Intelligence Artificielle 1

Cours 3 - Introduction à l'AI - Programmation (1/2)

Table des matières

Introduction à des tâches programmables reliées à l'intelligence artificielle (1/2)

Nous allons voir comment faire un modèle de "A à Z" avec la programmation.

Le principe reste identique pour les autres modules et modèles nécessitant une phase d'ingestion des données (entraînement).

Introduction à des tâches programmables reliées à l'intelligence artificielle (1/2)

Tâches courantes (en gras est abordé dans cette diapo) :

- **Importation des données, séparation des ensembles de données (train/test sets) et nettoyage des données**
- **La compilation d'un modèle préétabli avant l'entraînement/l'évaluation (étapes et distinctions brèves, vu en détail au cours AI 3) :**
  - **La topologie du modèle (couches, etc.)**
  - **La fonction de perte (Loss function)**
  - **L'optimisateur (Optimizer)**
  - **Métrique (Metric)**
- **L'entraînement d'un modèle à partir de données**
- **L'évaluation d'un modèle à partir de données**
- La représentation graphique des données et résultats d'évaluation
- La sauvegarde et chargement de modèles ML

Analyse, Importation, Séparation des ensembles de données (train/test sets), et Nettoyage

Les données sont une composante importante de l'apprentissage automatique ("Machine Learning", ML).

Un modèle apprend à partir des données fournies à celui-ci pour prédire les résultats ("prédictions") grâce aux tendances qu'il établit en entraînant sur les données.

Sur ce, le nettoyage des données est une des étapes les plus importantes d'un projet de Machine Learning.

Données - Analyse des données

Comment faire un nettoyage des données juste ?

En prennant le temps de regarder/analyser **les propriété des données**.

Données - Analyse des données

Les types d'informations pertinentes à regarder et comment :

- La distribution des données :
  - df.describe(), df.hist(), etc.
- L'importance des caractéristiques entre elles-mêmes :
  - Matrice de confusion ("Confusion Matrix").

Données - Analyse des données

N'hésitez pas à utiliser les techniques d'analyse de votre domaine d'application.

Généralement, il y a toujours un expert du domaine, un "SME" ("Subject Matter Expert"), qui appuit le processus de compréhension des données et de son domaine d'application (industrie, contexte, etc.).

