Avaliação de Modelos de Machine Learning
[1]

Prof. Dr. Giancarlo D. Salton

[1] Universidade Federal da Fronteira Sul
Campus Chapecó
gian@uffs.edu.br

outubro de 2023

A avaliação de modelos é uma etapa crítica no ciclo de vida do desenvolvimento de sistemas de *machine learning*. A avaliação de modelos fornece informações essenciais para a tomada de decisões informadas. Ela ajuda a determinar quão bem um modelo está performando em relação aos requisitos e objetivos específicos do problema. Permite comparar e selecionar entre diferentes modelos. Existem várias arquiteturas, algoritmos e hiperparâmetros disponíveis, e a avaliação ajuda a escolher aqueles que melhor se adequam ao problema em questão.

A avaliação também ajuda a identificar problemas como *overfitting* (ajuste excessivo), *underfitting* (ajuste insuficiente), vazamento de dados, entre outros. Compreender esses problemas é crucial para melhorar a qualidade do modelo. Além disso, fornece insights sobre como o modelo está tomando decisões. Em modelos de *machine learning* mais complexos, como redes neurais profundas, a avaliação ajuda a entender quais características são importantes para as previsões.

Outro aspecto bastante importante da avaliação é que ela evita a alocação desnecessária de recursos (tempo, computação, dinheiro) em modelos que não atendem aos requisitos ou que não trazem melhorias significativas em relação a modelos mais simples. Além disso a comunicação com partes interessadas (stakeholders) ao fornecer métricas compreensíveis e interpretações do desempenho do modelo ficam facilitadas, auxiliando na tarefa de se obter aceitação e suporte para a implementação do modelo.

Neste handout iremos estudar alguns métodos e algumas métricas de avaliação de modelos de *machine learning* com foco em redes neurais artificiais (RNAs).

Avaliação de Modelos

São 3 os propósitos da avaliação: determinar qual o melhor modelo; obter uma estimativa de como o modelo vai se comportar em produção; convencer os usuários de que ele funciona. A avaliação de modelos de *machine learning* utilizando métricas é fundamental para entender o desempenho do modelo, comparar diferentes modelos[2] e tomar decisões informadas sobre sua aplicabilidade em cenários específicos. O processo de treinamento e avaliação com métricas está

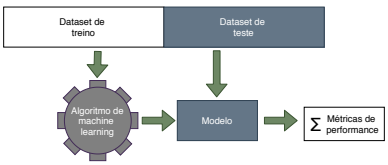

Figura 1: Imagem ilustrativa do processo de treinar e avaliar um modelo de *machine learning* utilizando um *dataset* de treino e um *dataset* de teste.

[2] Geralmente são treinados vários modelos com pequenas variações nos seus hiperparâmetros para que possamos encontrar o modelo que tenha a melhor performance sem entrar em *overfitting*.

resumido na Figura 1.

Métricas fornecem uma medida quantitativa do desempenho do modelo. Isso permite que você avalie o quão bem o modelo está realizando a tarefa para a qual foi treinado, seja classificação, regressão ou outra. Ao avaliar diferentes modelos, métricas oferecem uma base objetiva para comparação. Isso é crucial ao escolher entre modelos alternativos para determinar qual é mais adequado para um determinado problema.

Da mesma forma, métricas são utilizadas para ajustar os hiperparâmetros do modelo durante o treinamento para melhorar o desempenho. Por exemplo, você pode ajustar a taxa de aprendizado, a complexidade do modelo ou outros parâmetros com base nas métricas de desempenho. Além disso, ajudam a identificar se o modelo está *overfitting* (ajustando-se excessivamente aos dados de treinamento) ou *underfitting* (não se ajustando adequadamente). Isso é crucial para entender a capacidade de generalização do modelo.

A parte mais importante ao desenvolvermos um experimento para avaliar um modelo de *machine learning* é nos assegurarmos que que os exemplos utilizados para o teste do modelo não sejam os mesmos utilizados durante o treino. Assim, antes de iniciar um experimento de *machine learning*, precisamos separar uma parte do nosso *dataset* para compor o *dataset* de teste. Este conjunto de dados é mantido de fora do treino e é utilizado apenas para calcular as métricas de avaliação. Na Figura 2 e Figura 3 estão demonstradas duas divisões entre treino/validação/teste bastante utilizadas em *machine learning*.

Figura 2: Exemplo de divisão de *dataset* bastante comum: 50% para o treinamento; 20% para validação antes e durante o treino; 30% para o teste do modelo, onde são calculadas as métricas de desempenho.

Figura 3: Exemplo de divisão de *dataset* bastante utilizada quando há uma certa criticidade embutida na utilização do modelo: 40% para o treinamento; 20% para validação antes e durante o treino; 40% para o teste do modelo. Esta divisão deixa de fora um percentual maior de exemplos para tornar o treino e a avaliação propositalmente mais difícil.

Tipos de Erro

Quando trabalhamos com uma tarefa de classificação, a predição feita por um modelo de *machine learning* está correta ou está errada, não há meio termo. Se um modelo deveria predizer spam para uma determinada instância e prediz ham[3] a predição é considerada incorreta e, portanto, o modelo cometeu um erro[4].

