Capítulo 4

Avaliação de Modelos de Predição e Previsão Construídos por Algoritmos de Aprendizado de Máquina em Problemas de Cidades Inteligentes

Igor Garcia Sampaio, Flavia Bernardini, Aline Paes, Eduardo de Oliveira Andrade, José Viterbo

Abstract

The availability of open data is one of the fundamental pillars in Smart Cities. Such data can be used as input for computer systems providing new services to the population. In this context, sensor data along with machine learning approaches have been used to forecast the demand of several public services, such as electric power, gas, taxi, among others. In general, these problems are treated by supervised learning algorithms, whose goal is to find a model from training data. However, several algorithms can be considered to obtain a model, and even if a single algorithm is chosen, several parameters can be varied, which leads to obtaining multiple candidate models to solve a problem. In addition, one of the key features of machine learning lies in the fact that no algorithm is best for all situations. In this way, it becomes important to systematize the model evaluation process, especially in the context of Smart Cities, which may involve the use of machine learning approaches in batch or data flow. In Brazil, the process of evaluating classification and regression models is still a challenge for many solution developers. Thus, the purpose of this short course is to present the model evaluation process for prediction and forecast, built using batch machine learning algorithms and data flow. For this, tools and datasets available on the Internet are used to illustrate the modeling of the problem and the evaluation process of the built models.

Resumo

Em Cidades Inteligentes, um dos pilares fundamentais é a disponibilização de dados abertos, que podem ser utilizados como insumo por sistemas computacionais para a oferta de novos serviços para a população. Nesse contexto, dados de sensores juntamente com abordagens baseadas em aprendizado de máquina têm sido utilizados para previsão

de demanda de diversos serviços públicos, como energia elétrica, gás, taxi, dentre outros. Em geral, esses problemas são tratados por algoritmos de aprendizado supervisionado, cuja meta é encontrar um modelo a partir de dados de treinamento. Entretanto, diversos algoritmos podem ser considerados para a obtenção de um modelo, e, mesmo que um único algoritmo seja escolhido, diversos parâmetros podem ser variados, o que leva a obtenção de múltiplos modelos candidatos à solução de um problema. Além disso, uma das principais características do aprendizado de máquina está no fato de que nenhum algoritmo é melhor para todas as situações. Desta forma, se torna importante a sistematização do processo de avaliação de modelos, principalmente no contexto de Cidades Inteligentes, que podem envolver o uso de abordagens de aprendizado de máquina em lote ou em fluxo de dados. No Brasil, o processo de avaliação de modelos de classificação e regressão ainda é um desafio para muitos desenvolvedores de soluções. Assim, o objetivo deste minicurso é apresentar o processo de avaliação de modelos para previsão e predição, construídos utilizando algoritmos de aprendizado de máquina em lote e em fluxo de dados. Para isso, são utilizadas ferramentas e coleções de dados disponibilizados na internet para ilustrar a modelagem do problema e o processo de avaliação dos modelos construídos.

4.1. Introdução

No panorama da evolução tecnológica e da crescente urbanização, ganha corpo o debate sobre o contexto e as perspectivas em torno do movimento urbano tecnológico, mundialmente conhecido como Cidades Inteligentes (do inglês *Smart Cities*). Apesar da área chamar a atenção por parte da academia nos últimos anos, o termo não é novo [Harrison and Donnely 2011]. Ele surgiu em meados dos anos 1800, na criação de novas cidades no oeste americano, pelo modelo de autogoverno eficiente. A partir do ano 2000, a expressão passou a ser adotada por empresas do ramo tecnológico, que associaram o crescimento urbano com a necessidade de incorporação tecnológica.

Há muitas diferentes definições e frameworks apresentados na literatura para Cidades Inteligentes. No entanto, observa-se, em muitos desses trabalhos, que as TICs (Tecnologia da Informação e Comunicação) e o processamento de dados são transversais em diversos domínios de problemas nas cidades, e necessitam de técnicas para análise de tais dados. Nesse sentido, aprendizado de máquina tem sido amplamente utilizado em diversos problemas para oferecer novas ferramentas para o apoio à tomada de decisão e tem sido aplicado na solução de diversos problemas reais.

Há duas tarefas principais nos quais se enquadram os problemas resolvidos por aprendizado de máquina: preditivas e descritivas. Nas tarefas preditivas, a meta é encontrar uma função, também chamada de modelo ou hipótese, a partir de dados de treinamento. Tais dados são descritos pelos atributos de entrada. Os valores possíveis de rótulo formam um conjunto que define o atributo de saída da função. Tal atributo de saída é comumente conhecido como classe.

Já nas tarefas descritivas, a meta é explorar ou descrever um conjunto de dados. Em geral, nesse tipo de problema não há um atributo classe, ou seja, os dados não foram rotulados. Assim, os algoritmos de aprendizado utilizados nessas tarefas não utilizam o atributo classe e por isso, seguem a abordagem de aprendizado não-supervisionado. Neste

capítulo, o foco está apenas no aprendizado supervisionado.

Uma das principais características do aprendizado de máquina está no fato de que nenhum algoritmo ser melhor para todas as situações, e a teoria que dá embasamento a essa afirmação é o *Free-Lunch Theorem* [Wolpert 1996]. Assim sendo, para cada problema abordado, é necessário realizar experimentos para verificar qual(is) o(s) algoritmo(s) apresenta(m) melhor comportamento no conjunto de dados disponível. Assim sendo, uma sistematização do processo de avaliação é importante.

Com isso, o objetivo deste capítulo é apresentar conceitos importantes de aprendizado de máquina, incluindo aprendizado em lote e online. Alguns cenários de uso do aprendizado de máquina para previsão de demanda nas Cidades Inteligentes são apresentados. Descrevemos ainda como é realizada a avaliação dos conhecimentos extraídos pelos algoritmos de aprendizado em lote e online, bem como as ferramentas utilizadas na demonstração do processo de avaliação.

4.2. Cidades Inteligentes e TICs

Há muitos trabalhos na literatura que apresentam diferentes definições e frameworks para Cidades Inteligentes [Giffinger 2016, Oliveira and Campolargo 2015, Guedes et al. 2018, Anthopoulos 2015, Gil-Garcia et al. 2015]. Isso é devido a cada cidade ter uma perspectiva diferente sobre a qualidade de vida e do bem-estar de seus cidadãos. Por outro lado, algumas características importantes têm se destacado nas diferentes tentativas de definição das cidades inteligentes: (i) o cidadão é foco no processo de evolução das cidades [Giffinger 2016, Oliveira and Campolargo 2015]; (ii) as TICs têm papel fundamental no apoio aos diferentes domínios, principalmente no processo de tomada de decisão por parte de todos os atores de uma cidade, nos mais diversos eixos de serviços e domínios [Gil-Garcia et al. 2015, Anthopoulos 2015]; (iii) diversos frameworks têm sido propostos para nortear em quais componentes uma cidade pode focar para evoluir considerando o contexto de Cidades Inteligentes [Gil-Garcia et al. 2015, Guedes et al. 2018].

Em relação às características (i) e (ii), Oliveira e Campolargo [2015] afirmam que combinar informações providas por redes de sensores com aplicativos de smartphone permite a personalização dos serviços de uma cidade de acordo com sua posição, perfil e padrões de comportamento de seus cidadãos. A partir disso, é necessário evoluir o conceito de Cidades Inteligentes para focar nos cidadãos, suas necessidades, e uma colaboração aberta entre os cidadãos e as autoridades públicas — no caso de cidades, as prefeituras [Oliveira and Campolargo 2015]. Como as cidades estão cada vez maiores, a aproximação entre os cidadãos e as autoridades públicas e a personalização dos serviços de uma cidade para seus cidadãos somente é possível utilizando tecnologias mais avançadas no âmbito das TIC, em especial a Ciência da Computação e Sistemas de Informação. Ainda, tecnologias computacionais para permitir melhor entendimento de dados abertos também são fundamentais, como discutido em [Lima et al. 2018, Barcellos et al. 2017, Pinto et al. 2018].

Observa-se, nesse contexto, que as TICs e o processamento de dados oriundos dessas estruturas construídas, necessitam de técnicas para análise de tais dados. Com isso, o aprendizado de máquina tem sido amplamente utilizado em diversos problemas para oferecer novas ferramentas para o apoio à tomada de decisão. No entanto, é possível

observar em muitas situações a dificuldade de usar corretamente e saber avaliar os conhecimentos e padrões produzidos pelos algoritmos de aprendizado de máquina. Assim, entender tal avaliação é bastante importante no contexto das Cidades Inteligentes.

4.3. Aprendizado de Máquina Supervisionado em Lote e Online

No problema padrão de aprendizado supervisionado, a entrada para o algoritmo de aprendizado consiste de um conjunto S com N objetos ou exemplos $T_i, i = 1,...,N$, escolhidos de um domínio X com uma distribuição $\mathscr{D}$, desconhecida e arbitrária, da forma $\{(\mathbf{x}_1,y_1),...,(\mathbf{x}_N,y_N)\}$ para alguma função desconhecida $y = f(\mathbf{x})$. Os $\mathbf{x}_i$ são tipicamente vetores da forma (x_{i1}, x_{i2}, ..., x_{iM}) com valores discretos ou numéricos. x_{ij} refere-se ao valor do atributo j, denominado $\mathbf{X}_j$, do exemplo T_i, como mostra a Tabela 4.1. Os valores y_i referem-se ao valor do atributo Y, freqüentemente denominado classe.

Tabela 4.1. Conjunto de exemplos no formato atributo-valor

	X_1	X_2	...	X_M	Y
T_1	x_{11}	x_{12}	...	x_{1M}	y_1
T_2	x_{21}	x_{22}	...	x_{2M}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
T_N	x_{N1}	x_{N2}	...	x_{NM}	y_N

Os valores de y são tipicamente pertencentes a um conjunto discreto de classes $C_v, v = 1,...,N_{Cl}$, *i.e* $y \in \{C_1,...,C_{N_{Cl}}\}$, quando se trata de *classificação*, ou ao conjunto de números reais em caso de *regressão*. Dado um conjunto S de exemplos a um algoritmo de aprendizado, uma hipótese **h**, também denominada modelo, será induzida. Em problemas de classificação, a hipótese é também denominada classificador ou preditor. Em problemas de regressão, a hipótese é também denominada regressor ou previsor. A hipótese **h** consiste da *hipótese* feita sobre a verdadeira (mas desconhecida) função f. Dados novos exemplos **x**, o classificador, ou hipótese, **h** prediz o valor correspondente y.

No aprendizado de máquina em lote, o conjunto de dados S é inteiramente disponibilizado na fase de treinamento para construção da hipótese **h** [Ernst et al. 2005]. Esse tipo de aprendizado assume que a distribuição $\mathscr{D}$ é fixa, e portanto não há necessidade de alterar a hipótese **h** após a etapa de treinamento ter sido concluída. Assim, a amostra de dados (o conjunto S) e a sua ordem são conhecidos pelo algoritmo de aprendizagem.

No caso do aprendizado de máquina online, ou *stream learning*, apenas o conjunto de amostras possíveis é conhecido. O ambiente de aprendizagem pode mudar a todo momento, dependendo dos dados que estão sendo recebidos naquele instante [Bifet et al. 2011, Ben-David et al. 1997]. Além disso, é possível atualizar a hipótese **h** conforme novos casos são apresentados. No entanto, ainda está em fase de exploração na área de aprendizado de máquina a construção de métodos e técnicas que permitam identificar, com o passar do tempo, a característica da distribuição dos dados. Por outro lado, a abordagem de aprendizado em lote possui algoritmos muito mais estabelecidos na literatura, por ser explorada pela comunidade de aprendizado de máquina há décadas. Em geral, o poder de predição dos algoritmos em lote é melhor que os algoritmos online quando a distribuição é estacionária. Assim, ainda é necessário explorar ambas as abordagens quando os dados têm a potencialidade de serem disponibilizados em tempo

real. Por esse motivo, as abordagens em lote e online podem ser aplicadas a diferentes domínios, sendo preciso identificar qual delas utilizar em problemas específicos e, a partir daí, escolher os melhores modelos possíveis.

