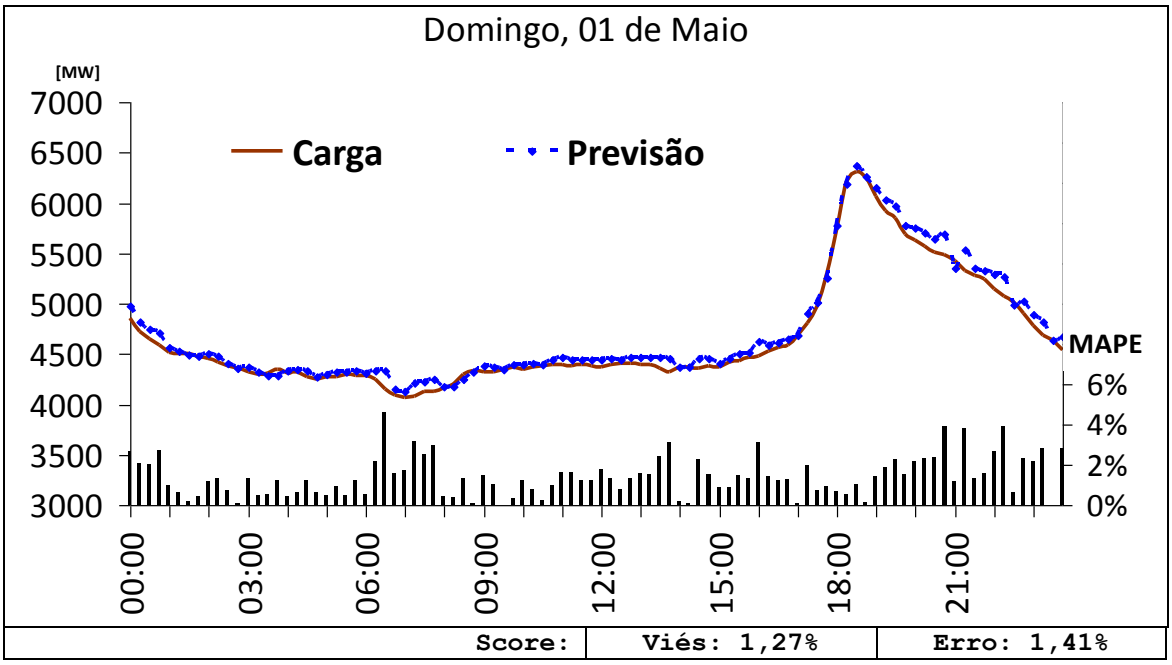


Figura 46 – Previsão 24 horas à frente, de 15 em 15 minutos

8.5 - Previsão 7 Dias à Frente, de 15 em 15 Minutos

A previsão de 7 dias à frente também oferece uma boa referência para avaliação da qualidade das demais previsões. Como a carga tem um forte fator ciclo semanal, essa previsão permite avaliar os valores de viés e erro para cada um dos dias da semana. Os gráficos da Figura 47 mostram os resultados.

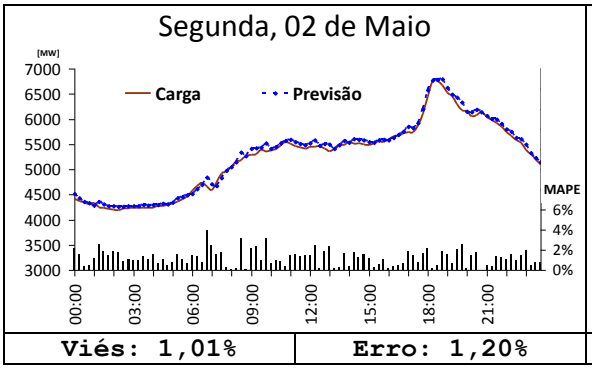


Figura 47 (a)

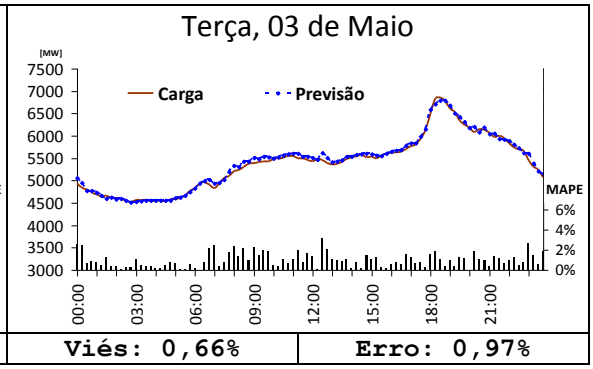


Figura 47 (b)

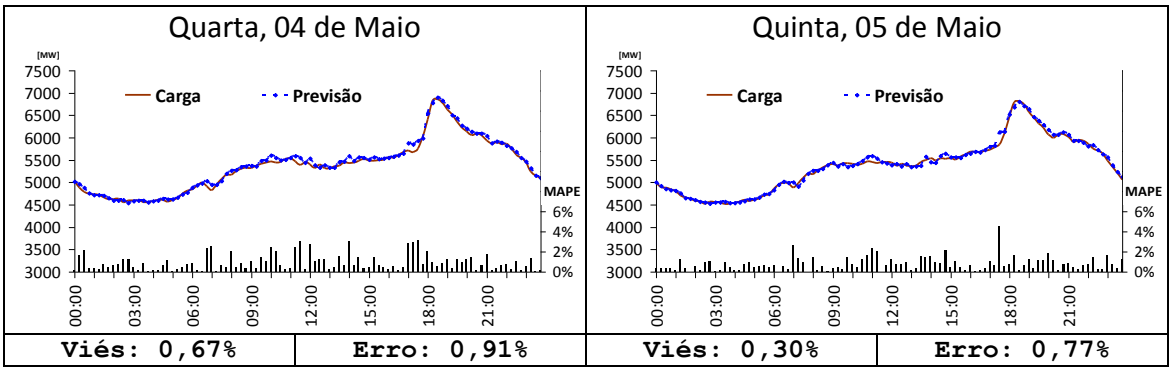


Figura 47 (c)

Figura 47 (d)

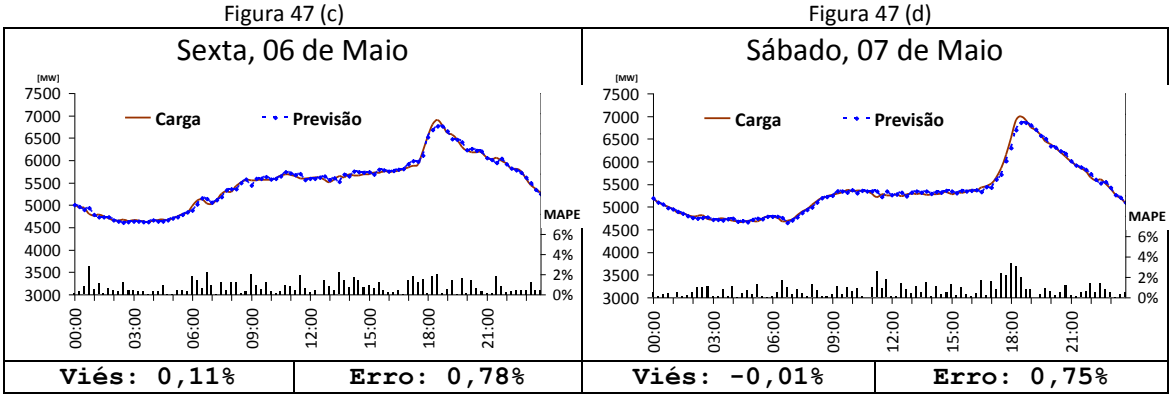


Figura 47 (e)

Figura 47 (f)

Figura 47 – Previsão até 7 dias à frente, de 15 em 15 minutos

Destes gráficos, percebe-se que os valores médios de erro ficam próximos de 1%, enquanto o viés é um pouco menor. A única exceção foi na segunda-feira, onde o viés ficou acima de 1%. Essa diferença talvez possa ser explicada por se tratar de um dia após um feriado nacional, mesmo que este tenha caído em um domingo.

No gráfico da quinta-feira, destaca-se um erro ligeiramente maior no início da subida rumo ao horário de pico. Como a carga aumenta muito neste horário, um pequeno erro na previsão exata do instante desta subida leva a um erro percentual maior. Apesar disso, pode-se dizer que a previsão se manteve fiel ao formato da curva.

8.6 - Previsão por Dia Tipo

Agrupar os perfis de carga de 24 horas em dias tipo é uma técnica muito utilizada na previsão de carga. Por isso, é interessante avaliar a qualidade da previsão para cada um desses tipos. A Tabela 24 apresenta os erros de previsão de cada dia tipo para todo o horizonte de previsão.

Dia Típico	Viés	Erro
Dia Útil Típico (segunda a sexta)	-0,07%	0,88%
Sábado Típico	0,06%	0,78%
Domingos e Feriados	0,16%	0,88%
Segunda-feira (feriado prolongado)	1,61%	1,89%
Sexta-feira (feriado prolongado)	1,47%	1,67%
Sábado (feriado prolongado)	-0,20%	0,86%

Tabela 24 – Desempenho da previsão para dias típicos

De um modo geral, o modelo obteve um bom desempenho em todos os dias típicos, apresentando viés próximo de zero e erro abaixo de 1%. Somente nas segundas e sextas-feiras de feriado prolongado, o erro e o viés aumentaram um pouco.

8.7 - Previsão de Feriados

Os feriados são um dos grandes desafios para a previsão de carga. Cada feriado tem uma característica diferente, podendo variar bastante conforme o dia da semana em que ocorre. A Figura 48 apresenta o resultado da previsão para todos os feriados presentes no horizonte de previsão.

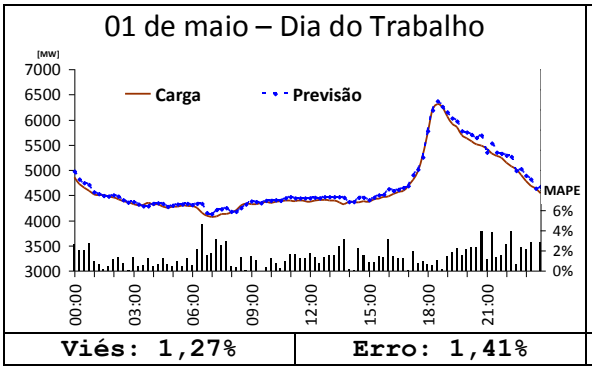


Figura 48 (a)

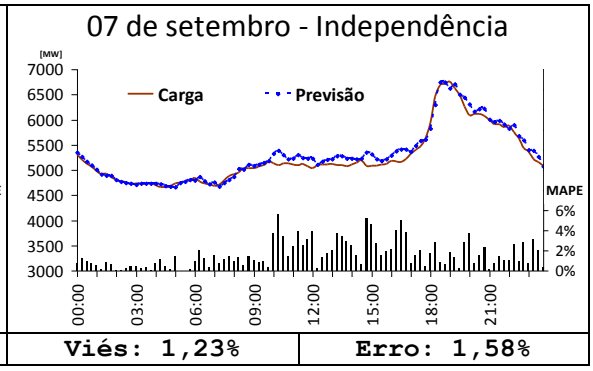


Figura 48 (b)

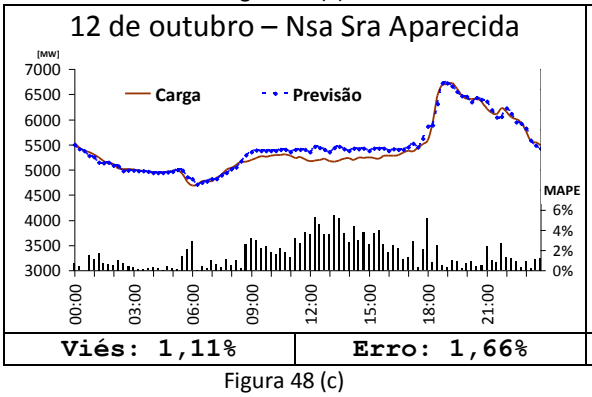


Figura 48 (c)

