SUMÁRIO

O QUE VEM POR AÍ?

Nesta aula iremos aprofundar os conceitos relacionados ao aprendizado supervisionado. Iremos realizar o pré-processamento do conjunto de dados, com eventuais limpezas, imputações, transformações e seleção dos atributos (engenharia de features), deixando-os prontos para os algoritmos de aprendizado de máquina.

Iremos aprender também algumas das métricas de avaliação mais utilizadas no mercado para avaliar os modelos testados e, com isso, auxiliar na tomada de decisão para a escolha do modelo definitivo e seguir com o deploy em produção.

HANDS ON

Nesta aula iremos focar em dois momentos importantes no fluxo do aprendizado supervisionado. O primeiro são técnicas para preparar os dados que serão o input dos algoritmos de machine learning. Esta é uma etapa muito importante, pois se os dados não forem bem tratados com as devidas limpezas (valores ausentes e outliers), realização de eventuais transformações, eliminação de vieses e aplicação de engenharia de features o modelo não irá fazer milagre.

Há um acrônimo famoso em ciência da computação: GIGO - Garbage In, Garbage Out. Ou seja, se o seu dado de entrada tem pouca qualidade, dificilmente o modelo irá conseguir extrair o máximo de informação dele.

O segundo diz respeito a métricas que mensuram o desempenho do modelo, colocando em prática o que o algoritmo visualizou na etapa de treinamento. O algoritmo irá tentar prever, ou deduzir, a classe para novos exemplos que não viu na etapa de treinamento, colocando em prática seu aprendizado.

SAIBA MAIS

No mundo do aprendizado de máquina supervisionado, em que modelos são treinados a partir de dados rotulados para realizar previsões ou tomar decisões, o pré-processamento de dados e a avaliação de modelos assumem papeis cruciais para garantir qualidade, confiabilidade e robustez dos resultados.

Pré-processamento: Alicerçando o Aprendizado

O pré-processamento de dados é a etapa fundamental que prepara os dados brutos para serem utilizados pelos algoritmos de aprendizado de máquina. Imagine um(a) construtor(a) iniciando uma obra: antes de erguer as paredes, ele(a) precisa limpar o terreno, nivelar a base e garantir que os materiais estejam em boas condições. No aprendizado de máquina, o pré-processamento garante que os dados estejam livres de inconsistências, erros e anomalias que podem comprometer o aprendizado do modelo.

Podemos citar técnicas como:

- **Tratamento de valores ausentes:** preencher ou remover valores ausentes de forma estratégica, evitando distorções nas análises. Técnicas como média, mediana ou modelos preditivos podem ser utilizadas para preencher os valores ausentes, minimizando o impacto na qualidade dos dados.
- **Detecção e remoção de outliers:** valores atípicos que se desviam significativamente da distribuição dos dados podem levar a modelos enviesados. O pré-processamento fornece ferramentas para identificar e remover esses outliers, garantindo que o modelo aprenda com a maioria representativa dos dados.
- **Normalização e padronização:** ajustar a escala dos dados para facilitar a interpretação e o aprendizado do modelo. Dados com escalas diferentes podem dificultar o aprendizado do modelo, pois algoritmos tendem a dar mais peso a características com valores maiores.
- **Codificação de dados categóricos:** converter dados categóricos em valores numéricos que os algoritmos possam entender.

- **Tratamento de inconsistências:** erros de digitação, formatos incorretos e discrepâncias entre diferentes fontes de dados podem prejudicar a confiabilidade do modelo. O pré-processamento oferece técnicas para identificar e corrigir essas inconsistências, garantindo a integridade dos dados.
    - Por exemplo, uma coluna de números inteiros pode ser carregada como string (texto), inviabilizando qualquer cálculo nesta coluna.
- **Transformação de características:** criar novas features a partir das existentes, extraindo informações adicionais dos dados, caso necessário.

Iremos utilizar ferramentas como algumas bibliotecas Python (Pandas, Scikit-learn), que oferecem funções e métodos para realizar diversas tarefas de pré-processamento, e ferramentas de visualização de dados (Matplotlib), que permitem explorar e analisar os dados visualmente, identificando padrões e potenciais problemas.

Ao dedicar tempo e atenção ao pré-processamento, garantimos que o modelo de aprendizado de máquina receba dados de alta qualidade, construindo uma base sólida para um aprendizado eficaz e confiável.

```
In[1] df=pd.DataFrame({"nome":['Peter','Bruce', "T'Challa"],
                       "simbolo": ['aranha', 'morcego', 'pantera'],
                       "idade": [22, pd.NaT, 25]})
In[2] df
Out[2]
```

	nome	simbolo	idade
0	Peter	aranha	22
1	Bruce	morcego	NaT
2	T'Challa	pantera	25

```
In[3] df.dropna()
Out[3]
```

	nome	simbolo	idade
0	Peter	aranha	22
2	T'Challa	pantera	25

Código-fonte 1 – Código em Pandas para eliminar linhas com valores nulos
Fonte: Elaborado pelo autor (2024)

Avaliação: Calibrando o Desempenho

Após o treinamento do modelo, surge a crucial etapa de avaliação. Aqui, o modelo é colocado à prova em um conjunto de dados de teste, diferente dos dados utilizados no treinamento, para avaliar sua capacidade de generalização para novos exemplos.

