Métricas de Avaliação e Divisão de Dados

Advanced Institute for Artificial Intelligence – AI2

https://advancedinstitute.ai

Métricas de Avaliação

Métricas são medidas quantificáveis empregadas para analisar resultados de um processo, ação ou estratégia específica.

Ao trazemos a definição para o mundo de ML, as métricas são essenciais para informar o desempenho de um determinado modelo em uma dada aplicação. Desta forma, a escolha da métrica avaliativa precisa, necessariamente, ser representativa e concordante, com o problema tratado.

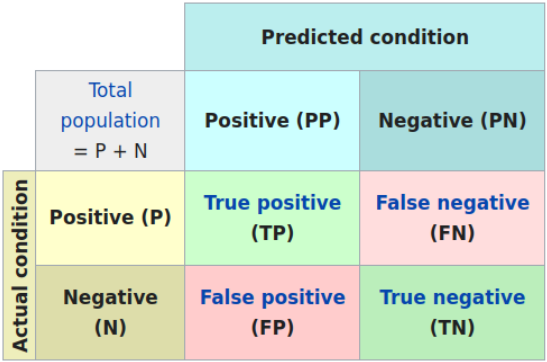
Para problemas de classificação, a literatura nos fornece algumas métricas bem estabelecidas e concisas que auxiliam no acompanhamento de desempenho dos modelos de ML, tanto em etapa de treinamento quanto de teste.

Tomemos como base para o desenvolvimento do tema, um problema de classificação binária cujo objetivo é separar pregos de parafusos em uma esteira industrial.

Dentre as diversas métricas na literatura algumas são mais empregadas, tais como:

- *Accuracy* - Acurácia;
- *Precision* - Precisão;
- *Recall* - Revocação;
- *F-Measure*;

As predições do modelo de ML geram a chamada matriz de confusão (ou contingência):



**FP $->$ superestimativa (alarme falso) - erro tipo I;

**FN $->$ subestimativa (erro grave) - erro tipo II;

Accuracy - Representa a medida de acerto do modelo de ML, ou seja, quantos exemplos foram preditos corretamente frente à população de dados avaliada:

$$ACC = \frac{TP + TN}{P + N}$$
$$= \frac{TP + TN}{TP + TN + FP + FN}$$

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Pecision - Representa a fração de instâncias rotuladas corretamente (TP) dentre todas as que foram classificadas como corretas, ou seja, verdadeiras positivas e falsas positivas:

$$PR = \frac{TP}{TP + FP}$$

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Recall - Representa a fração de instâncias rotuladas corretamente (TP) dentre todas as positivas da população de dados:

$$RC = \frac{TP}{P}$$
$$= \frac{TP}{TP + FN}$$

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Exemplo: Considere um modelo de ML para reconhecer cães (elemento relevante) em uma imagem. Ao processar uma imagem que contém dez gatos e doze cães, o modelo identifica oito cães.

Dos oito elementos identificados como cães, apenas cinco são realmente cães (TP), enquanto os outros três são gatos (FP).

Sete cães foram perdidos (FN) e sete gatos foram excluídos corretamente (TN).

A *precision* do modelo é então 5/8 (TP/elementos selecionados) enquanto sua *recall* é 5/12 (TP/elementos relevantes).

Em resumo: A *precision* pode ser vista como uma medida de **qualidade** e a *recall* como uma medida de **quantidade**.

Maior *precision* significa que um algoritmo retorna resultados mais relevantes do que irrelevantes, e alta *recall* significa que um algoritmo retorna a maioria dos resultados relevantes (sejam ou não irrelevantes também são retornados).

F-Measure - Combina *precision* e *recall* em uma média harmônica entre eles, sendo representada por:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$
$$= 2 * \frac{TP}{TP + FP + FN}$$

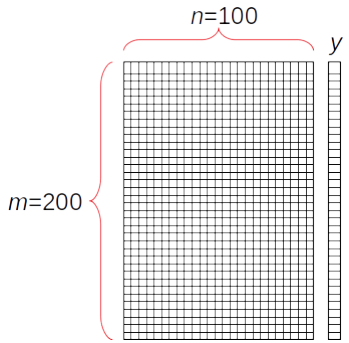
		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Divisão de Dados

No desenvolvimento de um modelo de *Machine Learning* levamos em consideração basicamente duas etapas, treinamento e teste. A primeira diz respeito ao processo de aprendizado do algoritmo para uma determinada tarefa, enquanto a segunda compreende a avaliação do modelo frente a dados ainda não vistos pelo mesmo. Matematicamente, o banco de dados $\mathcal{X}$ pode ser **particionado** da seguinte forma: $\mathcal{X} = \mathcal{X}^1 \cup \mathcal{X}^2$, em que $\mathcal{X}^1$ e $\mathcal{X}^2$ denotam os conjuntos de dados de **treinamento** e **teste**

Neste cenário, surge o problema de separação dos dados para as etapas mencionadas, ou seja, como separar os dados de forma que consigamos treinar um modelo eficientemente, e o mesmo não tenha a etapa de avaliação (teste) sub-representativa nos dados desconhecidos. Portanto, a separação deve ser cautelosa, analisando o conjunto de dados em algumas óticas, como por exemplo, o volume de dados e o balanceamento das classes, se falarmos de um problema de classificação.