Portanto, para tarefas de classificação de *targets* binários (**V**erdadeiro ou **F**also, 0 ou 1, etc), são 4 os possíveis resultados de uma previsão:

1. *True Positive* (TP) — verdadeiro positivo: ocorre nos casos em que o modelo prediz corretamente uma instância como positiva.

2. *True Negative* (TN) — verdadeiro negativo: ocorre nos casos em que o modelo prediz corretamente uma instância como negativa.

3. *False Positive* (FP) — falso positivo (Erro do Tipo I): são os casos em que o modelo prediz erroneamente uma instância como positiva

[3] Ham =Não-spam, na gíria do *machine learning*

[4] Contraste com uma tarefa de regressão onde um modelo deve predizer um número dentro de um intervalo contínuo. Caso ele deva predizer um valor de 0.1 mas prediz 0.099, ainda assim podemos aceitar a resposta como correta dentro de uma determinada margem de erro. Tarefas de regressão são, em geral, mais difíceis para modelos de *machine learning* do que tarefas de classificação.

quando ela é, na verdade, negativa. Ou seja, o modelo fez uma predição falsamente como positiva.

4. *False Negative* (FN) — falso negativo (Erro do Tipo II): são os casos em que o modelo prediz erroneamente uma instância como negativa quando ela é, na verdade, positiva. Ou seja, o modelo fez uma predição falsamente como negativa.

Note que não há uma regra para determinar qual é a classe positiva e qual a classe negativa. Geralmente uma das classe será, naturalmente, a classe positiva. Considere a tarefa de classificar emails em SPAM e HAM: como a classe que gera o alerta é a classe SPAM, podemos escolhe-la como a classe positiva sem precisar refletir muito sobre o assunto.

A partir dos resultados obtidos a partir dos *targets* e das predições do modelo, podemos montar uma matriz com estes resultados para facilitar a visualização do modelo. Esta matriz, chamada de "matriz de confusão" (ou *confusion matrix*), serve como base para a maioria das métricas que veremos à seguir.

		Predição	
		positive	**negative**
Target	**positive**	*TP*	*FN*
	negative	*FP*	*TN*

Tabela 1: Estrutura de uma matriz de confusão.

Nesta estrutura da matriz de confusão[5] temos os valores corretos dos *targets* representados nas linhas conforme índice à esquerda e os valores preditos pelo modelo nas colunas conforme índices no topo.

Considere o exemplo abaixo contendo *targets* de um *dataset* e as predições (PRED.) feitas por um modelo qualquer treinado para a tarefa daquele *dataset*. Na coluna RES. encontra-se o tipo de resultado da predição:

ID	***Target***	**Pred.**	**Res.**
1	spam	ham	FN
2	spam	ham	FN
3	ham	ham	TN
4	spam	spam	TP
5	ham	ham	TN
6	spam	spam	TP
7	ham	ham	TN
8	spam	spam	TP
9	spam	spam	TP
10	spam	spam	TP

ID	***Target***	**Pred.**	**Res.**
11	ham	ham	TN
12	spam	ham	FN
13	ham	ham	TN
14	ham	ham	TN
15	ham	ham	TN
16	ham	ham	TN
17	ham	spam	FP
18	spam	spam	TP
19	ham	ham	TN
20	ham	spam	FP

Tabela 2: Exemplo de um *dataset* de teste e as predições feitas pelo modelo.

A partir da coluna RES. podemos construir a matriz de confusão:

[5] As vezes a matriz de confusão pode ser representada com os valores negativos do *target* como primeiro índice das linhas e os valores negativos como primeiro índice das colunas:

$$\begin{matrix} TN & FP \\ FN & TP \end{matrix}$$

A biblioteca Scikit-Learn é famosa por utilizar este formato.

		Predição	
		spam	**ham**
Target	**spam**	6	3
	ham	2	9

Tabela 3: Estrutura da uma matriz de confusão obtida para o *dataset* sobre classificação de SPAM.

Entender a matriz de confusão e os tipos de erro associados é importante para uma análise aprofundada do desempenho do modelo em problemas de classificação. Isso permite ajustes no modelo, escolha de limiares de decisão apropriados(no caso de modelos que predizem probabilidades) e avaliação mais completa das consequências práticas das predições do modelo.

Métricas de classificação

O método padrão para avaliar um modelo de classificação é medir a taxa de erros (ou taxa de acertos[6]) em um *dataset* de teste. A taxa de erros pode ser facilmente calculada a partir da matriz de confusão:

[6] Mais conhecida como acurácia.

$$\text{taxa de erros} = \frac{(FP+FN)}{(TP+TN+FP+FN)} = \frac{(2+3)}{(6+9+2+3)} = 0.25 \tag{1}$$

$$\text{taxa de acertos} = \frac{(TP+TN)}{(TP+TN+FP+FN)} = \frac{(6+9)}{(6+9+2+3)} = 0.75 \tag{2}$$

Note que para o cálculo da taxa de erros, utilizamos a soma dos erros do Tipo I e do Tipo II (FP e FN) e dividimos pelo total de predições feitas pelo modelo. Por outro lado, para a acurácia, utilizamos os acertos do modelo divido pelo total de predições feitas.