Métricas:

- **Acurácia, Precisão, Revocação e F1-score:** para problemas de classificação, indicando a proporção de previsões corretas e a capacidade de identificar classes positivas e negativas.
- **Erro Médio Quadrático (MSE), Raiz Quadrada Média do Erro (RMSE) e Erro Percentual Médio Absoluto (MAPE):** para problemas de regressão, quantificando a diferença entre os valores previstos e os valores reais. Veja exemplos do cálculo do MSE e RMSE nos códigos-fonte 2 e 3, sem e com uso da biblioteca Scikit-Learn, respectivamente.

Técnicas como Validação cruzada: dividindo os dados em múltiplos subconjuntos para avaliar o modelo em diferentes cenários.

Ao avaliar o modelo:

- Identificamos seus pontos fortes e fracos.
- Comparamos diferentes modelos e selecionamos o melhor.
- Ajustamos os hiperparâmetros do modelo para otimizar seu desempenho.
- Monitoramos o desempenho do modelo em produção e o reajustamos conforme necessário.

```
import numpy as np

# Valores reais
reais = [100000, 150000, 200000, 250000, 300000]

# Valores previstos pelo modelo
prev = [95000, 142000, 210000, 245000, 315000]

# Cálculo do RMSE
diferencas = [r - p for r, p in zip(reais, prev)]
```

```
quadrados_das_diferencas = [d ** 2 for d in diferencas]
mse = np.mean(quadrados_das_diferencas)
rmse = np.sqrt(mse)

print(f"MSE: {mse}\nRMSE: {rmse}")
MSE: 87800000.0
RMSE: 9370.165420097983
```

Código-fonte 2 – Código para o cálculo do RMSE
Fonte: Elaborado pelo autor (2024)

```
import numpy as np
from sklearn.metrics import mean_squared_error

# Valores reais
reais = [100000, 150000, 200000, 250000, 300000]

# Valores previstos pelo modelo
prev = [95000, 142000, 210000, 245000, 315000]

# Cálculo do RMSE com scikit-learn
mse = mean_squared_error(reais, prev)
rmse = np.sqrt(mse)

print(f"MSE: {mse}\nRMSE: {rmse}")
MSE: 87800000
RMSE: 9370.165420097983
```

Código-fonte 3 – Código para o cálculo do RMSE com Scikit-Learn
Fonte: Elaborado pelo autor (2024)

Explicação da métrica:

- Sem o uso da biblioteca scikit-learn
  - É realizada a diferença entre os valores reais e preditos, elemento a elemento.
  - Cada valor é elevado ao quadrado (para remover eventuais números negativos).
  - Calcula-se a média para obter o MSE.
  - O RMSE é obtido com a raiz quadrada do MSE.
- Com o uso da biblioteca scikit-learn
  - **Importamos a função `mean_squared_error`:** essa função calcula o erro médio quadrático entre dois arrays (listas).
  - **Definimos as listas de valores:** mesma etapa dos exemplos anteriores.
  - **Cálculo do RMSE com `mean_squared_error`:**

- `mean_squared_error(valores_reais, valores_previstos)`: calcula o erro médio quadrático entre os valores reais e previstos.
- `np.sqrt(erro_medio_quadratico)`: calcula a raiz quadrada do erro médio quadrático, obtendo o RMSE.

Vantagens da utilização de bibliotecas:

- **Maior concisão:** o código se torna mais curto e legível, facilitando a compreensão e a reutilização.
- **Funcionalidades avançadas:** bibliotecas como scikit-learn oferecem diversas funções prontas para análise de dados e aprendizado de máquina, expandindo as possibilidades de processamento e análise.
- **Maior eficiência:** bibliotecas otimizadas podem realizar cálculos de forma mais rápida e eficiente, especialmente para grandes conjuntos de dados.

Recomendação: para projetos de aprendizado de máquina, a biblioteca scikit-learn se destaca como uma biblioteca completa e poderosa, oferecendo diversas ferramentas para pré-processamento, treinamento, avaliação e interpretação de modelos. No entanto, para cálculos simples e pontuais, a biblioteca NumPy pode ser uma opção mais leve e direta.

Lembre-se:

- O RMSE é apenas uma métrica de erro e deve-se utilizar outras métricas para analisar e avaliar o desempenho geral do modelo de aprendizado de máquina.
- A escolha da biblioteca ideal depende do contexto, da complexidade do projeto e das necessidades específicas do usuário.

O pré-processamento de dados e a avaliação de modelos são etapas indissociáveis do processo de aprendizado de máquina supervisionado. Ao dedicarmos o devido cuidado a essas etapas, garantimos que nossos modelos sejam robustos, confiáveis e capazes de gerar resultados precisos e relevantes para os problemas que desejamos solucionar. É como construir uma casa sólida: com uma base bem-preparada e acompanhamento constante, podemos ter certeza de que a construção resistirá ao tempo e às intempéries.