Suponha um problema de classificação em que o conjunto de dados $\mathcal{X}$ possui 200 instâncias (m), com 100 variáveis preditoras (n) cada, tal que $\boldsymbol{x}_i^j \in \mathbb{R}^n$, e 4 classes para a variável alvo, $y_i \in [0, 3]$.



Como proceder se:

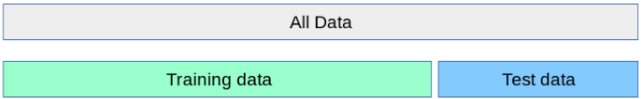
- Tivermos muitos registros com classes 0 e 1 (80% dos dados)?;
- Tivermos um banco de dados balanceado (25% para cada classe)?;
- Tivermos muitos registros com classes 0 e 1 (80% dos dados) e $m = 10.000$?.

Abordagens para reamostragem de dados:

- Hold-out;
- Cross-Validation;
- Nested-Cross-Validation*;
- Leave-One-Out Cross-Validation*;

São utilizadas em aplicações de: **estimativa de desempenho**, **seleção do modelo** e **ajuste de hiper-parâmetros**.

Hold-out (HO) é um método simples de separação de dados empregados em modelos de Machine Learning. Utiliza um percentual fixo para separar os dados em conjuntos de treinamento e teste, como por exemplo, 70% para treinamento e o restante para teste.



O método pode ser empregado para duas situações, treinamento e avaliação de modelos, ou, treinamento, avaliação e escolha do melhor modelo. Para qualquer aplicação vale ressaltar que os dados são mutuamente exclusivos, ou seja, pertencem ao mesmo conjunto de dados mas não para a mesma finalidade.

Passos do método:

1. Separar os dados de acordo com o percentual escolhido (geralmente 70-80% para treinamento);
2. Treina o modelo com um conjunto de hyper-parâmetros escolhidos;
3. Teste o modelo nos dados de teste;
4. Treino um modelo final em todo conjunto de dados se necessário (*deploy*).

Drawback:

- Não usa todos os dados para treinar o modelo;
- Não garante que instâncias críticas entrem para a partição de treino;
- Limitado para pequenos volumes de dados.

Como mitigar??

Cross-Validation (CV) é um método estatístico para avaliar e comparar algoritmos de aprendizado, utilizando a base do **Hold-out** (dados para treino e teste).

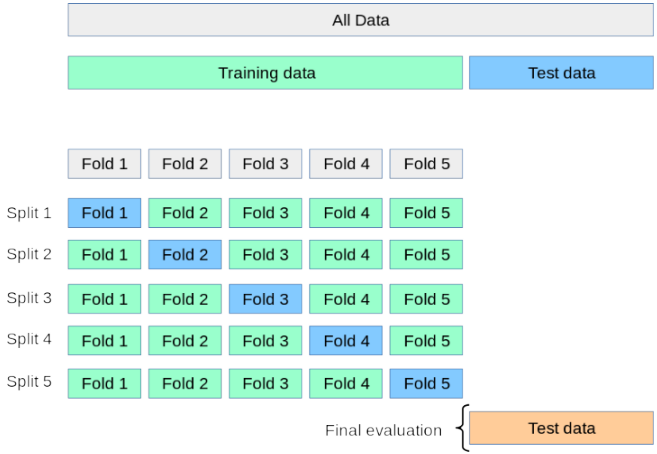
A diferença do CV para o HO está na criação de múltiplas divisões dos dados, ou seja, k partições são criadas para treinar e avaliar o modelo nos diferentes subconjuntos de dados, nomeando o método como k-**fold Cross-Validation**.

**É interessante para avaliar bancos de dados pequenos

**Para ML geralmente $k = 10$.

Passos do método:

1. Aleatorizar os dados para a divisão/estratificação;
2. Particionar os dados igualmente (ou o mais próximo possível) em k partições;
3. k iterações de treinamento e validação são feitas percorrendo as respectivas partições, $k-1$ para treinamento e 1 para validação/teste;
4. O desempenho (intermediário) dos modelos treinados corresponde à média das k iterações sob o conjunto de validação.


All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5
Final evaluation
Test data

Drawback:

- É computacionalmente custoso;
- Se não houver representatividade da distribuição de classes nas partições o modelo pode ter o aprendizado enviesado;
- Pode superestimar a confiança/desempenho do modelo.

Como mitigar??

Nested-Cross-Validation (NCV): Repete o procedimento e execução do $k-$fold CV dentro de cada partição k criada.

Leave-One-Out Cross-Validation (LOOCV): É um caso especial do $k-$fold CV, em que o k é igual ao número total de instâncias do banco de dados, ou seja, $k = m$. Uma única amostra é utilizada para testar o desempenho do modelo.