Données - Analyse des données

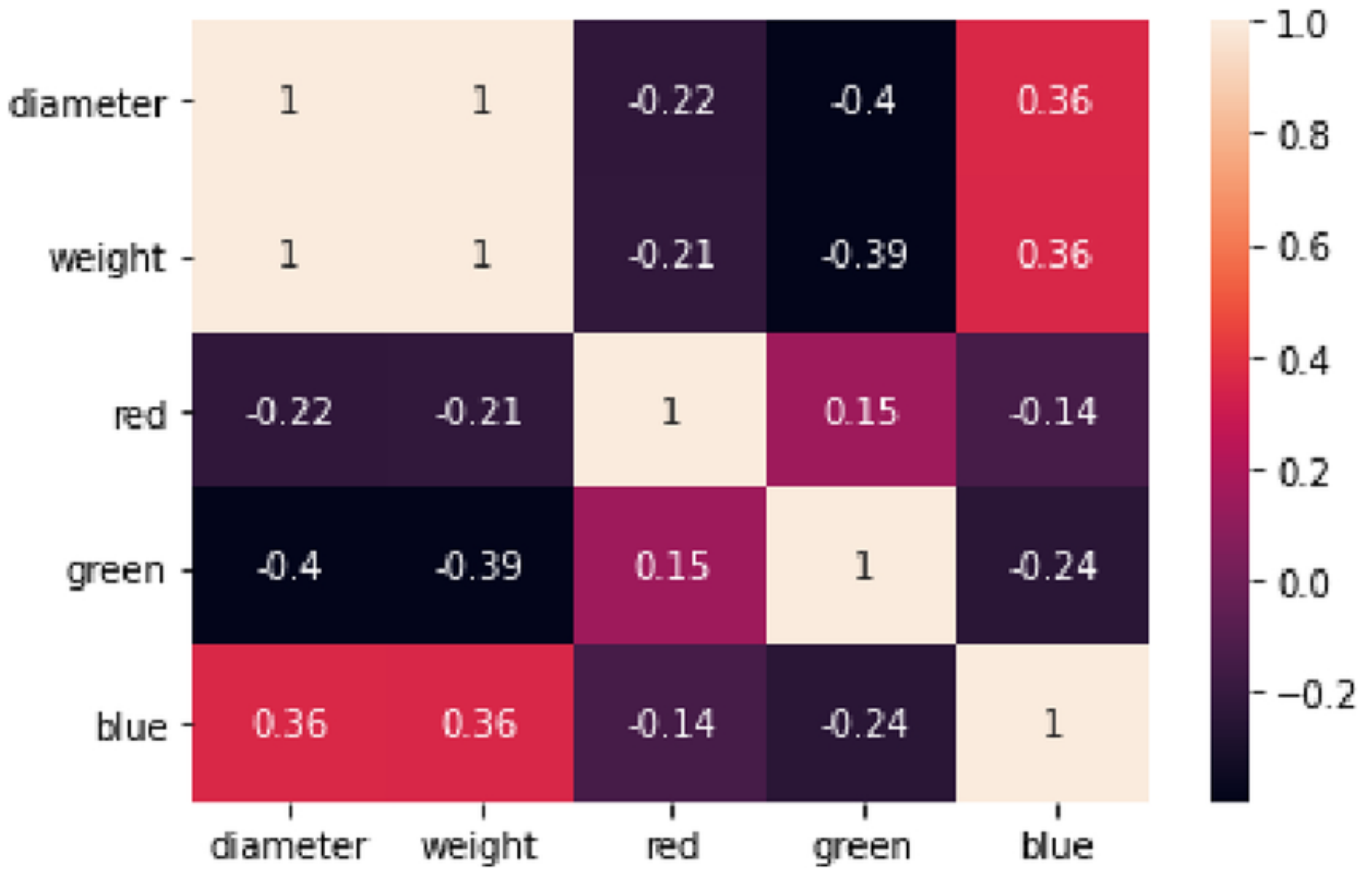

Figure: `https://www.researchgate.net/figure/Confusion-matrix-of-different-features_fig4_363518984`

Données - Importation et séparation des ensembles de données

L'importation des données dépends de la source et du contexte de développement.

Cela peut être aussi simple que d'importer un module.

Dans d'autres cas, il faut entammer un projet de développement logiciel complet pour la collecte et l'utilisation de la base de données dans une application.

Données - Importation et séparation des ensembles de données

La séparation des données est généralement faite de la manière suivante :

- 80% pour l'ensemble d'entraînement ("train set").
- 20% pour l'ensemble de test ("test set").