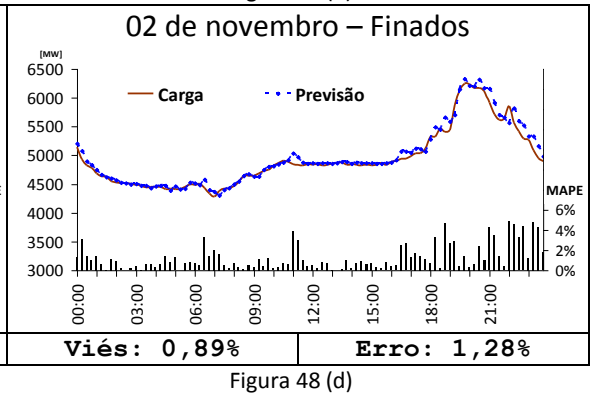


Figura 48 (d)

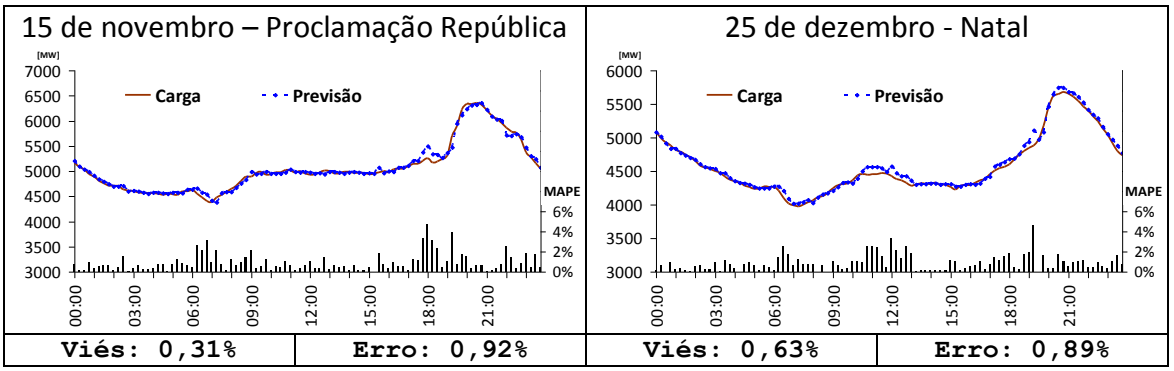


Figura 48 (e)

Figura 48 (f)

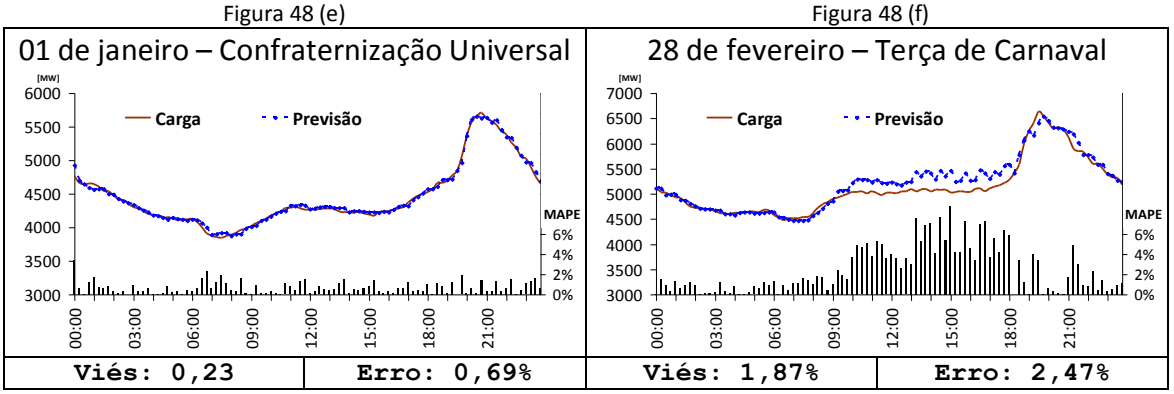


Figura 48 (g)

Figura 48 (h)

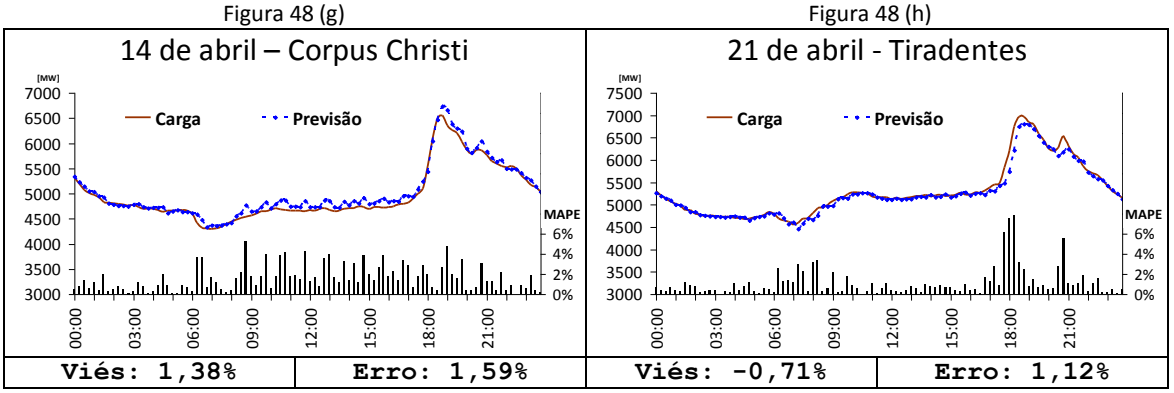


Figura 48 (i)

Figura 48 (j)

Figura 48 – Previsão da carga média em feriados de 15 em 15 minutos

Como era de se esperar, o erro de previsão nos feriados foi ligeiramente maior que o erro nos dias úteis. A terça-feira de carnaval foi o dia que apresentou o maior erro, chegando a 2,47%. Isso aconteceu, principalmente, pelo erro de previsão durante o dia, que foi pouco acima do real. Este comportamento também ocorreu em menor intensidade no dia 07 de setembro, 12 de outubro e 14 de abril (Corpus Christi).

8.8 - Previsão de Feriados Prolongados

Um feriado prolongado ocorre quando uma segunda-feira, uma sexta-feira, ou um sábado, ficam entre um feriado e um final de semana. Tornou-se costume no Brasil, em muitas escolas, empresas e órgãos públicos, diminuir o nível de atividades, e

consequentemente, o consumo de energia, nestes dias. De um modo geral, estes dias foram os que apresentaram os maiores erros. A explicação para isso talvez seja a pequena quantidade de ocorrências deste evento no histórico, se comparados aos demais tipos. Dentro do horizonte de previsão, ocorreram somente 3 feriados prolongados. A Figura 49 apresenta a sexta-feira após o feriado de Corpus Christi (quinta-feira), enquanto a Figura 50 apresenta uma segunda-feira, antes do feriado de 15 de novembro. Finalmente, a Figura 51 mostra o resultado da previsão da segunda-feira de carnaval.

