PYTHON

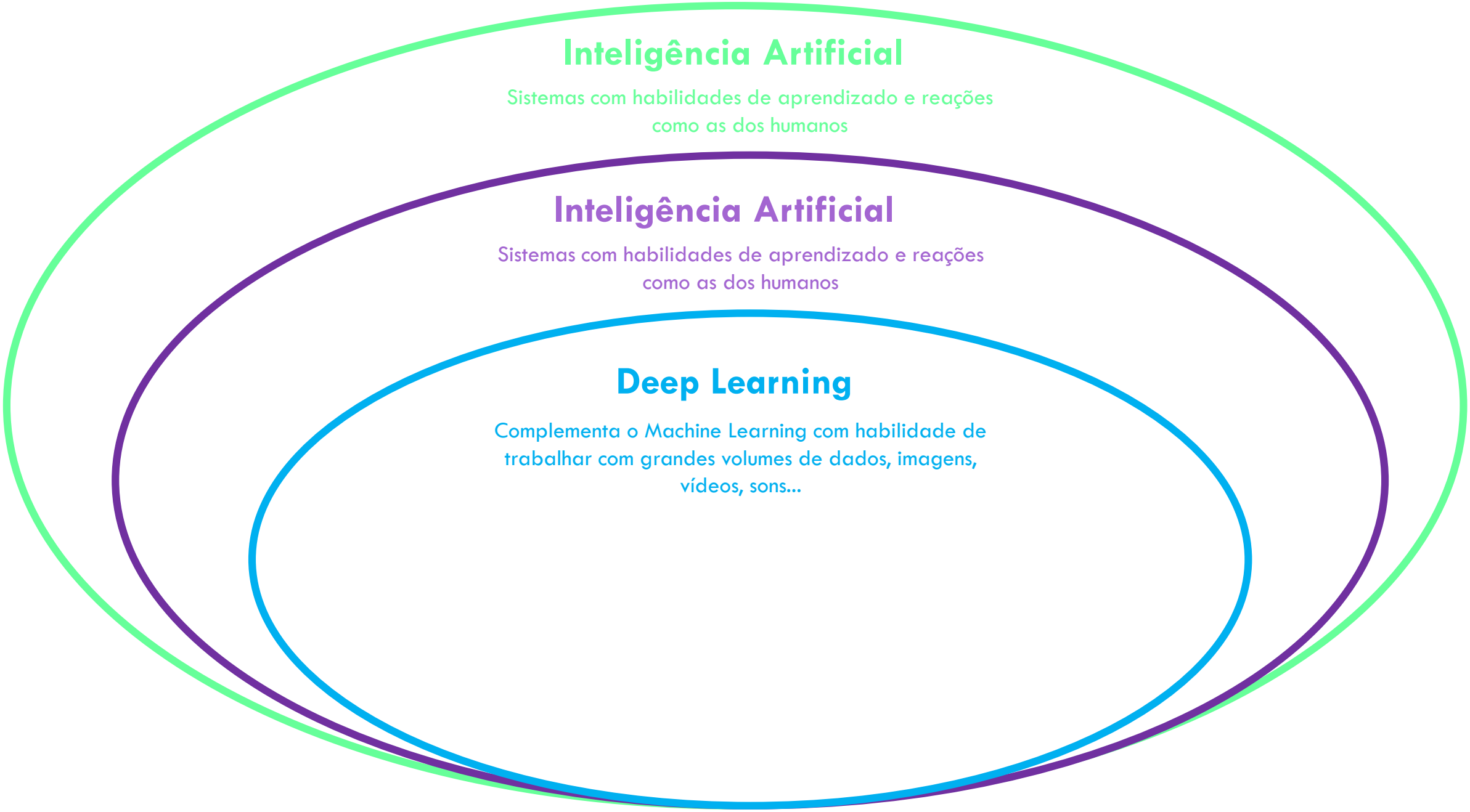
Professor: Eduardo Inocencio

SUMÁRIO

1. O que é Machine Learning
2. Modelos Supervisionados e Não Supervisionados
3. Biblioteca Scikit-Learning
4. Treino, teste e validação
5. Principais modelos
6. Regressão linear
7. Modelo de classificação
8. Modelo de clusterização

MACHINE LEARNING – O QUE É?