Données - Importation et séparation des ensembles de données

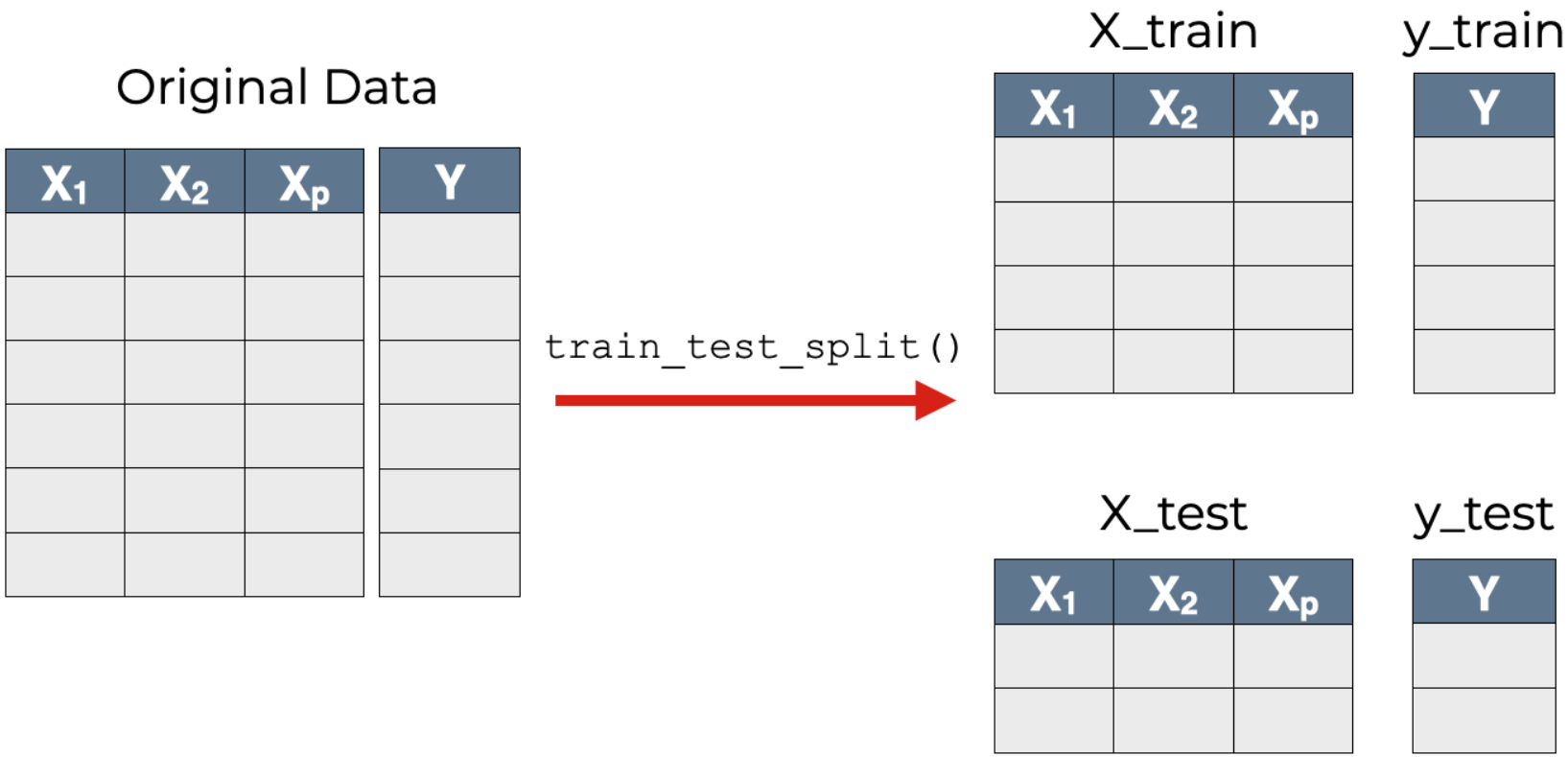

Figure: https://sharpsight.ai/blog/scikit-train_test_split/

Données - Importation et séparation des ensembles de données

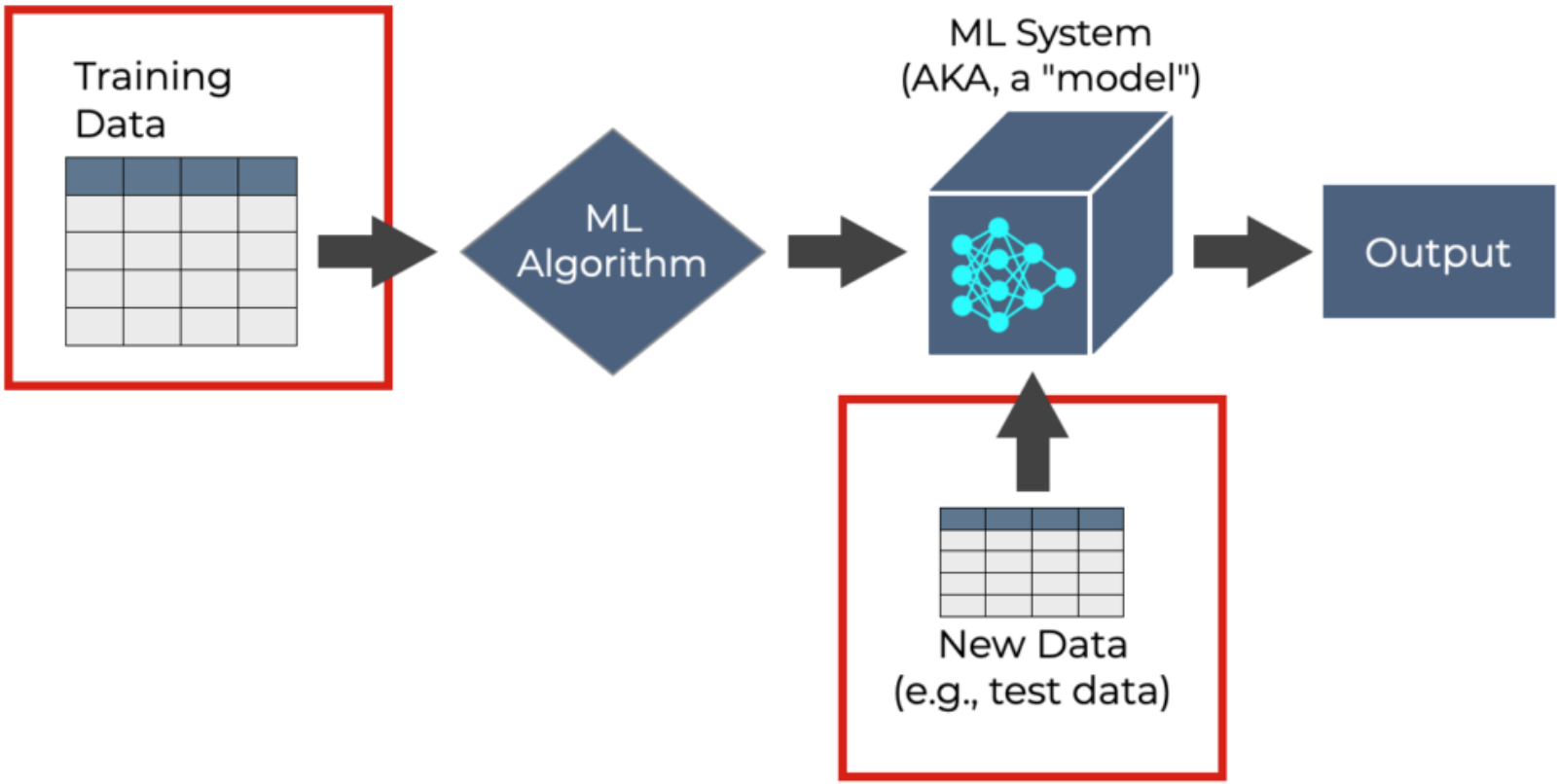

Figure: `https://sharpsight.ai/blog/scikit-train_test_split/`

Données - Importation et séparation des ensembles de données

Listing: https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/multi-class_classification_with_MNIST.ipynb?hl=en

```
1  # Copyright 2020 Google LLC. https://www.apache.org/licenses/LICENSE-2.0
2
3  # Import data
4  (x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

Données - Nettoyage des données

Listing: https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/multi-class_classification_with_MNIST.ipynb?hl=en