Ferramentas: A ferramenta Weka é bastante utilizada no contexto da abordagem de aprendizado em lote, por ter implementado os algoritmos mais tradicionais. Dentre eles, temos os algoritmos de indução de árvores de decisão e regressão (DTs, do inglês *Decision Trees*) [Quinlan 1986], e o algoritmo de retropropagação (*backpropagation*) para redes neurais perceptron multicamadas (*multi-layer perceptron*, ou MLP) [Haykin 2008], conforme levantamento apresentado em [Andrade et al. 2017], utilizados para regressão e classificação. Ambos são apresentados neste capítulo.

Já a ferramenta MOA tem sido amplamente utilizada no contexto de aprendizado online. Além disso, ela é interessante por ter sido implementada como uma extensão da ferramenta Weka, o que permite perceber como o framework que o Weka implementa é estendido na ferramenta MOA. Para acompanhar a lógica dos algoritmos utilizados na ferramenta Weka, foram utilizados no MOA os algoritmos de árvores de decisão e de construção de perceptron considerando a atualização dos modelos construídos, já que o aprendizado é online. São portanto utilizados os algoritmos VFDT (*Very Fast Decision Trees*) [Domingos and Hulten 2000] para classificação, FIMT-DD (*Fast Incremental Model Trees with Drift Detection*) [Ikonomovska et al. 2011] para regressão, e o algoritmo *Adatron* para aprendizado do modelo Perceptron na abordagem online [Anlauf and Biehl 1989], tanto para classificação quanto para regressão. Todos esses algoritmos são descritos a seguir.

4.3.1. Aprendizado em Lote — Árvores de Decisão

A característica principal das DTs está na sua capacidade de dividir um processo complexo de tomada de decisão em uma coleção de decisões mais simples, geralmente com uma solução mais fácil de compreender [Quinlan 1986]. A árvore de decisão geralmente é associada a problemas de classificação. Porém, a árvore de regressão é idêntica a uma árvore de decisão e está inserida no campo das DTs. A única diferença é que numa árvore de regressão, a saída é um valor numérico. Podemos observar dois exemplos de árvores distintas na Figura 4.1 e Figura 4.2, sendo que a primeira mostra uma árvore de decisão para um problema de decisão do domínio de transportes da Zona Oeste do Rio de Janeiro (classificação) e a segunda mostra uma árvore de regressão construída no domínio de consumo de energia elétrica.

No Weka, o algoritmo J48 é o mais amplamente utilizado para indução de árvores de decisão, por ser uma implementação do algoritmo C.45 [Quinlan 1993]. Já o M5P e o M5Rules são adaptações do C5 e do C5Rules para problemas de regressão.

4.3.2. Retropropagação para Multi-layer Perceptron

A MLP é uma rede neural alimentada para frente (*feedforward neural network*, em inglês) com neurônios conectados por diversas ligações, que possuem pesos associados. Estes neurônios estão distribuídos em diversas camadas [Riedmiller 1994]. A primeira camada geralmente é chamada de camada de entrada; as camadas intermediárias, de camadas escondidas; e a última camada, de camada de saída. Na Figura 4.3 podemos observar uma

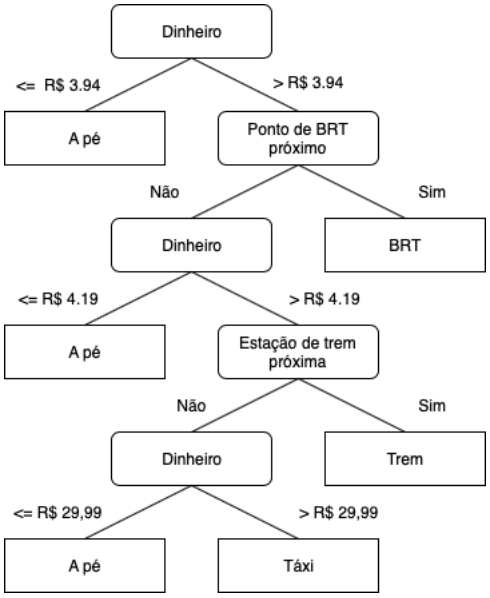


Figura 4.1. Exemplo de árvore de decisão no domínio de transportes na Zona Oeste do Rio de Janeiro

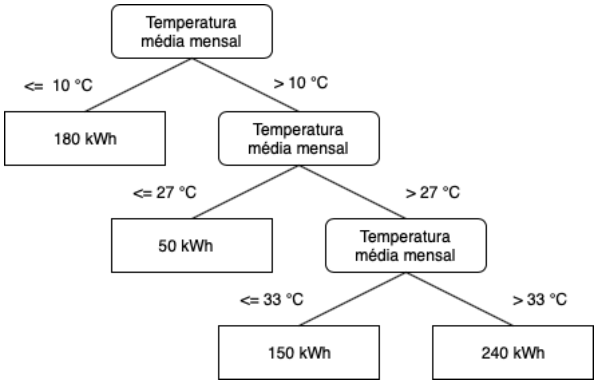


Figura 4.2. Exemplo de árvore de regressão no domínio de consumo de energia elétrica

arquitetura simples de uma MLP [Haykin 2008]. As ligações da rede MLP são completamente conectadas entre os pares de camadas adjacentes, e suas funções de ativação são contínuas e diferenciáveis [Choi and Choi 1992]. É importante observar que as redes MLP para problemas de regressão, em geral não há uma função de ativação na camada de saída, ao contrário do que acontece nos problemas envolvendo classificação. Porém, nesse caso costuma-se usar uma função de perda, como o erro quadrático médio (EQM), com a saída sendo composta de um conjunto de valores contínuos.

O treinamento, seja para obter um valor numérico como saída (problema de regressão) ou uma categoria (problema de classificação), consiste em ajustar os parâmetros ou pesos do modelo para minimizar o erro cometido nos dados disponíveis para o processo de treinamento. A retropropagação (*backpropagation*) é um algoritmo que envolve o cálculo de gradientes, sendo capaz, dessa maneira, de realizar as alterações necessárias de pesos na rede, relacionados ao erro cometido [Srinivasan et al. 2002]. Para isso, o erro é computado na saída e distribuído para trás em todas as camadas da rede neural, e assim

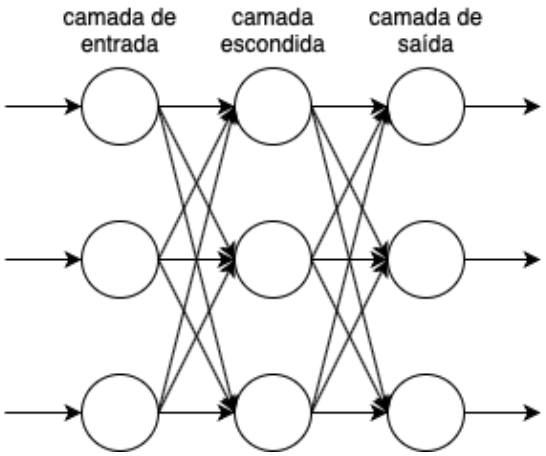


Figura 4.3. Representação de uma rede neural alimentada para frente com apenas três camadas

ocorrem os ajustes [Haykin 2008].

4.3.3. VFDT — Very Fast Decision Tree

O algoritmo foi projetado para lidar com grandes conjuntos e fluxos de dados e reduz bastante o tempo de treinamento das árvores de decisão [Domingos and Hulten 2000]. Então, é preciso processar cada amostra rapidamente, sem necessariamente armazená-las. Dessa maneira, seguindo o conceito de *data streaming*, procura-se elaborar árvores de decisões muito complexas com um custo computacional que não seja muito alto. Para a redução do custo computacional, é preciso encontrar o melhor atributo de teste para um determinado nó. Isso pode ser feito apenas com um pequeno subconjunto das amostras de treinamento que passam através deste nó. Assim, diante de um *data streaming*, as amostras iniciais serão utilizadas para escolher o teste do nó raiz. Quando o atributo para este nó for escolhido, as amostras seguintes serão passadas para as folhas correspondentes e utilizadas para escolher os atributos apropriados. Depois disso, o algoritmo continua prosseguindo recursivamente.

4.3.4. FIMT-DD — Fast Incremental Model Trees With Drift Detection

O algoritmo FIMT-DD para mineração em fluxo de dados e construção de modelos de árvores de regressão ocorre de uma maneira incremental e com divisões nessas árvores baseadas em cada atributo considerado [Wibisono et al. 2016]. Na Figura 4.4 temos um modelo de árvore gerado a partir de um *data streaming* de gás líquido [Osojnik et al. 2016]. O par u, y é correspondente às medidas de entrada e saída do sistema, respectivamente, para cada passo de tempo k.

Os nós de um modelo de árvore gerada são formados por testes em variáveis independentes, e as folhas contém as predições, análogo a como acontece com as árvores induzidas no aprendizado em lote e no algoritmo VFDT. O modelo precisa estar em monitoramento constante e atualizar toda vez que for detectada uma modificação. Se a mudança for verificada, a estratégia de adaptação será executada.

Os dados chegam em sequências de um determinado tamanho, e o algoritmo sim-

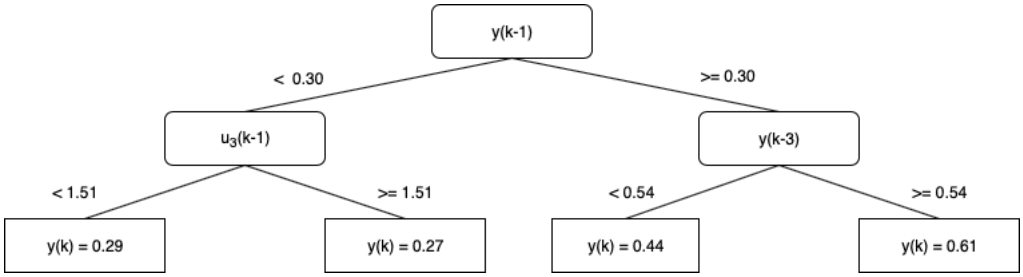


Figura 4.4. Exemplo de FIMT-DD no domínio de um separador de gás-líquido

plesmente inicia com uma folha em branco após a leitura dessas amostras. O algoritmo então, procura encontrar a divisão ótima para cada atributo e irá ranqueá-lo segundo uma métrica de avaliação. Feito isto, o FIMT-DD dividirá o atributo considerado com base no melhor atributo, de acordo com esta avaliação, gerando duas novas folhas, uma para cada ramo da divisão. Todo este processo que ocorre é estável e o risco de *overfitting* é baixo [Ikonomovska et al. 2011].

4.3.5. Perceptron Online

A primeira rede neural desenvolvida que consistiu em um algoritmo de aprendizado supervisionado, foi o *perceptron* [Rosenblatt 1958]. A sua arquitetura pode ser visualizada na Figura 6. O *perceptron* funciona como uma rede neural artificial básica, podendo receber diversos valores de entrada $(i_1, i_2, ..., i_k)$ e produzindo uma saída o através do produto com os pesos $(g_1, g_2, ..., g_k)$. Esta saída produzida é única e o viés b é adicionado para permitir que o classificador mude seu limite de decisão para a direita ou esquerda, por exemplo. No aprendizado *online*, a ideia é basicamente a mesma, seja para classificação ou regressão utilizando um fluxo de dados de entrada.