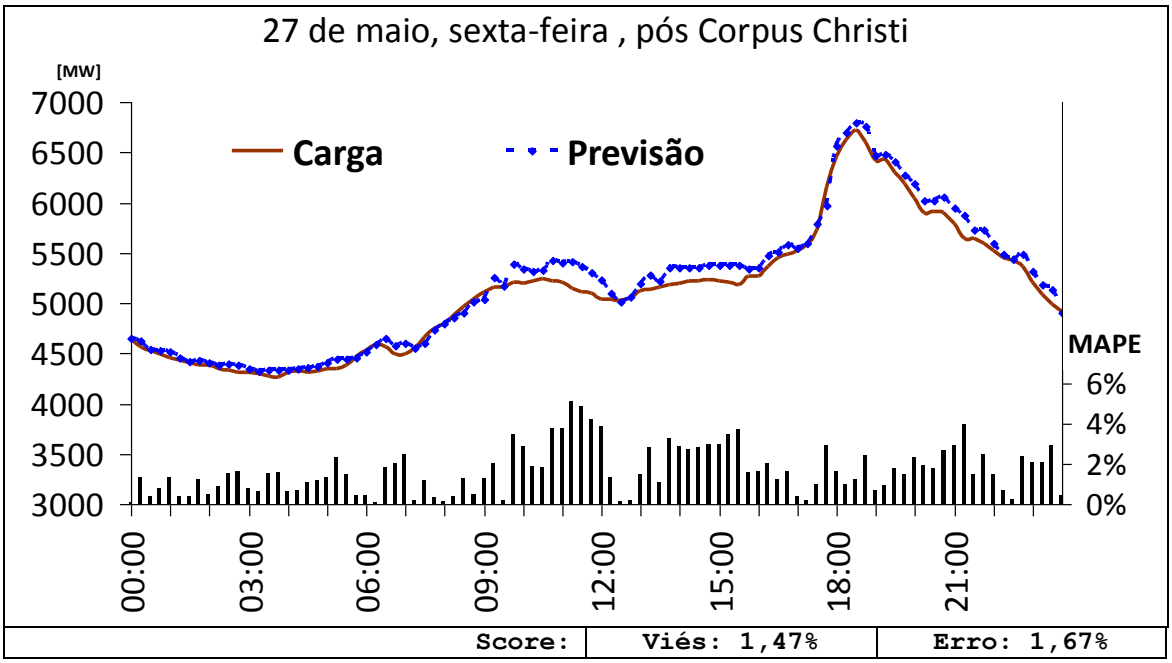


Figura 49 – Previsão de sexta-feira de feriado prolongado

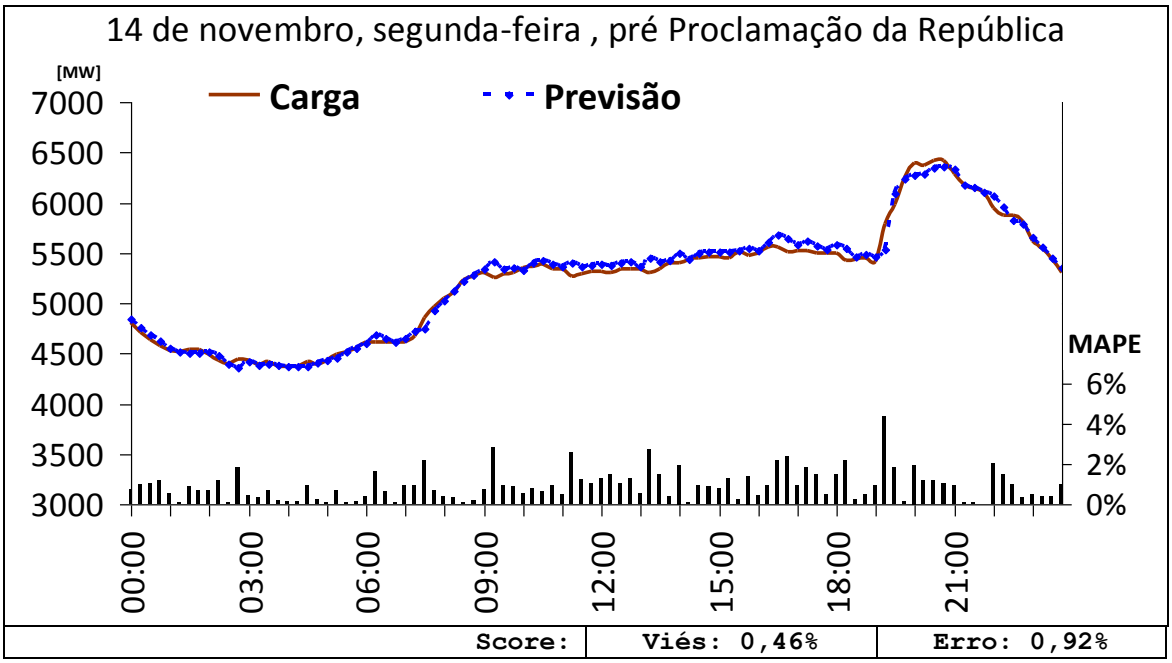


Figura 50 – Previsão de segunda-feira de feriado prolongado

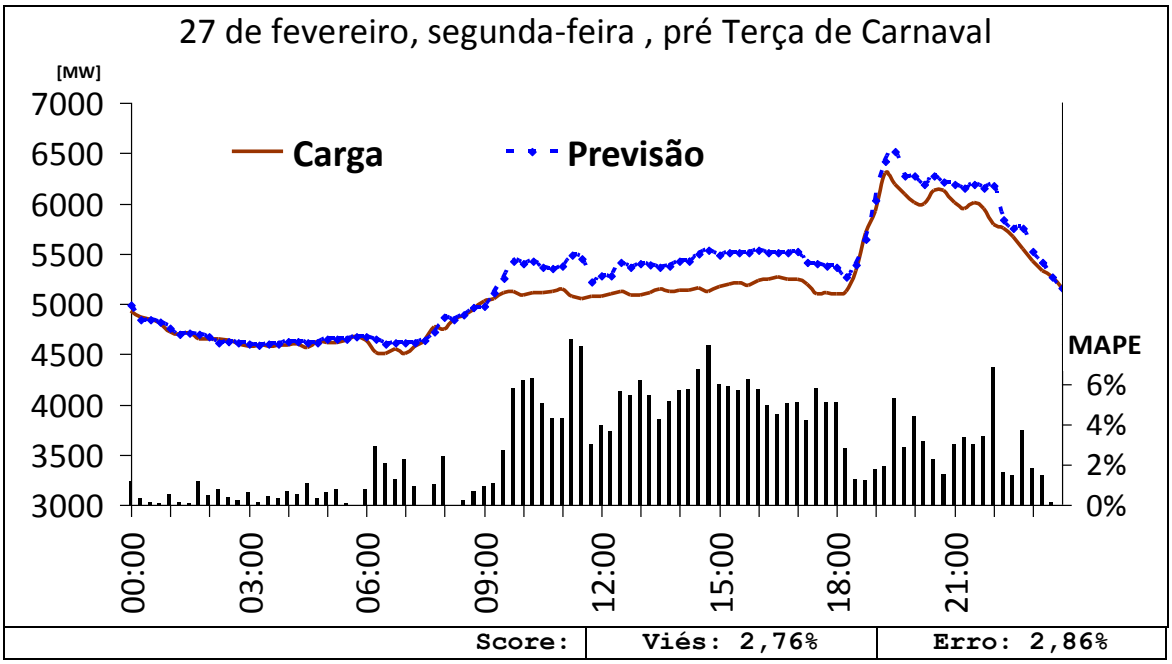


Figura 51 – Previsão de segunda-feira pré-carnaval

8.9 - Previsão de Início de Horário de Verão

Outro desafio para a previsão de carga é a vigência do horário de verão. A mudança nos relógios causa uma alteração significativa no perfil de carga. A Figura 52 mostra o resultado da previsão para as segundas-feiras da semana imediatamente antes da mudança; e também da primeira, segunda e terceira semana após a mudança.

Nota-se que no primeiro dia de horário de verão, o modelo apresentou um pico de erro após as 18:00h, exatamente no horário onde a vigência do horário de verão causa a maior modificação. Porém, nas semanas seguintes, o modelo assimilou bem a mudança, e voltou a acompanhar a curva real.

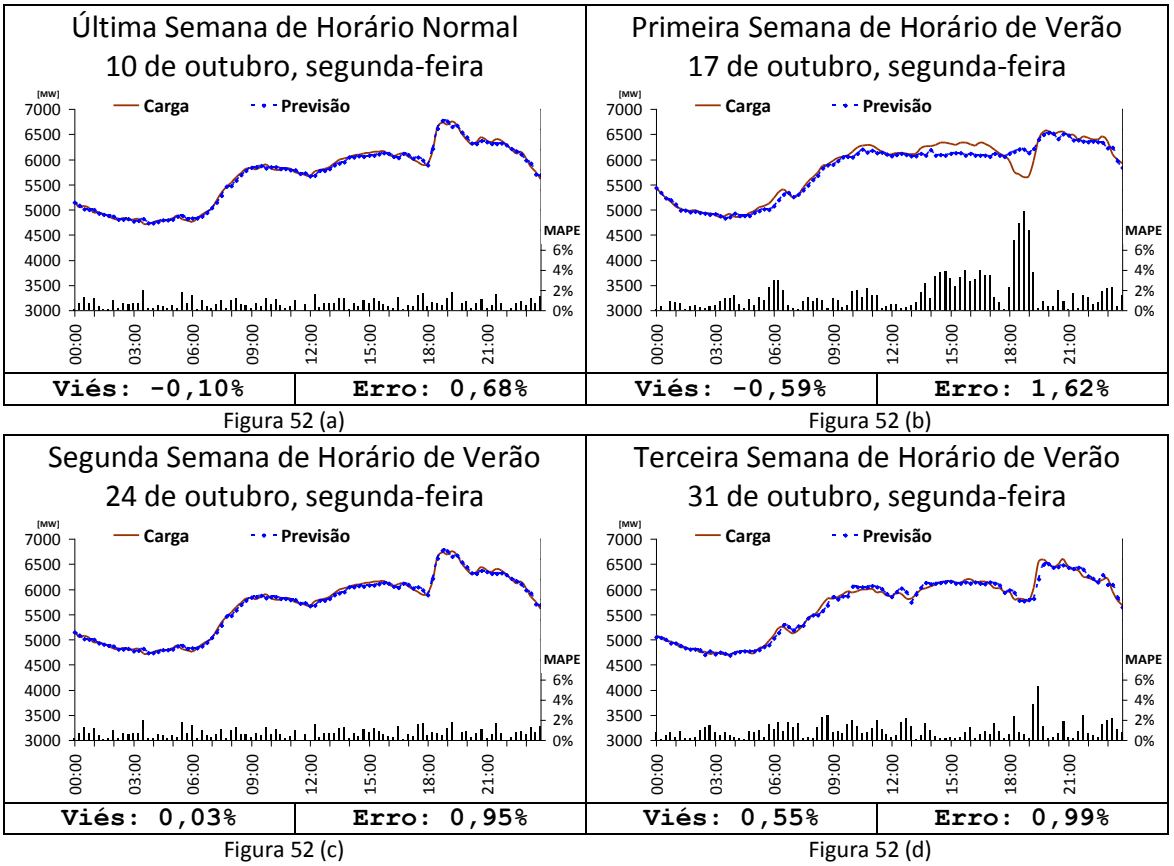


Figura 52 (a) Figura 52 (b)

Figura 52 (c) Figura 52 (d)

Figura 52 – Previsão de início de horário de verão

8.10 - Previsão de Término de Horário de Verão

De modo análogo ao item anterior, a Figura 53 mostra o desempenho da previsão no término do horário de verão. Ao contrário do início, no término do horário de verão não houve impacto significativo no desempenho do modelo. O erro na primeira semana, após o fim do horário de verão, foi semelhante ao da semana anterior. Na segunda semana, nota-se um comportamento atípico da curva, por ser a semana de carnaval. Na terceira semana, novamente o erro de previsão volta aos patamares esperados.

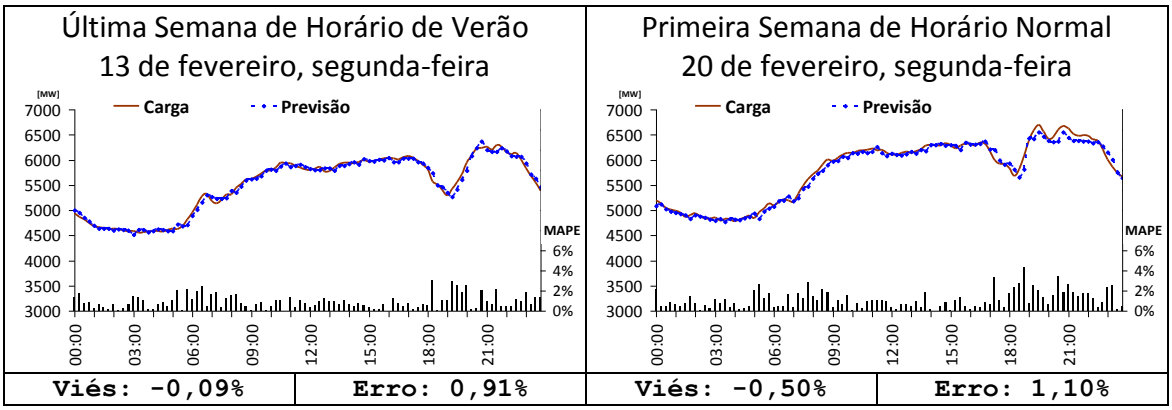


Figura 53 (a)

Figura 53 (b)

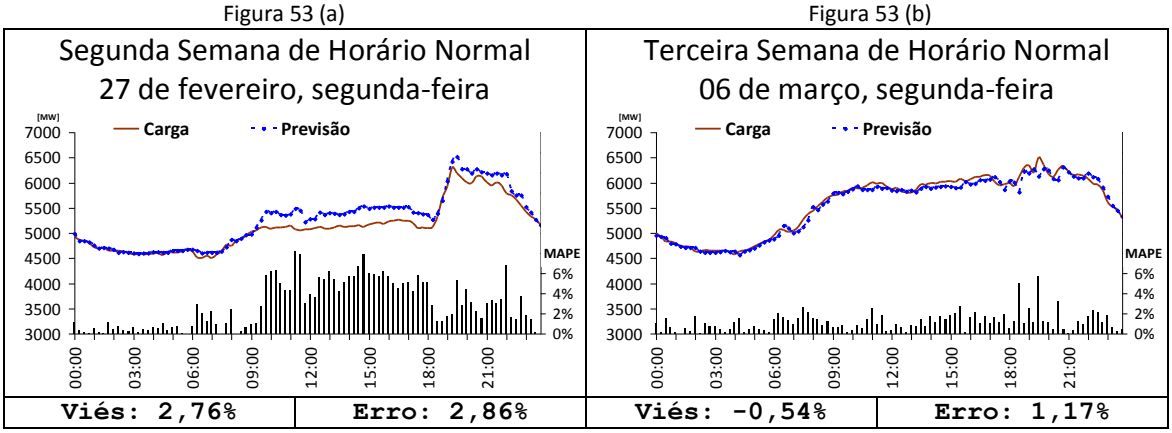


Figura 53 (c)

Figura 53 (d)

Figura 53 – Previsão de término de horário de verão

8.11 - Previsão da Carga Média Diária até 30 Dias à Frente

Além da previsão de 15 em 15 minutos, pode-se fazer também a análise da evolução da carga média diária. Este perfil de carga tem a forma de uma sequência de dentes, sendo o espaço entre eles representado pela carga mais baixa dos finais de semana. Quando algum dente parece deformado, indica a ocorrência de um feriado.

O valor da previsão da carga média diária foi calculado através da agregação dos valores previstos de 15 em 15 minutos. Nota-se que o erro neste caso é ligeiramente menor e bem mais consistente que o erro das previsões de 15 em 15 minutos. A maior resolução dos registros no histórico possibilita ao modelo acompanhar de forma mais próxima a evolução da curva de carga. Caso fosse feito o treinamento diretamente com um único registro de carga máxima por dia, o erro de previsão seria maior.

A Figura 54 apresenta o resultado da previsão no horizonte de 30 dias à frente, para o mês de maio. A quebra da periodicidade no formato dos dentes ocorrida no dia 26 foi causada pelo feriado de Corpus Christi.

