Description des choix : modélisation, estimation, logiciels

Table des matières

Partie 1 : Présentation

1.1 Introduction et objectif

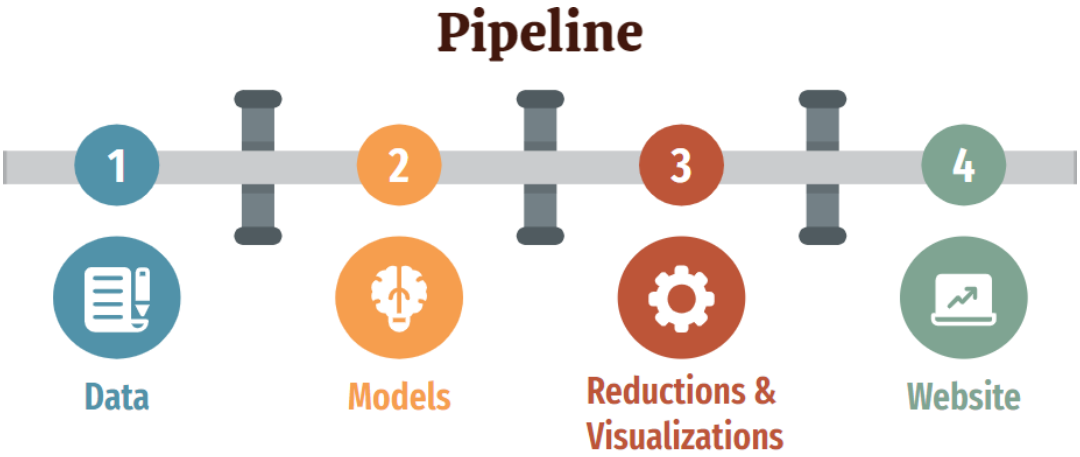
Dans le cadre de notre projet, notre commanditaire a exprimé le besoin de disposer d'un pipeline généralisé, capable de traiter n'importe quel jeu de données contenant des images. L'objectif est de permettre à ce pipeline de prendre les images, de pouvoir les classifier selon une caractéristique des images qu'on choisit. Par exemple si nos données sont des peintures, chaque peinture a un style artistique ou encore un artiste associé. Le pipeline se doit d'être totalement générique mais gardons cet exemple pour la suite de l'explication, pour des raisons de clarté :

Nous allons donc entraîner un modèle de classification qui va essayer de résoudre la tâche de reconnaître, pour une peinture donnée, le style de l'oeuvre. Nous allons ensuite extraire le *feature space* de nos données après entraînement. Pour expliquer simplement ce qu'est le feature space, imaginons que l'on demande à notre modèle après entrainement, de placer toutes les images sur un graphique. Intuitivement on comprend qu'il mettrait donc les peintures du même style relativement proches entre eux. Et des peintures qui auraient des styles diamétralement opposés, seraient même placés à des endroits très loins l'une de l'autre. Ce graphique là, c'est le feature space.
Attention : Le feature space est en grande dimension, nous devons donc appliquer des réductions de dimension en 2D pour pouvoir le visualiser et l'interpréter.

Le but est donc d'explorer le feature space, et de regarder si en entrainant sur le style des tableaux par exemple, le modèle n'a pas aussi bien classé les différents artistes entre eux. En effet, il nous suffit maintenant de colorier chaque point du graphique (correspondant à une peinture) par artistes, et nous pourrions déterminer des choses. Par exemple si on voit que les artistes sont bien séparés entre eux alors que nous avons entrainé le modèle sur le style du tableau, on pourait conclure que les artistes et les styles sont intimement liés et que les artistes ont donc globalement chacun une patte artistique propre à eux et très souvent ils gardent un seul style artistique pour toutes leurs œuvres. C'est une analyse plutot logique, mais cela permet notamment confirmer ces analyses par un œil informatique et donc totalement extérieur. Et surtout , grâce à notre pipeline nous pouvons généraliser ces analyses à n'importe quel jeu de données, et n'importe quelle tâche, et affirmer donc d'autres analyses moins évidentes.

Ainsi, nous avons été chargés de concevoir et développer un tel pipeline en utilisant les technologies prescrites et l'assistance de notre professeur Maximilien Servajean.

1.2 Visualisation du pipeline



1.3 Technologies et assistance

Pour mener à bien ce projet, nous avons utilisé les technologies qui nous ont été recommandées et nous avons bénéficié de l'aide précieuse de M. Servajean. Son expertise nous a permis de mieux comprendre les enjeux et les défis liés à la réalisation d'un pipeline généralisé, ainsi que les meilleures pratiques pour y parvenir. Grâce à cela, nous avons pu optimiser nos choix techniques et méthodologiques pour aboutir à une solution performante et adaptée aux besoins de notre commanditaire.