Apesar de muito popular, considerar as taxas de erro e acerto de forma bruta pode levar a decisões errôneas. Considere um cenário com um *dataset* de teste hipotético contendo 100 exemplos. Destes, 98 são de uma classe A e apenas 2 são classe B. Imagine agora, um modelo "burro" $\mathbb{M}1$ que sempre retorna A como predição, independente das *features* fornecidas como entrada:

```
def model_m1(features):
    return "A"
```

Um modelo assim, no *dataset* hipotético, teria uma acurácia altíssima, de 98%! Agora, considere que este modelo seja colocado em produção e continue somente predizendo somente A. Considere também que este modelo é utilizado para predição de *customer churn* (clientes que vão cancelar contrato e migrar para um concorrente) e a categoria B indica clientes que vão migrar. Em um exemplo destes,

pode não parecer muito quando observamos que apenas 2 clientes correm o risco de migrar para concorrência. Agora, imagine que estes 2 clientes correspondam a, digamos $> 80\%$ do faturamento da empresa. Caso eles migrem, há um risco de falência associado à esta saída. De nada adiantaria um modelo com 98% de acurácia neste caso. Seria muito mais interessante ter um modelo com 20% de acurácia mas que sempre acerte a classe B.

Além da taxa de erros e da acurácia, podemos extrair outras métricas úteis da matriz de confusão. As mais simples são

- Taxa de verdadeiros positivos: $\text{TPR} = \frac{TP}{(TP+FN)}$
- Taxa de verdadeiros negativos: $\text{TNR} = \frac{TN}{(TN+FP)}$
- Taxa de falsos positivos: $\text{FPR} = \frac{FP}{(TN+FP)}$
- Taxa de falsos negativos[7]: $\text{FNR} = \frac{FN}{(TP+FN)}$

[7] Para os exemplos contidos na Tabela 2, $\text{TPR} = 0.667$, $\text{TNR} = 0.818$, $\text{FPR} = 0.182$ e $\text{FNR} = 0.333$.

Além destas, há outras métricas mais complexas que são amplamente utilizadas no *machine learning* e também em outras áreas, a precisão (*precision*) e especificidade (*recall* ou revocação):

$$precision = \frac{TP}{(TP + FP)} \tag{3}$$

$$recall = \frac{TP}{(TP + FN)} \tag{4}$$

A precisão nos dá uma ideia de qual é a confiança do modelo ao fazer uma predição sobre a classe definida como sendo positiva, pois captura a frequência com a qual uma predição feita para esta classe esta correta. O *recall* nos diz qual a confiança de que todas as instâncias da classe definida como positiva foram realmente encontradas[8].

[8] Para os exemplos contidos na Tabela 2, $precision = 0.75$ e $recall = 0.667$.

A classificação de e-mails é um bom cenário de aplicação no qual as informações distintas fornecidas pela precisão e pela sensibilidade são úteis. O valor de precisão nos diz quão provável é que um e-mail legítimo seja marcado como spam e, presumivelmente, excluído: $25\% = (1 - precision)$. Por outro lado, o *recall* nos diz quão provável é que um e-mail de spam seja perdido pelo sistema e acabe em nossa caixa de entrada: $33,333\%(1 - recall)$. Ter ambos os números é útil, pois nos permite pensar sobre ajustar o modelo para um tipo de erro ou outro. É melhor para um e-mail legítimo ser marcado como spam e excluído, ou para um e-mail de spam acabar em nossa caixa de entrada? O desempenho registrado na Tabela 2 mostra que este sistema

tem uma probabilidade ligeiramente maior de cometer o segundo tipo de erro do que o primeiro.

Há ainda uma outra métrica muito útil que serve como alternativa à acurácia e a taxa de erros quando os dados do *dataset* não são balanceados, a *f1-metric*:

$$\text{F}_1\text{-measure} = 2 \times \frac{(\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})} \tag{5}$$

que é a média harmônica da precisão e do *recall*[9]. A *f1-metric* sempre terá um valor no intervalo definido pela precisão e pelo *recall* utilizados para calculá-la.

[9] Para o *dataset* da Tabela 2, $f1 = 0.706$

Em alguns casos, apenas calcular as métricas pode transmitir uma ideia errônea sobre o modelo, especialmente quando alguns erros são mais importantes do que os outros. Se associarmos um valor, seja um "peso" ou um valor monetário, aos erros e acertos nestes casos, podemos ter uma visão mais clara do que está acontecendo. Considere a estrutura de uma matriz de *profit* e possíveis valores associados a cada um dos resultados de uma tarefa de classificação binária:

		Predições	
		Positivo	Negativo
target	Positivo	TP_{Profit}	FN_{Profit}
	Negativo	FP_{Profit}	TN_{Profit}

(a)

		Predições	
		Positivo	Negativo
target	Positivo	140	−140
	Negativo	−700	0

(b)

Tabela 4: Estrutura da matriz de *profit* (a) e exemplo de matriz de *profit* com os lucros e perdas de cada tipo de acerto e cada tipo de erro (b).