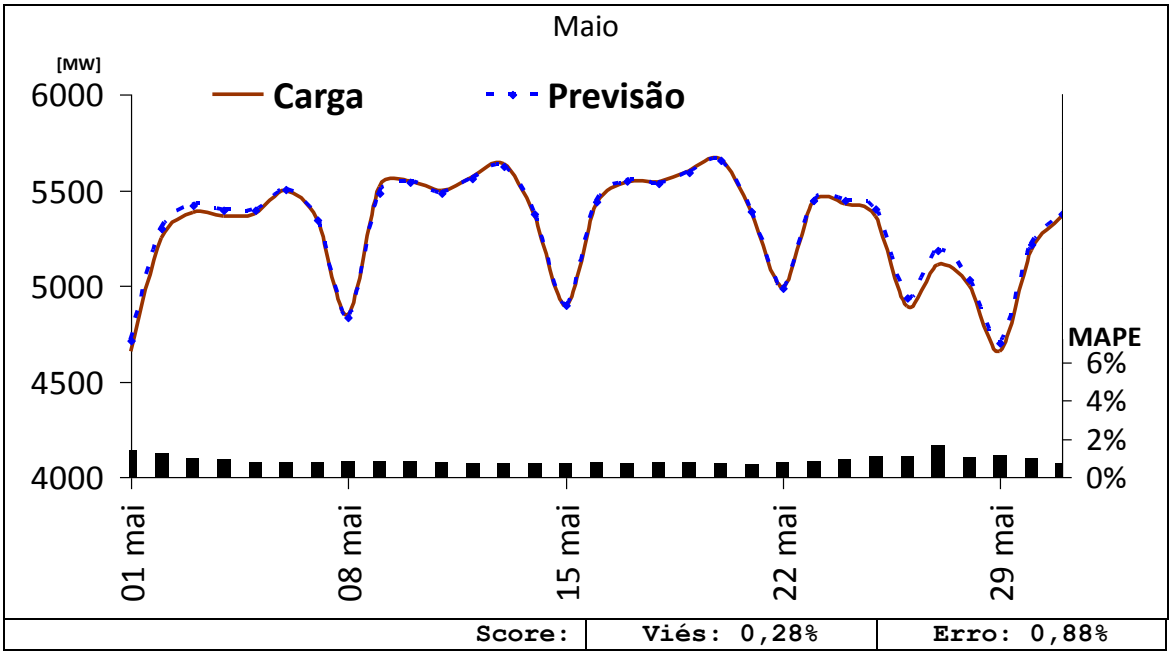


Figura 54 – Previsão da carga média diária até 30 dias à frente

8.12 - Previsão da Carga Média Diária até 1 Ano à Frente

A Figura 55 mostra o desempenho da floresta aleatória nos meses seguintes, no horizonte de previsão de até 1 ano à frente. Verifica-se que o previsor manteve um bom desempenho em todos os meses do ano. O maior erro foi no mês de setembro, que apresentou erro médio de 1,91%.

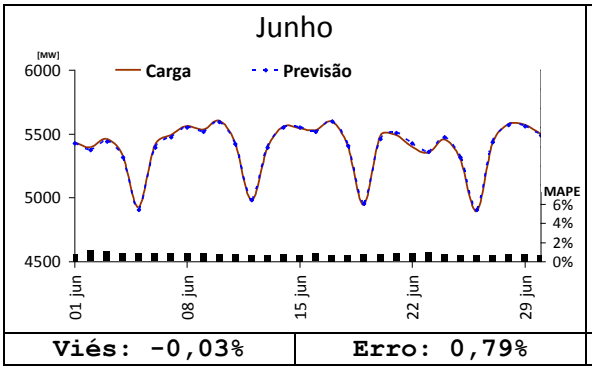


Figura 55 (a)

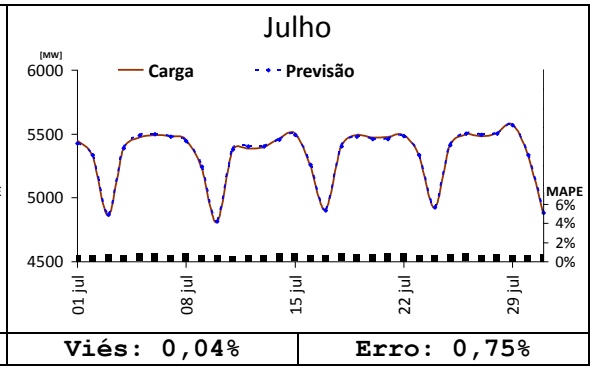
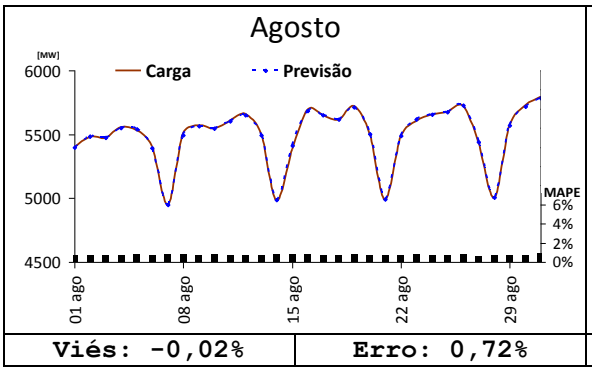
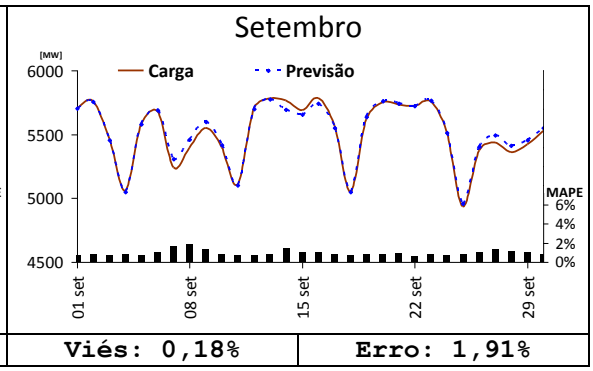


Figura 55 (b)



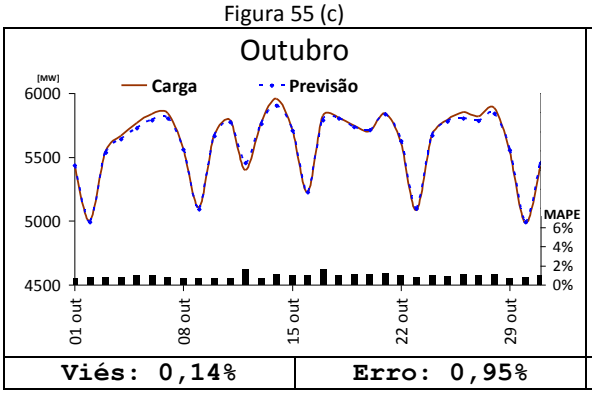
Viés: -0,02%	Erro: 0,72%

Figura 55 (c)



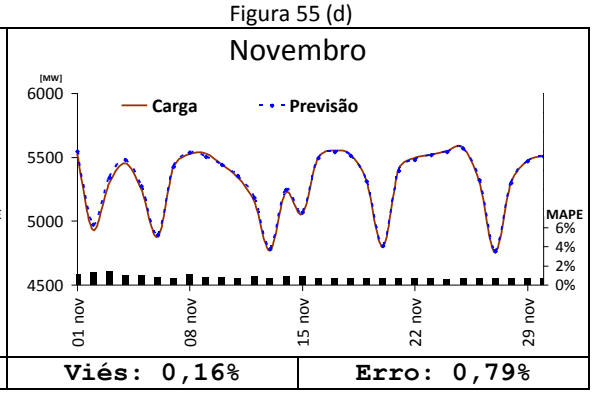
Viés: 0,18%	Erro: 1,91%

Figura 55 (d)



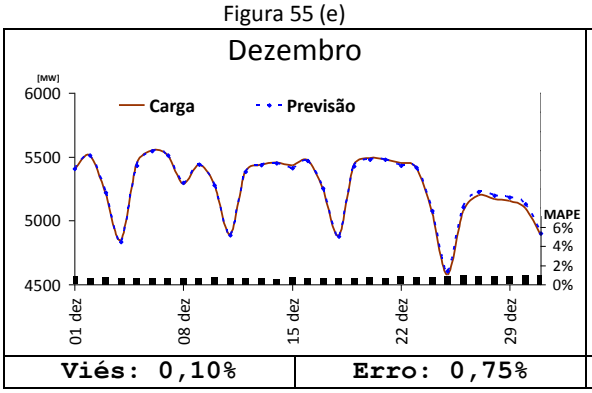
Viés: 0,14%	Erro: 0,95%

Figura 55 (e)



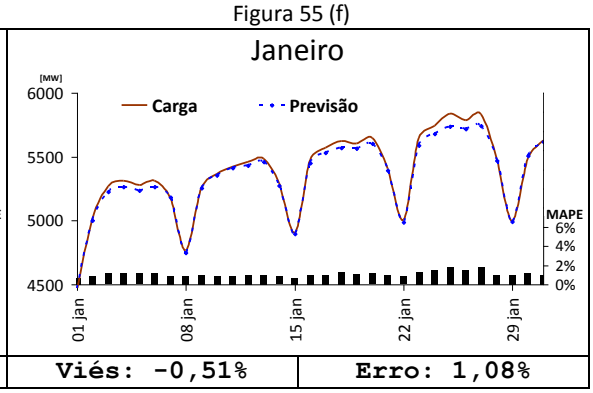
Viés: 0,16%	Erro: 0,79%

Figura 55 (f)



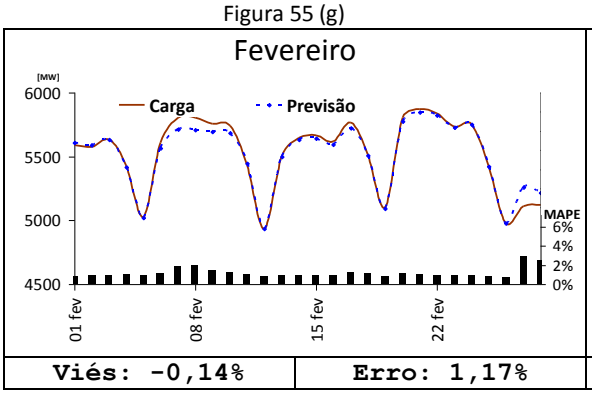
Viés: 0,10%	Erro: 0,75%

Figura 55 (g)



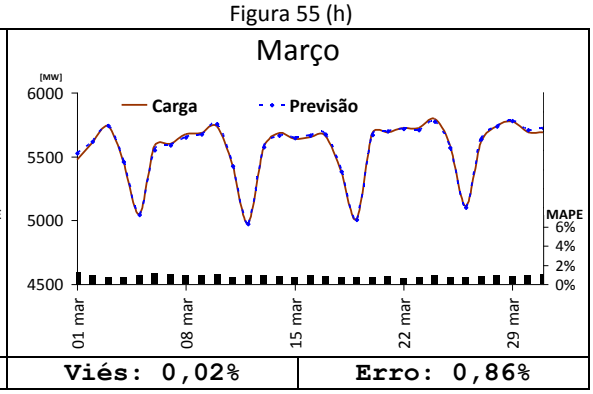
Viés: -0,51%	Erro: 1,08%

Figura 55 (h)



Viés: -0,14%	Erro: 1,17%

Figura 55 (i)



Viés: 0,02%	Erro: 0,86%

Figura 55 (j)

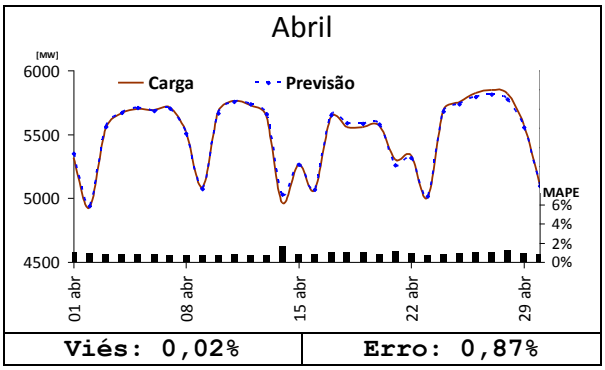


Figura 55 (k)

Figura 55 – Previsões da carga média diária até 1 ano à frente

8.13 - Erro de Previsão da Carga Média Diária até 1 Ano à Frente

A Figura 56 resume o erro de previsão da carga média diária em todos os 365 dias à frente. Conclui-se que, dentro deste horizonte de previsão, o modelo não perdeu desempenho, mantendo o viés próximo de zero e o erro médio próximo de 1%. Poucos foram os dias com erros maiores que 1,5%, sendo a maioria deles relacionados com a ocorrência de feriados.

