Zoidberg 2.0

T-DEV 810

Alcaro Matthieu,
Lambert Emilien,
Marcel Léo,
Qedira Fares,
Vallera Antonio

Synthèse des connaissances

Introduction

Le projet zoidberg 2.0 est une introduction aux technologies d'intelligences artificielles. Nous travaillons avec un ensemble de données composés de photos d'examens radiologiques de patients atteints de pneumonies de types virales ou bactériologiques, ainsi que de patients sains sans aucune maladie.

Dans ce projet, nous avons expérimenté les différentes familles d'algorithmes supervisées ainsi que les différentes étapes nécessaires au fonctionnement tel que la gestion des données, ou encore les méthodes de transformation pour rendre le jeu de données plus facile à traiter par les algorithmes.

Objectifs

L'objectif du projet est d'expérimenter les différentes technologies de machine learning et d'appréhender les concepts nécessaires à la mise en place d'un algorithme d'apprentissage. Pour cela nous avons travaillé sur les points suivants :

- L'ensemble de données
- Les algorithmes d'apprentissage
- Les transformations
- Les statistiques

Ensembles de données

Données Mnist :

La base de données Mnist comporte un ensemble de fichiers binaires qui représentent des séries de nombres de 0 à 9, en noir et blanc, représentés par des nuances de gris. Cet ensemble de données est utilisé dans le bootstrap du projet Zoidberg, mais nous l'avons également utilisé dans ce projet pour comparer les performances des différents algorithmes avec des données différentes.

Données Zoidberg 2.0 :

Les données du Zoidberg 2.0 sont assez diverses, nous avons fait, voire les photos des connaissances qui sont des docteurs et il s'avère que de nombreuses photos sont des cas assez rare en médecine ou des patients avec des particularités, comme des sutures métalliques sur le sternum chez des patients qui ont eu une sternotomie (sternum ouvert en deux pour accéder au cœur, par exemple, ou aux poumons). Ainsi sur les images, il n'y a pas uniquement des tâches ou des marques liées à la maladie, mais d'autres types de pathologie qui pourraient induire en erreur ou perturber l'apprentissage. Nous avons également remarqué que certaines des photo avaient un situs inversus « cœur à droite »

La taille des photos n'est pas régulière et le ratio hauteur largeur est différent sur quasiment toutes les photos. Ce qui peut poser problème à certains algorithmes qui nécessitent des images de taille identique entre elles.

Enfin, les dimensions ne sont pas les mêmes pour toutes les photos. Toutes les photos sont en noir et blanc, mais certaines sont en vrais noir et blanc type "grayscale" avec 1 dimension et d'autre photo sont en noir et blanc, mais au format RGB a 3 dimensions. Ce qui pose également des problèmes avec certains algorithmes, car il faut charger des images de dimensions similaires.

Algorithmes

SVC :

Le SVC (Support Vector Classification) est un algorithme appartenant à l'ensemble des SVM (Support Vector Machines). Le SVC prend plusieurs hyperparamètres, notamment le gamma. Plus le gamma est grand, plus l'algorithme essaie de s'adapter aux données de façon précise. On peut d'ailleurs constater qu'augmenter le gamma de façon trop conséquente conduit à un "overfitting" et donc un modèle moins précis.

SVC Linear :

Le SVC Linear est un SVC avec un paramètre kernel 'Linear'

MLP Classifier :

Le MLP Classifier met en œuvre un algorithme de perceptron multicouche (MLP) qui s'entraîne par rétro propagation.

MLP s'entraîne sur deux tableaux : le tableau X de taille (n_samples, n_features), qui contient les échantillons d'entraînement représentés sous forme de vecteurs de caractéristiques en virgule flottante ; et le tableau y de taille (n_samples,) qui contient les valeurs cibles (étiquettes de classe) pour les échantillons d'entraînement.

Prédiction KNN :

Nearest Neighbors met en œuvre l'apprentissage non supervisé des plus proches voisins. Il agit comme une interface uniforme pour trois différents algorithmes de plus proches voisins: BallTree, KDTree, et un brute-force algorithm. Le choix de l'algorithme de recherche des voisins est contrôlé par le paramètre 'algorithm', qui doit être l'un de ['auto', 'ball_tree', 'kd_tree', 'brute']. Lorsque la valeur par défaut "auto" est utilisée, l'algorithme tente de déterminer la meilleure approche à partir des données d'apprentissage.