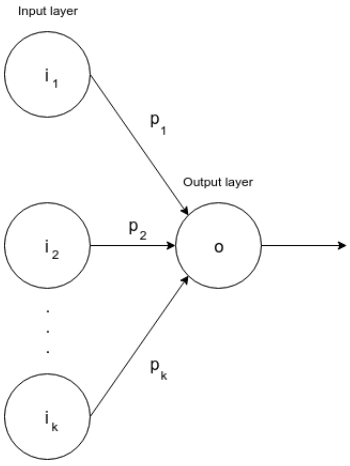


Figura 4.5. Arquitetura da rede neural perceptron

4.4. Metodologia de Avaliação dos Modelos

O conhecimento do domínio a ser pesquisado durante a aplicação de algoritmos de aprendizado de máquina a problemas reais, geralmente é provido pelo conjunto de dados, a

partir do qual a inferência de um modelo preditivo é realizada. Em alguns casos, as próprias características das técnicas existentes e do problema que está sendo solucionado podem ser consideradas para auxiliar na escolha da técnica a ser escolhida sobre um novo conjuntos de dados.

Entretanto, diversos algoritmos podem ser considerados candidados à solução de um dado problema. Mesmo que um único algoritmo seja escolhido, pode ser necessário realizar ajustes em seus parâmetros, o que leva a obtenção de múltiplos modelos para os mesmos dados.

A partir disso, existem diversas métricas de desempenho aplicáveis a problemas tanto de classificação quanto de regressão [Faceli et al. 2011]. Além disso, a ordem de apresentação dos dados é fundamental quando se considera o *stream learning* [Bifet et al. 2018]. Assim, para avaliar algoritmos de aprendizado supervisionados, utiliza-se um framework para avaliação de classificadores e regressores, que engloba:

1. Quais medidas de performances, ou métricas de avaliação, podem ser úteis, e quais são mais indicadas em algumas situações específicas;

2. Quando um conjunto inteiro está disponível para aprendizado em lote, quais técnicas de amostragem são comumente utilizadas para avaliação do poder preditivo dos modelos;

3. Como realizar os testes de hipóteses, para verificação de quais são os melhores algoritmos para um dado problema ou classe de problemas.

O framework aqui apresentado é baseado em [Demšar 2008, Faceli et al. 2011, Japkowicz and Shah 2011a, Bifet et al. 2018].

4.4.1. Métricas de Avaliação

Dado um conjunto de exemplos S, esse conjunto de exemplos deve ser utilizado na fase de treinamento e na fase de teste. Para cada fase, é utilizado um subconjunto desse conjunto de exemplos, os quais são identificados como [Bernardini 2006]:

Conjunto de Treinamento S_{tr}: Esse conjunto é a principal entrada dos algoritmos de aprendizado. É a partir dele que o modelo é construído e, portanto, deve ser um subconjunto representativo do conjunto original, ou seja, que tenha uma distribuição o mais semelhante possível do conjunto original, para que seja realizada inferência sobre S.

Conjunto de Teste S_{teste}: Esse conjunto é utilizado para avaliar o modelo construído. O conjunto de teste não deve ser apresentado ao algoritmo de aprendizado durante a construção do modelo. Idealmente, esse conjunto não deve ter exemplos em comum com o conjunto de treinamento.

4.4.1.1. Métricas para Problemas de Classificação

Utilizando o classificador (hipótese) construído **h** e as decisões tomadas pelo classificador no conjunto de teste $S_{teste} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, constrói-se uma matriz bidimensional, cujas dimensões são denominadas *classe verdadeira* e *classe predita*. A essa matriz dá-se o nome de matriz de confusão, mostrada na Tabela 4.2 para N_{Cl} classes. Cada elemento $M(C_i, C_j)$ da matriz, definido pela Equação 1, indica o número de exemplos que pertencem à classe C_i e foram preditos como pertencentes à classe C_j. Nessa equação, $||h(\mathbf{x}) = C_j||$ é igual a 1 se a igualdade $h(\mathbf{x}) = C_j$ for verdadeira (o classificador acertou a classificação do exemplo **x**), ou é igual a 0 se a igualdade for falsa (o classificador errou a classificação do exemplo **x**). O número de predições corretas para cada classe são os apresentados na diagonal principal da matriz de confusão, ou seja, os valores associados a $M(C_i, C_i)$. Todos os outros elementos da matriz $M(C_i, C_j)$, para $i \neq j$, são referentes ao número de erros cometidos em cada classe.

Tabela 4.2. Matriz de confusão

Classe Verdadeira	Predita C_1	Predita C_2	...	Predita $C_{N_{Cl}}$
C_1	$M(C_1, C_1)$	$M(C_1, C_2)$	...	$M(C_1, C_{N_{Cl}})$
C_2	$M(C_2, C_1)$	$M(C_2, C_2)$	...	$M(C_2, C_{N_{Cl}})$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$C_{N_{Cl}}$	$M(C_{N_{Cl}}, C_1)$	$M(C_{N_{Cl}}, C_2)$	...	$M(C_{N_{Cl}}, C_{N_{Cl}})$

$$M(C_i, C_j) = \sum_{\forall(\mathbf{x},y) \in S | y = C_i} ||h(\mathbf{x}) = C_j|| \tag{1}$$

Para simplificar, considere um problema de classificação com duas classes, geralmente rotulados como sendo exemplos positivos "+" e negativos "-" de um conceito. Na Tabela 4.3 é ilustrada a matriz de confusão para problemas com somente dois valores no atributo classe, onde T_P é o número de exemplos positivos classificados corretamente, F_N é o número de exemplos positivos classificados erroneamente, F_P é o número de exemplos negativos classificados erroneamente, e T_N é o número de exemplos negativos classificados corretamente, de um total de $N = (T_P + F_N + F_P + T_N)$ exemplos.

Tabela 4.3. Matriz de confusão para problemas de classificação com somente duas classes

Classe Verdadeira	Predita +	Predita -
+	Verdadeiros Positivos T_P	Falsos Negativos F_N
-	Falsos Positivos F_P	Verdadeiros Negativos T_N

A acurácia e a taxa de erro, definidas pelas Equações 2 e 3 respectivamente, são duas das medidas mais utilizadas para avaliar a performance de uma hipótese. Outras

medidas que também podem ser utilizadas para avaliar um classificador são sensibilidade ou *recall* (Equação 4), precisão (Equação 5) e F_1 (Equação 6). Deve ser observado que, para serem utilizadas tais medidas em problemas com mais de dois valores para o atributo classe, o processo que se utiliza geralmente é tornar cada classe como sendo a classe positiva e a união de todas as outras é considerada a classe negativa. Entretanto, para facilitar a utilização das medidas de *recall* e precisão, foram utilizadas neste trabalho as Equações 7 e 8, respectivamente[1].

$$Acc(\mathbf{h}) = \frac{T_P + T_N}{N} \tag{2}$$

$$Err(\mathbf{h}) = 1 - Acc(\mathbf{h}) \tag{3}$$

$$Recall(\mathbf{h}) = \frac{T_P}{T_P + F_N} \tag{4}$$

$$Prec(\mathbf{h}) = \frac{T_P}{T_P + F_P} \tag{5}$$

$$F_1(\mathbf{h}) = \frac{2 \times Prec(\mathbf{h}) \times Recall(\mathbf{h})}{Prec(\mathbf{h}) + Recall(\mathbf{h})} \tag{6}$$

$$Recall(\mathbf{h}) = \frac{\sum_{v=1}^{N_{Cl}} T_{P_{C_v}}}{\sum_{v=1}^{N_{Cl}} T_{P_{C_v}} + F_{N_{C_v}}} \tag{7}$$

$$Prec(\mathbf{h}) = \frac{\sum_{v=1}^{N_{Cl}} T_{P_{C_v}}}{\sum_{v=1}^{N_{Cl}} T_{P_{C_v}} + F_{P_{C_v}}} \tag{8}$$

4.4.1.2. Métricas para Problemas de Regressão

No caso de problemas de regressão, para um conjunto de N exemplos, ou objetos no conjunto de teste S_{teste}, o erro da hipótese $\mathbf{h}$ pode ser calculado pela distância entre o valor y_i conhecido e aquele predito pelo modelo, $\mathbf{h}(x_i)$, para cada exemplo $x_i \in S_{teste}$. As medidas de erro mais conhecidas são (i) Erro Quadrático Médio, ou MSE (*Mean Squared Error*), definido pela Eq. 9; Erro Médio Absoluto, ou MAE (*Mean Absolute Error*), definido pela Eq. 10; e Raiz Quadrada do Erro Médio Absoluto, ou RMSE (Root Mean Squared Error), definido pela Eq. 11. As métricas MSE, MAE e RMSE devem sempre apresentar valores não negativos. Quanto menor o valor dessas métricas, melhor o modelo será.

[1] A definição das medidas *Recall* e *Prec* para classes com mais de dois valores foi baseada na definição utilizada no KDD-Cup de 2005 — `http://www.acm.org/sigs/sigkdd/kdd2005/kddcup.html`.

$$MSE(\mathbf{h}) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \mathbf{h}(x_i)\right)^2 \tag{9}$$

$$MAE(\mathbf{h}) = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \mathbf{h}(x_i)\right| \tag{10}$$

$$RMSE(\mathbf{h}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \mathbf{h}(x_i)\right)^2} \tag{11}$$

No entanto, quando o atributo de interesse não está no intervalo $[0,1]$, é comum utilizar métricas normalizadas, dentre as quais se destacam a Raiz Quadrada do Erro Relativo, ou RRSE (*Root Relative Squared Error*), definida pela Eq. 12; e Porcentagem do Erro Médio Absoluto, ou MAPE (Mean Absolute Percentage Error), definida pela Eq. 13. Em ambas as equações, $\bar{y}$ é dado pela Eq. 14.

$$RRSE(\mathbf{h}) = \sqrt{\frac{\sum_{i=1}^{N}\left(y_i - \mathbf{h}(x_i)\right)^2}{\sum_{i=1}^{N}\left(y_i - \bar{y}\right)^2}} \tag{12}$$

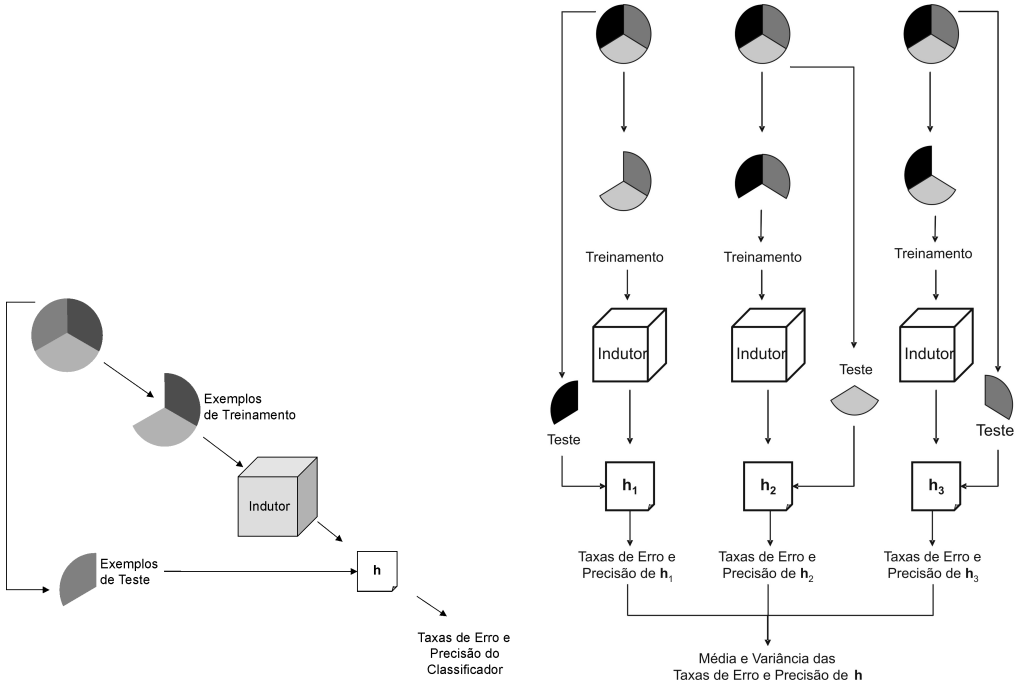
$$MAPE(\mathbf{h}) = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|y_i - \mathbf{h}(x_i)\right|}{\left|y_i - \bar{y}\right)|} \tag{13}$$