- **Machine Learning (ML),** ou Aprendizagem de Máquina, é um método computacional que utiliza algoritmos e métodos estatísticos para analisar conjuntos de dados, descobrir padrões e realizar definições ou predições.
- **ML** é uma sub-área de Inteligência Artificial.


Inteligência Artificial
Sistemas com habilidades de aprendizado e reações como as dos humanos
Inteligência Artificial
Sistemas com habilidades de aprendizado e reações como as dos humanos
Deep Learning
Complementa o Machine Learning com habilidade de trabalhar com grandes volumes de dados, imagens, vídeos, sons...

MACHINE LEARNING – O QUE É?

Muitos avanços estão sendo realizados na Ciência e Tecnologia devido ao desenvolvimento de novos métodos de análise utilizando aprendizado de máquina.

ML tem sido usado em diversas áreas, tanto em pesquisas acadêmicas (Medicina, Biologia, Astronomia, Ciências Sociais, ...) quanto no mundo corporativo (em prevenção à fraudes em bancos, estratificação de população em sistemas de saúde, sistemas de recomendação em e-commerce, ...)

APLICAÇÕES EM DIFERENTES ÁREAS

Sistemas Financeiros: Prevenção de fraudes e geração de insights.

Saúde: Identificação de tendências, confirmações e novidades para diagnósticos e tratamentos. Descoberta de novos medicamentos.

Marketing e Vendas: Recomendações de produtos e serviços através de consultas e compras anteriores.

Ciência: Eficiência na análise de dados e resultados de estudos científicos.

APLICAÇÕES EM DIFERENTES ÁREAS

Indústria: Descoberta de novos materiais, técnicas de fabricação, prevenção de falhas, aperfeiçoamento na produção ...

Governo: Geração de insights, análises socioeconômicas, detecção de fraudes ...

Transporte: Identificação de padrões e tendências para rotas de transporte.

MACHINE LEARNING – ONDE ESTÁ SENDO USADA?

MACHINE LEARNING – ONDE ESTÁ SENDO USADA?

MACHINE LEARNING – ONDE ESTÁ SENDO USADA?

Google vai usar inteligência artificial para detectar vírus em e-mails

Sistema tem capacidade para verificar 300 bilhões de anexos por semana

Pesquisadores holandeses usam inteligência artificial para prever o comportamento de asteroides

Inteligência artificial é utilizada na operação de rebocadores no Porto

Saúde

Nova tecnolo

Inteligência artificial descobre novo tipo de antibiótico

A halicina se mostrou capaz de combater bactérias consideradas super resistentes pela OMS.

Machine learning picks out hidden vibrations from earthquake data

Technique may help scientists more accurately map vast underground geologic structures.

Inteligência artificial vai identificar futuros criminosos no Reino Unido

Plataforma desenvolvida pela polícia de West Midlands custou cerca de 55 milhões de reais e pode trazer segurança aos cidadãos

