

***INTELIGÊNCIA ARTIFICIAL***

**UA02 / LABORATÓRIO # 4**

**ASSUNTO**: TREINAMENTO E TESTES

**Materiais de Apoio**

**Site oficial do Scikit Learn:**

**scikit-learn.org/**

**Um bom tutorial de EDA:**

**https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6**

**Parte 1 – Importação de Bibliotecas e Leitura de Dataset**

# Importação de Bibliotecas

import pandas as pd

from sklearn import datasets, linear\_model

from matplotlib import pyplot as plt

import numpy as np

from sklearn.model\_selection import KFold # Para estratégia de Cross-Validadion (K-Folds)

from sklearn.model\_selection import train\_test\_split # Para estratégia Train/Test Split

# Leitura de Dataset - Diabetes, do SKLearn

columns = "age sex bmi map tc ldl hdl tch ltg glu".split() # Declara os nomes das colunas

diabetes = datasets.load\_diabetes() # Carrega o dataset diabetes de sklearn

df = pd.DataFrame(diabetes.data, columns=columns) # Carrega o dataset como um data frame

y = diabetes.target # define a variavel target variable (varivavel dependente, no modelo de regressão)

**Parte 2 – Train/Test Split**

# Treinamento e Testes - Estratégia: Train/Test Split (20% para Testes)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df, y, test\_size=0.2)

print (X\_train.shape, y\_train.shape)

print (X\_test.shape, y\_test.shape)

(353, 10) (353,)

(89, 10) (89,)

**Parte 3 – Modelo de Regressão Linear**

# Construção e Aplicação do Modelo: Regressão Linear

lm = linear\_model.LinearRegression()

model = lm.fit(X\_train, y\_train)

predictions = lm.predict(X\_test)

predictions[0:5]

array([143.87724381, 152.24462251, 182.90064091, 166.35506734,

165.69649022])

# Exibição do Modelo: Scatter Plot

plt.scatter(y\_test, predictions)

plt.xlabel("True Values")

plt.ylabel("Predictions")

Text(0, 0.5, 'Predictions')

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

# Exibição de percentual de acurácia

print ("Score:", model.score(X\_test, y\_test))

Score: 0.5265609153881694

**Desafio 1: experimente outros percentuais de splitting e veja como fica a acurácia? Por que melhora (ou piora)?**

**Parte 4 – Cross-Validation (K-Folds)**

# Treinamento e Testes - Estratégia: Cross Validation (2 folds)

X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]]) # Cria um array como dataset exemplo

y = np.array([1, 2, 3, 4]) # Cria um outro array como dataset exemplo

kf = KFold(n\_splits=2) # Define a separação (split) em 2 folds (k=2)

kf.get\_n\_splits(X) # Retorna o numero de iterações de separação (splitting) do cross-validator

print(kf)

KFold(n\_splits=2, random\_state=None, shuffle=False)

for train\_index, test\_index in kf.split(X):

    print("TRAIN:", train\_index, "TEST:", test\_index) # Linhas do array usadas para TRAIN e TEST

TRAIN: [2 3] TEST: [0 1]

TRAIN: [0 1] TEST: [2 3]

**Desafio 2: experimente outras quantidades de splits para ver como o cross validator trabalha.**

**Bom Trabalho!!**