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{14}$$

4.4.2. Técnicas de Amostragem

Em muitos problemas nos quais se deseja aplicar algoritmos de aprendizado de máquina, obtém-se um conjunto S com N objetos, conforme dito anteriormente, o qual deve ser empregado na indução do modelo, e também na sua avaliação. O desempenho preditivo do modelo precisa então ser calculado, por meio da taxa de acerto ou de erro, no caso de classificação, ou por meio de outras métricas para classificação ou para regressão, descritas anteriormente (Seção 4.4.1). No entanto, calcular o desempenho nos mesmos objetos empregados em seu treinamento leva a estimativas otimistas, uma vez que todos os algoritmos de aprendizado de máquina tentam melhorar, de alguma forma, o seu desempenho preditivo nesses objetos durante a fase indutiva. Portanto, deve-se utilizar métodos de amostragem para obter estimativas de desempenho preditivo mais confiáveis, definindo subconjuntos de treinamento e de teste. Os dados de treinamento devem ser empregados na indução e no ajuste do modelo, enquanto os exemplos de testes simulam a apresentação de objetos novos ao modelo, os quais não foram vistos em sua indução. Essa divisão entre treinamento e teste visa assegurar que as medidas de desempenho sejam aplicadas em um conjunto de exemplos diferente daquele utilizado no treinamento (aprendizado). Na Figura 4.6, é possível visualizar duas das principais técnicas de amostragem existentes, descritos nas subseções a seguir — o Holdout e o método de Validação Cruzada por r partições, mais conhecido por *K-Fold Cross-Validation*.

Figura 4.6. Técnicas de amostragem [Bernardini 2006]



(a) Holdout

(b) Validação Cruzada por r partições, com $r = 3$

4.4.2.1. Holdout

Nesse método, um conjunto separado de instâncias é reservado para avaliar o desempenho do classificador. Este conjunto é diferente do conjunto de treinamento usado pelo algoritmo de aprendizado. Divide-se então o conjunto de dados S em uma proporção de p para treinamento e $1-p$ para teste, como mostrado na Figura 4.7(a). Normalmente, emprega-se $p = \frac{2}{3}$. O desempenho do classificador em um conjunto de testes separado é geralmente um bom indicador de seu desempenho de generalização. Garantias formais sobre o desempenho do conjunto de testes em termos de intervalos de confiança teóricos podem ser fornecidas neste caso [Faceli et al. 2011].

Uma das maiores vantagens de uma estimativa de erro utilizando Holdout reside na sua independência do conjunto de treinamento. Como a estimativa é obtida em um conjunto de testes em separado, algumas generalizações concretas nessa estimativa podem ser obtidas. Além disso, há outra diferença crucial entre uma estimativa de erro de validação e uma estimativa reamostrada. Uma estimativa de validação pertence à saída do classificador pelo algoritmo de aprendizado, dado os dados de treinamento. Portanto, qualquer generalização feita sobre essa estimativa se aplicará essencialmente a qualquer classificador com o desempenho determinado do conjunto de testes.

4.4.2.2. Validação Cruzada

No método de validação cruzada por r partições, o conjunto de exemplos é dividido em r subconjuntos de tamanho aproximadamente igual. Os objetos de $r-1$ partições são utilizados no treinamento de um modelo, o qual é então testado na partição restante. Esse processo é repetido r vezes, utilizando em cada ciclo uma partição diferente para teste. O desempenho final do modelo é dado pela média dos desempenhos observados sobre cada subconjunto de teste. Este processo é ilustrado pela Figura 4.7(b), empregando-se $r = 3$.

Uma variação desse metódo aplicável especificamente para problemas de classificação é o r-fold cross-validation estratificado, que mantém, em cada partição, a proporção de exemplos de cada classe semelhante à proporção contida no conjunto de dados total. Se, por exemplo, o conjunto de dados original tem 20% dos objetos na classe c_1 e 80% na classe c_2, cada partição também procura manter essa proporção, apresentando 20% de seus exemplos na classe c_1 e 80% na classe c_2.

4.4.2.3. Séries Temporais

Uma série temporal é uma série de observações registradas sequencialmente no decorrer do tempo [Wooldridge 2015]. Séries temporais e séries de fluxo de dados têm um tratamento diferenciado, pois o comportamento dos dados pode mudar no decorrer do tempo, como mostrado na Figura 4.7. Nesse caso, como os dados ou observações são registrados ao longo do tempo, o dado em geral possui uma característica que é sua frequência temporal.

Em problemas de regressão, nos quais não há uma ordem nas observações, técnicas de amostragem podem ser utilizadas. No entanto, quando se trata de séries temporais, a ordem dos dados é fundamental. Isso acontece pois uma característica muito importante desse tipo de dados é que as observações vizinhas são dependentes, e o interesse pode estar em analisar e modelar essa dependência. Com isso, algumas considerações devem ser feitas a respeito da amostragem de séries temporais. Em [Mello and Ponti 2018] é apresentada uma discussão aprofundada de como detectar a dependência da série e, caso exista, transformações na série temporal podem ser realizadas. No entanto, essa questão foge do escopo deste capítulo. Assim, discutimos a amostragem das séries temporais considerando as técnicas de holdout e validação cruzada.

No caso da técnica de holdout, na qual o conjunto de dados é dividido em treinamento e teste, tais dados não podem ser divididos de maneira aleatória. Deve-se respeitar a ordem dos dados na divisão dos conjuntos de treinamento e teste, pois não se pode utilizar um dado do futuro para predizer dado do passado. No caso da validação cruzada, tal técnica não pode ser aplicada em problemas nos quais se utiliza as séries temporais.

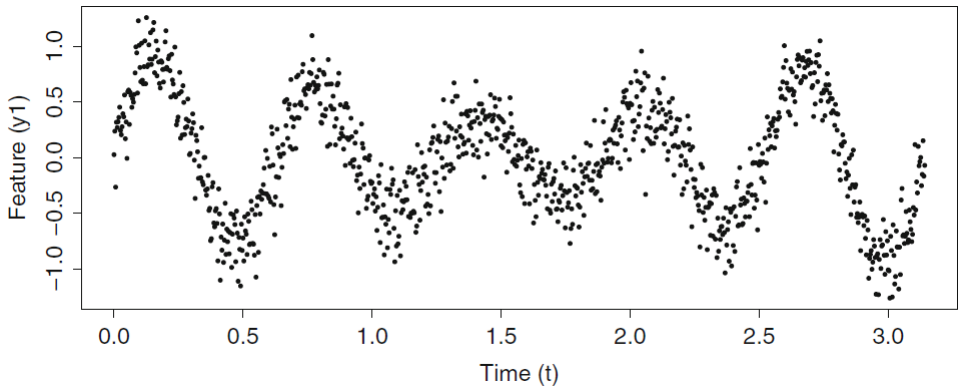


Figura 4.7. Exemplo de uma série temporal com dependência no tempo [Mello and Ponti 2018].

4.4.3. Teste de Hipóteses

Em Aprendizado de Máquina existe uma constatação geral, usualmente conhecida como *Free-Lunch Theorem* [Wolpert 1996], que declara que não existe um modelo universalmente melhor que todos os demais, uma vez que existem modelos que funcionam muito bem para alguns domínios e muito mal para outros. Entretanto, a pergunta que segue a partir desse teorema é como definir se um modelo é de fato melhor que o outro em um certo domínio. Esse mesmo questionamento surge ao se propor um novo método de aprendizado, seja ele genérico ou aplicado a um domínio específico. Para tanto, uma ferramenta comumente utilizada é a análise estatística de significância, realizada por meio de um teste de hipóteses que busca evidências para rejeitar uma suposição. Tal suposição é chamada de *hipótese nula* e usualmente é o oposto daquilo que se deseja comprovar. Por exemplo, se o que se quer provar é que um método A é igual a um método B, em termos de desempenho, então a hipótese nula é que A é igual a B. Para aplicar um teste estatístico, é necessário definir uma amostra de comparação. No caso de métodos de Aprendizado de Máquina, a amostra consiste nos resultados preditivos obtidos a partir do conjunto de teste, de acordo com alguma medida de desempenho. Caso o método de avaliação utilizado seja o $k-$*fold cross validation* e a medida de desempenho seja a acurácia, por exemplo, as amostras serão compostas pela acurácia obtida em cada uma das partições (*folds*) de teste.

Diversos testes estatísticos podem ser utilizados para prover mais evidências para a decisão de aceitar ou não uma hipótese, porém nem todos são pertinentes para todas as situações. Para decidir qual deles usar, primeiro deve-se observar se a comparação está sendo feita entre dois métodos em um mesmo domínio, entre dois métodos para múltiplos domínios, ou entre diversos métodos e diversos domínios [Japkowicz and Shah 2011b]. A seguir, descrevemos brevemente possibilidades de testes estatísticos para cada um dos casos elicitados.

Teste de hipóteses para comparar dois métodos em um mesmo domínio Para o primeiro caso, em que precisamos verificar se um método é melhor que o outro em um domínio, o mais comum é aplicar a estatística paramétrica de teste t. Por ser paramétrico, para utilizar o t-test alguns requerimentos devem ser respeitados: que as amostras seguem

a distribuição normal ou que seu tamanho é maior que 30; que a amostra é representativa da população; e que ambas as amostras (as amostras oriundas dos dois métodos) possuem variância igual. O t-test tem como objetivo verificar se a diferença nas médias e desvio padrão da métrica de desempenho de duas amostras correpondentes são iguais (p.ex., a acurácia média para cada conjunto de teste, considerando o mesmo conjunto para os dois métodos que se deseja comparar). Nesse caso, a hipótese nula é que a diferença entre as médias é zero. Para tanto, calcula-se a estatística t definida por 15

$$t = \frac{\overline{D} - \mu_D}{\sqrt{\frac{s_d^2}{n}}} \tag{15}$$

onde $\overline{D}$ é a diferença entre as médias dos valores de medida de desempenho observadas em n amostras (p.ex., as acurácias computadas para os n folds) e s_d^2 é a variância das diferenças amostrais. A hipótese nula declara que μ_D é zero e ela pode ser rejeitada com nível de significância 0.001 e $n-1$ graus de liberdade se o valor de t for maior que uma constante α observada na tabela *t-student*. Por exemplo, com 9 graus de liberdade (correspondente a 10 folds), a hipótese nula pode ser rejeitada se t for maior que $4,781$.