Outros tratamentos que cientistas podem fazer antes da etapa de treinamento é o tratamento de valores ausentes, detecção e remoção de outliers, normalização, padronização e codificação de dados categóricos (seja feature ou classe). Vejamos cada um em detalhes a seguir.

Tratamento de valores ausentes:

Dados ausentes podem prejudicar o desempenho do modelo. Abordagens comuns para lidar com essa situação incluem:

- **Remoção:** eliminar linhas ou colunas com valores ausentes, caso a quantidade seja significativa.
- **Imputação:** preencher os valores ausentes com valores estimados, como média, mediana ou valores previstos por um modelo auxiliar.

Exemplo de Imputação:

```
import numpy as np
from sklearn.impute import SimpleImputer

# Dados com valores ausentes
dados = np.array([[1, 2, 3, np.nan],
                  [4, np.nan, 6, 7],
                  [8, 9, 10, 11]])

# Imputação com a média das colunas
imputador=SimpleImputer(missing_values=np.nan, strategy='mean')
dados_imputados = imputador.fit_transform(dados)

print(f"Dados após imputação: {dados_imputados}")
array([[ 1. ,  2. ,  3. ,  9. ],
       [ 4. ,  5.5,  6. ,  7. ],
       [ 8. ,  9. , 10. , 11. ]])
```

Código-fonte 4 – Código para lidar com valores faltantes
Fonte: Elaborado pelo autor (2024)

Identificação de outliers:

Em datasets reais, outliers (valores atípicos) podem surgir por diversos motivos, como erros de coleta de dados, falhas em sensores ou comportamentos excepcionais. Esses outliers podem distorcer o aprendizado do modelo, levando a previsões incorretas e decisões equivocadas.

A detecção e remoção de outliers é uma etapa crucial no pré-processamento de dados, garantindo que o modelo seja treinado com base em informações confiáveis e representativas da população real. Uma das técnicas que pode ser utilizada é a Distância Interquartílica:

- **Distância interquartílica (IQR):** divide os dados em quartis e identifica pontos que se distanciam significativamente do intervalo entre o primeiro e o terceiro quartil (código-fonte 5). Procure relembrar o cálculo dos quartis e veja como usar a função `percentile` do numpy na documentação oficial.

```
import numpy as np

dados = np.array([1, 2, 3, 4, 5, 100])

q1 = np.percentile(dados, 25)
q3 = np.percentile(dados, 75)
iqr = q3 - q1

# Limite inferior e superior
limite_inf = q1 - 1.5 * iqr
limite_sup = q3 + 1.5 * iqr

outliers=np.where((dados<limite_inf)|(dados>limite_sup))[0]

print(f"Outliers por IQR: {outliers}")
Outliers por IQR: [5]
```

Código-fonte 5 – Código identificando outlier
Fonte: Elaborado pelo autor (2024)

Por padrão, utilizamos para o cálculo do limite superior e inferior um fator multiplicador de 1,5 para a diferença entre o quartil 3 e 1. (`q3` e `q1`). Será considerado outlier qualquer valor que esteja fora destes limites e na variável outliers haverá uma lista com todos estes valores. O resultado será a posição na lista (posição 5 na lista, o 6º elemento - relembrando que o primeiro elemento da lista está na posição zero).

Removendo Outliers:

Após identificar os outliers, a decisão de removê-los depende do contexto e da natureza dos dados.

- **Remoção:** eliminar os outliers do dataset antes de treinar o modelo.

- **Tratamento alternativo:** manter os outliers no dataset mas utilizar técnicas robustas no aprendizado deles, como modelos robustos ou algoritmos de estimação robusta.

Caso a opção seja remover, a continuação do exemplo no código-fonte 5 fica assim, indicando o índice da lista a ser removido.

```
print(f"Nova lista sem outlier: {np.delete(dados, 5)}")
Nova lista sem outlier: array([1, 2, 3, 4, 5])
```

Código-fonte 6 – Código removendo o outlier
Fonte: Elaborado pelo autor (2024)

Observações:

- A remoção de outliers deve ser feita com cautela, pois pode levar à perda de informações importantes.
- É importante analisar os outliers e entender suas causas antes de removê-los.
- Técnicas de detecção e remoção de outliers devem ser utilizadas em conjunto com outras técnicas de pré-processamento para garantir a qualidade dos dados.

Normalização e Padronização: ajustar a escala dos dados facilita o aprendizado do modelo, evitando que características com valores maiores dominem o processo.

- **Normalização:** ajusta os dados para um intervalo específico, como entre 0 e 1.
- **Padronização:** ajusta os dados com média 0 e desvio padrão 1.

No aprendizado supervisionado, a normalização dos dados é uma técnica de pré-processamento que visa ajustar a escala das variáveis em um dataset para um intervalo específico, geralmente entre 0 e 1 ou -1 e 1. Essa normalização garante que todas as variáveis contribuam de forma similar para o processo de aprendizado do modelo, evitando que características com valores maiores dominem o processo e prejudiquem a convergência do algoritmo.

Quando a normalização é particularmente importante?