1.4 Entraînement du pipeline et base de données utilisée

Bien que le pipeline développé soit conçu pour être général et s'adapter à divers jeux de données, il est essentiel de le tester et de l'entraîner. Pour ce faire, nous avons utilisé la base de données WikiArt, qui recense plusieurs dizaines de milliers de tableaux. Cette base de données représente un excellent terrain d'entraînement pour notre modèle, puisqu'elle contient une grande variété de données et permet d'évaluer l'efficacité de notre pipeline dans la création de clusters pertinents.

En exploitant cette base de données, nous avons pu affiner les performances de notre pipeline et nous assurer qu'il est capable de gérer efficacement différentes tâches données. Cela nous a également permis d'identifier d'éventuelles améliorations et d'adapter notre solution aux exigences spécifiques de notre commanditaire.

1.5 Choix du modèle d’apprentissage

Pour répondre aux besoins de notre projet, nous avons choisi d'utiliser un réseau de neurones en raison de sa capacité à traiter efficacement les données d'images. Les réseaux de neurones sont des modèles d'apprentissage automatique inspirés du fonctionnement du cerveau humain, capables d'apprendre à partir de données et de reconnaître des motifs complexes.

Nous avons opté pour une architecture à base de convolutions, également connue sous le nom de réseau de neurones convolutif (CNN). Les CNN sont particulièrement adaptés au traitement des images, car ils exploitent les motifs spatiaux locaux présents dans les données. Les couches de convolution permettent de capturer des caractéristiques visuelles pertinentes, telles que les formes, les textures et les contours, en analysant des régions spécifiques de l'image.

Pour améliorer les performances de notre modèle, nous avons effectué un fine-tuning sur ResNet50, une architecture de réseau de neurones pré-entraînée. Le fine-tuning consiste à adapter un modèle pré-entraîné sur une tâche spécifique en poursuivant l'entraînement avec de nouvelles données. En utilisant ResNet50 comme point de départ, nous avons pu bénéficier des connaissances préalables apprises sur un vaste ensemble de données d'images, ce qui a permis d'accélérer et d'améliorer notre processus d'apprentissage.

1.6 Conclusion de la présentation

En somme, le développement de ce pipeline généralisé a été mené avec succès grâce à l'utilisation de technologies appropriées et à l'aide apportée par Maximilien. La base de données WikiArt a servi de

support pour l'entraînement de notre modèle, nous permettant d'optimiser ses performances et de le préparer pour une utilisation par notre commanditaire. Nous sommes convaincus que ce pipeline répondra aux besoins et aux attentes, tout en offrant une solution polyvalente et efficace pour la classification et la création de clusters de données.

Partie 2 : Outils

2.1 Langage Python

Pour mener à bien notre projet, nous avons opté pour l'utilisation du langage de programmation Python. Ce choix s'explique par la popularité et la polyvalence de Python dans le domaine du traitement de données et de l'apprentissage automatique. En effet, Python est un langage de haut niveau, open-source et multiplateforme qui facilite la réalisation de projets complexes grâce à sa syntaxe claire et concise. De plus, Python dispose d'un vaste écosystème de bibliothèques et de frameworks dédiés au traitement de données, ce qui en fait un outil idéal pour notre pipeline.

2.1 Librairie Pytorch

Pour développer et entraîner notre réseau de neurones dans le cadre de notre projet, nous avons choisi d'utiliser la bibliothèque PyTorch. PyTorch est une bibliothèque open-source de machine learning qui offre une grande souplesse et une excellente performance. Elle est largement utilisée dans la communauté de l'apprentissage automatique en raison de sa facilité d'utilisation et de sa flexibilité, et c'est pourquoi Maximilien Servajean nous l'a fortement conseillée au détriment de Keras et Tensorflow, d'autres librairies permettant aussi l'apprentissage automatique.
PyTorch est construit sur le langage Python, ce qui permet une intégration fluide avec les autres bibliothèques et outils Python. Nous avons pu tirer parti de la puissance de PyTorch pour concevoir et former notre réseau de neurones, en exploitant ses fonctionnalités avancées telles que la gestion automatique des gradients et le calcul sur GPU. Grâce à PyTorch, nous avons pu expérimenter différentes architectures de réseau, ajuster les hyperparamètres et obtenir des performances optimales pour notre tâche spécifique.

2.2 Framework Flask

Pour lier notre code Python à notre site web, nous avons utilisé le framework Flask. Flask est un framework web minimaliste mais puissant qui facilite la création d'applications web en Python. Il offre des fonctionnalités robustes pour la gestion des requêtes HTTP, la gestion des sessions utilisateur, la création de routes et la génération de réponses dynamiques.
En utilisant Flask, nous avons pu créer une interface utilisateur conviviale pour notre projet. Nous avons pu définir des routes pour chaque fonctionnalité de notre application, ce qui nous a permis de gérer facilement les interactions entre le frontend et le backend. Nous avons également pu intégrer des visualisations générées avec Bokeh dans notre site web, offrant ainsi une expérience interactive aux utilisateurs.