Deve-se destacar a importância do baixo valor de viés obtido, de 0,20%. Por estar este tipo de previsão diretamente ligado com a compra de energia, um viés próximo de zero significa que o valor de energia contratado é muito próximo do valor da energia efetivamente utilizada.

Desta forma, a utilização deste modelo para a atividade de compra de energia, que é fundamental para as concessionárias, pois influencia diretamente na sua rentabilidade econômico-financeira, faz com que não haja necessidade de contratação de energia extra no mercado e que nem ocorra desperdício com a contratação de uma energia não utilizada.

Assim, o modelo acertou a energia que deveria ser contratada para o próximo ano.

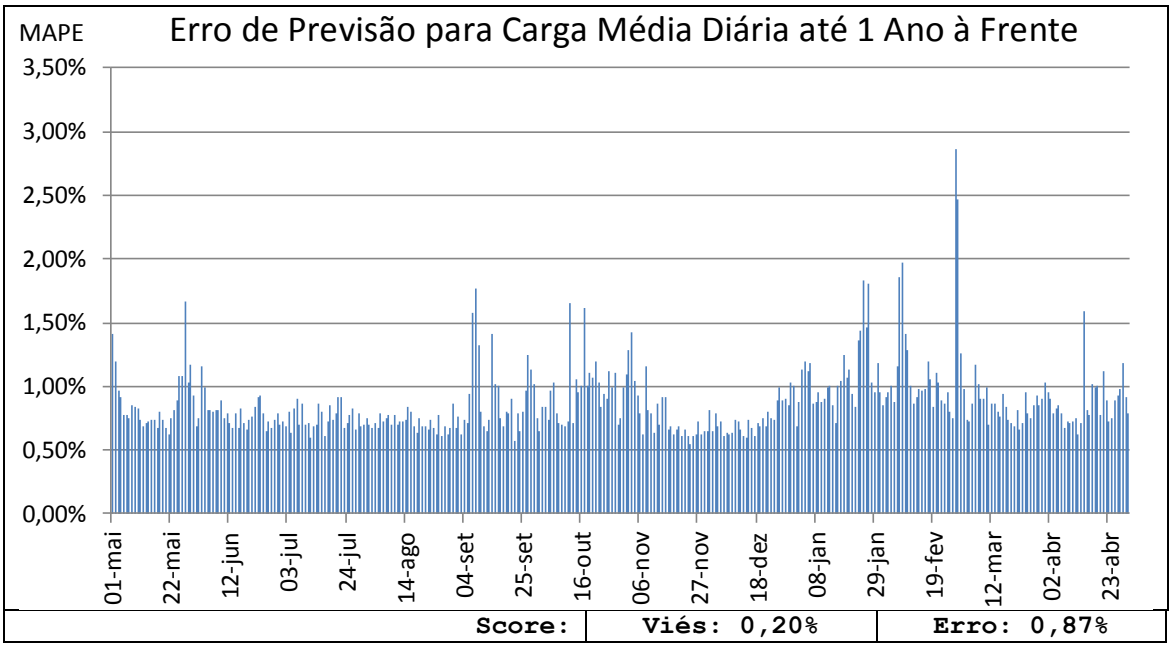


Figura 56 – Erro de previsão da carga média diária até 1 ano à frente

8.14 - Previsão da Carga Máxima Diária até 30 Dias à Frente

Enquanto as previsões dos valores de carga média são importantes para os contratos de energia, as previsões de valores de carga máxima são necessárias para o planejamento e dimensionamento dos equipamentos da rede.

Prever o valor de carga máxima é relativamente mais difícil que a previsão do valor de carga média diária. Enquanto esta última se beneficia da abundância de dados coletados de 15 em 15 minutos, o valor da carga máxima é representado por somente um registro por dia.

O formato da curva de carga máxima diária também é semelhante a uma série de dentes, com largura de 7 dias. Seu ponto mais baixo ocorre nos domingos e feriados. A Figura 57 mostra o resultado da previsão para os primeiros 30 dias. Como previsto, o erro é um pouco maior que o erro da carga média, ultrapassando 1%. O dia de maior erro foi o domingo dia 29, logo após o feriado de Corpus Christi.

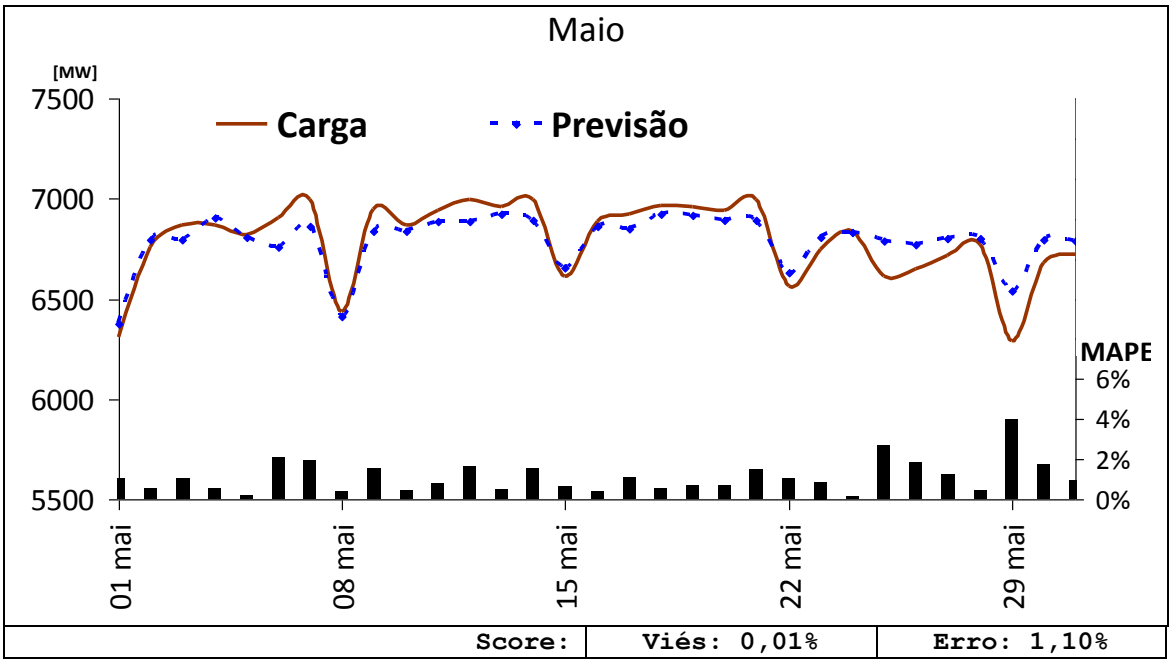


Figura 57 – Previsão da carga máxima diária até 30 dias à frente

8.15 - Previsão da Carga Máxima Diária até 1 Ano à Frente

Enquanto a previsão de carga média diária mostrou uma boa consistência, a previsão da carga máxima apresentou uma oscilação um pouco maior. A Figura 58 apresenta os gráficos de cada um dos meses seguintes.

A influência dos feriados no pico de carga é maior do que na previsão da carga média. Por exemplo, o feriado de 07 de setembro caiu em uma quarta-feira. Apesar do erro neste dia ter sido baixo, a carga teve um comportamento atípico dois dias depois, na sexta-feira. Como o valor de carga máxima foi bem menor que o esperado, o erro foi maior. Isso indica que possivelmente algumas empresas tenham optado por trocar o feriado da quarta-feira pela sexta-feira.

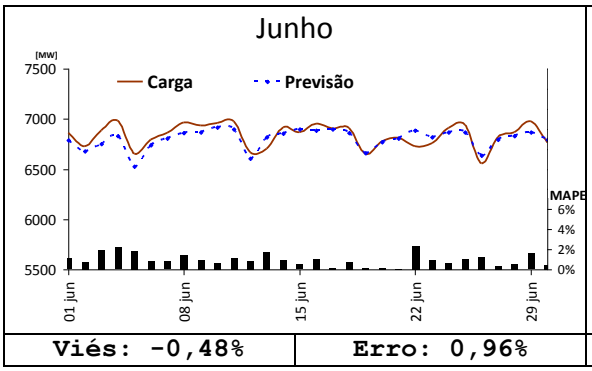


Figura 58 (a)

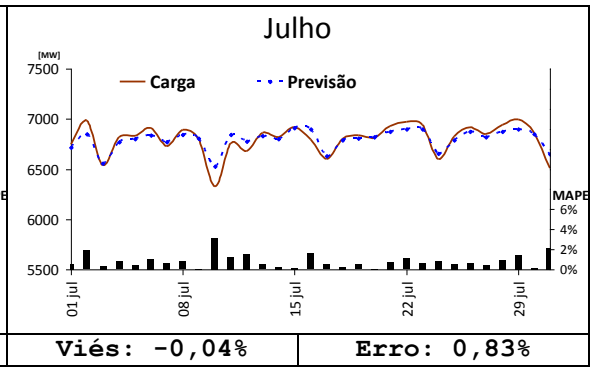
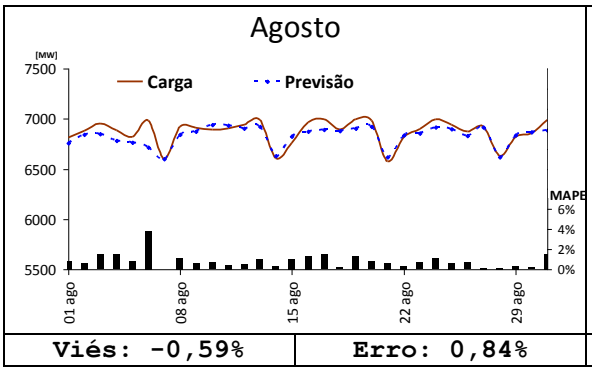
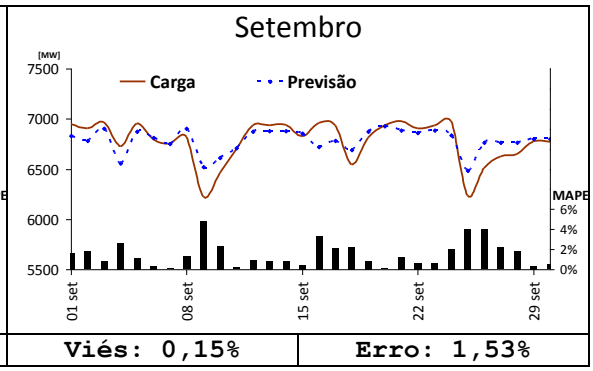


Figura 58 (b)



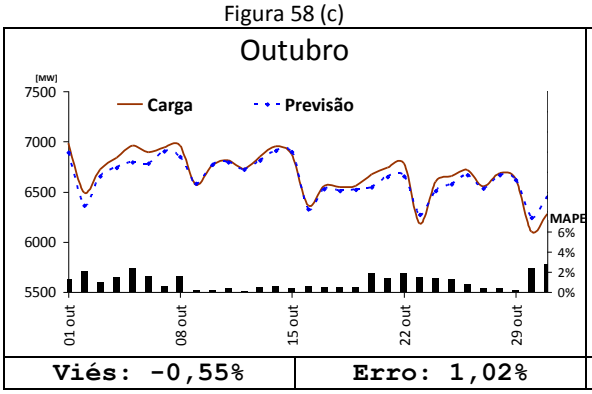
Viés: -0,59%	Erro: 0,84%

Figura 58 (c)



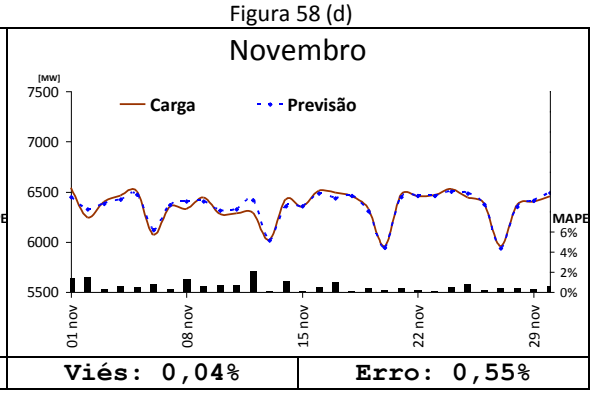
Viés: 0,15%	Erro: 1,53%

Figura 58 (d)