```
1  # Copyright 2020 Google LLC. https://www.apache.org/licenses/LICENSE-2.0
2
3  # Normalisation - the task of scaling values in a same level between 0 and 1.
4
5  # For this, we divide by 255 because the color spectrum is between 0 and 255.
6  x_train_normalized = x_train / 255.0
7  x_test_normalized = x_test / 255.0
```

Données - Nettoyage des données

Liste **non exhaustive** de manipulations :

- Normalisation (généralement fait dans tous les projets avec de l'IA).
- Remplacer les valeurs vides par :
  - Une valeur spécifique.
  - La moyenne des valeurs de toutes les données.
- Supprimer les rangées avec trop de valeurs vides ou impures.
- Retirer des colonnes (caractéristiques) inutiles.
  - La matrice de confusion aide à cette prise de décision.
- Etc.

La compilation d'un modèle avant l'entraînement/l'évaluation : étapes et distinctions brèves

Il y a quatre (4) composantes importantes lors de la compilation d'un modèle avant l'entraînement ou l'évaluation de celui-ci :

- La topologie du modèle (couches, etc.)
- La fonction de perte (Loss function)
  - Différence entre la fonction de perte (Loss function) et la fonction de coût (Cost function)
- L'optimisateur (Optimizer)
- Métrique (Metric)

La topologie du modèle (couches, etc.) - Résumé bref

Un modèle de réseaux de neurones est constitué de plusieurs couches de manière séquentielle.

Cette partie est complexe, il est généralement possible de se baser sur l'état de l'art (littérature) et d'utiliser les découvertes prouvées à cet effet.

La topologie du modèle (couches, etc.)

Listing: https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/multi-class_classification_with_MNIST.ipynb?hl=en

```
1  # Copyright 2020 Google LLC. https://www.apache.org/licenses/LICENSE-2.0
2
3  # All models in this course are sequential.
4  model = tf.keras.models.Sequential()
5
6  # The features are stored in a two-dimensional 28X28 array. Flatten that
7  # two-dimensional array into a one-dimensional 784-element array.
8  model.add(tf.keras.layers.Flatten(input_shape=(28, 28)))
9
10 # Define the first hidden layer.
11 model.add(tf.keras.layers.Dense(units=32, activation='relu'))
12
13 # Define a dropout regularization layer.
14 model.add(tf.keras.layers.Dropout(rate=0.2))
15
16 # Define the output layer. The units parameter is set to 10 because
17 # the model must choose among 10 possible output values (representing
18 # the digits from 0 to 9, inclusive).
19 # Don't change this layer.
20 model.add(tf.keras.layers.Dense(units=10, activation='softmax'))
```

La fonction de perte (Loss function) - Résumé bref

Cette fonction permet à chaque étape (epoch) de calculer et quantifier la marge d'erreur entre la valeur et la prédiction.

Un problème d'optimisation cherche à minimiser une fonction de perte.

Les problèmes courants et populaires sont dans le domaine de la classification (MNIST) ou de la régression (Real Estate $)

Les loss functions peuvent être réservées à seulement la classification ou la régression :

- Classification : Categorical cross-entropy, Hinge Loss, Log Loss
- Régression : Mean Absolute Error (MAE ou L1 Loss), Mean Squared Error (MSE ou L2 Loss), Uber Loss

Différence entre la fonction de perte (Loss function) et la fonction de coût (Cost function) - Résumé bref

Quelle est la différence entre la “Loss function” (i.e. L1 Loss) et la “Cost function” (i.e. MAE) ?

La “Loss function” est pour une prédiction et sa valeur réelle (“label”).

La “Cost function” est pour la totalité des observations (dataset / jeu de données) par aggrégation des valeurs des “Loss function”.

Des abus de langue peuvent être fait avec loss et cost, mais vous êtes invité.e.s à seulement vous rappeler que **les retours des functions de loss/cost permettent de mieux guider l’optimisateur**.

L'optimisateur (Optimizer) - Résumé bref

Un optimisateur permet d'obtenir graduellement le minimum le plus petit possible vis-à-vis les retours de la loss function après chaque étape (epoch) en définissant le réarrangement des paramètres pour y arriver.

Le but de l'optimisateur (et l'acte d'optimisation) est de trouver les paramètres optimaux pour le modèle qui réduit le minimum de la loss function le plus possible.

Métrique (Metric) - Résumé bref

La métrique permet de faire une évaluation du modèle et savoir s'il est compétent ou non (dépendamment des standards du milieu d'application).
Types de métriques possibles :

- Classification (maximiser = meilleur) : Précision (Accuracy)
- Régression (minimiser = meilleur) : MAE, MSE, Root Mean Squared Error (RMSE), R2

La fonction de perte (Loss function), l'optimisateur (Optimizer), et la métrique (Metric)

Listing: https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/multi-class_classification_with_MNIST.ipynb?hl=en