1. **Algoritmos Sensíveis à Escala:** algoritmos como Regressão Linear, K-Nearest Neighbors (KNN) e Redes Neurais Artificiais com função de ativação sigmoide ou tanh são sensíveis à escala dos dados. Nesses casos, a normalização garante que as características com valores maiores não dominem o aprendizado, levando a um modelo mais preciso e robusto.
2. **Variáveis com Escalas Diferentes:** se as variáveis em seu dataset possuem escalas significativamente diferentes, como idade em anos e renda mensal, a normalização garante que nenhuma variável domine o processo de aprendizado devido à sua magnitude. Isso torna o modelo mais justo e evita vieses na análise.
3. **Melhoria da Convergência:** em alguns casos, a normalização pode auxiliar na convergência dos algoritmos de otimização, especialmente em redes neurais artificiais. Ao normalizar os dados, os gradientes utilizados na atualização dos pesos da rede ficam em uma escala similar, facilitando o processo de aprendizado e otimização.

Quando a normalização pode não ser necessária?

1. **Algoritmos Invariantes à Escala:** alguns algoritmos, como Árvores de Decisão e Random Forest, são invariantes à escala dos dados, ou seja, seu desempenho não é afetado pela normalização. Nesses casos, a normalização pode ser dispensável, dependendo da análise e das características do dataset.
2. **Dados Já Normalizados:** se os dados já estiverem em uma escala natural e compreensível para o algoritmo, como probabilidades entre 0 e 1 ou valores angulares em graus, a normalização pode ser desnecessária.
3. **Perda de Informação:** a normalização pode levar à perda de certa informação sobre a escala original dos dados. Em alguns casos, essa perda de informação pode ser irrelevante para o problema em questão, mas em outros pode ser crucial para a interpretação dos resultados.

```
from sklearn.preprocessing import normalize

# Dados a serem normalizados/padronizados
dados = np.array([2, 3, 5, 6, 7, 4, 8, 7, 6])

# Normalização entre 0 e 1
dados_normalizados = normalize([dados], norm='max')

print(f"Dados normalizados: {dados_normalizados}")
Dados normalizados:
[[0.25  0.375 0.625 0.75  0.875 0.5   1.    0.875 0.75 ]]
```

Código-fonte 7 – Normalização dos dados de uma lista
Fonte: Elaborado pelo autor (2024)

```
from sklearn.preprocessing import StandardScaler

dados = np.array([[1, 2, 3], [4, 5, 6],[7, 8, 9]])

# Padronização com média 0 e desvio padrão 1
padronizador = StandardScaler()
dados_padronizados = padronizador.fit_transform(dados)

print(f"Dados padronizados: {dados_padronizados}")
[[-1.22474487 -1.22474487 -1.22474487]
 [ 0.          0.          0.        ]
 [ 1.22474487  1.22474487  1.22474487]]
```

Código-fonte 8 – Padronização dos dados
Fonte: Elaborado pelo autor (2024)

Codificação de Dados Categóricos: transformando variáveis categóricas em valores numéricos, sejam features ou classes, para que algoritmos de aprendizado de máquina possam processá-las.

- **Codificação binária:** cria uma nova variável binária para cada categoria.
- **Codificação one-hot encoding:** cria uma nova variável para cada categoria, com valor 1 para a categoria presente e 0 para todas as demais.
- **Codificação de label encoding:** atribui um valor numérico inteiro para cada categoria.

```
from sklearn.preprocessing import OneHotEncoder, LabelEncoder

# Dados categóricos
dados_categóricos = np.array(['A', 'B', 'C', 'A', 'B'])

# Codificação one-hot encoding
codific_one_hot = OneHotEncoder()
```

```
dados_one_hot=codific_one_hot.fit_transform(dados_categóricos.
reshape(-1, 1))

print(f"Dados one-hot encoding: {dados_one_hot.toarray()}")
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]]

# Codificação de label encoding
codificador_label = LabelEncoder()
dados_label=codificador_label.fit_transform(dados_categóricos)

print(f"Dados label encoding: {dados_label}")
[0 1 2 0 1]
```

Código-fonte 9 – Codificação com one-hot encoding e label encoding
Fonte: Elaborado pelo autor (2024)

No one-hot encoding, podemos ver que foram criadas 3 colunas, uma para cada classe (classes A, B e C), nesta ordem. Na variável `dados_categóricos`, o primeiro elemento é da classe A, então atribui 1 para a primeira coluna e zero para as demais ([1. 0. 0.]). O segundo elemento é da classe B, então atribui 1 para a segunda coluna e zero para as demais ([0. 1. 0.]) e assim por diante.

No label encoding foi atribuído um valor numérico para cada classe. As classes A, B e C ficaram com os valores 0, 1 e 2, respectivamente.

Avaliação de Modelos: Medindo o Desempenho

A avaliação de modelos é crucial para determinar a qualidade e a confiabilidade do modelo treinado. Há uma frase famosa de Peter Drucker, consultor de gestão, educador e autor austríaco-americano: "Se você não pode medir, você não pode gerenciar".