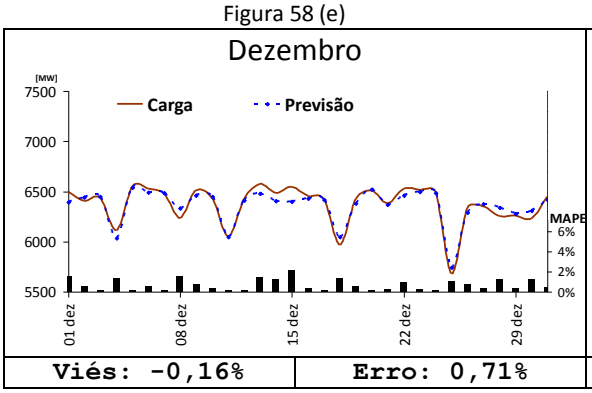
Viés: -0,55%	Erro: 1,02%

Figura 58 (e)



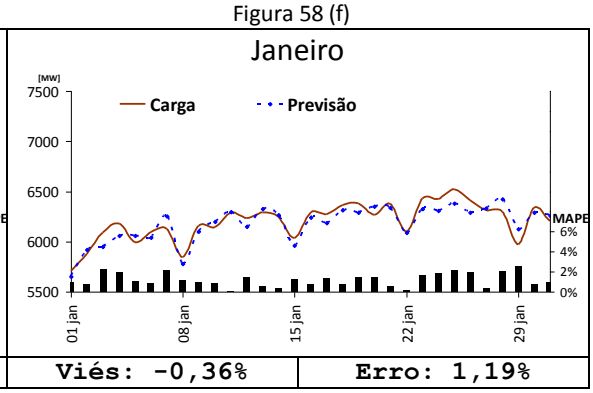
Viés: 0,04%	Erro: 0,55%

Figura 58 (f)



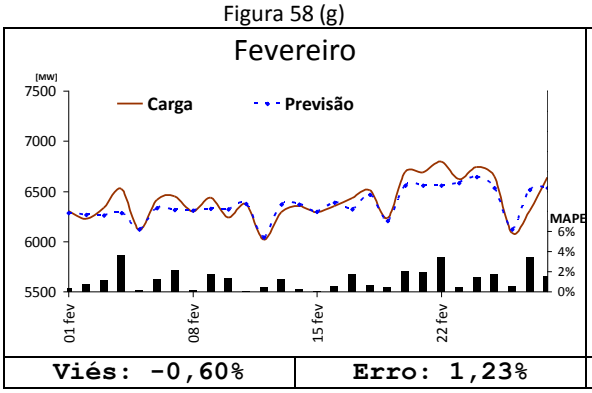
Viés: -0,16%	Erro: 0,71%

Figura 58 (g)



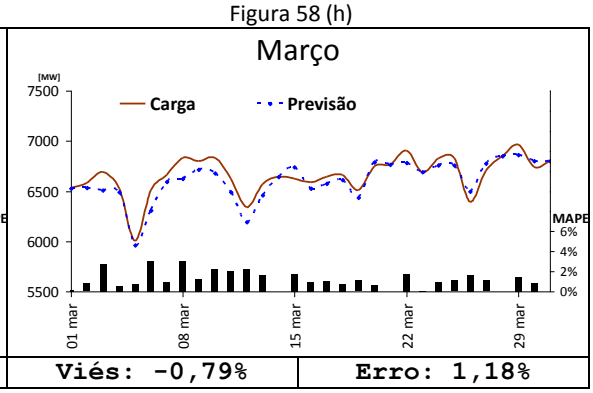
Viés: -0,36%	Erro: 1,19%

Figura 58 (h)



Viés: -0,60%	Erro: 1,23%

Figura 58 (i)



Viés: -0,79%	Erro: 1,18%

Figura 58 (j)

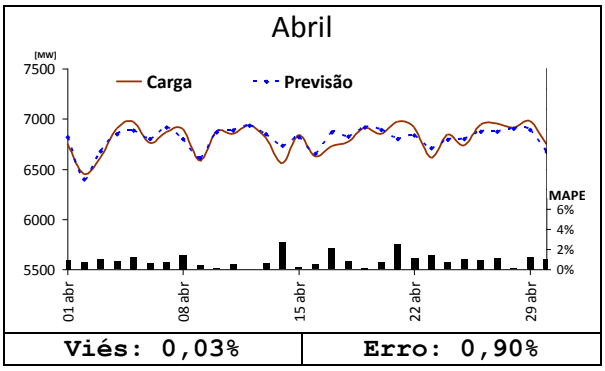


Figura 58 (k)

Figura 58 – Previsões da carga máxima diária até 1 ano à frente

Como dito anteriormente, a carga máxima está ligada ao planejamento das redes elétricas, para que estas estejam preparadas para suprir a demanda de carga, e também à operação, para que para que valores operacionais não sejam excedidos, e para que as intervenções na rede sejam agendadas. Em um tempo de geração distribuída, esta previsão ajuda a determinar em que pontos a rede mais necessidade deste reforço. Este conceito foi utilizado em [116].

8.16 - Erro de Previsão da Carga Máxima Diária até 1 Ano à Frente

A Figura 59 resume o erro de previsão da carga máxima para os 365 dias do horizonte de previsão. Apesar de a média ter ficado em 1%, alguns dias tiveram erros acima de 2%.

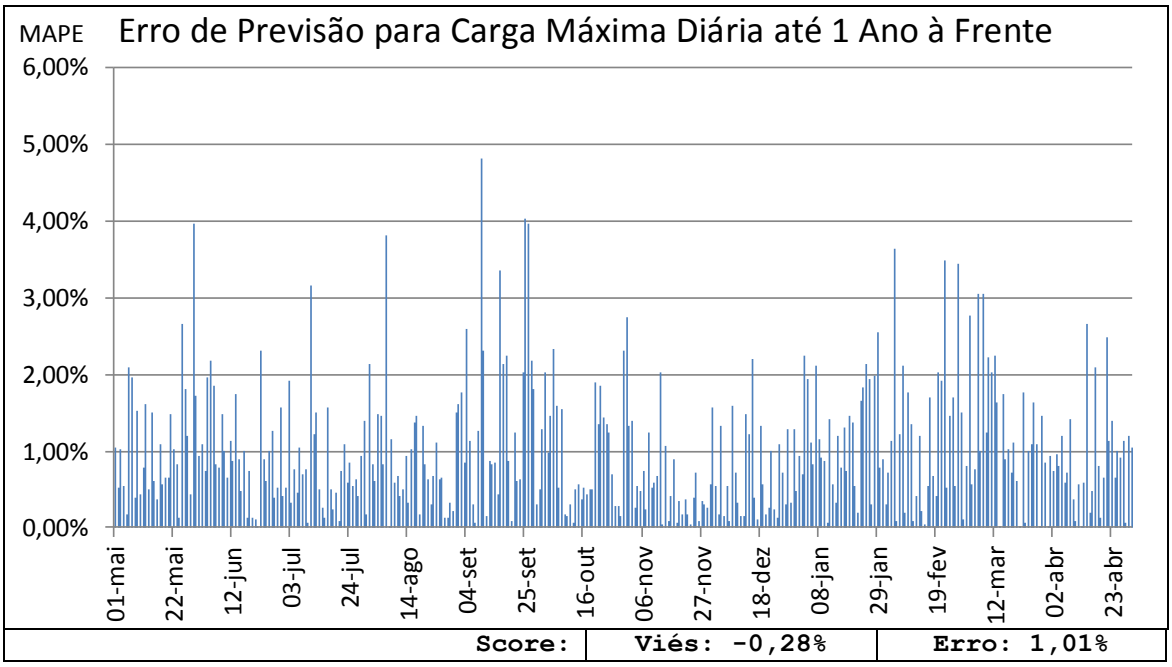


Figura 59 – Erro de previsão da carga máxima diária, 1 ano à frente

Apenas um valor ultrapassou de maneira significativa o erro de 4%. O valor previsto foi de 6523MW, enquanto o valor real da carga foi de 6222MW, um erro de aproximadamente 300MW. Isso ocorreu no dia 9 de setembro, um sábado, logo após o feriado de independência. Possivelmente, devido ao feriado da quinta-feira, o pico de carga do sábado foi mais baixo que o usual. Porém, como o valor do erro foi positivo, não haveria um grande impacto na operação sistêmica, já que o valor previsto foi maior que o valor efetivamente registrado.

8.17 - Viés da Previsão da Carga Máxima Diária até 1 Ano à Frente

É interessante observar que o viés da carga máxima deve ser analisado de maneira especial, separando os erros positivos (quando a previsão é maior que a carga real) dos erros negativos (quando a previsão é menor que a carga real). Os erros positivos impactam menos a operação do sistema, se comparados com os erros negativos, pois estes implicam em um valor maior do que o previsto para aquele horário, podendo causar, por exemplo, sobrecarga em alguns pontos do sistema elétrico.

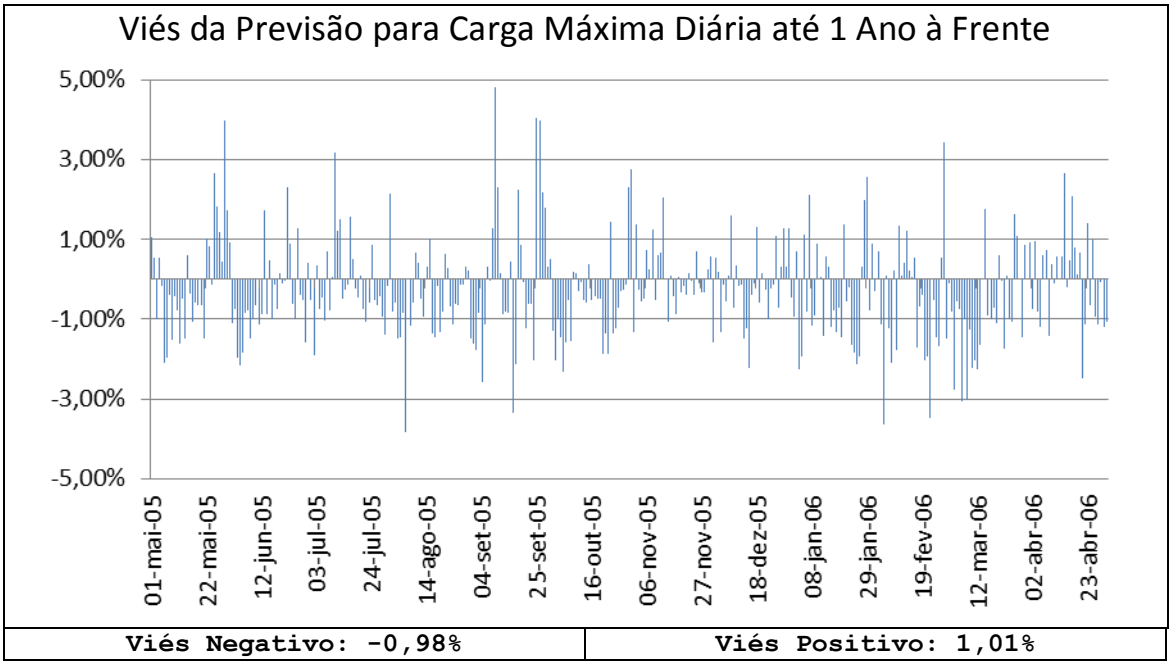


Figura 60 – Viés de previsão da carga máxima diária, 1 ano à frente

A Figura 60 mostra o viés da previsão da carga máxima para os 365 dias do horizonte de previsão. A média dos erros negativos (aqueles que influenciam mais a operação das redes elétricas) teve um valor médio de -0,98%, e somente em 4 dias o

valor ultrapassou 3%. Estes resultados mostram a qualidade da previsão para o planejamento das redes elétricas e para as diversas ações necessárias para a correta operação do sistema, pois os equipamentos e dispositivos da rede são dimensionados para suportar valores temporários bem maiores, ao redor de 10% de sua capacidade.