Machine learning predicts behavior of biological circuits

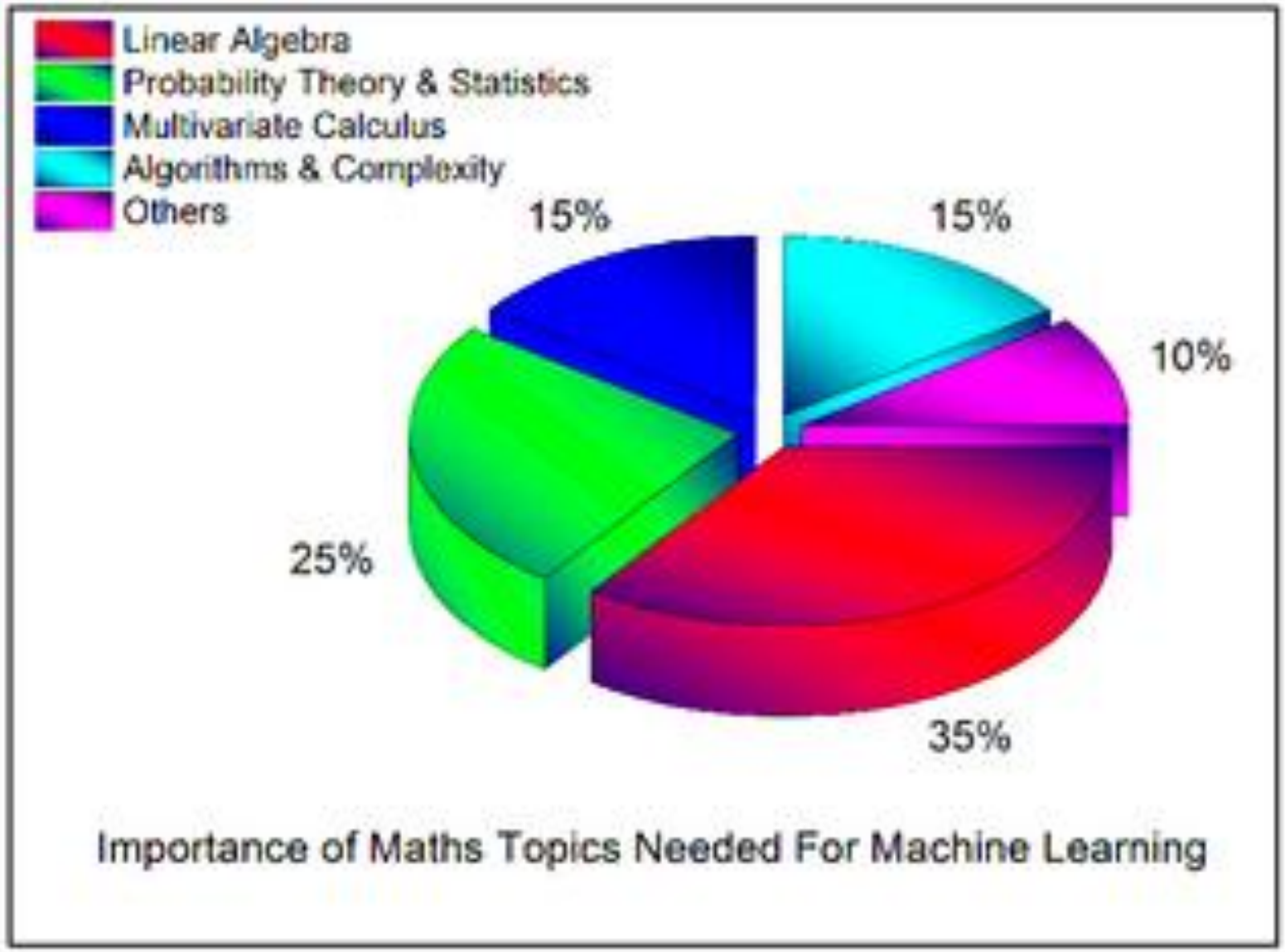
Neural networks cut modeling times of complex biological circuits to enable new insights into their inner workings

Startup reduz desperdício em bares e restaurantes com ajuda de inteligência artificial

A empresa entrega para o cliente o levantamento e indicações do que pode estar gerando as perdas. Faz também um relatório para gerenciar os processos na prática.

A MATEMÁTICA DO MACHINE LEARNING

1. **Estatística:** Descritiva, Probabilística, Bayesiana e Regressões.
2. **Álgebra Linear:** Vetores e Matrizes, Sistemas Lineares, Estimativa dos mínimos quadrados, transformação linear, auto vetores e autovalores.
3. **Cálculo Multivariado:** Funções de várias variáveis, Derivadas Parciais, Integrais Múltiplas, Equações Diferenciais, Geometria Multivariada, Vetor Direcional e Gradiente.



https://towardsdatascience.com/the-mathematics-of-machine-learning

LINGUAGENS DE PROGRAMAÇÃO UTILIZADAS

FORMAS DE APRENDIZAGEM DE MÁQUINA

Supervisionada: Interação de um agente externo. O algoritmo possui dados de entrada e de saída para treinamento (Ex.: Análise de crédito).

Não Supervisionada: Tipo de aprendizagem auto-organização. Não existe uma resposta ou modelo de referência para treinar o algoritmo (Ex.: Associação ou agrupamento de produtos com similaridades).

Aprendizagem por Reforço: Recebe informações do ambiente, que indica o erro, mas não a forma de melhorar a ação e o desempenho.

