K-NEARST NEIGHBORS ALGORITHM

PART 1 - DISTANCE CALCULATION

- The core of KNN inolves measure distances between data points. In this step W'ill create a function to compute the Euclidian Distance between two points.

THE EUCLIDIAN DISTANCE BETWEEN TWO POINTS

$$\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

- Calculate the Euclidian Distance between two points:
- Parameters:

  point 1: list or arrays-like (coordinates of the first point)

  point 2: list or arrays-like (coordinates of the second point)

- Returns: distance: float

In [19]:
```
def euclidian_distance(point1, point2):
    #Ensure both points have same dimension
    assert len(point1) == len(point2), 'Point must have the same dimensio

    #Compute the Euclidian Distance:
    # 1
    e_distance = 0
    for i in range(len(point1)):
        dif = (point2[i] - point1[i])**2 # (q - p) --> 'q' query point; t
        e_distance += dif
    e_distance = e_distance**0.5

    """
    # 2
    e_distance = (sum((q[i] - p[i])**2 for i in range(len(p))))**0.5

    # 3
    e_distance = sum((p1 - p2)**2 for p1, p2 in zip(point1, point2))**0.5
    """
    return e_distance
```

PART 2 - FIND THE NEAREST NEIGHBORS

```
    """
    Find the 'K' nearest neighbors of a query point within a
    dataset.
    Parameters:
```

```
    x_train: list/vector or array-like
        # Training dataset containg features
    query_point: list/vector or array-like
        # Coordinates of the query point
    K: int
        # Number of neighbors to find
Returns:
neighbors: list
    # List object with the indices of the 'K' nearest
neighbors
"""
```

In [24]:
```
def find_neighbors(x_train, query_point, k): # Parameters: 'x_train' - tr
    distance = []

    # calculate the distance between the query point to each point in the
    for i, data_point in enumerate(x_train):
        distance = euclidian_distance(query_point, data_point)
        distance.append((i, distance)) # add to the 'distance' empty list

    # Sort distances in ascending order
    distance.sort(key=lambda x: x[1]) # will organize the values of 'dist
                                      # will organize by the value of the
    # get the indices of the 'k' nearest neighbors
    neighbors = []
    top_k = distance[ : k] # list slicing - top_k stores tuples (index, d

    for item in top_k: # scroll through the top_k list, while 'item' assu
        index = item[0] # then, get the first positioning value at the tu
        neighbors.append(index) # and add this index (the positioning of
    return neighbors # Therefore, neighbors is a list that stores the ind
```

PART 3 - PREDICTING THE CLASS

```
"""
    Predicting the class of a query point based on the
majority class among its nearest neighbors
    Parameters:
    x_train: list/vector or array-like
        Training dataset containing features
    y_train: list/ vector or array-like
        Training dataset containing labels.
    query_point: list/vector or array-like
        Coordinates of the query point
    k: int
    number of neighbors to consider
    returns:
    predicted_class: int or str
        Predicted class label for the query point
"""
```

In [ ]:
```
def predict(x_train, y_train, query_point, k):
    # It will try to predict a label (1 or 0) of a query_point
    # Performs the actual classification by comparing the query to each x

    neighbors = find_neighbors(x_train, query_point, k)
    # Calls the function that returns the indexes of 'k' nearest neighbor
```

```
    # These indexes will be used to find the corresponding labels (1 or 0

    neighbors_labels = []
    for i in neighbors:
        neighbors_labels.append(y_train[i])
    # Here, acessing y_train, I get the labels (1 or 0) of the correspond

    count_ones = neighbors_labels.count(1) # how many ones '1' are presen

    if count_ones > len(neighbors_labels) // 2:
    # if the number of ones it's more than half of the 'neighbors_labels'
        predicted_class = 1 # therefore, the query point is classified as
    else:
        predicted_class = 0 # else, if it's minority, it's classified as

    return predicted_class
```

CLASSIFICATION

- LOAD/LOOK/CLEAN THE DATA

In [80]:
```
data_load = pd.read_csv('news_labeled_dataset_II.csv')
```

In [81]:
```
print(f'Before Cleaning: {data_load.shape}')
```

```
Before Cleaning: (300, 6)
```

In [71]:
```
data_load = data_load.drop(index=[150, 151, 152])
# remove these specif empty lines - used to divide the dataset into two p
```

In [73]:
```
data_fraud = data_load.iloc[:150]
data_nfraud = data_load.iloc[150:]
```

In [88]:
```
data_load.iloc[149]
```

Out[88]:
```
id                                                        150
label                                                       1
titulo          Transporte escolar em Santa Catarina é alvo de...
data                                                 01/08/2023
texto_noticia   Transporte escolar em Santa Catarina é alvo de...
link_noticia    https://www.tvbv.com.br/transporte-escolar-em-...
Name: 149, dtype: object
```