2.2 Librairie Bokeh

Afin de produire des rendus graphiques de qualité pour la visualisation des clusters générés par notre pipeline, nous avons utilisé la librairie Bokeh. Bokeh est une bibliothèque Python dédiée à la création de visualisations interactives pour le Web. Elle permet de générer facilement des graphiques élégants et dynamiques à partir de données complexes, en offrant une grande flexibilité et un contrôle précis sur les éléments visuels.

Bokeh s'intègre parfaitement avec le langage Python et d'autres bibliothèques de traitement de données, ce qui facilite son utilisation dans le cadre de notre projet. Grâce à Bokeh, nous avons pu créer des visualisations attrayantes et informatives pour présenter les résultats de notre pipeline de manière claire et intuitive.

En utilisant la combinaison de Python, PyTorch, Bokeh et Flask, nous avons pu développer un pipeline complet, de la formation du modèle de réseau de neurones à la visualisation des résultats sur un site web convivial. Cette combinaison de technologies puissantes nous a permis d'exploiter pleinement les fonctionnalités avancées de PyTorch, de créer des visualisations attrayantes avec Bokeh et de lier le tout à notre site web grâce à Flask.

Partie 3 : Répartition des tâches et méthodologie

3.1 Organisation du travail

Afin d'optimiser notre travail et de progresser efficacement sur les différents aspects du développement de ce pipeline, nous avons choisi de nous répartir les tâches. Cette organisation nous a permis de travailler simultanément sur les divers éléments nécessaires à la réalisation de notre projet et d'assurer une meilleure coordination au sein de notre équipe.

3.2 Entraînement du modèle et extraction du feature space

L'un des membres de notre équipe a été chargé de l'entraînement de notre modèle de réseaux de neurones sur le jeu de données WikiArt. Son objectif principal était de fournir le meilleur modèle possible et de trouver l'architecture idéale, puis d'extraire le feature space associé. Pour ce faire, il s'est appuyé sur des modèles préexistants (ResNet50) qu'il a améliorés en utilisant notre base de données. Cette approche a permis de gagner du temps et d'assurer la qualité et la pertinence du modèle entraîné.

3.3 Réduction de dimension

Pendant ce temps, un autre membre de notre équipe s'est concentré sur la réduction de dimension des données pour permettre une représentation graphique des résultats du modèle. Plusieurs méthodes ont été utilisées pour cette tâche, telles que :

- t-SNE (t-distributed Stochastic Neighbor Embedding) : une technique de réduction de dimension non linéaire qui préserve les relations locales entre les points de données.
- UMAP (Uniform Manifold Approximation and Projection) : une méthode de réduction de dimension qui préserve à la fois les relations locales et globales, offrant des représentations plus cohérentes et interprétables.
- ACP (Analyse en Composantes Principales) : une méthode linéaire de réduction de dimension qui projette les données sur les axes principaux de variation, maximisant la variance tout en minimisant la perte d'information.
- ACP puis t-SNE : ACP jusqu'a environ 30 composantes (dimensions) et t-SNE jusqu'à 2 dimensions, pour des raisons de performances.

Le but de cette étape était de réduire les résultats du modèle à deux dimensions, facilitant ainsi leur représentation graphique et leur interprétation.

3.4 Site web

Un autre membre de l'équipe s'est focalisé sur le site web et de le rendre surtout joli, tâche très difficile surtout pour des gens pour qui ce n'est pas du tout le métier.

3.5 Application Flask / Bokeh

Enfin, un des étudiants a travaillé à lier le résultat du feature space pour le rendre en graphique bokeh sur le site web, et ce en utilisant donc le framework Flask.

Partie 4 : Hypothèse sur les méthodes de réduction de dimension

4.1 Supposition initiale

Au début de notre projet, nous avons émis l'hypothèse que les méthodes de réduction de dimension UMAP et t-SNE seraient plus performantes pour notre modèle par rapport à l'ACP. Cette supposition reposait sur le fait qu'UMAP et t-SNE sont des techniques non linéaires, tandis que l'ACP est une méthode linéaire.

Les méthodes non linéaires, comme UMAP et t-SNE, sont souvent considérées comme plus adaptées pour traiter des données complexes et de grande dimension, car elles peuvent capturer et préserver les structures locales et globales inhérentes aux données. En revanche, l'ACP, en tant que méthode linéaire, peut parfois être moins efficace pour traiter des données présentant des relations non linéaires.