O conjunto de dados muda a todo instante, demandando contínuo processo de adaptação (Ex.: movimentação de robôs).


Supervisionada
Classificação
Regressão
Variável resposta é discreta?
Previsão ou predição de um valor numérico.

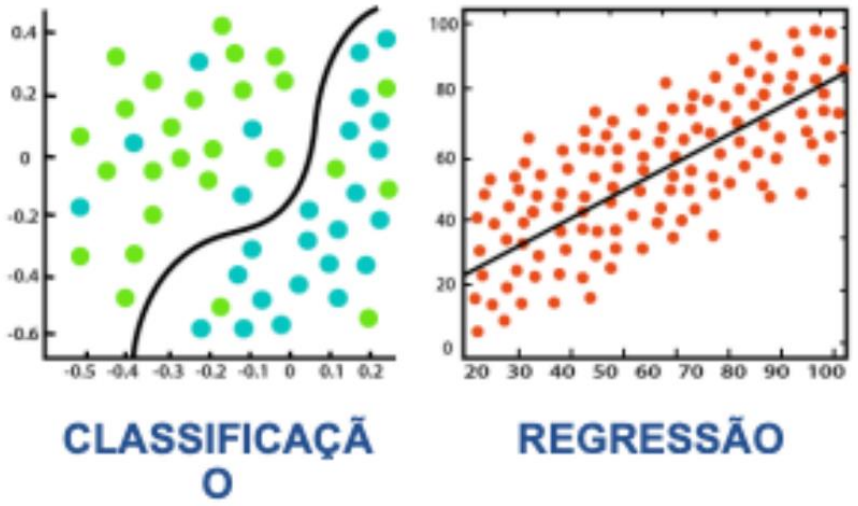

NÃO SUPERVISIONADA
Agrupamento (Clustering)
Associação
Redução de dimensionalidade
Agrupar dados por características (idade, gostos, ...)
Obter regras com os dados (pais com crianças compram mais doces).
Escolha das melhores variáveis para otimizar o processo.

MODELOS SUPERVISIONADOS

No aprendizado supervisionado a classe (variável) que queremos predizer já foi rotulada nos dados que serão utilizados para treinar o modelo.

O aprendizado supervisionado pode ser utilizado para:

- Predição de uma categoria (classificação)
- Predição de um número (regressão)



MODELOS SUPERVISIONADOS

Principais algoritmos de classificação/regressão:

- Naive Bayes
- Árvore de decisão
- Random Forest
- K-Nearest Neighbours (KNN)
- Support Vector Machine (SVM)

MODELOS NÃO SUPERVISIONADOS

No aprendizado não supervisionado, não temos previamente os rótulos nos dados, de modo que o algoritmo irá procurar por padrões no conjunto de dados e agrupá-los segundo esses padrões.

O aprendizado não supervisionado pode ser utilizado para:

Criar grupos similares (Clusterização)

Diminuir número de dimensões em um conjunto de dados.

MODELOS NÃO SUPERVISIONADOS

Principais algoritmos Não Supervisionados:

K-means (agrupamento)

DBSCAN (agrupamento)

Isolation Forest (detecção de anomalia)

Auto Encoder (redes neurais)

MACHINE LEARNING – CRIANDO MODELOS

1. **Pense** para quem e porquê você está criando o modelo.
2. **Escolha** a pergunta que estamos tentando responder.
3. **Selecione** o conjunto de dados para responder esta pergunta.
4. **Identifique** como medir o resultado.

Para todas as anteriores é preciso de conversa e entendimento.

ENTENDIMENTO DO NEGÓCIO: 4P'S

Para analisar dados e modelá-los e é importante que estejamos sempre alinhados com as necessidades do negócio e/ou problema a ser resolvido.

Para isso, podemos utilizar a estratégia dos 4P's.

Problema – Qual a dor do negócio que seu estudo vai endereçar?

Potencial – Qual o potencial ganho que se obterá com o projeto?

Produto – Qual será o produto que você entregará? Qual modelo?

Proposta – Como seu modelo irá resolver a dor inicial?