Esta avaliação é feita mediante uma mensuração (medição). Vamos explorar algumas das métricas mais importantes:

Métricas para Problemas de Classificação:

- **Acurácia:** porcentagem de previsões corretas. Também conhecida como taxa de acerto, é a métrica mais simples e intuitiva para avaliar o desempenho de um modelo de classificação. Ela representa a proporção de

previsões corretas feitas pelo modelo em relação ao **total de amostras no conjunto de teste.**

- **Cálculo da Acurácia:** a acurácia é calculada pela seguinte fórmula:

$$\text{Acurácia} = (\text{Número de previsões corretas}) \,/\, (\text{Número total de amostras})$$

- **Interpretação da Acurácia:**
  - **Alto valor de acurácia:** indica que o modelo está fazendo um bom trabalho em geral, classificando a maioria das amostras corretamente.
  - **Baixo valor de acurácia:** indica que o modelo está com dificuldades em classificar as amostras corretamente, necessitando de ajustes ou de um algoritmo mais adequado.
- **Limitações da Acurácia:**
  - **Falta de detalhes:** a acurácia não fornece informações sobre como o modelo está errando. Ela não distingue entre diferentes tipos de erros, como classificar incorretamente um positivo como negativo ou vice-versa.
  - **Sensibilidade à distribuição das classes:** em datasets com classes desbalanceadas, ou seja, quando uma classe possui muito mais amostras que a outra, a acurácia pode ser enganosa. Um modelo que sempre prevê a classe majoritária terá uma alta acurácia, mesmo que não esteja classificando corretamente as amostras da classe minoritária.

Precisão: relevância das previsões positivas. A precisão, também conhecida como valor preditivo positivo (VP), reflete a proporção de classificações positivas feitas pelo modelo que foram acertadas, ou seja, a confiabilidade das previsões positivas do modelo.

- **Cálculo da Precisão:** a precisão é calculada pela seguinte fórmula:

$$\text{Precisão} = (\text{Verdadeiros Positivos}) \,/\, (\text{Verdadeiros Positivos} + \text{Falsos Positivos})$$

- **Interpretação da Precisão:**

- **Alto valor de precisão:** indica que o modelo está fazendo previsões positivas confiáveis, ou seja, a maioria das amostras que ele classifica como positivas realmente são positivas.
- **Baixo valor de precisão:** indica que o modelo está fazendo muitas previsões positivas incorretas, classificando como positivas amostras que na verdade são negativas.

- **Aplicações da Precisão:**
  - **Cenários com custos altos para falsos positivos:** em cenários nos quais os custos de um falso positivo são altos, como em diagnósticos médicos ou detecção de fraudes, a precisão é uma métrica crucial. Um modelo com alta precisão garante que a maioria das previsões positivas sejam verdadeiras, minimizando os custos e riscos associados aos falsos positivos.

Revocação: abrangência das previsões positivas.

- A revocação, também conhecida como sensibilidade ou recall, reflete a proporção de resultados positivos reais que foram identificados pelo modelo. É uma métrica que indica, entre as amostras positivas existentes, quantas o modelo conseguiu classificar corretamente.
- **Cálculo da Revocação:** a revocação é calculada pela seguinte fórmula:

$$\text{Revocação} = (\text{Verdadeiros Positivos}) / (\text{Verdadeiros Positivos} + \text{Falsos Negativos})$$

- Interpretação da Revocação:
  - **Alto valor de revocação:** indica que o modelo está identificando a maioria dos resultados positivos reais, não deixando escapar muitos casos positivos.
  - **Baixo valor de revocação:** indica que o modelo está perdendo muitos resultados positivos reais, classificando-os como negativos.
- Aplicações da Revocação:
  - **Cenários com custos altos para falsos negativos:** em cenários nos quais os custos de um falso negativo são altos, como na detecção de doenças ou na identificação de criminosos, a revocação é uma métrica crucial. Um modelo com alta revocação garante que a maioria dos casos

positivos seja identificada, minimizando os custos e riscos associados aos falsos negativos.

F1-Score: a Métrica de equilíbrio entre precisão e revocação.

- Métricas como acurácia, precisão e revocação fornecem informações valiosas sobre diferentes aspectos do desempenho do modelo, mas nem sempre oferecem uma visão completa. É nesse contexto que a métrica **F1-Score** se destaca, **combinando** as informações de **precisão** e **revocação** em um único valor, proporcionando uma avaliação mais equilibrada e informativa.
- **Compreendendo o F1-Score:** o F1-Score, também conhecido como F1 medida ou F1 pontuação, é uma média harmônica ponderada entre a precisão e a revocação representada pela seguinte fórmula:

$$\text{F1Score} = 2 * (\text{Precisão} * \text{Revocação}) / (\text{Precisão} + \text{Revocação})$$

- **Interpretando o F1-Score:**
  - **Valores entre 0 e 1:** o F1-Score varia entre 0 e 1, em que 1 representa o melhor desempenho possível (modelo perfeito) e 0 indica o pior desempenho (modelo que sempre erra).
  - **Interpretação intuitiva:** um valor alto de F1-Score indica que o modelo está obtendo um bom equilíbrio entre precisão e revocação, classificando corretamente a maioria das amostras positivas e negativas.
  - **Considerando custos e prioridades:** em cenários com custos distintos para falsos positivos e falsos negativos, o F1-Score pode ser ponderado para dar mais peso à métrica que reflete o custo mais alto, ajustando a importância relativa da precisão e da revocação na avaliação final.
- Vantagens do F1-Score:
  - **Equilíbrio entre Precisão e Revocação:** o F1-Score evita a dominância de uma métrica sobre a outra, fornecendo uma visão mais holística do desempenho do modelo.
  - **Útil em Cenários com Classes Desbalanceadas:** em datasets com classes desbalanceadas, em que uma classe possui muito mais

amostras que a outra, o F1-Score é menos sensível à distribuição das classes do que a acurácia, oferecendo uma avaliação mais justa do modelo.