Pour calculer la valeur la plus adaptée du k (le nombre de voisins utilisé par l'algorithme), il faut prendre la racine carré du nombre total d'image d'entrainement (train). Dans notre cas, on laisse le choix de l'algorithme a la valeur par défaut pour utiliser le plus adapté. Cela permet d'entraîner au mieux possible l'algorithme et d'avoir une bonne précision lors de la phase de test.

Naive Bayes :

Les méthodes Naive Bayes sont un ensemble d'algorithmes d'apprentissage supervisé basés sur l'application du théorème de Bayes avec l'hypothèse "naïve" de l'indépendance conditionnelle entre chaque paire de caractéristiques étant donné la valeur de la variable de classe. Le théorème de Bayes établit une relation prenant en compte la variable de classe et un vecteur de caractéristiques.
Les classificateurs Naive Bayes ont très bien fonctionné dans de nombreuses situations du monde réel, notamment dans la classification des documents et le filtrage. Ils nécessitent une petite quantité de données d'entraînement pour estimer les paramètres nécessaires.

Gaussian NB implémente l'algorithme Gaussian Naive Bayes pour la classification. La vraisemblance des caractéristiques est supposée être gaussienne

Decision Tree Classifier :

DecisionTreeClassifier est une classe capable d'effectuer une classification multi-classes sur un ensemble de données.

Comme les autres classificateurs, Decision Tree Classifier prend en entrée deux tableaux : un tableau d'entrainement, clairsemé ou dense, de forme (images_training, labels_training) contenant les échantillons d'apprentissage, et un tableau de test de valeurs entières, de forme (images_tests, labels_tests), contenant les étiquettes de classe pour les échantillons d'apprentissage.

Dans le cas où il y a plusieurs classes avec la même et la plus haute probabilité, le classifieur prédit la classe avec l'indice le plus bas parmi ces classes. Comme alternative à la sortie d'une classe spécifique, la probabilité de chaque classe peut être prédite, ce qui est la fraction d'échantillons d'entraînement de la classe.

Nous pouvons également exporter l'arbre au format Graphviz en utilisant l'exportateur export_graphviz afin d'avoir un résultat visuel de la classification faite par l'algorithme.

Les paramètres utilisés sont 'entropy' pour le critère de classification (donc par rapport au gain), 'max_depth' qui correspond au nombre de niveau dans l'arbre (plus ce nombre est grand, plus la classification sera précise et ce jusqu'à un certain point, même si le graphique de résultat devient moins lisible), 'min_sample_split' indique en combien d'échantillon diviser le tableau d'entrainement ainsi que 'random_state' indiquant le niveau d'aléatoire.

Random Tree forest :

Random Forest est un modèle d'ensemble composé de nombreux arbres de décision, des sous-ensembles aléatoires de caractéristiques et le vote moyen pour faire des prédictions.

Le fonctionnement de l'algorithme est donc similaire à celui du Decision Tree Classifier avec en paramètre supplémentaire (n_estimators) qui correspond au nombre d'arbres qu'il va exécuter. Plus ce paramètre sera grand plus les performances seront élevées mais le code sera ralenti.

Pour récupérer un graphique en résultat, on récupère un graphique aléatoire parmi les différents Decision Tree Classifier effectuer par le Random Tree Forest.

Extremely Randomized Trees

Le classificateur ExtraTrees (Extremely Randomized Trees) fonctionne à l'identique par rapport au Random Tree Forest, Extra Trees teste des divisions aléatoires sur une fraction des caractéristiques (contrairement à Random Forest, qui teste toutes les divisions possibles sur une fraction des caractéristiques).

Transformations

Nystroem :

La méthode Nystroem est une méthode de “Kernel approximation”. Cette méthode est utilisée lorsque le dataset de classification comporte une trop grande quantité de données

PCA :

Le PCA est une méthode de réduction de dimensions. Le nombre de variables d'entrée ou de caractéristiques d'un dataset correspond à sa dimensionnalité.
La réduction de dimensions fait référence aux techniques qui réduisent le nombre de variables d'entrée dans un dataset.