Capítulo 9
Conclusões

O problema de previsão da carga elétrica nos sistemas de potência possui grandes desafios, que vão desde a qualidade dos registros históricos, até os diferentes fatores que influenciam a carga.

Não é raro encontrar históricos com dados inconsistentes, incompletos, com erros de medição ou descontinuidades. Num mesmo sistema, podem existir diferentes fontes de dados, com diferentes tecnologias, implementadas em diferentes épocas e com diversas frequências de medição.

O comportamento da carga elétrica é influenciado por diversos fatores, cada um causando um impacto nos diversos horizontes de previsão. Além dos ciclos tradicionais, como o diário e o semanal, a carga também sofre influência de feriados, estações do ano, mudanças de temperatura, vigência de horário de verão, e até de hábitos e costumes sociais.

Desenvolver técnicas de modelagem que permitam superar estas dificuldades é um desafio que começou a ser superado somente nos últimos tempos, graças ao desenvolvimento de técnicas avançadas de mineração de dados com Big Data e aprendizado de máquina.

Este trabalho apresentou todo o processo de desenvolvimento de um modelo de previsão de cargas, propondo o uso e o aprimoramento de técnicas de Big Data para a geração de um modelo de Florestas Aleatórias (*Random Forests*).

As Florestas Aleatórias funcionam pelo princípio dos modelos combinados, onde vários modelos com baixa correlação são combinados para produzir uma saída com alto grau de precisão. Dentre as principais vantagens das florestas aleatórias, pode-se citar a pequena quantidade de parâmetros para ajuste, a baixa intervenção humana no seu treinamento, sua capacidade de tratar dados com ruído, sua facilidade

e rapidez de treinamento, e principalmente, sua ótima precisão. Este bom desempenho deriva diretamente de sua capacidade de não sofrer sobreajuste (overfitting). Assim, pode-se tentar aumentar o tamanho do modelo, buscando diminuir o erro, sem o perigo de perder desempenho.

No estudo de caso, observou-se a consistência dos resultados, nos diversos horizontes de previsão. Os erros se mantiveram baixos nas diversas situações, tanto em dias típicos de semana e fim de semana, como em dias atípicos como feriados. O modelo mostrou-se capaz de gerar boas previsões tanto para valores de carga de 15 em 15 minutos, como também para a carga média diária.

Além de erros baixos, a técnica utilizada gerou um viés bem próximo de zero. Isso mostra seu valor principalmente na contratação antecipada de energia, onde o objetivo é prever de modo preciso, a quantidade de energia a ser consumida em um determinado horizonte, evitando-se tanto a compra de energia no mercado *spot*, como a contratação de excedentes de energia não utilizada.

9.1 - Principais Contribuições

A principal contribuição deste trabalho foi a proposição de alterações na aplicação das Florestas Aleatórias para a previsão de carga elétrica. A aplicação de técnicas de mineração com Big Data possibilita gerar um modelo capaz de prever valores de 15 em 15 minutos, para diversos horizontes de previsão, com grande exatidão, e sem a necessidade de grandes ajustes ou da intervenção humana. Os valores gerados podem ainda ser agrupados e filtrados de maneira conveniente, para utilização em diversas áreas diferentes de uma empresa.

Neste trabalho utilizaram-se dados reais de uma conhecida distribuidora de energia nacional, sendo que os resultados da previsão foram considerados muito bons. Dentre as outras contribuições deste trabalho, destacam-se:

1) O ganho de velocidade no treinamento, devido à modificação proposta no processo de partição, que dispensou a ordenação dos dados;

2) O uso do critério de parada ótima no processo de busca pelo subespaço aleatório de atributos, fornecendo uma fundamentação teórica a um antigo parâmetro empírico;
3) O tratamento de atributos nulos com a lógica de três estados, tornando o método robusto quando a uma base de dados incompleta, contendo inconsistências e ruído;
4) O ótimo desempenho do modelo de previsão, principalmente para aplicação em contratos de compra e venda de energia;
5) A possibilidade de previsão em diversos horizontes, e com diversas resoluções;
6) A consistência da precisão mesmo para dias atípicos, como feriados, feriados prolongados e durante o horário de verão;
7) A facilidade de programação, e a direta aplicação do modelo, sem a necessidade de grande preparação ou transformação dos dados brutos.

9.2 - Propostas para Trabalhos Futuros

Apesar da técnica de Floresta Aleatória ser facilmente adaptada para um ambiente distribuído, com processamento paralelo, este trabalho não chegou a implementar as rotinas necessárias para geração das árvores de decisão em diferentes máquinas. Isso reduziria significativamente o tempo gasto para criação do modelo.

Sugere-se também, para futuros trabalhos, a aplicação de técnicas de derivação conceitual para o ajuste contínuo do modelo, conforme houver maior disponibilidade de dados históricos. A alocação dinâmica de mais árvores ao modelo, e a eliminação de outras que passem a apresentar desempenho fraco, pode trazer benefícios aos resultados.

Outra possibilidade interessante é o estudo do comportamento do modelo com deriva conceitual durante uma época de grande crise, como ocorrido em 2001, onde a atividade econômica passou por grandes mudanças. Este também é um desafio que deve ser enfrentado no futuro.

Referências

[1] HS Hippert, CE Pedreira, RC Souza "Neural networks for short-term load forecasting: A review and evaluation" IEEE Transactions on Power Systems, 16.1 (2001) 44-55

[2] Y Chen, et al. "Short-term load forecasting: similar day-based wavelet neural networks" IEEE Transactions on Power Systems, 25.1 (2010) 322-330

[3] KH Kim, HS Youn, YC Kang "Short-term load forecasting for special days in anomalous load conditions using neural networks and fuzzy inference method" IEEE Transactions on Power Systems, 15.2 (2000) 559-565

[4] P Bunnoon, K Chalermyanont, C Limsakul "A Computing Model of Artificial Intelligent Approaches to Mid-term Load Forecasting: a state-of-the-art-survey for the researcher" International Journal of Engineering and Technology, 2.1 (2010) 94

[5] D Bassi, O Olivares "Medium term electric load forecasting using TLFN neural networks" International Journal of Computers Communications & Control, 1.2 (2006) 23-32

[6] M Ghiassi, DK Zimbra, H Saidane "Medium term system load forecasting with a dynamic artificial neural network model" Electrical Power System Research, 76.5 (2006) 302-316

[7] MC Falvo, R Lamedica "Meteorological parameters influence for medium term load forecasting" Transmission and Distribution Conference and Exhibition IEEE PES (2005/2006) 1296-1301

[8] MC Falvo, R Lamedica "A Knowledge based system for medium term load forecasting" Transmission and Distribution Conference and Exhibition IEEE PES (2005/2006) 1291-1295

[9] HM Al-Hamadi, SA Soliman “Long-term/mid-term electric load forecasting based on short-term correlation and annual growth” Electric Power Systems Research, 74.3 (2005) 353-361

[10] EA Feinberg, D Genethliou “Load forecasting” Applied Mathematics for Restructured Electric Power Systems (2005) 269-285

[11] H Hahn, S Meyer-Nieberg, S Pickl “Electric load forecasting methods: Tools for decision making” European Journal of Operational Research, 199.3 (2009) 902-907

[12] D Wang, Z Sun “Big data analysis and parallel load forecasting of electric power user side” Proceedings of the CSEE, 35.3 (2015) 527-537

[13] P Wang, B Liu, T Hong “Electric load forecasting with recency effect: A big data approach” International Journal of Forecasting, 32.3 (2016) 585-597

[14] M Kezunovic, L Xie, S Grijalva “The role of big data in improving power system operation and protection” Bulk Power System Dynamics and Control-IX Optimization, Security and Control of the Emerging Power Grid (IREP), 2013 IREP Symposium, IEEE (2013) 1-9

[15] S Sagiroglu, D Sinanc “Big data: A review” Collaboration Technologies and Systems (CTS), 2013 International Conference, IEEE (2013) 42-47

[16] PD Diamantoulakis, VM Kapinas, GK Karagiannidis “Big data analytics for dynamic energy management in smart grids” Big Data Research, 2.3 (2015) 94-101

[17] M Mayilvaganan, M Sabitha “A cloud-based architecture for Big-Data analytics in smart grid: A proposal” Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference, IEEE (2013) 1-4

[18] Y Song, G Zhou, Y Zhu “Present status and challenges of big data processing in smart grid” Power System Technology, 37.4 (2013) 927-935

[19] A Deihimi, O Orang, H Showkati “Short-term electric load and temperature forecasting using wavelet echo state networks with neural reconstruction” Energy, 57. 4 (2013) 382-401

[20] Y Simmhan, S Aman, and A Kumbhare “Cloud-based software platform for big data analytics in smart grids” Computing in Science & Engineering, 15.4 (2013) 38-47

[21] BG Malkiel,.EF Fama “Efficient capital markets: A review of theory and empirical work” The Journal of Finance, 25.2 (1970) 383-417

[22] A Labrinidis, and HV Jagadish “Challenges and opportunities with big data” Proceedings of the VLDB Endowment, 5.12 (2012) 2032-2033

[23] D Laney “3D Data Management: Controlling data volume, velocity and variety” META Group Research, Note 6 (2001) 70

[24] IBM “What is big data?” Consultado em fev 2016, em https://www.ibm.com/analytics/us/en/big-data/

[25] M Kezunovic, et al. “The big picture: Smart research for large-scale integrated smart grid solutions” IEEE Power and Energy Magazine, 10.4 (2012) 22-34

[26] K Michael, KW Miller “Big data: New opportunities and new challenges” Computer, 46.6 (2013) 22-24

[27] P Michalik, J Stofa, I Solotova “Concept definition for big data architecture in the education system” Applied Machine Intelligence and Informatics (SAMI), 12th International Symposium IEEE (2014) 331-334

[28] S Kaisler, F Armour, JA Espinosa, W Money “Big Data: issues and challenges moving forward” System Sciences HICC 2013, 46th Annual Hawaii International Conference (2013) 995-1004

[29] U Fayyad, G Piatetsky-Shapiro, and P Smyth “From data mining to knowledge discovery in databases” AI magazine, 17.3 (1996) 37

[30] P Chapman, et al. “CRISP-DM 1.0 Step-by-step data mining guide” SPSS (2000)

[31] Han, Jiawei, Jian Pei, Micheline Kamber “Data mining: concepts and techniques” Elsevier (2011)

[32] J Pitt, et al “Transforming big data into collective awareness” Computer, 46.6 (2013) 40-45

[33] H Chen, RHL Chiang, VC Storey “Business intelligence and analytics: From big data to big impact” MIS Quarterly, 36.4 (2012)

[34] R Kune, et al. “The anatomy of big data computing” Software: Practice and Experience, 46.1 (2016) 79-105

[35] VN Vapnik “The Nature of Statistical Learning Theory” New York, Springer Verlag, (1995)

[36] KU Sattler, O Dunemann “SQL database primitives for decision tree classifiers.” Proceedings of the Tenth International Conference on Information and Knowledge Management, ACM (2001) 379-386

[37] E Rahm, HH Do “Data cleaning: Problems and current approaches” IEEE Data Eng. Bull., 23.4 (2000) 3-13

[38] T Dasu, T Johnson “Exploratory data mining and data cleaning” John Wiley & Sons (2003)

[39] R Kimball, J Caserta “The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data” John Wiley & Sons (2011)

[40] DJ Hand, H Mannila, P Smyth “Principles of data mining” MIT Press (2001)

[41] IH Witten, et al. “Data Mining: Practical machine learning tools and techniques” Morgan Kaufmann (2016)

[42] P Barnaghi, A Sheth, C Henson “From data to actionable knowledge: Big data challenges in the web of things” IEEE Intelligent Systems, 28.6 (2013) 6-11