- **Aplicações em Diversos Domínios:** o F1-Score é amplamente utilizado em diferentes áreas do aprendizado de máquina, como classificação de texto, análise de imagens e detecção de fraudes.

- Limitações do F1-Score:
  - **Dependência da Precisão e Revocação:** o F1-Score depende diretamente da precisão e da revocação; portanto, sua interpretação está condicionada à qualidade dessas métricas.
  - **Possibilidade de Conflitos:** em alguns casos, pode haver um conflito entre a precisão e a revocação, em que aumentar uma leva à diminuição da outra. Nesses casos, a escolha do F1-Score como métrica principal pode mascarar esse conflito.

Matriz de Confusão: em tarefas de classificação, esta matriz apresenta uma visão detalhada do desempenho de um modelo em relação às classes que ele precisa distinguir. Essa matriz tabular revela a quantidade de amostras de cada classe que foram classificadas corretamente e incorretamente pelo modelo, permitindo uma análise profunda dos erros e acertos dele.

		Predito	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Figura 1 – Matriz de Confusão
: Fonte: Elaborado pelo autor (2024)

1. Estrutura da Matriz de Confusão: a Matriz de Confusão é composta por linhas e colunas, cada uma representando uma classe. As linhas indicam a classe real das amostras, enquanto as colunas indicam a classe para a qual o modelo as classificou. De forma geral ela apresenta esta estrutura:

- **Diagonais principais:** representam as amostras que foram classificadas corretamente pelo modelo.

- **Elementos fora da diagonal principal:** representam as amostras que foram classificadas incorretamente pelo modelo.

2. Interpretando a Matriz de Confusão:

- **Verdadeiros Positivos (VP):** número de amostras da classe real que foram classificadas corretamente como positivas pelo modelo.
- **Verdadeiros Negativos (VN):** número de amostras da classe real que foram classificadas corretamente como negativas pelo modelo.
- **Falsos Positivos (FP):** número de amostras da classe real que foram classificadas incorretamente como positivas pelo modelo. Também chamado de Erro Tipo 1.
- **Falsos Negativos (FN):** número de amostras da classe real que foram classificadas incorretamente como negativas pelo modelo. Também chamado de Erro Tipo 2.

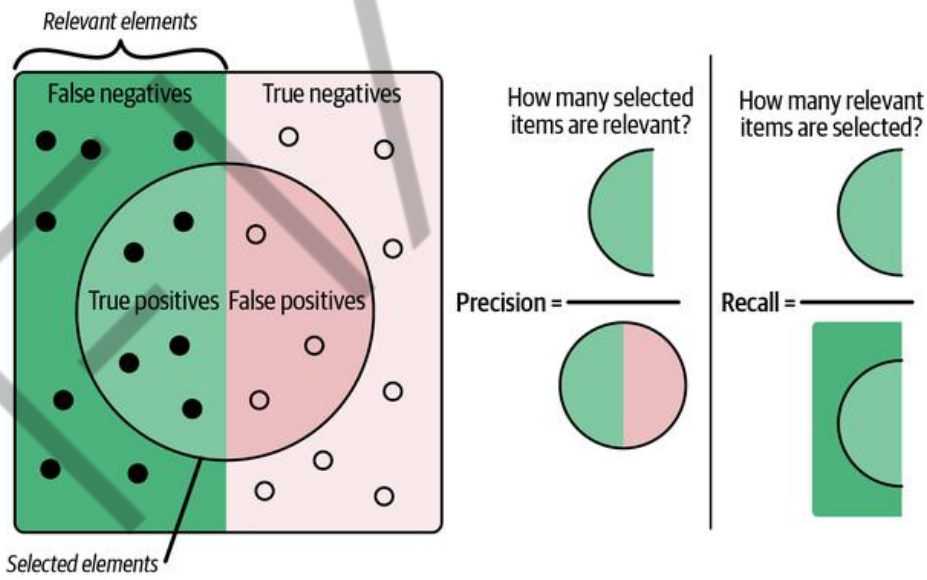

Figura 2 – Representação gráfica das métricas Precisão e Revocação (Recall)
Fonte: Moses et al. (2022)

3. Métricas Derivadas da Matriz de Confusão:

A partir da Matriz de Confusão, diversas métricas podem ser calculadas para avaliar o desempenho do modelo, como vimos anteriormente:

- **Acurácia:** $(\text{VP} + \text{VN}) / (\text{Total de amostras})$

- **Precisão:** $\text{VP} / (\text{VP} + \text{FP})$
- **Revocação:** $\text{VP} / (\text{VP} + \text{FN})$
- **F1-Score:** $2 * (\text{Precisão} * \text{Revocação}) / (\text{Precisão} + \text{Revocação})$