		Predições	
		Positivo	Negativo
target	Positivo	43	17
	Negativo	3	37

(a)

		Predições	
		Positivo	Negativo
target	Positivo	57	3
	Negativo	10	30

(b)

Tabela 5: Exemplo de matriz de confusão para uma árvore de decisão (a) com average class accuracy$_{HM}$ = 80.761% e para um modelo k-NN (b) com average class accuracy$_{HM}$ = 83.824%

Se escolhermos o modelo exclusivamente por uma métrica (neste caso a average class accuracy$_{HM}$), escolheríamos o modelo KNN pois average class accuracy$_{HM}$ = 83.824%. Se analisarmos os erros cometidos pelos dois modelos, perceberemos que a Árvore de Decisão, mesmo tendo uma métrica geral mais baixa, obteve um lucro maior comparado ao outro modelo:

		Predições	
		Positivo	Negativo
target	Positivo	6 020	−2 380
	Negativo	−2 100	0

(a)

		Predições	
		Positivo	Negativo
target	Positivo	7 980	−420
	Negativo	−7 000	0

(b)

Tabela 6: Árvore de decisão (a) com lucro de 1 540 e kNN (b) com lucro de 560.

A *Average Class Accuracy* é uma métrica usada para avaliar o desempenho de modelos de classificação multiclasse. Ao contrário da acurácia tradicional, que considera apenas a taxa global de acertos, a acurácia média por classe fornece uma visão mais granular do desempenho do modelo em cada classe individualmente.

Para cada classe no problema de classificação, calcule a acurácia separadamente. A acurácia por classe é a proporção de instâncias corretamente classificadas para aquela classe em relação ao total de instâncias dessa classe.

$$\text{Average Class Accuracy} = \frac{\text{N}^o \text{ de instâncias corretamente classificadas para a classe}}{\text{Número total de instâncias da classe}} \tag{6}$$

Outra forma de calcular a *Average Class Accuracy* é utilizando o *recall* das classes:

$$\text{average class accuracy} = \frac{1}{|levels(t)|} \sum_{l \in levels(t)} \text{recall}_l \tag{7}$$

onde: $levels(t)$ é o conjunto de classes que podem ser utilizadas na classificação; $|levels(t)|$ é o número de classes em $levels(t)$.

Uma vez que você tem a acurácia para cada classe, calcule a média dessas acurácias para obter a média harmônica da *Average Class Accuracy*:

$$\text{Acurácia Média por Classe} = \frac{\sum \text{Acurácia por Classe}}{\text{Número de Classes}} \tag{8}$$

Assim como a *Average Class Accuracy*, podemos utilizar o *recall* para o cálculo dessa versão com média harmônica:

$$\text{average class accuracy}_{\text{HM}} = \frac{1}{\frac{1}{|levels(t)|} \sum_{l \in levels(t)} \frac{1}{\text{recall}_l}} \tag{9}$$

Essas métricas são particularmente úteis quando há desequilíbrio nas classes, ou seja, quando algumas classes têm mais instâncias do que outras. A acurácia tradicional pode ser enganosa em situações desse tipo, pois um modelo pode ter uma acurácia alta simplesmente prevendo a classe majoritária em todos os casos. A Acurácia Média

por Classe fornece uma visão mais equilibrada do desempenho em todas as classes.

É importante observar que, embora a Acurácia Média por Classe seja informativa, pode ser complementada com outras métricas específicas de classe, como precisão, recall ou F1-score, dependendo dos requisitos específicos do problema. Essas métricas individuais por classe podem ajudar a entender melhor o comportamento do modelo em diferentes contextos.

Métricas sobre scores

Alguns modelos de *machine learning* não predizem uma classe diretamente quando utilizados para tarefas classificação. É o caso, por exemplo, da regressão logística. A regressão logística (Equação 10) utiliza a função sigmoide[10] para produzir a saída do modelo.

$$y = \frac{1}{1 + e^{-(\mathbf{W}^T\mathbf{x}+b)}} \tag{10}$$

[10] Note que a função sigmoide é a mesma função de ativação utilizada pelos neurônios das RNAs. Uma forma de interpretar os neurônios da RNA é que cada um deles é uma regressão linear independente.

Apesar desta saída ter uma interpretação similar à uma probabilidade[11], sempre que uma regressão linear produz um predição, devemos setar um *threshold* para dizer a partir de que valor de saída uma classe interpretada como 1 e até onde ela é 0. Supondo um *threshold* de 0.5[12], nossa predição fica:

[11] Lembre-se que a função sigmoide produz um número no intervalo $[0,1]$, que é o mesmo intervalo das probabilidades.

[12] Na prática, 0.5 é o *threshold default* da regressão logística.

$$threshold(score, 0.5) = \begin{cases} positivo & \text{if } threshold \geq 0.5 \\ negativo & \text{demais casos} \end{cases} \tag{11}$$

Assumir que o *threshold* padrão é o suficiente para todos os casos de predição, pode nos induzir à predições errôneas. Por isso, precisamos otimizar o *threshold* do nosso modelo tanto para aumentar a performance do modelo quanto para sobre ajustar o modelo para um tipo de erro ou outro[13].

[13] Veja discussão sobre tipos de erro na Seção .