[43] X Wu, et al. “Top 10 algorithms in data mining” Knowledge and Information Systems, 14.1 (2008) 1-37

[44] P Refaeilzadeh, L Tang, H Liu “Cross-validation” Encyclopedia of Database Systems, Springer US (2009) 532-538

[45] TG Dietterich and EB Kong “Machine learning bias, statistical bias, and statistical variance of decision tree algorithms” Technical Report, Department of Computer Science, Oregon State University (1995)

[46] L Breiman “ Bias, variance, and arcing classifiers” Technical Report 460, Statistics Department, University of California (1996)

[47] P Geurts “Contributions to decision tree induction: bias/variance tradeoff and time series classification”, Tese de Doutorado, University of Liège Belgium. (2002)

[48] Q Mu, Y Wu, X Pan, L Huang, X Li “Short-term load forecasting using improved similar days method” Asia-Pacific Power and Energy Engineering Conference (APPEEC), IEEE (2010) 1-4

[49] W Charytoniuk, MS Chen, PV Olinda “Nonparametric Regression Based Short-Term Load Forecasting” IEEE Transactions on Power Systems, 13 (1998) 725-730

[50] T Haida, S Muto “Regression Based Peak Load Forecasting using a Transformation Technique” IEEE Transactions on Power Systems, 9 (1994) 1788-1794

[51] H Mori, N Kosemura “Optimal Regression Tree Based Rule Discovery for Short-Term Load Forecasting” Proceedings of IEEE Power Engineering Society Transmission and Distribution Conference, 2 (2001) 421–426

[52] MY Cho, JC Hwang, CS Chen “Customer Short-Term Load Forecasting by using ARIMA Transfer Function Model” Proceedings of the International Conference on Energy Management and Power Delivery, 1 (1995) 317-322

[53] TWS Chow, CT Leung “Nonlinear Autoregressive Integrated Neural Network Model for Short-Term Load Forecasting” IEE Proceedings on Generation, Transmission and Distribution, 143 (1996) 500-506

[54] HT Yang, CM Huang, CL Huang “Identification of ARMAX Model for Short-Term Load Forecasting: An Evolutionary Programming Approach” IEEE Transactions on Power Systems, 11 (1996) 403–408

[55] K Ho, YY Hsu, FF Chen, TE Lee, CC Liang, TS Lai, and KK Chen “Short-Term Load Forecasting of Taiwan Power System” Applied Mathematics for Power Systems using a Knowledge Based Expert System, IEEE Transactions on Power Systems, 5 (1990) 1214–1221

[56] S Rahman “Formulation and Analysis of a Rule-Based Short-Term Load Forecasting Algorithm” Proceedings of the IEEE, 78 (1990) 805-816

[57] S Rahman, O Hazim “Load Forecasting for Multiple Sites: Development of an Expert System-Based Technique” Electric Power Systems Research, 39 (1996) 161–169

[58] AG Bakirtzis, V Petridis, SJ Kiartzis, MC Alexiadis, AH Maissis “A Neural Network Short-Term Load Forecasting Model for the Greek Power System” IEEE Transactions on Power Systems, 11 (1996) 858–863

[59] H Chen, CA Canizares, A Singh “ANN-Based Short-Term Load Forecasting in Electricity Markets” Proceedings of the IEEE Power Engineering Society Transmission and Distribution Conference, 2 (2001) 411–415

[60] A Khotanzad, RA Rohani, TL Lu, A Abaye, M Davis, DJ Maratukulam “ANNSTLF – A Neural-Network-Based Electric Load Forecasting System” IEEE Transactions on Neural Networks, 8 (1997) 835-846

[61] A Khotanzad, RA Rohani, D Maratukulam “ANNSTLF – Artificial Neural Network Short-Term Load Forecaster – Generation Three” IEEE Transactions on Neural Networks, 13 (1998) 1413-1422

[62] DB Fogel “An Introduction to Simulated Evolutionary Optimization” IEEE Transactions on Neural Networks, 5 (1994) 3-14

[63] AD Papalexopoulos, S Hao, TM Peng “An Implementation of a Neural Network Based Load Forecasting Model for the EMS” IEEE Transactions on Power Systems, 9 (1994) 1956-1962

[64] M Peng, NF Hubele, GG Karady “Advancement in the Application of Neural Networks for Short-Term Load Forecasting” IEEE Transactions on Power Systems, 7 (1992) 250-257

[65] JW Taylor, R Buizza “Neural Network Load Forecasting With Weather Ensemble Predictions” IEEE Transactions on Power Systems, 17 (2002) 626-632

[66] SJ Kiartzis, AG Bakirtzis “A Fuzzy Expert System for Peak Load Forecasting: Application to the Greek Power System” Proceedings of the 10th Mediterranean Electrotechnical Conference, 3 (2000) 1097-1100

[67] V Miranda, C Monteiro “Fuzzy Inference in Spatial Load Forecasting” Proceedings of IEEE Power Engineering Winter Meeting, 2 (2000) 1063-1068

[68] SE Skarman, M Georgiopoulous “Short-Term Electrical Load Forecasting using a Fuzzy ARTMAP Neural Network” Proceedings of SPIE (1998) 181-191

[69] HT Yang, CM Huang “A New Short-Term Load Forecasting Approach using Self-Organizing Fuzzy ARMAX Models” IEEE Transactions on Power Systems, 13 (1998) 217-225

[70] BJ Chen, MW Chang, CJ Lin “Load Forecasting using Support Vector Machines: A Study on EUNITE Competition 2001” Technical Report, Department of Computer Science and Information Engineering, National Taiwan University (2002)

[71] N Christiani, JS Taylor “An Itroduction to Support Vector Machines and Other Kernel-Based Learning Methods” Cambridge University Press, Cambridge (2000)

[72] Y Li, T Fang “Wavelet and Support Vector Machines for Short-Term Electrical Load Forecasting” Proceedings of International Conference on Wavelet Analysis and its Applications, 1 (2003) 399-404

[73] M Mohandes “Support Vector Machines for Short-Term Electrical Load Forecasting” International Journal of Energy Research, 26 (2002) 335-345

[74] L Breiman, J Friedman, CJ Stone, RA Olshen “CART - Classification and regression trees” CRC press (1984)

[75] S Sindhu, S Sivatha, S Geetha, A Kannan “Decision tree based light weight intrusion detection using a wrapper approach” Expert Systems with Applications, 39.1 (2012) 129-141

[76] L Rokach, O Maimon “Top-down induction of decision trees classifiers-a survey” IEEE Transactions on Systems, Man, and Cybernetics, Part C, 35.4 (2005) 476-487

[77] Q Ding “Long-term load forecast using decision tree method” Power Systems Conference and Exposition 2006, IEEE PES (2006)

[78] MS Chen, J Han, PS Yu “Data mining: an overview from a database perspective” IEEE Transactions on Knowledge and data Engineering, 8.6 (1996) 866-883

[79] JZ Kolter, MA Maloof “Dynamic weighted majority: A new ensemble method for tracking concept drift” International Conference Data Mining 2003, IEEE (2003)

[80] JS Vitter, “Random sampling with a reservoir”. ACM Transactions on Mathematical Software (TOMS), 11.1 (1985) 37-57

[81] B Efron, R Tibshirani “Improvements on cross-validation: the 632+ bootstrap method” Journal of the American Statistical Association, 92.438 (1997) 548-560

[82] B Hendrickson, RW Leland “A Multi-Level Algorithm for Partitioning Graphs” SC, 95.28 (1995)

[83] TS Ferguson “Optimal Stopping and Applications” Mathematics Department, UCLA

[84] TS Ferguson “Who solved the secretary problem?” Statistical Science, 4 (1989) 282-296

[85] TP Hill “Knowing When to Stop” American Scientist, 97, (2009) 126-133

[86] Y Chow, H Robbins, D Siegmund “Great expectations: the theory of optimal stopping” Dover (1991)

[87] PR Freeman “The secretary problem and its extensions: A review” International Statistical Review, 51 (1983) 189-206

[88] AV Gnedin, “A Solution to the Game of Googol” Annals of Probability, 22 (1994) 1588-1595

[89] JN Bearden “A new secretary problem with rank-based selection and cardinal payoffs” Journal of Mathematical Psychology, 50 (2006) 58-59

[90] JN Bearden, A Rapoport, RO Murphy “Sequential observation and selection with rank-dependent payoffs: An experimental test” Management Science, 52 (2006) 1437-1449

[91] JP Gilbert, F Mosteller “Recognizing the maximum of a sequence” Journal of the American Statistical Association, 61 (1966) 35-73

[92] TG Dietterich “An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization” Machine Learning, 40.2 (2000) 139–157

[93] Y Freund, RE Schapire “A desicion-theoretic generalization of on-line learning and an application to boosting” Computational Learning Theory, Springer (1995) 23-37

[94] H Drucker, et al. “Boosting and other ensemble methods” Neural Computation, 6.6 (1994) 1289-1301

[95] V Svetnik, et al. “Random forest: a classification and regression tool for compound classification and QSAR modeling” Journal of Chemical Information and Computer Sciences, 43.6 (2003) 1947-1958

[96] C Strobl, et al. “Bias in random forest variable importance measures: Illustrations, sources and a solution” BMC Bioinformatics, 8.1 (2007) 25

[97] TG Dietterich “Ensemble methods in machine learning” Multiple Classifier Systems, 1857 (2000) 1-15

[98] L Breiman “Random forests” UC Berkeley TR567 (1999)

[99] SW Kwok, C Carter “Multiple decision trees” Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence, North-Holland Publishing Co. (1990) 327–338

[100] L Breiman “Bagging predictors” Machine Learning, 24.2 (1996) 123-140

[101] B Efron “Bootstrap methods: another look at the jack knife” The Annals of Statistics, (1979) 1–26

[102] Y Amit, D Geman, K Wilder “Joint induction of shape features and tree classifiers” IEEE Transactions on Pattern Analysis and Machine Intelligence, 19.11 (1997) 1300-1305

[103] TK Ho “The random subspace method for constructing decision forests” IEEE Transactions on Pattern Analysis and Machine Intelligence, 20.8 (1998) 832–844

[104] P Panov, S Džeroski “Combining bagging and random subspaces to create better ensembles” Advances in Intelligent Data Analysis VII (2007) 118–129

[105] G Louppe, P Geurts “Ensembles on random patches” Machine Learning and Knowledge Discovery in Databases, Springer (2012) 346-361

[106] L Breiman “Random Forests” Machine Learning, 45.1 (2001) 5-32

[107] L Breiman “Manual on setting up, using, and understanding random forests v3.1” Statistics Department, University of California Berkeley (2002)

[108] A Cutler, G Zhao “Pert - Perfect random tree ensembles” Computing Science and Statistics, 33 (2001) 490–497

[109] G Zhao “A new perspective on classification” PhD thesis, Utah State University, Department of Mathematics and Statistics (2000)

[110] L Wehenkel “Discretization of continuous attributes for supervised learning: Variance evaluation and variance reduction” Proceedings of the Seventh International Fuzzy Systems Association World Congress, 1 (1997)

[111] P Geurts, L Wehenkel “Investigation and reduction of discretization variance in decision tree induction” Machine Learning, ECML2000, Springer (2000) 162-170

[112] P Geurts, D Ernst, L Wehenkel “Extremely randomized trees” Machine Learning, 63.1 (2006) 3-42

[113] JJ Rodriguez, LI Kuncheva, CJ Alonso “Rotation forest: A new classifier ensemble method” IEEE Transactions on Pattern Analysis and Machine Intelligence, 28.10 (2006) 1619-1630

[114] LI Kuncheva, JJ Rodríguez “An experimental study on rotation forest ensembles” Multiple Classifier Systems, Springer (2007) 459–468

[115] A Saffari, et al. “On-line random forests” Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference, IEEE (2009)

[116] CI de Almeida Costa, VC do Nascimento, G Lambert-Torres, LEB da Silva: “Control model for distributed generation and network automation for microgrids operation” Electric Power Systems Research, 127 (2015) 151-159