In [46]:
```
!pip install nltk
nltk.download('stopwords')
```

```
Requirement already satisfied: nltk in /opt/conda/lib/python3.10/site-pa
ckages (3.9.1)
Requirement already satisfied: regex>=2021.8.3 in /opt/conda/lib/python
3.10/site-packages (from nltk) (2024.11.6)
Requirement already satisfied: joblib in /opt/conda/lib/python3.10/site-
packages (from nltk) (1.2.0)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.10/site-pa
ckages (from nltk) (4.65.0)
Requirement already satisfied: click in /opt/conda/lib/python3.10/site-p
ackages (from nltk) (8.1.3)
```

```
[nltk_data] Downloading package stopwords to /home/jovyan/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[46]: True

In [92]:
```
import re # regular expressions to handle with text patterns
import string # to help with cleaning punctuiations
import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer # convert tex
from sklearn.model_selection import train_test_split # training and test
from sklearn.neighbors import KNeighborsClassifier # the KNN from Scikit-
from sklearn.metrics import accuracy_score # to evaluate how accurate the
from nltk.corpus import stopwords

def read_data(file_name):
    # Load the dataset with comma delimimiter
    data = pd.read_csv(file_name, delimiter=',')
    return data

def clean_text(text):
    text = str(text).lower() #converts everything into string, then into
    text = re.sub(f'[{string.punctuation}]', '', text) # "replace(pattern
    text = re.sub(r'\s+', ' ', text).strip() # raw string r' '; whitespac
    return text # returns the clean text in: lowercase + without punctuat

def preprocess_text_data(data, custom_stopwords=None):

    data['full_text'] = data['titulo'].fillna('') + ' ' + data['texto_not
    # creates a new colum where it join the 'title' and 'text' columns fr
    data['full_text'] = data['full_text'].apply(clean_text)

    stop_words_pt = stopwords.words('portuguese')
    stop_words_pt.extend(["notícia", "notícias", "portal", "site", "jorna
    "coluna", "colunista", "colunistas", "redação", "editor", "edição", "
    "publicado", "imprensa", "fontes",
    "veja", "leia", "clique", "aqui", "mais", "saiba", "compartilhe", "co
    "comentários", "sem", "link", "página", "continuar", "continuar lendo
    "hoje", "ontem", "amanhã", "agora", "enquanto", "durante", "antes", "
    "semana", "mês", "ano", "anos", "segunda", "terça", "quarta", "quinta
    "sábado", "domingo", "manhã", "tarde", "noite",
    "disse", "afirmou", "declarou", "falou", "informou", "relatou", "come
    "segundo", "conforme", "explicou", "completou",
    "inclusive", "portanto", "contudo", "ainda", "apenas", "entre", "sobr
    "mediante", "tudo", "todos", "cada", "várias", "diversos", "algum", "

    vectorizer = TfidfVectorizer(stop_words = stop_words_pt, max_features
    X = vectorizer.fit_transform(data['full_text']) # [fit: learn from da
    Y = data['label'].fillna('')
    return X, Y, vectorizer

def split_data(X, Y):
    # 0.8 train / 0.2 test
    x_train, x_test, y_train, y_test = train_test_split(
    # Spliting with the labels (0 and 1) / using strtify
        X,
        Y,
        train_size = 0.8,
        shuffle = True,
        random_state = 42,
```

```
        stratify = Y
    )

    return x_train, x_test, y_train, y_test

def fit_model(x_train, y_train, k=5):
    # “These 240 news articles are labeled: some are about fraud, others
    # learn to distinguish between them, using the words/features they co

    # Implement KNN classifier
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(x_train, y_train)

    # Predict on the train set (evaluate how well the training data was m
    preds_in_train = knn.predict(x_train)

    #calculate the accuracy of preds on the train data
    train_accuracy = accuracy_score(y_train, preds_in_train)

    return train_accuracy, knn
```

In [96]:

```
#Run the Flow

# read the data
data = pd.read_csv('news_labeled_dataset_II.csv')

# convert text into numbers
X, Y, vectorizer = preprocess_text_data(data)

# split the data (80/20)
x_trainval, x_test, y_trainval, y_test = split_data(X, Y)

# split the data again, now, the train data for validation(70/30) (divisã
x_train, x_val, y_train, y_val = train_test_split(
    x_trainval,
    y_trainval,
    train_size = 0.7,
    random_state = 42,
    stratify = y_trainval
)

# fit on train data
train_accuracy, knn = fit_model(x_train, y_train)
print(f'KNN accuracy on train data: {train_accuracy}')

clas_preds = knn.predict(x_val) # classification predictions
val_accuracy = accuracy_score(y_val, clas_preds) # compares the predicts
print(f'Validation accuracy: {val_accuracy}')

#JUST PRESS THE BUTTON AND SEE WHAT HAPPENS...
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/feature_extraction/text.
py:409: UserWarning: Your stop_words may be inconsistent with your prepr
ocessing. Tokenizing the stop words generated tokens ['lendo'] not in st
op_words.
  warnings.warn(
KNN accuracy on train data: 0.9345238095238095
Validation accuracy: 0.7638888888888888
```

In [77]:
```
print(data['label'].iloc[150])
```

```
nan
```

DEU CERTOOOOOOOOOOOOOOOOOOOOO!!! OBRIGADO MEU DEUS POR TUDO! VAMOS EM NOME DO SENHOR JESUS! AMÉM! SEMPRE!

JUST KEEP GOING!

29/04/2025 18:53

In [ ]:
```

```