Para nos auxiliar no entendimento das métricas, vamos atualizar a tabela de predição de SPAM para que esta inclua os scores produzidos por algum modelo, na Tabela 7:

As métricas *ROC* (*Receiver Operating Characteristic*) e AUC (*Area Under the Curve*) são comumente usadas para avaliar o desempenho de modelos de classificação binária. A curva *ROC* é uma representação gráfica da capacidade de um modelo de distinguir entre classes positivas e negativas em diferentes *thresholds*. Ela é construída traçando

ID	*target*	Pred.	Score	Res.	ID	*Target*	Pred.	Score	Res.
7	ham	ham	0.001	TN	5	ham	ham	0.302	TN
11	ham	ham	0.003	TN	14	ham	ham	0.348	TN
15	ham	ham	0.059	TN	17	ham	spam	0.657	FP
13	ham	ham	0.064	TN	8	spam	spam	0.676	TP
19	ham	ham	0.094	TN	6	spam	spam	0.719	TP
12	spam	ham	0.160	FN	10	spam	spam	0.781	TP
2	spam	ham	0.184	FN	18	spam	spam	0.833	TP
3	ham	ham	0.226	TN	20	ham	spam	0.877	FP
16	ham	ham	0.246	TN	9	spam	spam	0.960	TP
1	spam	ham	0.293	FN	4	spam	spam	0.963	TP

Tabela 7: Exemplo de predições, seus SCORES e o resultado das predições em termos dos elementos da matriz de confusão quando *threshold* = 0.5.

a taxa de verdadeiros positivos (TPR ou Sensibilidade) contra a taxa de falsos positivos (FPR ou $1 - Especificidade$) em vários limiares de decisão. Cada ponto na curva representa um *threshold*. A ideia por tras da *ROC* é que antes que o modelo faça uma predição de Falso Positivo, todos os Positivos Verdadeiros já tenham sido encontrados. Assim sendo, podemos utilizar esta curva como um guia para escolhermos os *thresholds* do nosso modelo.

		Predições	
		Positivo	Negativo
target	Positivo	4	4
	Negativo	2	10

(a)

		Predições	
		Positivo	Negativo
target	Positivo	7	2
	Negativo	4	7

(b)

Tabela 8: Matriz de confusão para as predições utilizando a tabela de exemplos de predições e *scores* com *threshold* = 0.75 (a) e *threshold* = 0.25 (b).

Note que conforme a TPR diminui, a TNR aumenta (e vice-versa) e a captura deste *tradeoff* é o propósito da *ROC*.

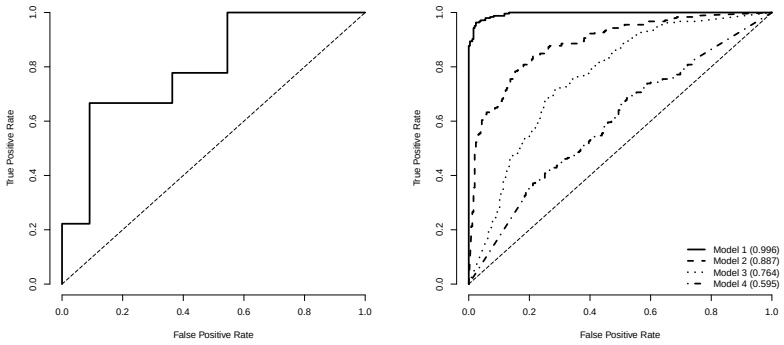

Figura 4: (a) Exemplo da *ROC* para o exemplo de predição de SPAM e HAM com *scores*; (b) seleção de várias *ROC* para diferentes modelos treinados para a mesma tarefa.

Sobre o a *ROC* ainda podemos calcular uma outra métrica, chamada de *Área Under de Curve* (*AUC*, ou *ROC-index*). A *AUC* pode ser calculada utilizando o método do trapézio:

$$\text{AUC} = \sum_{i=2}^{|\mathbf{T}|} \frac{(FPR(\mathbf{T}[i]) - FPR(\mathbf{T}[i-1])) \times (TPR(\mathbf{T}[i]) + TPR(\mathbf{T}[i-1]))}{2} \tag{12}$$

Existem ainda outras métricas menos conhecidas e que são utilizadas em situações mais pontuais, como a *Kolmogorov-Smirnov statistic*(KS-statistic ou KS).Esta é uma medida que quantifica a diferença cumulativa entre a distribuição empírica de um conjunto de dados e uma distribuição teórica de referência. Essa estatística é frequentemente utilizada para testar se uma amostra segue uma distribuição específica.

A ideia por trás do teste KS é comparar a distribuição acumulada empírica (ECDF) dos dados observados com a distribuição acumulada teórica, que pode ser, por exemplo, uma distribuição normal, exponencial, ou qualquer outra distribuição específica. O procedimento básico envolve, primeiro, ordenar os dados em ordem crescente. Depois calcula-se a ECDF para os dados observados e compara-se a ECDF dos dados observados com a distribuição acumulada teórica. A KS é então calculada como a maior diferença absoluta entre a ECDF dos dados observados e a distribuição acumulada teórica.

$$D = \max(|F_n(x) - F(x)|) \tag{13}$$

onde: $F_n(x)$ é a função de distribuição acumulada empírica dos dados observados; $F(x)$ é a função de distribuição acumulada teórica. A estatística D é comparada a valores críticos de uma tabela KS ou é usada para calcular um p-valor associado. Se o p-valor for menor que um nível de significância escolhido, há evidências para rejeitar a hipótese nula de que os dados seguem a distribuição teórica.

