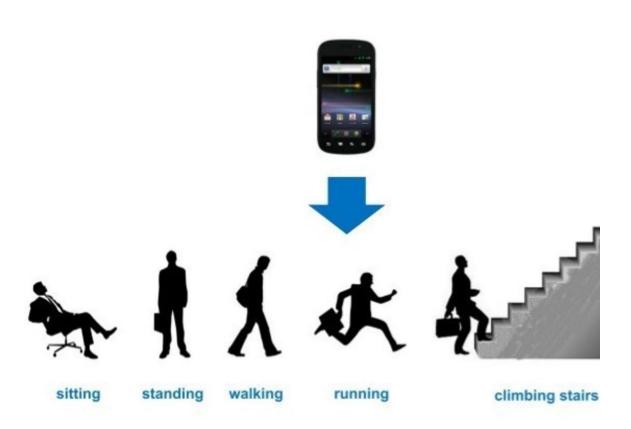
## **Tópicos de Ciência de Dados**

## Trabalho Laboratorial

## Classificação de Atividades Humanas



## B. Elaboração de um conjunto de scripts e funções em Python, NumPy, SciPy e Scikit-learn para realizar as tarefas de Aprendizagem Computacional e Avaliação

- 1. Data splitting e métricas de exactidão em machine learning
  - 1.1. Usando o scikit-learn, desenvolva um conjunto de funções para **data splitting** usando dois cenários:
    - 1.1.1. Train-Test (TT) e Train-Validation-Test data split (https://scikit-

<u>learn.org/stable/modules/generated/sklearn.model\_selection.train\_t</u> <u>est\_split.html</u>)

1.1.2. K-fold data split

(https://scikit-

<u>learn.org/stable/modules/generated/sklearn.model\_selection.KFold.</u> html

1.2. Usando o scikit-learn, desenvolva um conjunto de funções para cálculo de **métricas de exactidação**, nomeadamente as seguintes

(https://scikit-learn.org/stable/modules/model\_evaluation.html):

- 1.2.1. Matriz de confusão
- 1.2.2. Recall
- 1.2.3. Precision
- 1.2.4. F1-score
- 2. Experiências iniciais com um classificador simples, i.e., k-Nearest Neighbours (kNN)
  - 2.1. Usando:
    - as funções anteriores
    - o algoritmo kNN

(https://scikit-

<u>learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#</u>)

- o dataset Iris

(https://scikit-

<u>learn.org/stable/auto\_examples/datasets/plot\_iris\_dataset.html</u>)

avalie a capacidade de previsão do algoritmo neste dataset, com k=1 e restantes parâmetros por omissão (e.g., métrica de distância, etc.), e usando todo o conjunto de features, nos seguintes cenários:

- 2.1.1. Train-only, TT 70-30 e 10x10-fold cross-validation (10CV)
- 2.1.2. Train-only, TVT 40-30-30 e 10x10CV, fazendo variar k na gama  $\{1, 3, 5, ..., 15\}$ .
- 2.1.3. Analise os resultados em termos de **bias-variance** e **underfitting- overfitting**
- Repita a experiência anterior usando o algoritmo ReliefF para obter o ranking de features e seleccionar o modelo ideal (em termos de parâmetros e features a utilizar)
  - 2.2.1. Utilize a métrica F1-score como critério óptimo para a escolha de features e parâmetros a utilizar

- 2.2.2. Visualize o "gráfico do cotovelo" relativo ao desempenho do modelo (no conjunto de validação) à medida que se vão adicionando features
- 2.2.3. Apresente os resultados alcançados no conjunto de validação para todas as combinações de features e parâmetros testados. Tire conclusões em termos do dilema bias-variance e underfitting-overfitting.
- 2.2.4. Apresente os resultados do modelo ideal no conjunto de teste, compare-os com os resultados no conjunto de validação e tire conclusões.
- 2.3. Repita 2.2, implementando o algoritmo forward feature selection (FFS)
- 2.4. Repita 2.3, restringindo a classe iris-versicolor a 30 amostras e a classe irisviriginica a 10 amostras. Analise os resultados em termos do impacto da class imbalance.
- 3. Repita o ponto 2 no dataset de actividades humanos utilizado neste trabalho, restringindo a selecção de features ao algoritmo ReliefF
- 4. Repita o ponto 2 no dataset de actividades humanos utilizado neste trabalho, usando data splitting TVT (apenas TVT; CV não é para fazer), uma rede neuronal feedforward (MLP) com 3 camadas, número variável de neurónios na camada escondida, função de activação logística em todos os neurónios, batch learning e as features seleccionadas nas alíneas anteriores:

(https://scikit-learn.org/stable/modules/neural\_networks\_supervised.html)

- 4.1. Com velocidade de aprendizagem fixo
- 4.2. Com velocidade de aprendizagem variável
- 4.3. Com coeficiente de momentum
- 4.4. Discuta os resultados de 4.1 a 4.3, e estes com os resultados dos pontos anteriores (2 a 3: kNN, etc.)
- 5. Desenvolva de raiz a sua própria rede neuronal (de acordo com o especificado na alínea 4), usando para treino o algoritmo de retropropagação do erro.

(https://towardsdatascience.com/how-to-build-your-own-neural-network-from-scratch-in-python-68998a08e4f6

https://www.youtube.com/watch?v=IMZfiFltrLI)