MACHINE LEARNING – BOAS PRÁTICAS

1. Simplesmente comece!
2. Defina o problema de negócio explicitamente.
3. Estabeleça as métricas de sucesso (acurácia, precisão, recall...)
4. Colete os dados
5. Conheça os seus dados, explore e tire insights
6. Crie novas variáveis (*features*)
7. Escolha o algoritmo mais adequado para os seus dados e para seu projeto
8. Considere os problemas possíveis e teste antes do lançamento
9. Implante e automatize

CICLO DE PROCESSOS

CICLO DE PROCESSOS

Pré-processamento dos dados: verificação de dados faltantes, imputação de valores, interpolação, criação de novas variáveis, normalização.

Treinamento: separação de amostra de teste e treino, aplicação de modelos.

Avaliação: o modelo teve boa performance? Ele resolveu o problema?

AMOSTRAS DE TREINO E TESTE

Quando trabalhamos com modelos supervisionados precisamos sempre separar nossas amostras de Treino e Teste, para evitar *overfitting*.

- **Amostra de Treino**: amostra utilizada para ensinar o modelo como fazer a classificação/ regressão.
- **Amostra de Teste:** amostra na qual aplicamos o modelo já treinado, conferindo a qualidade do modelo obtido.

SCIKIT-LEARN

A biblioteca Scikit-learn é uma das principais bibliotecas de machine learning em Python.

Com ela podemos desenvolver projetos com modelagem estatística e modelos de classificação, regressão, agrupamentos e redução de dimensionalidade.

Iremos abordar brevemente dois modelos: regressão linear simples, e modelo de classificação.

SCIKIT-LEARN DATASETS

» Similar ao Seaborn, o Scikit-Learn também possui datasets embutidos que podem ser úteis para estudar as funcionalidades da biblioteca.

`load_boston([return_X_y])`	Load and return the boston house-prices dataset (regression).
`load_iris([return_X_y])`	Load and return the iris dataset (classification).
`load_diabetes([return_X_y])`	Load and return the diabetes dataset (regression).
`load_digits([n_class, return_X_y])`	Load and return the digits dataset (classification).
`load_linnerud([return_X_y])`	Load and return the linnerud dataset (multivariate regression).
`load_wine([return_X_y])`	Load and return the wine dataset (classification).
`load_breast_cancer([return_X_y])`	Load and return the breast cancer wisconsin dataset (classification).

```
from sklearn.datasets import load_wine

wine = load_wine(return_X_y=True)
```

```
from sklearn.datasets import load_boston

boston = load_boston(return_X_y=True)
```

» Os dados retornados serão do tipo array (Numpy) e terão um metadados se não utilizarmos o argumento return_X_y=True.

SCIKIT – LEARN TREINO E TESTE

» O Sklearn possui uma função própria para fazer a separação entre amostra de treino e teste: a função *train_test_split()*.

```
X,y = boston
```

Separando o dataset em variáveis **preditoras** (X) e variável **resposta** (y)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

```
len(X)
```

```
506
```

Confirmando o tamanho do conjunto de dados inteiro e das amostra de treino e teste.

```
len(X_train)
```

```
379
```

```
len(X_test)
```

```
127
```

SCIKIT – LEARN TREINO E TESTE

- A função train_test_split() separa, por default, 25% dos dados para teste.
- Para modificar a proporção, usamos o argumento test_size.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=0)
```

```
print('Treino: %d'%len(X_train))
print('Teste: %d'%len(X_test))
```

```
Treino: 354
Teste: 152
```

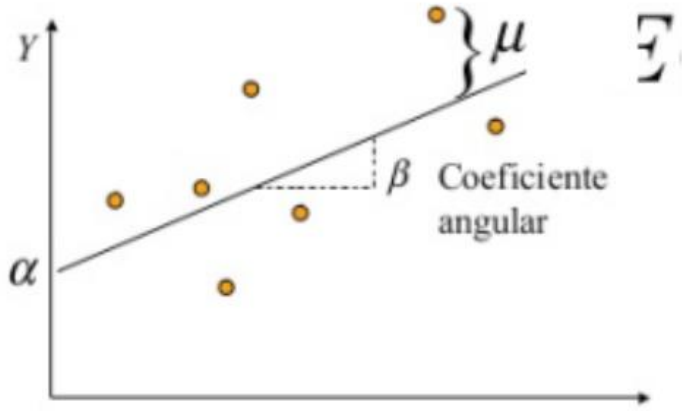
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
                                                    random_state=0)
```

```
print('Treino: %d'%len(X_train))
print('Teste: %d'%len(X_test))
```

```
Treino: 303
Teste: 203
```