Na prática, para calcular a KS, primeiro determinamos a distribuição cumulativa dos *scores* produzidos para os *targets* positivos e negativos, e depois calculamos a estatística D:

$$CP(positive, ps) = \frac{\text{num positive test instances with score} \leq ps}{\text{num positive test instances}} \tag{14}$$

$$CP(negative, ps) = \frac{\text{num negative test instances with score} \leq ps}{\text{num negative test instances}} \tag{15}$$

O *KS* é calculado determinando a maior diferença entre as distribuições cumulativas para os *targets* positivos e negativos.

$$\text{KS} = \max_{ps} \left(CP(positive, ps) - CP(negative, ps) \right) \tag{16}$$

O teste *KS* é não paramétrico e relativamente fácil de entender e implementar. No entanto, é mais sensível a diferenças nas caudas das distribuições do que em suas partes centrais. Para conjuntos de dados grandes, outros testes de aderência à distribuição podem ser mais apropriados.

O *Gain* e o *Lift* são métricas comumente usadas em problemas de classificação para avaliar o desempenho de modelos de pontuação, como modelos de escore de crédito ou modelos de resposta a campanhas de marketing.

O *Gain* é uma métrica que compara o desempenho do modelo com um modelo aleatório. Ele representa a proporção acumulativa de casos positivos (por exemplo, respostas a uma campanha) capturados à medida que os exemplos são classificados em ordem decrescente de pontuação do modelo. A fórmula é dada por:

$$\text{Gain} = \frac{\text{Número acumulado de positivos capturados no decil D}}{\text{Número total de positivos}} \times 100 \tag{17}$$

O *Lift* é derivado do *Gain* e indica o quão melhor o modelo está em comparação com um modelo aleatório. Pode ser calculado como o ganho dividido pelo ganho esperado sob condições aleatórias. A fórmula é dada por:

$$\text{Lift} = \frac{\text{Percentual de exemplos positivos no decil D}}{\text{Percentual de casos teste}} \tag{18}$$

O *Cumulative Gain* é a soma acumulativa do ganho à medida que percorremos os dados classificados pelo modelo. Ele mostra como a captura de casos positivos se acumula à medida que mais exemplos são considerados. A fórmula é semelhante à do *Gain*, mas é calculada acumulativamente. O *Cumulative Lift* é a soma acumulativa do *Lift*. Ele expressa o ganho relativo acumulado em relação a um modelo aleatório. A fórmula é semelhante à do *Lift*, mas é calculada acumulativamente.

Essas métricas são frequentemente utilizadas em situações em que a taxa de positivos é baixa e o foco está na identificação eficiente dos

casos positivos. Por exemplo, em campanhas de marketing direcionadas, onde o objetivo é identificar um subconjunto de clientes mais propensos a responder. O *Gain* e o *Lift* fornecem uma visão prática de como o modelo melhora as chances de identificar casos positivos em comparação com uma abordagem aleatória.

Métricas de Regressão

Além de métricas de classificação, ainda há as métricas de regressão as quais medem algum grau de erro em relação às predições feitas pelo modelo.

A Soma dos Erros Quadráticos (*Sum of Squared Errors - SSE*) é a soma dos quadrados das diferenças entre os valores preditos pelo modelo e os valores reais da variável dependente. A SSE mede a soma dos quadrados das diferenças entre os valores reais e os valores previstos. Quanto menor a SSE, melhor o modelo está em ajustar os dados. A penalização quadrática dá mais peso a grandes erros, destacando previsões significativamente erradas. A fórmula do *SSE* é:

$$\text{SSE} = \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2 \tag{19}$$

onde t_i é o *target* correspondente ao exemplo i do *dataset* de teste; $\mathbb{M}(\mathbf{d}_i))$ é a predição feita pelo modelo; n é o número de exemplos do *dataset* de teste.

Erro Quadrático Médio (*Mean Squared Error - MSE*) é a média da SSE e é uma métrica amplamente usada para avaliar a qualidade das previsões em problemas de regressão. O MSE é a média da SSE e fornece uma medida média dos erros de previsão. Assim como a SSE, o MSE penaliza mais fortemente grandes erros, tornando-o sensível a outliers. No entanto, como é uma média, o MSE é mais fácil de interpretar. O *MSE* é calculado como:

$$\text{MSE} = \frac{\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2}{n} \tag{20}$$

onde t_i é o *target* correspondente ao exemplo i do *dataset* de teste; $\mathbb{M}(\mathbf{d}_i))$ é a predição feita pelo modelo; n é o número de exemplos do *dataset* de teste.