Um teste estatístico não paramétrico alternativo, ainda para o caso (1), é o teste de *McNemar*, que, por ser não-paramétrico, não assume nenhuma suposição sobre a natureza da distribuição das amostras. Diferente do teste t, aqui, observa-se como os dois métodos se comportam em termos de classificação discordante dos exemplos, ou seja, considerando (ci) o número de exemplos classificados incorretamente pelo primeiro método e incorretamente pelo segundo método e (cii) o número de exemplos classificados incorretamente pelo segundo método, mas classificados corretamente pelo primeiro método. Então, é aplicada a estatística χ^2 de McNemar computada como na equação 16.

$$\chi^2_{mcnemar} = \frac{(|ci - cii| - 1)^2}{ci + cii} \tag{16}$$

Caso $ci + c2 \geq 20$ e $\chi^2_{mcnemar}$ exceda a estatística $\chi^2_{1,1-\alpha}$, então podemos rejeitar a hipótese nula com confiança $1-\alpha$. Caso $ci + cii < 20$, então o teste de McNemar não pode ser utilizado. Observe, no entanto que, como o teste de McNemar é baseado em comparações de erros de classificação, ele somente é passível de ser utilizado com dados categóricos (problemas de classificação), mas não em amostras contínuas (problemas de regressão).

Teste de hipótese para comparar dois métodos em vários domínios No segundo caso, quando queremos observar a diferença entre dois métodos em múltiplos domínios, costuma-se utilizar o teste não paramétrico de *Wilcoxon*. Para aplicar tal teste, primeiro é necessário calcular a diferença absoluta entre os valores observados na métrica de desempenho para o par de métodos, considerando cada domínio. Em seguida, tais valores de diferença são ordenados, onde o menor número de ordem (1) contempla a menor diferença que ainda seja maior que 1 (valores idênticos são descartados). Em seguida, os números de ordem recebem o sinal da diferença, para que, finalmente os valores positivos e negativos sejam somados entre si. O teste de Wilcoxon compara a soma dos números positivos com a soma dos números negativos, seleciona o menor deles (M), e o compara

com uma constante V_α. Se V_α for maior ou igual que M, então a hipótese nula pode ser rejeitada com confiança α.

Como esse teste demanda uma certa quantidade de passos, consideraremos um exemplo, para seu melhor entendimento. Suponha, então, que tenhamos cinco domínios $D1$, $D2$, $D3$, $D4$, e $D5$, e estamos induzindo modelos a partir dos métodos $M1$ e $M2$. A Tabela 4.4 apresenta os resultados obtidos em uma medida arbitrária pelos dois métodos para cada domínio, as diferenças entre as medidas, e a ordenação das diferenças. A partir da tabela, temos a soma de 11 para os valores positivos e 4 para os valores negativos, logo o valor mínimo é 4. Considerando 3 graus de liberdade (número de domínios -1) e $\alpha = 0,05$, temos que $V_\alpha = 1$. Como $4 > 1$, não podemos rejeitar a hipótese que o desempenho de $M1$ é igual ao desempenho de $M2$.

Tabela 4.4. Exemplo para o teste de hipótese de Wilcoxon, considerando três domínios e dois métodos

	$M1$	$M2$	$\|M1-M2\|$	$M1-M2$	**Ordem**	$\pm$ **Ordem**
D1	0,75	0,67	0,12	0,12	1	+2
D2	0,68	0,50	0,18	0,18	3	+4
D3	0,55	0,70	0,15	-0,15	2	-3
D4	0,49	0,53	0,04	-0,04	2	-1
D5	0,80	0,50	0,30	0,30	5	+5

Teste de hipótese para comparar diversos métodos em diversos domínios Neste terceiro caso, queremos observar se existe diferença significativa entre os resultados obtidos por métodos distintos, considerando múltiplos domínios. Assim como no caso anterior, nos focamos aqui em um teste não paramétrico, chamado de teste de *Friedman*, que é aplicável ao caso em questão. A hipótese nula é de que todos os métodos alcançam o mesmo desempenho e rejeitar essa hipótese significa que ao menos um par de métodos podem ser ditos como estatisticamente distintos, em termos de desempenho. Para computar a estatística de Friedman, os valores da métrica de desempenho são observados e, para cada domínio, os métodos são ordenados de acordo com os resultados obtidos. Assim, para cada domínio, o método com o melhor desempenho naquele domínio recebe um valor de 1, o método com o segundo melhor desempenho recebe o valor 2, e assim por diante. Os valores de ordenação são em seguida somados para cada método e tal soma é utilizada para computar a estatística de Friedman, como na Eq. 17, onde m é o número de métodos sendo comparados, n é quantidade de domínios e $soma_j$ é a soma dos valores de ordenação para cada método j.

$$\chi_F^2 = \left[\frac{12}{n \times m \times (m+1)} \times \sum_{j+1}^{m} (soma_j)^2 \right] - 3 \times n \times (m+1) \tag{17}$$

O valor da estatística de Friedman é então comparado a um valor crítico de acordo com a tabela de valores superiores de Friedman, observando um nível de significância. Se o valor computado for maior que o valor crítico, a hipótese nula pode ser rejeitada.

É importante notar que, caso a hipótese nula seja rejeitada, é ainda necessário identificar qual o par de classificadores que apresenta diferença estatisticamente significante.

Vale ressaltar que alguns pesquisadores defendem um uso criterioso de testes estatísticos por achar que eles sobrevalorizam resultados quantitativos, em detrimento de análises qualitativas [Demšar 2008], ou ainda devido à falta de uma análise mais cuidadosa de medidas estatísticas adicionais, como o poder do teste e o tamanho do efeito [Neumann et al. 2018].

Teste de hipótese para comparar em um mesmo domínio diversos métodos Por último, temos a comparação na qual procuramos verificar se diversos métodos distintos (três ou mais) possuem origem em um mesmo domínio. Mais uma vez temos um teste não paramétrico, conhecido pelo nome de seus autores, o teste de $Kruskal-Wallis$, um teste que resumidamente seria uma extensão do $Wilcoxon-Mann-Whitney$. A hipótese nula ou H_0 é de que todos os métodos pertencem a uma mesma distribuição, sejam as amostras do mesmo tamanho ou não. Rejeitar H_0 significa que um método é estocasticamente dominante sobre outro. Porém, não verifica-se pelo teste onde ocorre esta dominância, independente do número de pares considerados. Também podemos utilizar a hipótese alternativa ou H_a, na qual é possível demonstrar que a mediana da população de um grupo de amostras é diferente da mediana da população de ao menos outro grupo de amostras. Na Eq. 18, devemos ignorar a que grupo cada dado pertence e classificar os dados de 1 até N e atribuir quaisquer valores que sejam repetidos a média dos *ranks* que eles teriam recebido, caso não fossem repetidos. O número total de observações incluindo todos os grupos é dado por N, n_i é o número de observações do grupo i, $\bar{r_i}$. é a classificação média de todas as observações no grupo i (Eq. 19), $\bar{r}$ é a média de todos os r_{ij} (Eq. 20) e finalizando, r_{ij} é a classificação de observação j do grupo i.

$$H = (N-1)\frac{\sum_{i=1}^{g} n_i(\bar{r}_{i\cdot} - \bar{r})^2}{\sum_{i=1}^{g}\sum_{j=1}^{n_i}(r_{ij} - \bar{r})^2} \tag{18}$$

$$\bar{r_{i\cdot}} = \frac{\sum_{j=1}^{n_i} r_{ij}}{n_i} \tag{19}$$

$$\bar{r} = \frac{1}{2}(N+1) \tag{20}$$

4.5. Análise Experimental em Conjuntos de Dados para Problemas de Predição e Previsão no Contexto de Cidades Inteligentes

O objetivo desta seção é apresentar três análises experimentais que envolvem a construção de modelos em lote e online em problemas de cidades. São utilizados conjuntos de dados reais, nos cenários de previsão de demanda e mobilidade urbana, para mostrar o funcionamento do framework de avaliação apresentado.

4.5.1. Previsão de Tarifas de Táxi

Com a tecnologia impulsionando cada vez mais o transporte alternativo, uma área em Nova York que está ficando para trás é a sua icônica Yellow Cab. Com o Uber, o Lyft, o Via e outros aplicativos que definem o ritmo do passeio, a Yellow Cab se uniu ao Google para se tornar mais centrada em dados, solicitando a previsão do valor estimado da tarifa usando alguns recursos para determinar o valor esperado da tarifa [1].

À primeira vista, pode parecer que depende simplesmente da distância percorrida. No entanto, os taxistas em Nova York cobram valores variados por outros fatores, como passageiros adicionais ou pagamento por cartão de crédito em vez de dinheiro.

Nesse sentido, prever o valor de uma corrida é importante no contexto da mobilidade urbana. Para mostrar o uso de aprendizado de máquina para este problema, utilizamos a um conjunto de dados da cidade de Nova Iorque [2]. Essa base foi tratada e pré-processada. A fim de reduzir o custo computacional, optou-se por utilizar apenas uma amostra desses dados (15485 registros, ou exemplos). Na Tabela 4.5 é possível ver um recorte do conjunto de dados utilizado.

Tabela 4.5. Recorte do dataset de previsão de tarifa de táxi

vendor id	rate code	passenger count	trip time (in secs)	trip distance	payment type	fare amount
CMT	1	2	1370	9.2	CSH	29.5
CMT	1	1	963	7.7	CSH	23
CMT	1	1	311	0.8	CSH	5.5

4.5.2. Previsão de Demanda de Energia Elétrica

Com uma crescente em direção aos padrões de consumo de energia, uma tarefa bastante desafiadora para os pesquisadores de Aprendizado de Máquina tem sido criar uma solução de modelagem para previsão de consumo de eletricidade para usuários individuais. Este tipo de problema é contextualizado no ambiente de cidades inteligentes, por isso será utilizado neste trabalho. Porém, o foco não será no desenvolvimento de novas técnicas, mas sim, saber avaliá-las do ponto de vista dos algoritmos de aprendizado utilizados para este tipo de problema.

O conjunto de dados escolhido nesse domínio [Kelly and Knottenbelt 2014], é de acesso público e contém informações sobre as potências de energia elétricas consumidas de cada canal (aparelhos eletrodomésticos) e os intervalos de tempo estão disponíveis em diversos arquivos que estão separados e são relativos à cada residência contida no conjunto. Nesse conjunto, há dados de 5 residências, com um total de 54 aparelhos eletrodomésticos conectados. Porém, também de forma a simplificar o custo computacional, optou-se por selecionar apenas uma faixa de tempo e apenas um canal (um eletrodoméstico), o que totalizou 14.172 exemplos. Na Tabela 4.6, é possível ver um recorte do conjunto de dado utilizado. Nessa tabela, o atributo Timestamp é relativo ao registro de

[1] https://nycdatascience.com/blog/student-works/predicting-nyc-yellow-cab-taxi-fare/

[2] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

data e hora do sistema operacional unix. Já o atributo Potência se refere à potência do eletrodoméstico em um determinado instante de tempo, dada em Watts.

Tabela 4.6. Recorte do dataset de consumo de energia

Timestamp	Potência (W)
1364796004	239
1364796010	239

4.5.3. Classificação de Comportamento de Tráfego Urbano na cidade de São Paulo

É de conhecimento comum que o tráfego de veículos na cidade de São Paulo é classificado como um dos piores do mundo, agravado pela grande dependência da malha rodoviária no Brasil. Assim, é de grande importância desenvolver soluções que consigam auxiliar o planejamento e roteamento de veículos. Com esse foco, foi utilizada uma base de dados da Universidade da Califórnia [Dheeru and Karra Taniskidou 2017].