REGRESSÃO LINEAR

» A regressão linear é um modelo matemático em que tentamos predizer um valor y (variável resposta) baseado em diversas outras variáveis X (variáveis preditoras).



$$y = \alpha + \beta X + \mu$$

y: variável de interesse
X: variáveis preditoras
α: interceptação do eixo y
β: inclinação da reta ajustada
μ: erro da predição

REGRESSÃO LINEAR

» O dataset Boston possui 13 variáveis preditoras e uma variável resposta (preço das casas em Boston).

1. CRIM - per capita crime rate by town
2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS - proportion of non-retail business acres per town.
4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
5. NOX - nitric oxides concentration (parts per 10 million)
6. RM - average number of rooms per dwelling
7. AGE - proportion of owner-occupied units built prior to 1940
8. DIS - weighted distances to five Boston employment centres
9. RAD - index of accessibility to radial highways
10. TAX - full-value property-tax rate per $10,000
11. PTRATIO - pupil-teacher ratio by town
12. B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT - % lower status of the population
14. MEDV - Median value of owner-occupied homes in $1000's

» Iremos treinar um modelo de regressão que envontre o valor de MEDV baseado nas 13 variáveis preditoras.

REGRESSÃO LINEAR

» Na biblioteca Scikit-Learn há diversos modelos para regressão linear: LinearRegression, Ridge, RidgeCV, SGDRegressor e outros.

» Para o nosso estudo iremos utilizar o modelo LinearRegression.

```
from sklearn.linear_model import LinearRegression
```

» Iniciando o modelo de regressão:

```
model = LinearRegression()
```

REGRESSÃO LINEAR

» Para treinar o modelo, aplicamos a função fit() aos dados X e y.

```
model.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

» Para predizer os valores de y utilizando já o modelo treinado, aplicamos o modelo aos dados de teste.

```
predito = model.predict(X_test)
```

```
predito
```

```
array([24.58155243, 24.51629253, 29.71379915, 12.51132696, 21.34965428,
       19.05443022, 20.94614567, 20.95753329, 19.54644456, 20.53025981,
        6.96153725, 17.1707288 , 16.85608802,  5.74921859, 40.74378524,
       32.62964196, 22.88997064, 37.11387241, 30.94054261, 23.12796161,
```

REGRESSÃO LINEAR

» Agora que já treinamos e aplicamos o modelo aos dados de teste, podemos verificar qual os coeficientes da reta ajustada.

» coef_ é o valor de b que multiplica cada uma das variáveis preditoras.

» intercept_ é o valor onde reta intercepta o eixo y.

```
model.coef_
```

```
array([-1.03747356e-01,  5.58589924e-02,  5.88240770e-02,  2.50523544e+00,
       -1.90284888e+01,  3.25353601e+00, -3.22150522e-03, -1.57603462e+00,
        2.58716068e-01, -1.14681299e-02, -1.10777478e+00,  5.50051783e-03,
       -5.59569992e-01])
```

```
model.intercept_
```

```
45.481419593250976
```

REGRESSÃO LINEAR

» Agora que já treinamos e aplicamos o modelo aos dados de teste, podemos verificar qual os coeficientes da reta ajustada.
» coef_ é o valor de b que multiplica cada uma das variáveis preditoras.
» intercept_ é o valor onde reta intercepta o eixo y.

```
model.coef_
```

```
array([-1.03747356e-01,  5.58589924e-02,  5.88240770e-02,  2.50523544e+00,
       -1.90284888e+01,  3.25353601e+00, -3.22150522e-03, -1.57603462e+00,
        2.58716068e-01, -1.14681299e-02, -1.10777478e+00,  5.50051783e-03,
       -5.59569992e-01])
```

```
model.intercept_
```

```
45.481419593250976
```

REGRESSÃO LINEAR

» Para sabermos quão boa foi nossa estimativa, podemos utilizar a função r2_score().
» Quanto mais próximo de 1 o r2_score for, mais precisa está sendo nossa predição.