A Raiz do Erro Quadrático Médio (*Root Mean Squared Error - RMSE*) é a raiz quadrada do MSE e fornece uma medida do erro médio em unidades da variável dependente. O *RMSE* é a raiz qua-

drada do MSE e tem a mesma unidade da variável dependente. É uma métrica mais intuitiva, pois retorna a medida de erro em uma escala semelhante aos valores originais. Isso facilita a interpretação e a comparação entre modelos. A equação do *RMSE* é:

$$\text{RMSE} = \sqrt{MSE} \tag{21}$$

O MAE é a média das diferenças absolutas entre os valores reais e os valores previstos. O MAE é menos sensível a outliers do que o MSE, já que não penaliza grandes erros de forma quadrática. Ele fornece uma medida mais robusta da precisão[14] média do modelo.

[14] Não confundir com a precisão utilizada em classificação!

$$\text{MAE} = \frac{\sum_{i=1}^{n} abs(t_i - \mathbb{M}(\mathbf{d}_i))}{n} \tag{22}$$

onde t_i é o *target* correspondente ao exemplo i do *dataset* de teste; $\mathbb{M}(\mathbf{d}_i))$ é a predição feita pelo modelo; n é o número de exemplos do *dataset* de teste; e *abs* é a operação de valor absoluto.

O R^2 é uma métrica independente do domínio. Esta métrica calcula a proporção da variabilidade total na variável dependente que é explicada pelo modelo. Um R^2 de 1 indica que o modelo explica toda a variabilidade, enquanto um R^2 de 0 indica que o modelo não oferece nenhuma melhoria em relação à simples média dos valores observados. É uma medida de ajuste global do modelo aos dados. Sua equação é:

$$R^2 = 1 - \frac{\text{SSE}}{\text{total sum of squares}} \tag{23}$$

$$\text{total sum of squares} = \frac{1}{2}\sum_{i=1}^{n} \left(t_i - \bar{t}\right)^2 \tag{24}$$

onde: onde t_i é o *target* correspondente ao exemplo i do *dataset* de teste; $\bar{t}$ é a média calculada entre todos os *targets* no *dataset* de teste; n é o número de exemplos do *dataset* de teste.

Cada métrica tem suas próprias vantagens e limitações, e a escolha entre elas depende do contexto específico do problema. Por exemplo, se a interpretabilidade é crucial e se quer evitar penalizar outliers, o MAE pode ser preferido. Se a ênfase está em destacar grandes erros, o MSE ou o RMSE podem ser mais apropriados. O R^2 fornece uma visão geral da qualidade do ajuste do modelo em relação à variabilidade total dos dados.

Outras Métricas Interessantes

Conforme a tarefa para qual o modelo esteja sendo treinado, outras métricas que refletem melhor a solução para o problema podem ser utilizadas. Desta forma, existem outras métricas mais "exóticas" também podem ser requeridas conforme a necessidade, como a *perplexity* em modelos de linguagem, o *BLEU score* para sistemas de tradução e *Top-K accuracy* para problemas de classificação multiclasse.

A *perplexity* (perplexidade) é frequentemente usada em tarefas de linguagem natural, enquanto a entropia cruzada categórica é uma métrica comum em problemas de classificação. Em termos mais simples, a perplexidade é a inversa geométrica da probabilidade normalizada da sequência. Um modelo de linguagem ideal terá uma perplexidade próxima de 1, indicando que o modelo não está surpreso e faz previsões muito precisas. No entanto, na prática, perplexidades muito baixas podem ser difíceis de alcançar. O cálculo da perplexidade é realizado sobre a *Categorical Cross Entropy*[15]

[15] $H(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i)$

$$\text{Perplexidade} = \exp(\text{Categorical Cross Entropy}) \tag{25}$$

onde: exp é a operação exponencial, ou seja, $exp(x) = e^(- x)$.

A *Categorical Cross Entropy* mede a "distância" entre as distribuições de probabilidade real e prevista. Quanto menor a *Categorical Cross Entropy*, melhor o modelo está em prever as classes corretas. A perplexidade é uma medida inversa da probabilidade média atribuída pelo modelo a eventos reais. Quanto menor a perplexidade, melhor o modelo está em fazer previsões precisas. Lembre-se de que a interpretação exata da perplexidade pode variar dependendo do contexto específico da aplicação, e é sempre uma boa prática considerar a utilização de várias métricas para avaliar o desempenho do modelo de forma abrangente.

Além disso, a perplexidade ode ser usada no ajuste de hiperparâmetros durante o treinamento do modelo. O objetivo é minimizar a perplexidade, o que, por sua vez, significa maximizar a probabilidade das sequências observadas. No entanto, a perplexidade é sensível ao tamanho do vocabulário. Modelos com vocabulários menores podem ter perplexidades mais baixas mesmo que não capturem tão bem a complexidade da linguagem.

A métrica *BLEU* (*Bilingual Evaluation Understudy*) é uma métrica de avaliação amplamente utilizada para avaliar a qualidade de traduções automáticas em tarefas de processamento de linguagem natural, especialmente em tradução automática. Foi proposta como uma métrica para avaliar a qualidade de traduções automáticas em

comparação com as traduções de referência, ouro padrão humano.