Statistiques

Scores :

Algorithme	Scores (précision)	Paramètres algorithme	Temps d'exécution (seconde)
SVC	bacteria: 0.55 (55%) normal: 0.85 (85%) virus: 0.40 (40%)	/	934.9
SVC (avec PCA)	bacteria: 0.52 (52%) normal: 0.71 (71%) virus: 0.35 (35%)	/	26.5
SVC LINEAR	0.74741536590310 (74%)	/	1355.9
SVC LINEAR (avec PCA)	0.63085343604297 (63%)	/	18.0
MLP CLASSIFIER	0.38782051282051 (38%)	solver= lbfgs alpha= 1e-5 hidden_layer_sizes= (784, 3) random_state= 1	118.6
KNN	0.45993589743589 (46%)	k= 70 (calculer par la fonction)	3.5
NAIVE BAYES	0.46794871794871 (47%)	/	10.8
DECISION TREE CLASSIFIER	0.52083333333333 (52%)	criterion= entropy max_depth= 5 min_sample_split= 2 random_state= 0	44.4
RANDOM TREE FOREST	0.55288461538461 (55%)	n_estimators= 100 max_depth= 5 min_sample_split= 2 random_state= 0	16.2
EXTREMELY RANDOMIZED TREES	0.60096153846153 (60%)	n_estimators= 100 max_depth= 10 min_sample_split= 2 random_state= 0	13.8

Graphiques / visuels :

Le pdf Tree_Graph représente le résultat de l'algorithme Decision Tree Classifier avec la classification réalisée pour 5 niveaux dans l'arbre et un critère d' 'entropy'.

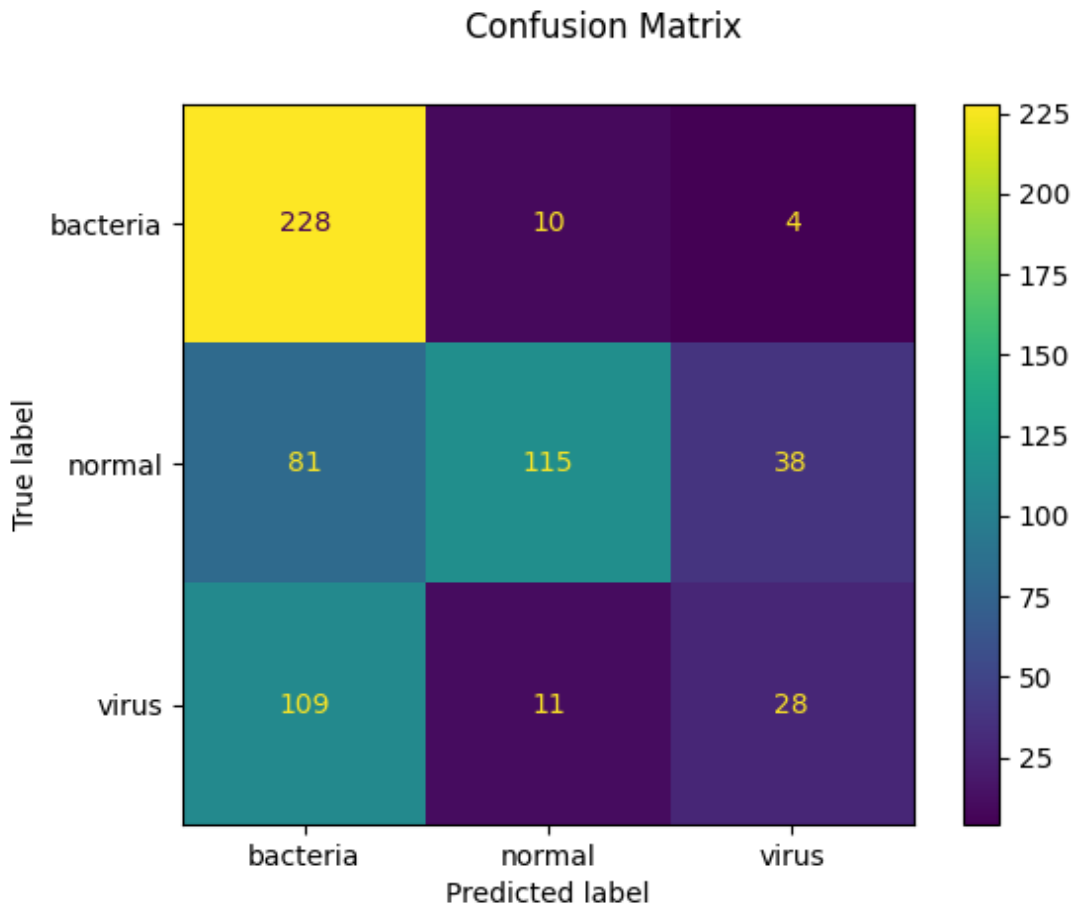
Le pdf Tree_Forest_Graph représente un des résultat de l'algorithme Random Tree Forest, en effet cet algorithme utilise plusieurs fois l'algorithme Decision Tree Classifier mais ne récupère qu'un arbre de résultat de manière aléatoire par rapport au nombre d' "arbre" générer par la "forêt". Dans notre cas 1 résultat parmi les 100 généré.