```
from sklearn.metrics import r2_score
```

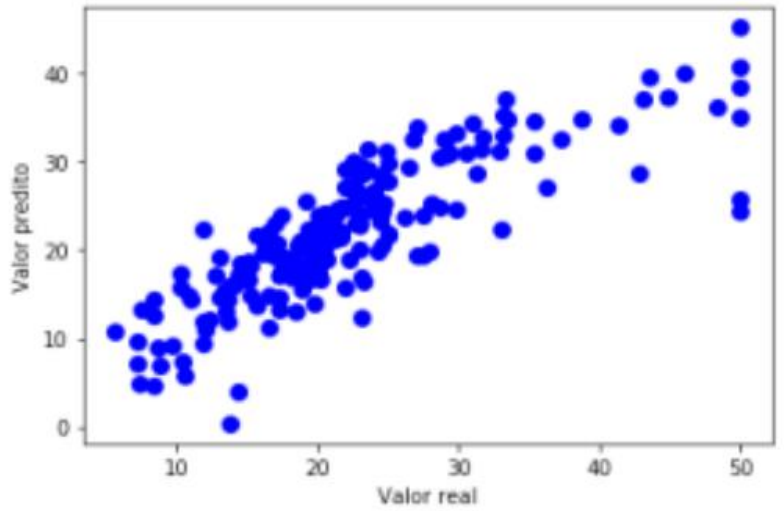
```
r2_score(y_test, predito)
```

```
0.6882607142538016
```

REGRESSÃO LINEAR

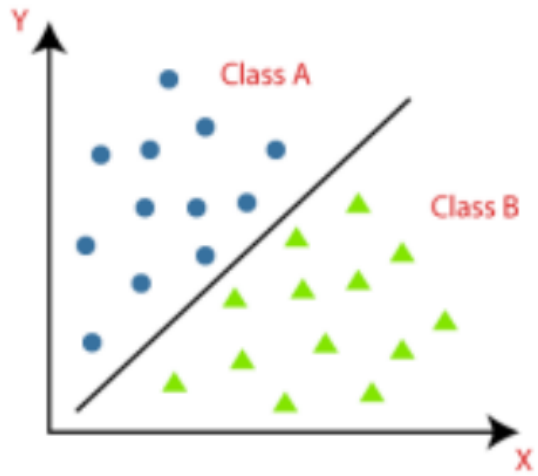
» Podemos ainda visualizar a relação entre os valores reais e os valores preditos:

```
plt.scatter(y_test, predito, color='blue', linewidth=3)
plt.xlabel('Valor real')
plt.ylabel('Valor predito')
```

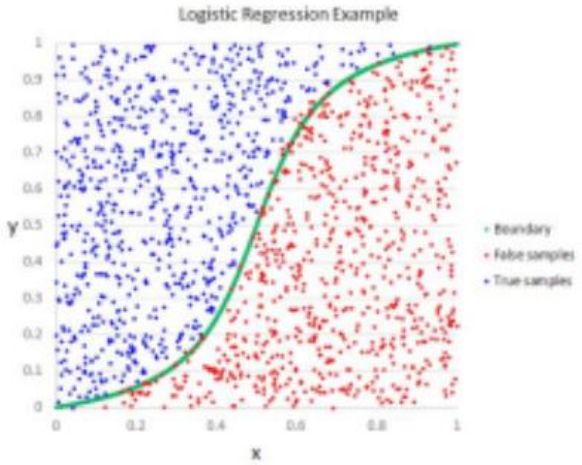


CLASSIFICAÇÃO

- Algoritmos de classificação são muito úteis em diversas tarefas:
  - Separar estrelas e Galáxias
  - Aprovação de crédito
  - Classificar diagnósticos
  - Separar tipos de flores
- Existem diversos tipos de algoritmos para classificação:
  - Regressão logística
  - KNN
  - Naive Bayes
  - Árvores de decisão
  - Redes neurais
- Iremos utilizar trabalhar o modelo de Regressão Logística.



REGRESSÃO LINEAR



» É um modelo de classificação muito comum.

» Não é um algoritmo de regressão.

» É usado para estimar valores discretos (em geral, valores binários) com base em determinado conjunto de variáveis independentes.

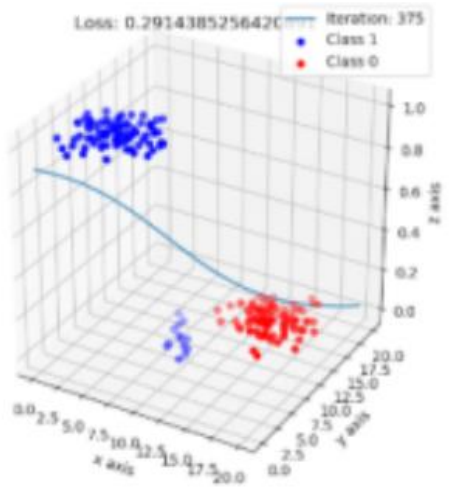
» A Regressão Logística prevê a probabilidade de ocorrência de um evento, ajustando os dados a uma função logit.

$$f(x) = log(\frac{x}{1-x})$$

REGRESSÃO LOGÍSTICA

» O modelo de regressão logística irá sempre retornar uma probabilidade de um certo dado pertencer (ou não) à uma dada classe.

» Dentre os casos de uso, podemos citar:
- » Detecção de spams
- » Predição de Diabetes
- » Cálculo de churn
- » Se um cliente irá clicar ou não em uma propaganda
- » Estudos de mercado



REGRESSÃO LOGÍSTICA

- Existem três tipos de Regressão Logística:
  - Regressão logística binária: usada em classificações binárias (ex.: spam e não-spam, sim e não).
  - Regressão logística multinomial: três ou mais categorias em que não há ordem nas categorias (ex.: classificação de vinhos).
  - Regressão logística ordinária: três ou mais categorias onde a ordem da classificação importa (ex.: as notas de um restaurante.)

REGRESSÃO LOGÍSTICA

» Na biblioteca do Scikit-Learn temos o modelo LogisticRegression para a modelagem de regressão logística.

```
from sklearn.linear_model import LogisticRegression
```

» Nesse exemplo usaremos o dataset clássico Iris.

```
from sklearn.datasets import load_iris
iris = load_iris(return_X_y=True)
```

» Iremos classificar as espécies das flores em 3 categorias: Iris setosa, Iris virginica e Iris versicolor.

REGRESSÃO LOGÍSTICA

» Separando as variáveis preditoras (X) da variável resposta (y), que é a classificação que queremos obter.

```
X,y = iris
```

» Separamos nossas amostras de treino e teste

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=0)
```

» Note que deixamos 30% para a amostra de teste.

REGRESSÃO LOGÍSTICA

» Criando o modelo de classificação

```
clf = LogisticRegression()
```

» Treinando o modelo de classificação na amostra de treino

```
clf.fit(X_train,y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

REGRESSÃO LOGÍSTICA

» Aplicando agora o modelo treinado nos dados de teste

```
predito = clf.predict(X_test)
```

```
predito
```

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 2, 1, 0, 0, 2, 2,
       0, 0, 2, 0, 0, 1, 1, 0, 2, 2, 0, 2, 2, 2, 0, 2, 1, 1, 2, 0, 2, 0,
       0])
```

» Para verificar as probabilidades calculadas para cada item do dataset, usamos a função predict_proba().

```
clf.predict_proba(X_test)
```

Probabilidade de cada classe, para cada item do dataset.

```
array([[1.01514149e-03, 1.68979201e-01, 8.30005658e-01],
       [2.77325788e-02, 8.19480925e-01, 1.52786497e-01],
       [9.35029110e-01, 6.49599289e-02, 1.09613940e-05],
       [2.42826242e-04, 3.78006486e-01, 6.21750688e-01],
```

REGRESSÃO LOGÍSTICA

» Para avaliar o resultado, podemos usar a função classification_report e obter rapidamente as principais métricas de avaliação.

```
from sklearn.metrics import classification_report
```

```
print(classification_report(y_test, predito))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      0.72      0.84        18
           2       0.69      1.00      0.81        11

    accuracy                           0.89        45
   macro avg       0.90      0.91      0.88        45
weighted avg       0.92      0.89      0.89        45
```