A *BLEU* calcula a precisão dos *n-grams* (sequências de n palavras) nas traduções geradas em relação às traduções de referência. Quanto mais n-gramas da tradução candidata coincidirem com os n-gramas das referências, maior será a pontuação de precisão. Assim, penaliza as traduções candidatas mais curtas em comparação com as referências para evitar que o modelo escolha traduções muito curtas para obter pontuações mais altas. Isso é conhecido como a penalização de brevidade. A fórmula geral para o cálculo da pontuação *BLEU* é dada por:

$$BLEU = \text{Brevity Penalty} \times \exp\left(\sum_{n=1}^{N} \frac{1}{N} \log \text{precision}_n\right) \tag{26}$$

onde: N é o máximo de n-gramas considerados; precision_n é a precisão dos n-gramas; e o *Brevity Penalty* é aplicado para evitar favorecer traduções muito curtas.

A *BLEU* é amplamente utilizado para comparar diferentes sistemas de tradução automática ou diferentes configurações de modelos. Apesar de sua ampla aplicação, é importante notar que a *BLEU* é uma métrica automática e pode não correlacionar perfeitamente com avaliações humanas. Ela é uma ferramenta útil, mas não substitui a avaliação humana completa. Outro ponto a ser considerado, é que a pontuação *BLEU* é mais alta quando considera mais n-gramas. Portanto, é importante considerar o valor de N ao interpretar a pontuação.

Além disso, *BLEU* pode não ser ideal quando há uma grande discrepância no vocabulário entre as traduções de referência e a tradução candidata, como quando palavras são substituídas pos sinônimos na tradução. Em algumas aplicações, pode ser útil personalizar ou ajustar a métrica *BLEU* para melhor se adequar a domínios específicos ou requisitos de avaliação. Embora a *BLEU* seja uma métrica valiosa, é recomendável usá-la em conjunto com outras métricas e, quando possível, complementar a avaliação com avaliações humanas para uma compreensão mais completa do desempenho do modelo de tradução automática.

A métrica *Top-k Accuracy* é uma medida de desempenho usada em problemas de classificação multiclasse. Em vez de avaliar apenas se a classe predita é a correta, a *top-k accuracy* considera se a classe correta está entre as k principais classes com as maiores probabilidades previstas pelo modelo.

A fórmula geral para calcular a *top-k accuracy* é a seguinte:

$$\text{Top-k Accuracy} = \frac{\text{N}^o\text{de previsões corretas em que a classe correta está entre as k principais}}{\text{N}^o \text{ total de exemplos no } \textit{dataset}} \tag{27}$$

onde: k é um parâmetro que determina quantas previsões principais serão consideradas. Por exemplo, para *top-1 accuracy*, $k = 1$, o que significa que apenas a classe com a probabilidade mais alta é considerada e é equivalente a acurácia tradicional.

A *top-k accuracy* é útil em cenários em que não é estritamente necessário que o modelo preveja a classe exata, mas é aceitável se a classe correta estiver entre as principais classes previstas. Isso é especialmente relevante quando lidamos com problemas em que há incerteza sobre a classe correta ou quando o modelo deve fazer previsões de classes semânticas semelhantes.

A *top-k accuracy* pode ser interpretada como a porcentagem de amostras para as quais a classe correta está entre as k previsões mais prováveis feitas pelo modelo. Quanto maior a *top-k accuracy*, melhor o modelo está em capturar a classe correta, mesmo que não seja a principal previsão.

É importante escolher k com base no contexto do problema e nas expectativas específicas do aplicativo. Em alguns casos, a *top-5 accuracy* (onde $k = 5$) é usada, permitindo mais flexibilidade para considerar várias previsões principais. Em geral, a *top-k accuracy* oferece uma visão mais abrangente da capacidade de um modelo de classificação de lidar com incerteza nas previsões.

Considerações Finais

A avaliação de modelos de *machine learning* é parte importante do desenvolvimento de sistemas de aprendizado de máquina, pois permite determinar quão bem um modelo está realizando suas previsões em dados não vistos. Os dados geralmente são divididos em conjuntos de treinamento e teste. O modelo é treinado no conjunto de treinamento e avaliado no conjunto de teste para verificar como ele generaliza para dados não vistos.

Existem várias métricas para avaliar o desempenho de um modelo, dependendo do tipo de problema. Algumas métricas comuns incluem *precision*, *recall*, *F1-score*, *ROC* e *AUX*, erro absoluto médio (*MAE*), erro quadrático médio (*MSE*) e muitas outras.

O objetivo final é criar um modelo que generalize bem para dados não vistos. Avaliações repetidas em conjuntos de teste independentes

podem ajudar a garantir isso. Além disso, modelos podem sofrer de *overfitting* (ajuste excessivo aos dados de treinamento) ou *underfitting* (incapacidade de capturar os padrões nos dados). A avaliação ajuda a identificar esses problemas. Além disso, ao avaliar modelos de machine learning, é essencial considerar a natureza do problema e escolher as métricas apropriadas.

Referências

John D. Kelleher and Brendan Tierney. *Data Science*. MIT Press, 1 edition, 2018a.

John D. Kelleher and Brendan Tierney. *Deep Learning*. MIT Press, 1 edition, 2018b.

John D. Kelleher, Brian Mac Namee, and Aoife D'Arcy. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples and Case Studies*. MIT Press, 2 edition, 2020.