Prédictions :

Au regard de nos différents essais et des paramètres appliqués sur les algorithmes, on peut observer qu'au niveau de la meilleure précision, le SVC Linear est le plus performant comparé au MLP Classifier qui est le moins performant. Si on compare par rapport au plus rapide, le KNN est le plus rapide et le SVC Linear le moins rapide en temps d'exécution. Néanmoins, en appliquant des paramètres différents sur les algorithmes on peut observer des performances et des temps d'exécution différents pour chaque algorithme.

D'un autre point de vue, avec les bon paramètres et une bonne configuration, le réseau de neurones (MLP Classifier) doit être le plus efficace et performant pour la classification des images.

Matrices de confusion :

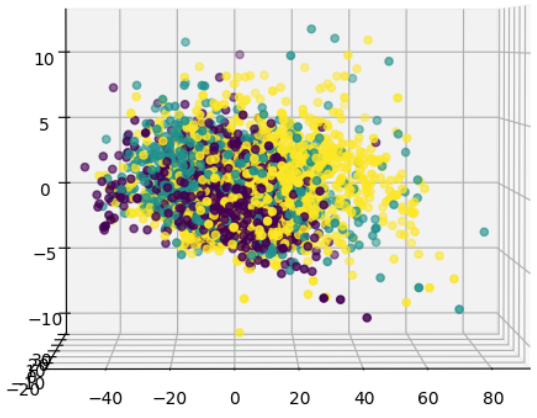


Matrice de confusion obtenue après l'exécution de l'algorithme SVC.

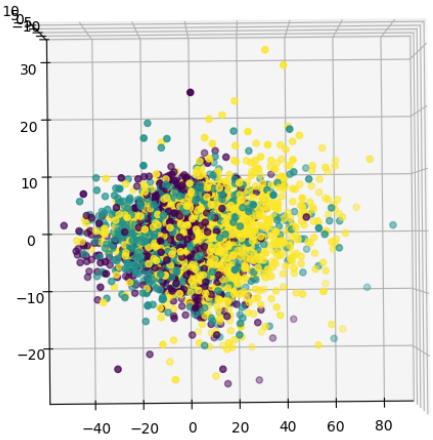
PCA 3D :

Voici une visualisation de notre training dataset réduit en 3 dimensions grâce au PCA.

Vu de face



Vu du dessus



Conclusion

Au cours de ce projet, nous avons appris à manipuler 8 types d'algorithmes d'apprentissage supervisé, ainsi que les différents types de fonctions de transformation associés. Nous avons également compris l'importance d'avoir un jeu de données spécifiques adapté. Comme l'importance d'avoir la même dimension ou encore d'avoir la même résolution d'image (X par Y). Sauf pour le MLP Classifier qui lui n'a pas besoin que les photo aient un format particulier.

Nous avons appris à interpréter les résultats grâce à des indicateurs tels que la matrice de confusion et les temps d'exécution ou encore les résultats de prédiction.

Également, nous avons expérimenté le fait de visualiser et d'interpréter les données sous forme graphique pour en extraire des informations pertinente et ainsi améliorer les paramètres des fonctions d'apprentissage et ainsi augmenter les performances ainsi que nos résultats de prédiction.