Na base de dados supracitada, o atributo de classe foi definido como um valor real. Porém, para ilustrar um caso envolvendo um problema de classificação binária, que ocorre frequentemente em problemas decisórios reais, tal atributo foi discretizado em duas classes, usando o método de discretizaçao em grupos $k-bins$ [Dougherty et al. 1995]. Assim, o valor de 0 indica baixa desaceleração, enquanto o valor de 1 indica alta desaceleração.

Além do atributo de classe, outros atributos foram definidos, considerando intervalos de 30 minutos. Alguns deles são: a quantidade de ônibus imobilizados; a quantidade de caminhões quebrados, quantidade de acidentes com vítimas, quantidade de incêndios, falta de luz (o que pode atrapalhar os semáforos), manifestações, alagamentos, semáforos com problemas, entre outros, totalizando 18 atributos (incluindo o de classe). A Tabela 4.7 apresenta um recorte da base de dados utilizada, onde tanto linhas de exemplos como colunas de atributos foram omitidas, para melhor visualização. É interessante notar que pode ocorrer *slowndow* não pelo acontecimento de algum evento, mas apenas por ser horário de pico, como pode ser observado no segundo exemplo da tabela.

Tabela 4.7. Recorte em linhas e colunas da base de dados de Classificação de Comportamento do Tráfego Urbano na Cidade de São Paulo

Hour	Immob_bus	Broken_Truck	Fire	Flooding	Manifest.	...	Slowdown
7:00	0.0	0.0	0.0	0.0	0.0	...	0
			...				
9:30	0.0	0.0	0.0	0.0	0.0	...	1
			...				
10:30	0.0	0.0	0.0	0.0	0.0	...	0

4.5.4. Ferramentas Utilizadas

Escolhemos o WEKA (Waikato Environment for Knowledge Analysis) [Hall et al. 2009] como ferramenta para construção dos modelos para classificação (classificador) e regressão (regressor). Tal ferramenta foi escolhida devido ao fato de permitir aos pesquisadores fácil acesso às técnicas mais avançadas em aprendizado de máquina. Além disso, o

WEKA não só é uma caixa de ferramentas contendo a implementação de diversos algoritmos de aprendizado consolidados na literatura, mas também é um framework que permite que pesquisadores e profissionais possam implementar novos algoritmos sem ter que se preocupar com a infraestrutura de suporte para manipulação de dados e avaliação do esquema. Além disso, a ferramenta possibilita a extração das principais métricas de avaliação dos algoritmos, descritas na Seção 4.4.1.

Para o aprendizado online ou *stream learning*, escolhemos uma ferramenta também desenvolvida pelos pesquisadores da Universidade de Waikato, o MOA (Massive Online Analytics) [Bifet et al. 2011]. O MOA é um ambiente de software para implementar algoritmos e executar experimentos para aprendizado online a partir de fluxos de dados em evolução. O MOA foi projetado para lidar com os problemas desafiadores de ampliar a implementação de algoritmos de última geração para tamanhos de conjuntos de dados reais e tornar os algoritmos comparáveis nas configurações de fluxo de dados de referência. Ele contém uma coleção de algoritmos offline e online para classificação, agrupamento e mineração de gráficos, bem como ferramentas para avaliação. Para pesquisadores, a estrutura fornece insights sobre vantagens e desvantagens de diferentes abordagens e permite criação de conjuntos de dados de streaming de referência por meio de configurações armazenadas, compartilhadas e repetidas para os feeds de dados. Os profissionais podem usar a estrutura para comparar facilmente os algoritmos e aplicá-los aos conjuntos de dados e configurações do mundo real. O MOA suporta interação bidirecional com o WEKA, o Waikato Environment for Knowledge Analysis. Além de fornecer algoritmos e medidas para avaliação e comparação, o MOA é facilmente extensível com novas contribuições e permite a criação de cenários de referência.

4.5.5. Problemas Abordados e Resultados Obtidos

4.5.5.1. Aprendizado em Lote

Como visto anteriormente, utilizamos duas bases de dados para problemas de regressão — demanda de táxi (S_{DTaxi}) e consumo de energia ($S_{CEnergia}$) — e uma base de dados de um problema de classificação — classificação de comportamento de tráfego ($S_{CTrafego}$). O objetivo, nesta seção, é observar o comportamento de algumas métricas para cada conjunto de dados utilizando os algoritmos de aprendizado em lote. Utilizamos, no Weka, os algoritmos M5P e M5 Rules para os conjuntos de dados S_{DTaxi} e $S_{CEnergia}$ (regressão), e os algoritmos MLP e J48 para o conjunto de dados $S_{CTrafego}$. Como o objetivo aqui é ilustrar como se realiza a avaliação para cada domínio em específico, os resultados não são comparados entre os domínios.

Para a amostragem da base de dados, escolhemos a validação cruzada, conhecida como *k-fold*, com *k=10*, para o conjunto de dados de classificação e para o conjunto de dados de demanda de táxi. A técnica Holdout foi utilizada para o conjunto de dados de energia, respeitando a ordem de apresentação dos dados na divisão entre conjunto de treinamento e teste.

Resultados para o conjunto S_{DTaxi}: Carregamos o conjunto S_{DTaxi}, com 15485 instâncias, no WEKA, e executamos os algoritmos M5P e M5 Rules, uma implementação de árvores de regressão no Weka, utilizando os parâmetros *default* dos algoritmos.

O atributo utilizado para ser predito é *fare amount*. Os resultados obtidos são mostrados na Tabela 4.8. Pode-se observar, nesses resultados, que os erros relativos estão bem baixos. Nesse caso, o algoritmo M5 Rules apresentou os melhores resultados para metade das métricas. É interessante observar que os resultados das métricas normalizadas são mais fáceis de serem analisadas, pois o atributo *fare amount* não foi normalizado, e seu domínio é o intervalo numérico $[2.5, 175]$. No geral, é possível dizer que o modelo em questão obteve resultados satisfatórios, visto que suas métricas de avaliação estão baixas. No entanto, isso é esperado, já que para este problema, os dados estavam muito bem estruturados e seus atributos faziam sentido para a tentativa de predição de uma tarifa de táxi.

Tabela 4.8. Valores das métricas para o conjunto de dados S_{DTaxi}

Métrica	M5P	M5 Rules
MAPE	5,05%	4,80%
MAE	0,566	0,541
RMSE	2,76	2,81
RRSE	28,32%	28,80%

Resultados para o conjunto $S_{CEnergia}$: Carregamos o conjunto de dados $S_{CEnergia}$, com 14.173 instâncias, no WEKA e executado, usando a mesma técnica de amostragem e o mesmo algoritmo de regressão. O atributo a ser predito é o *potência*. Os resultados obtidos são mostrados na Tabela 4.9. Pode-se observar, nesses resultados, que os erros relativos também estão altos, porém menores que para o conjunto de dados anterior. Nesse caso, o algoritmo de árvore de regressão também apresentou os melhores resultados para todas as métricas. É interessante observar que os resultados das métricas normalizadas são mais fáceis de serem analisadas, pois o atributo *hour* não foi normalizado, e seu domínio é o intervalo numérico $[163, 2642]$. No geral, é possível dizer que o modelo em questão obteve resultados razoavelmente satisfatórios, visto que suas métricas de avaliação estão relativamente baixas. No entanto, outras possibilidades de modelagem do problema ainda podem ser consideradas. Além disso, técnicas de construção de atributos a partir dos atributos disponibilizados podem ser utilizadas, já que somente a hora da leitura foi utilizada como atributo de entrada.

Tabela 4.9. Valores das métricas para o conjunto de dados $S_{CEnergia}$

Métrica	M5P	M5 Rules
MAPE	3,71%	5,04%
MAE	12,315	14,9026
RMSE	41,3821	46,8329
RRSE	23,2402%	26,3014%

Resultados para o conjunto $S_{CTrafego}$: Para o problema de classificação de desaceleração (comportamento) de tráfego urbano, foi utilizado o procedimento de validação cruzada estratificada com 10 folds, e o software Weka. Existem 67 exemplos da classe 0 e 68 exemplos da classe 1, totalizando 135 exemplos. Tal divisão favorece a utilização

Tabela 4.10. Valores preditivos médios obtidos a partir da execução do algoritmo J48 para o problema de classificação de tráfego urbano

Métrica	Valor Médio Pesado
Acurácia	70,37 %
Precisão	0,707
Sensitividade	0,704
F-measure	0,703
AUC ROC	0,735

Tabela 4.11. Matriz de confusão obtida a partir do algoritmo J48 para o problema de classificaçao de comportamento de tráfego urbano

0	**1**	**<– classificado como**
51	16	0
24	44	1

de técnicas clássicas, sem termos que abordar métodos para lidar com problemas de desbalanceamento de classes. Aqui também são utilizadas variações dos mesmos métodos apresentados anteriormente: J48 (algoritmo para indução de árvores de decisão) e MLP (algoritmo para indução de rede neural com múltiplas camadas, do inglês, *multi-layer perceptron*), com os parâmetros padrão disponibilizados pelo software Weka. Apesar de serem métodos tradicionais e comumente utilizados, sua escolha favorece a abordagem de problemas tanto de classificação quanto de regressão.

A Tabela 4.10 apresenta os resultados preditivos médios obtidos a partir da execução do algoritmo J48, cuja árvore final possui 34 folhas. Tais valores estão acima de 70%, constituindo valores de acerto aceitáveis.

A Tabela 4.11 exibe os valores da matriz de confusão, onde é possível perceber que a diagonal principal (exemplos classificados corretamente) possui os maiores valores, porém ainda existem um número relativamente alto de exemplos classificados como pertencentes à classe 0 embora sejam da classe 1 e vice-versa.

Ademais, o uso da árvore de decisão favorece a interpretabilidade do modelo aprendido. Podemos, por exemplo, obter a seguinte regra a partir de um dos ramos da árvore: `se Hour = 9:30 e Immobilized_bus <= 1, então classe = 0; senão se Immobilized_bus > 1, então classe = 1`. Esse tipo de conhecimento extraído do modelo favorece o entendimento do especialista da aplicação sobre possíveis erros do modelo, para que ele seja eventualmente melhorado, e ainda para que os órgãos reguladores saibam com exatidão como basear seu processo decisório.

Para esse problema, o algoritmo MLP obteve resultado piores que a árvore de decisão, como pode ser observado nos resultados exibidos na Tabela 4.12 e em sua matriz de confusão, na Tabela 4.13. Apenas o valor de área sob a curva ROC teve um resultado ligeiramente melhor do que o obtido pelo J48. Embora o MLP obtenha resultados muito bons em diversos problemas, aqui vemos uma instância do *Free-Lunch Theorem*: Não existe um método que seja melhor que os demais em todos os tipos de probemas.

Tabela 4.12. Valores preditivos médios obtidos a partir da execução do algoritmo MLP para o problema de classificação de tráfego urbano

Métrica	Valor Médio Pesado
Acurácia	66,67 %
Precisão	0,667
Sensitividade	0,667
F-measure	0,667
AUC ROC	0,753

Tabela 4.13. Matriz de confusão obtida a partir do algoritmo MLP para o problema de classificaçao de comportamento de tráfego urbano

0	1	<– classificado como
44	23	0
22	46	1

Teste de Hipóteses para o problema de classificação: Para verificar se os resultados preditivos obtidos eram estatisticamente diferentes, foram executadas as análises estatísticas t-test e McNemar, a partir do software online *GraphPad*[2]. No caso do t-test, foram executadas as análises para as medidas de acurácia e F1, computadas a partir do conjunto de teste. Para o caso da análise de McNemar, foram observados as predições concordantes e discordantes nos conjuntos de testes. Considerando um p-value de 0,05, nenhuma das duas análises apontou diferença significativa para os resultados observados. Nesse caso, o especialista da aplicação poderia se sentir confortável em utilizar quaisquer um dos modelos, ou aprofundar a escolha de forma qualitativa, observando, por exemplo, qual a classe mais relevante e qual modelo consegue obter melhores resultados em tal classe.