4. Aplicações da Matriz de Confusão:

- **Identificação de erros do modelo:** a Matriz de Confusão permite identificar quais classes o modelo está classificando com mais dificuldade, auxiliando na investigação das causas dos erros e na implementação de medidas corretivas.
- **Avaliação de modelos em diferentes classes:** em datasets com classes desbalanceadas, a Matriz de Confusão permite avaliar o desempenho do modelo individualmente para cada classe, identificando possíveis vieses ou dificuldades em classificar classes minoritárias.
- **Comparação de modelos:** a Matriz de Confusão pode ser utilizada para comparar o desempenho de diferentes modelos de classificação em um mesmo dataset, auxiliando na escolha do modelo mais adequado para o problema em questão.

Exemplo: imagine um modelo treinado para classificar cachorros (elemento de interesse) ou gatos. Ao processar uma imagem que contém 12 cães e 10 gatos, o programa identifica 8 cães. Dos 8 elementos identificados como cães, apenas 5 são realmente cães (verdadeiro positivo), enquanto os outros 3 são gatos (falso positivos). E 7 cães (12 - 5) foram 'perdidos', ou seja, não entraram na classificação correta (falso negativo). Da mesma forma que 7 gatos (10 - 3) foram corretamente excluídos (verdadeiro negativo). Repare que a diagonal principal da matriz foram as classificações corretas para cachorros e gatos.

A confusão deste modelo é:

	Cachorro (Predito)	**Gato (Predito)**	**Total**
Cachorro (Real)	VP = 5	FN = 7	12
Gato (Real)	FP = 3	VN = 7	10
Total	8	14	22

Tabela 1 – Exemplo de Matriz de confusão para um modelo de classificação de cachorros
Fonte: Elaborado pelo autor (2024)

E as métricas derivadas:

- **Acurácia** = (VP + VN) / (Total de amostras) = (5+7) / (12+10) = 12/22 = **54,5%**
- **Precisão** = VP / (VP + FP) = 5 / (5+3) = 5/8 = **62,5%**
- **Revocação** = VP / (VP + FN) = 5 / (5+7) = 5/12 = **41,67%**
- **F1-Score** = 2 * (Precisão * Revocação) / (Precisão + Revocação) = 2*((⅝)*(5/12)) / (⅝ + 5/12)) = **50%**

Você irá trabalhar mais estes conceitos realizando as atividades! Veja também as videoaulas para fixar o conteúdo.

Aplicações:

- Essa matriz indica que o modelo é mais preciso em identificar cachorros do que gatos.
- Analisando os erros (FP e FN), podemos investigar as causas das falhas do modelo, como imagens mal iluminadas ou animais em poses atípicas.
- Com base na análise da matriz, podemos ajustar o modelo para melhorar seu desempenho, como utilizando técnicas de pré-processamento de imagens ou treinando o modelo com mais dados.

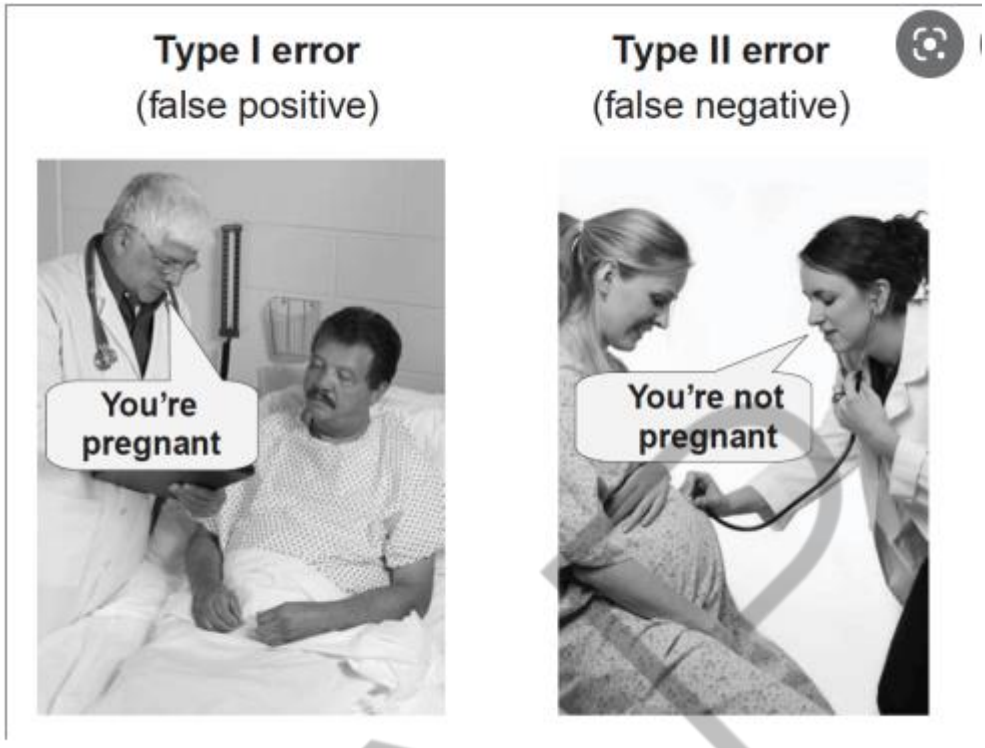

Figure 3.1 Type I and Type II errors

Figura 3 – Descrição dos erros Falso Positivo e Falso Negativo
Fonte: Ellis (2010)

Felizmente, as bibliotecas já deixaram o cálculo destas métricas todas prontas para nós. Veja como é simples obter estas métricas com a biblioteca Scikit-Learn. No exemplo do código-fonte 10, é importante frisar que as classes preditas (y_pred) devem ser obtidas pelo modelo. Aqui foi colocado desta forma para efeito de exemplificação.