```
1  # Copyright 2020 Google LLC. https://www.apache.org/licenses/LICENSE-2.0
2
3  def create_model(my_learning_rate):
4
5      model = tf.keras.models.Sequential()
6      model.add(tf.keras.layers.Flatten(input_shape=(28, 28)))
7      model.add(tf.keras.layers.Dense(units=32, activation='relu'))
8      model.add(tf.keras.layers.Dropout(rate=0.2))
9      model.add(tf.keras.layers.Dense(units=10, activation='softmax'))
10
11     # Construct the layers into a model that TensorFlow can execute.
12     # Notice that the loss function for multi-class classification
13     # is different than the loss function for binary classification.
14     model.compile(
15         optimizer=tf.keras.optimizers.Adam(learning_rate=my_learning_rate),
16         loss="sparse_categorical_crossentropy",
17         metrics=['accuracy'])
18
19     return model
```

L'entraînement d'un modèle à partir de données

Il suffit de "fitter" le modèle avec les données X (features), y (labels), le nombre d'epochs et d'autres hyper-paramètres pour arranger les composantes du modèle en question.

L'action d'entraîner permet au modèle d'itérer les epochs pour réduire la Loss, optimiser les paramètres (optimisateur) et potentiellement améliorer la métrique.

L'entraînement d'un modèle à partir de données

Listing: https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/multi-class_classification_with_MNIST.ipynb?hl=en

```
# Copyright 2020 Google LLC. https://www.apache.org/licenses/LICENSE-2.0

def train_model(model, train_features, train_label, epochs,
                batch_size=None, validation_split=0.1):
    """Train the model by feeding it data."""

    history = model.fit(x=train_features, y=train_label, batch_size=batch_size,
                        epochs=epochs, shuffle=True,
                        validation_split=validation_split)

    # To track the progression of training, gather a snapshot
    # of the model's metrics at each epoch.
    epochs = history.epoch
    hist = pd.DataFrame(history.history)

    return epochs, hist
```

L'évaluation d'un modèle à partir de données

L'évaluation est une fonction simple du modèle pour permettre d'évaluer celui-ci avec de nouvelles données inconnues au modèle.

En temps normal, les données d'entraînement (train set) sont connues du modèle, donc il faut utiliser des données de test (test set) sinon c'est de la "triche" au sens de la littérature et de la communauté de l'intelligence artificielle puisqu'il connait déjà le contenu.

L'évaluation d'un modèle à partir de données

Listing: https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/multi-class_classification_with_MNIST.ipynb?hl=en

```
1  # Copyright 2020 Google LLC. https://www.apache.org/licenses/LICENSE-2.0
2
3  # Evaluate against the train set.
4  print("\n Evaluate the new model against the train set:")
5  my_model.evaluate(x=x_train_normalized, y=y_train, batch_size=batch_size)
6  # Evaluate against the test set.
7  print("\n Evaluate the new model against the test set:")
8  my_model.evaluate(x=x_test_normalized, y=y_test, batch_size=batch_size)
9
10 """
11  Evaluate the new model against the train set:
12 15/15 ------------------ 0s 2ms/step - accuracy: 0.9727 - loss: 0.0995
13
14  Evaluate the new model against the test set:
15 3/3 -------------------- 0s 3ms/step - accuracy: 0.9567 - loss: 0.1503
16 """
```

Bibliographie

- https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/multi-class_classification_with_MNIST.ipynb?hl=en
- https://stephenallwright.com/loss-function-vs-cost-function/
- https://stephenallwright.com/l1-loss-function/
- https://stephenallwright.com/interpret-mae/
- https://sourestdeeds.github.io/blog/overfitting-and-underfitting/
- https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/