Como foi visto ao decorrer dos problemas abordados, utilizar métricas para analisar o seu conjunto de dados, gera uma visão mais completa do desempenho do mesmo. O importante é entender o quê cada medição significa e quais são suas restrições, para então decidir que ação tomar.

4.5.5.2. Testes Online

Muitos tipos de dados são adquiridos sequencialmente ao longo do tempo. Em vez de esperar que os dados sejam coletados, as análises de streaming permitem identificar padrões e tomar decisões com base neles à medida que os dados começam a chegar. Quando os dados não são estacionários e os padrões mudam com o tempo, as análises de streaming se adaptam. E em escalas em que o armazenamento de dados brutos se torna impraticável, as análises de streaming permitem persistir apenas representações menores e mais direcionadas. Com base nisso, este trabalho apresentará o comportamento dos dois datasets, previamente avaliados com a ferramenta WEKA, em uma outra ferramenta, o MOA. Diferente do WEKA, o MOA avalia os dados a partir de uma simulação de *streaming* de dados, utilizando o conjunto de dados completo.

[2] `https://www.graphpad.com/`

No MOA é necessários inserir alguns parâmetros antes da execução propriamente dita do modelo. Utilizamos somente o algoritmo FIMT-DD por questão de tempo de execução. Para o aprendizado utilizamos o algoritmo da seguinte maneira no MOA: `trees.FIMTDD -s VarianceReductionSplitCriterion`, que é denominado *learner* na ferramenta. Para o *streaming* foi utilizado o padrão `ArffFileStream`, e por fim, para a avaliação foi utilizado o `BasicRegressionPerformanceEvaluator`. A frequência de amostragem utilizada foi de 10 (a performance do modelo é calculada a cada 10 exemplos lidos).

Resultados para o conjunto S_{DTaxi}: O conjunto de dados de demanda de táxi S_{DTaxi}, com 15485 instâncias foi carregado no MOA, e o algoritmo FIMT-DD foi executado. O atributo a ser predito é o *fare amount*. Os resultados obtidos são mostrados na Tabela 4.14. O MAE médio foi de 1,28, ou seja, o algoritmo errou 1,28 (para mais ou para menos) em média o valor do atributo *fare amount*. O RMSE médio foi de 2,62, ou seja, os valores que estão fora da curva de melhor ajuste estão a uma distância média de 2,62 dos valores contidos na curva de melhor ajuste. Percebe-se então que no caso deste conjunto de dados, o modelo construído a partir do WEKA é melhor que o feito pelo MOA, uma vez que as métricas de comparação foram melhores (menores).

Tabela 4.14. Valores das métricas para o conjunto de dados S_{DTaxi}

Métrica	FIMT-DD
MAE	0,72
RMSE	1,66

Resultados para o conjunto $S_{CEnergia}$: Após a análise do dataset de demanda de táxi, o dataset de consumo de energia, com 14173 instâncias foi carregado no MOA e executado, usando a mesma frequência de amostragem e o mesmos parâmetros. O atributo a ser predito é o *potência*. Os resultados obtidos são mostrados na Tabela 4.15. Com isso, pode-se fazer uma avaliação quanto às métricas. O MAE médio foi de 16,99, ou seja, o algoritmo errou 16,99 (para mais ou para menos) em média o valor do atributo *potência*. O RMSE médio foi de 58,14, ou seja, os valores que estão fora da curva de melhor ajuste estão a uma distância média de 58,14 dos valores contidos na curva de melhor ajuste. Observa-se que no caso deste conjunto de dados, o modelo construído a partir do WEKA é muito melhor que o feito pelo MOA, uma vez que as métricas de comparação foram melhores. Um corte do gráfico do MAE ao longo das amostras pode ser visto na Figura 4.8 abaixo.

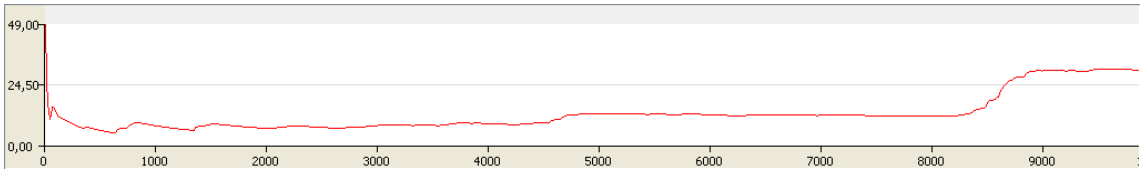


Figura 4.8. Gráfico do MAE ao longo das amostras para o conjunto de dados $S_{CEnergia}$.

Uma análise interessante é que um pouco depois da amostra de número 8000, o erro absoluto começa a aumentar. Isso se dá pelo fato do conjunto de dados escolhido ter

uma variação muito grande no atributo *potência* nessa faixa, o que faz com que o algoritmo prediga de forma errada as amostras subsequentes. Isso não acontece no WEKA, pois quando separamos o conjunto de dados em 2/3 para treinamento (por volta de 9448 amostras), o algoritmo aprende essa variação no atributo *potência* durante a fase de treinamento, e portanto, consegue predizer de forma correta, o restante das amostras.

Tabela 4.15. Valores das métricas para o conjunto de dados $S_{CEnergia}$

Métrica	FIMT-DD
MAE	16,99
RMSE	58,14

Resultados para o conjunto $S_{CTrafego}$: Para analisar o problema de classificação de comportamento de tráfego urbano de forma online, podemos considerar que os valores seriam inseridos a cada meia hora, para prever a classe da próxima meia hora. Para manter o padrão na utilização dos métodos, também optamos por fazer uso de um algoritmo online adaptativo baseado em árvore e outro baseado em um algoritmo que faça uso do método de gradiente para otimização dos parâmetros, assim como é feito com redes neurais. Para o treinamento da árvore de decisão, utilizou-se como aprendiz o algoritmo `trees.HoeffdingAdaptiveTreeClassifLeaves`. Esse método utiliza o algoritmo de janelas de tamanho adaptativo ADWIN para substituir as folhas por novas folhas, quando necessário devido à natureza do stream. O tamanho das janelas é calculado estatisticamente, para que não haja mudança no valor médio da variável de classe dentro da janela [Bifet and Gavalda 2007]. Para a avaliação (*evaluator*) foi utilizado o método `WindowClassificationPerformanceEvaluator`, avaliando os dados de forma *prequential*. A frequência de amostragem utilizada foi de 1. Os valores de medidas de avaliação obtidos podem ser observados na Tabela 4.16.

Tabela 4.16. Valores das medidas de avaliação para o aprendizado online de árvore de decisão, para o problema de classificação de comportamento de tráfego

Métrica	Valor
Acurácia	63,70 %
Kappa	27,74 %
Kappa temporal	-226, 67
Kappa M	26,87 %

Para o treinamento online de um método que use o gradiente como otimizador, optou-se pelo aprendiz *functions.SGD*. Os demais parâmetros permaneceram com seus valores default, como antes. Os valores de medidas de avaliação obtidos podem ser observados na Tabela 4.17.

Em ambos os casos, a medida reportada comparável com o treinamento offline, a saber, a acurácia, obteve aqui valores piores do que ambos os algoritmos offline. Embora o método que aprende função tenha obtido um resultado ligeiramente melhor do que o método que aprende árvore, ele ainda apresenta um valor pior do que o pior método offline. Uma das razões é o baixo número de exemplos para esse problema, que afeta mais a abordagem online, pois os exemplos são considerados "aos poucos".

Tabela 4.17. Valores das medidas de avaliação para o aprendizado online de uma função, com otimização baseada em gradiente, para o problema de classificação de comportamento de tráfego

Métrica	Valor
Acurácia	65,19 %
Kappa	30,40 %
Kappa temporal	-213, 33
Kappa M	29,85 %

Dessa forma, deve-se avaliar cuidadosamente quando é necessário utilizar um algoritmo online e quando uma abordagem offline é suficiente. No caso de problemas em que todos os exemplos estejam disponíveis de antemão, ou que a chegada de novos exemplos não acarrete uma mudança na distribuição original da amostra, uma abordagem offline pode ser suficiente. Em outros casos, em que os exemplos de fato sejam disponibilizados como streams de dados, ou que aconteça mudança na distribuição de acordo com os novos exemplos, torna-se necessário recorrer a uma abordagem online. Para o problema aqui abordado, todos os exemplos estavam disponíveis de antemão e não foi observado mudança de distribuição com o tempo. Sendo assim, a abordagem offline já seria suficiente e nos valemos da abordagem online meramente por questões ilustrativas.

4.6. Considerações Finais

Neste capítulo, foram apresentados conceitos de aprendizado de máquina em lote e online, potencialmente utilizáveis em diversos problemas no contexto de Cidades Inteligentes. Foram exploradas duas ferramentas para as tarefas de aprendizado — a ferramenta Weka, para aprendizado em lote, e o MOA, para aprendizado online. Observa-se que, para os problemas de regressão, o algoritmo online apresentou melhores resultados, e para o problema de classificação, os algoritmos em lote apresentaram melhores resultados. Como esses estudos foram para ilustrar a aplicação do framework de avaliação apresentado, esses resultados mostram, na prática, que não há um método ou algoritmo de aprendizado que seja melhor em todos os casos. Além disso, mostramos também que em diferentes ferramentas, diferentes métricas estão disponíveis para serem analisadas. Entretanto, os resultados podem ser melhor avaliados por meio da análise da saída dos modelos obtidos. Entendemos que este capítulo é importante pois, a avaliação de modelos construídos utilizando aprendizado de máquina ainda se apresenta como um desafio, fonte de muitas dúvidas, em diversas aplicações. Assim, ao apresentar um framework que pode ser utilizado em diferentes cenários, pode trazer uma melhor visão na comunidade de Sistemas de Informações do Brasil em como realizar essas avaliações.

Referências

[Andrade et al. 2017] Andrade, E. O., Viterbo, J., and Nader, C. V. (2017). Um levantamento do uso de aprendizado profundo em análise de sentimentos. In *14th National Meeting on Artificial and Computational Intelligence (ENIAC)*, pages 85–96. Brazilian Conference on Intelligent Systems.

[Anlauf and Biehl 1989] Anlauf, J. and Biehl, M. (1989). The adatron: an adaptive per-

ceptron algorithm. *EPL (Europhysics Letters)*, 10(7):687.

[Anthopoulos 2015] Anthopoulos, L. (2015). Understanding the smart city domain: A literature review. In Rodríguez-Bolívar, editor, *Transforming City Governments for Successful Smart Cities, Public Administration and Information Technology*, pages 9–21.

[Barcellos et al. 2017] Barcellos, R., Viterbo, J., Miranda, L., Bernardini, F., Maciel, C., and Trevisan, D. (2017). Transparency in practice: using visualization to enhance the interpretability of open data. In *Proc. 18th Annual International Conference on Digital Government Research, DG.O 2017*, pages 139–148.

[Ben-David et al. 1997] Ben-David, S., Kushilevitz, E., and Mansour, Y. (1997). Online learning versus offline learning. *Machine Learning*, 29(1):45–63.