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score

y_test = [1, 0, 1, 0]

# Previsões do modelo (código de obtenção das previsões do
modelo)
y_pred = [0, 1, 1, 1]

# Métricas de avaliação
print(f'Acurária: {accuracy_score(y_test, y_pred)}')
print(f'Precisão: {precision_score(y_test, y_pred)}')
print(f'Revocação: {recall_score(y_test, y_pred)}')
print(f'F1-Score: {f1_score(y_test, y_pred)}')

Acurácia: 0.25
Precisão: 0.3333
```

```
Revocação: 0.5
F1-Score: 0.4
```

Código-fonte 10 – Cálculos das métricas acurácia, precisão, revocação e f1-score com a biblioteca scikit-learn
Fonte: Elaborado pelo autor (2024)

Em resumo, a Matriz de Confusão é uma ferramenta essencial para avaliar o desempenho de modelos de classificação em diferentes cenários. Ao fornecer uma visão detalhada dos acertos e erros do modelo em relação às classes, a Matriz de Confusão permite identificar problemas, comparar modelos e tomar decisões informadas para aprimorar a performance do modelo de aprendizado de máquina.

MERCADO, CASES E TENDÊNCIAS

Saiba porque não deveríamos dizer 'obrigado' ou 'bom dia' a ferramentas de IA como o ChatGPT: embora especialistas vejam pouco valor em ser gentil com entidades não conscientes, outros argumentam que isso pode influenciar na qualidade das respostas. Saiba mais aqui.

A Microsoft apresentou sua nova inteligência artificial VASA-1, capaz de criar vídeos ultrarrealistas a partir de uma única foto e um arquivo de áudio. Veja aqui.

O QUE VOCÊ VIU NESTA AULA?

Nesta aula exploramos duas etapas cruciais no aprendizado supervisionado: o pré-processamento de dados e as métricas de desempenho de modelos. Vimos diversas técnicas para pré-processar os dados, como lidar com valores ausentes e outliers, transformação dos dados com normalização e padronização, bem como codificar os dados categóricos.

Vimos também as formulações de algumas métricas para tarefas de classificação (acurácia, precisão, revocação e f1-score) e regressão (MSE e RMSE) e como calculá-las com o auxílio de bibliotecas.

Ao dominar o pré-processamento de dados e as métricas de desempenho, você estará pronto(a) para construir modelos de aprendizado supervisionado mais robustos, precisos e confiáveis.

REFERÊNCIAS

ELLIS, P. D. **The essential guide to effect sizes:** statistical power, meta-analysis, and the interpretation of research results. Cambridge: Cambridge University Press, 2010.

MOSES, B.; GAVISH, L.; VORWERCK, M. **Data quality fundamentals:** a practitioner's guide to building trustworthy data pipelines. Sebastopol: O'Reilly Media, 2022.

NUMPY. **Numpy.delete — NumPy v1.26 manual**. 2024. Disponível em: <https://numpy.org/doc/stable/reference/generated/numpy.delete.html>. Acesso em: 04 jul. 2024.

NUMPY. **Numpy.Mean — NumPy v1.26 manual**. 2024. Disponível em: <https://numpy.org/doc/stable/reference/generated/numpy.mean.html>. Acesso em: 04 jul. 2024.

NUMPY. **Numpy.Percentile — NumPy v1.26 manual**. 2024. Disponível em: <https://numpy.org/doc/stable/reference/generated/numpy.percentile.html>. Acesso em: 04 jul. 2024.

SCIKIT-LEARN. **Precision-Recall**. 2024. Disponível em: <https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html>. Acesso em: 04 jul. 2024.

SCIKIT-LEARN. **Sklearn.Metrics.Confusion_matrix**. 2024. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html>. Acesso em: 04 jul. 2024.

SCIKIT-LEARN. **Sklearn.Preprocessing.LabelEncoder**. 2024. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html>. Acesso em: 04 jul. 2024..

SCIKIT-LEARN. **Sklearn.Preprocessing.OneHotEncoder**. 2024. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html>. Acesso em: 04 jul. 2024.

PALAVRAS-CHAVE

Palavras-chave: Pré-processamento. Limpeza dos dados. Transformações dos dados. Normalização e Padronização dos dados. Tratamento de dados ausentes. Seleção de features. Métricas de avaliação. Acurácia. Precisão. Revocação. F1-Score. Matriz de Confusão. Erro médio quadrático. Raiz quadrada média do erro. Pandas. Numpy. Scikit-Learn. Python.