[Bernardini 2006] Bernardini, F. (2006). Combinacão de classificadores simbólicos utilizando medidas de regras de conhecimento e algoritmos genéticos. Tese de Doutorado — ICMC/USP.

[Bifet and Gavalda 2007] Bifet, A. and Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM.

[Bifet et al. 2018] Bifet, A., Gavaldà, R., Holmes, G., and Pfahringer, B. (2018). *Machine Learning for Data Streams With Practical Examples in MOA*. MIT Press.

[Bifet et al. 2011] Bifet, A., Holmes, G., Pfahringer, B., Read, J., Kranen, P., Kremer, H., Jansen, T., and Seidl, T. (2011). MOA: a real-time analytics open source framework. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 617–620. Springer.

[Choi and Choi 1992] Choi, J. Y. and Choi, C.-H. (1992). Sensitivity analysis of multilayer perceptron with differentiable activation functions. *IEEE Transactions on Neural Networks*, 3(1):101–107.

[Demšar 2008] Demšar, J. (2008). On the appropriateness of statistical tests in machine learning. In *Workshop on Evaluation Methods for Machine Learning in conjunction with ICML*, page 65.

[Dheeru and Karra Taniskidou 2017] Dheeru, D. and Karra Taniskidou, E. (2017). UCI machine learning repository.

[Domingos and Hulten 2000] Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM.

[Dougherty et al. 1995] Dougherty, J., Kohavi, R., and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *Machine Learning Proceedings 1995*, pages 194–202. Elsevier.

[Ernst et al. 2005] Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556.

[Faceli et al. 2011] Faceli, K., Lorena, A. C., Gama, J., Carvalho, A. C. P. d. L., et al. (2011). Inteligência artificial: Uma abordagem de aprendizado de máquina.

[Giffinger 2016] Giffinger, R. (2016). Smart cities — ranking of european medium-sized cities. Final report, Centre of Regional Science, Vienna University of Technology, October 2007. Available at `http://www.smart-cities.eu/download/smart_cities_final_report.pdf`.

[Gil-Garcia et al. 2015] Gil-Garcia, J. R., Pardo, T., and Nam, T. (2015). What makes a city smart? identifying core components and proposing an integrative and comprehensive conceptualization. *Information Polity*, 20(1):61–87.

[Guedes et al. 2018] Guedes, A., Alvarenga, J., Goulart, M., Rodriguez, M., and Soares, C. (2018). Smart cities: The main drivers for increasing the intelligence of cities. *Sustainability*, 10:1–19.

[Hall et al. 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

[Harrison and Donnely 2011] Harrison, C. and Donnely, I. (2011). A theory of smart cities. In *Proc. 55th Annual Meeting of the International Society for the Systems Sciences*.

[Haykin 2008] Haykin, S. (2008). *Neural Networks and Learning Machines*. Prentice Hall, 3rd edition.

[Ikonomovska et al. 2011] Ikonomovska, E., Gama, J., and Džeroski, S. (2011). Learning model trees from evolving data streams. *Data mining and knowledge discovery*, 23(1):128–168.

[Japkowicz and Shah 2011a] Japkowicz, N. and Shah, M. (2011a). *Evaluating learning algorithms: a classification perspective*. Cambridge University Press.

[Japkowicz and Shah 2011b] Japkowicz, N. and Shah, M., editors (2011b). *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press.

[Kelly and Knottenbelt 2014] Kelly, J. and Knottenbelt, W. (2014). Uk-dale: A dataset recording uk domestic appliance-level electricity demand and whole-house demand. *ArXiv e-prints*, 59.

[Lima et al. 2018] Lima, P. C. R., Barcellos, R., Bernardini, F., and Viterbo, J. (2018). Using geocoding and topic extraction to make sense of comments on social network pages of local government agencies. In *Electronic Government — 17th IFIP WG 8.5 International Conference, EGOV 2018*, pages 263–274.

[Mello and Ponti 2018] Mello, R. F. and Ponti, M. A. (2018). *Machine Learning: A Practical Approach on the Statistical Learning Theory*. Springer.

[Neumann et al. 2018] Neumann, N. M., Plastino, A., Pinto Junior, J. A., and Freitas, A. A. (2018). Is p-value<0.05 enough? two case studies in classifiers evaluation. *Anais do Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*, pages 94–103.

[Oliveira and Campolargo 2015] Oliveira, A. and Campolargo, M. (2015). From smart cities to human cities. In *48th Hawaii International Conference on System Sciences (HICSS)*, pages 2336–2344. IEEE.

[Osojnik et al. 2016] Osojnik, A., Panov, P., and Džeroski, S. (2016). Modeling dynamical systems with data stream mining. *Computer Science and Information Systems*, (00):9–9.

[Pinto et al. 2018] Pinto, H. D. S., Bernardini, F., and Viterbo, J. (2018). How cities categorize datasets in their open data portals: an exploratory analysis. In *Proc. 19th Annual International Conference on Digital Government Research, DG.O 2018*.

[Quinlan 1986] Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

[Quinlan 1993] Quinlan, J. R. (1993). *C4. 5: programs for machine learning*. Elsevier.

[Riedmiller 1994] Riedmiller, M. (1994). Advanced supervised learning in multi-layer perceptrons-from backpropagation to adaptive learning algorithms. *Computer standards and interfaces*, 16(3):265–278.

[Rosenblatt 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[Srinivasan et al. 2002] Srinivasan, N., Ravichandran, V., Chan, K., Vidhya, J., Ramakirishnan, S., and Krishnan, S. (2002). Exponentiated backpropagation algorithm for multilayer feedforward neural networks. In *Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on*, volume 1, pages 327–331. IEEE.

[Wibisono et al. 2016] Wibisono, A., Jatmiko, W., Wisesa, H. A., Hardjono, B., and Mursanto, P. (2016). Traffic big data prediction and visualization using fast incremental model trees-drift detection (fimt-dd). *Knowledge-Based Systems*, 93:33–46.

[Wolpert 1996] Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390.

[Wooldridge 2015] Wooldridge, J. M. (2015). *Introductory econometrics: A modern approach*. Nelson Education.

Biografia dos Autores

Igor Garcia Ballhausen Sampaio - http://lattes.cnpq.br/0452937415654599

Aluno de mestrado em Computação pela Universidade Federal Fluminense. Graduado em Engenharia de Telecomunicações pela mesma universidade, membro associado e representante estudantil da Sociedade Brasileira de Computação. Foi representante discente no Colegiado do Curso de Graduação em Engenharia de Telecomunicações da Universidade Federal Fluminense. Bolsista (CNPq) de Iniciação Científica em Desenvolvimento Tecnológico e Inovação no período de graduação, desenvolvendo sistemas para o monitoramento e análise de consumo energético sob a orientação do Prof. Dr. José Viterbo Filho. Atualmente trabalha com técnicas de aprendizado de máquina para problemas de localização indoor e identificação e classificação de objetos. Conhecimento em operações de rede de acesso móvel de telecomunicações. Possui 7 anos de experiência no ensino de matemática e física para alunos do ensino médio e fundamental.

Flavia Cristina Bernardini - http://lattes.cnpq.br/5935862634033333

Flavia Bernardini possui graduação em Ciência da Computação UNESP (1999) e mestrado e doutorado em Ciência da Computação pelo ICMC/USP (2002 e 2006). É professora associada do Instituto de Computação da Universidade Federal Fluminense (UFF). É uma das responsáveis pelo Núcleo de Análise de Dados para a Cidadania (D4Ctz), e atua como colaboradora no ADDLabs e no LabESI. Tem experiência na coordenação e execução de projetos de pesquisa e desenvolvimento desde 2007, envolvendo principalmente o desenvolvimento de sistemas que utilizam Inteligência Artificial, com maior ênfase em Aprendizado de Máquina, para diversos problemas em diferentes domínios de aplicação. Nos últimos 5 anos, tem voltado seus interesses para a temática de Cidades Inteligentes. Nesse contexto, possui diversos trabalhos publicados, com diversas orientações de graduação, mestrado e doutorado concluídas e em andamento. Participa do grupo de Indicadores para Cidades Inteligentes da Rede Brasileira de Cidades Inteligentes e Humanas. Também, coordena um projeto de ensino voltado para a educação básica, envolvendo programação, pensamento computacional e robótica educativa.

Aline Marins Paes Carvalho - http://lattes.cnpq.br/0506389215528790

Aline Paes é professora adjunta do Instituto de Computação da Universidade Federal Fluminense (UFF). É mestre e doutora em Engenharia de Sistemas e Computação, com ênfase em Inteligência Artificial, pela COPPE-Sistemas, UFRJ, tendo feito estágio de doutoramento (sanduíche) por um ano no Imperial College London, UK, sob a supervisão do Professor Stephen Muggleton. Foi bolsista do CNPq de pós-doutorado júnior na COPPE-Sistemas, UFRJ, sob a supervisão do professor Valmir Carneiro Barbosa. Aline Paes atua na área de Ciência da Computação, com ênfase em Inteligência Artificial, com interesses e contribuições nos seguintes temas: aprendizado de máquina relacional, integrado a técnicas neurais, estatísticas e lógicas, atualização e adaptação de modelos por aprendizado online, revisão de teorias e aprendizado por transferência, IA explicável, indução de programas, processamento de linguagem natural, jogos e IA social. Seu nome era Aline Marins Paes até 2010.

Eduardo de Oliveira Andrade - http://lattes.cnpq.br/3122564652271949

Aluno de mestrado em Computação pela Universidade Federal Fluminense. Graduado em Ciência da Computação pela Universidade Federal do Rio de Janeiro (2015). Atualmente realiza pesquisas na área de Segurança da Informação, Aprendizado de Máquina e Aprendizado Profundo. Foi membro do Grupo de Resposta a Incidentes de Segurança (GRIS), onde atuou na detecção, resolução e prevenção de incidentes de segurança na UFRJ. Além disso, ofereceu suporte acadêmico aos estudantes de computação e demais alunos interessados nos assuntos relacionados à segurança na UFRJ. Ministrou workshops e cursos aos novos membros do próprio GRIS, apoiado pelo prof. Dr. Gabriel P. Silva. Em 2011 foi bolsista do Programa de Atividades Extracurriculares de Apoio aos Laboratórios de Informática de Graduação (PAEALIG), apoiado pela prof. Dr. Claudine P. Dereczynski. Executou o gerenciamento da rede e máquinas dos laboratórios e deu suporte aos alunos e professores de Geografia, Geologia e Meteorologia com a instalação de softwares de suas respectivas áreas e configuração de controle de acesso dos diversos usuários.

José Viterbo Filho - http://lattes.cnpq.br/8721187139726277

José Viterbo é graduado em Engenharia Elétrica (com ênfase em Computação) pela Escola Politécnica da Universidade de São Paulo, possui Mestrado em Computação, pela Universidade Federal Fluminense, e doutorado em Informática, pela Pontifícia Universidade Católica do Rio de Janeiro. Atualmente é Professor Adjunto no Instituto de Computação da Universidade Federal Fluminense (IC/UFF). É coordenador do Laboratório de Sistemas de Tempo Real e Embarcados (LabTempo) e pesquisador colaborador no Laboratório de Documentação Ativa e Design Inteligente (ADDLabs) e no Laboratório de Gestão em Tecnologia da Informação e Comunicação (GTecCom), na mesma universidade. Além disso, é Diretor de Publicações da Sociedade Brasileira de Computação (SBC). Atua no Programa de Pós-Graduação em Computação da UFF (PPGC/UFF), onde desenvolve pesquisas na área de computação ubíqua, inteligência coletiva, análise e gestão de dados abertos.