Chapitre 4 : Méthodologie

1. Introduction

Au cours des chapitres précédents, nous avons exploré les différents types d'algorithmes d'apprentissage profond et d'apprentissage automatique, ainsi que la notion de classification. Basés sur ces concepts, ce chapitre se concentre sur l'expérimentation de la construction de modèles de réseaux de neurones artificiels dans deux scénarios : avec compression de données en l’utilisation de l’algorithme des components principales (ACP) et sans compression de données. Nous appliquerons également différents types d'algorithmes d'optimisation. Les résultats des tests et des évaluations seront enregistrés afin de déterminer l'efficacité des techniques utilisées et de comparer les modèles construits.

1. Notre objet

Cette étude vise à mettre en évidence l'importance de l'utilisation des données statistiques dans la construction des modèles de prédiction supervisée. Nous menons une analyse comparative pour évaluer l'impact de la compression des données sur les algorithmes de classification profonde, ainsi que l'étendue de l'impact des algorithmes d'optimisation. Pour ce faire, nous avons utilisé un ensemble de données comprenant environ 5 000 enregistrements sur lesquels nous avons réalisé ces tests.

1. Prétraitement

Le prétraitement des données, également connu sous le nom de nettoyage des données, représente une phase cruciale dans l'analyse de données. Cette étape englobe la transformation des données brutes en données exploitables en éliminant les erreurs, les valeurs manquantes, les doublons, et en les standardisant dans un format approprié. Les étapes suivantes illustrent le processus de prétraitement des données que nous avons effectué :

* **collection des données**

Pour cette étude, j'ai exploré et choisi un jeu de données approprié sur la plateforme **Kaggle**, en tenant compte de sa pertinence par rapport aux objectifs spécifiques de mon cas d'étude. Après avoir récupéré le jeu de données, j'ai effectué des ajustements pour l'adapter aux exigences et aux paramètres définis dans le cadre de mon étude.

* **Importation des données**

Pour l'importation de données dans l'environnement Python, il est courant de recourir à la bibliothèque **Pandas**, renommée pour sa capacité à gérer divers formats de fichiers de données. Plus précisément, la fonction **read\_csv()** de Pandas est souvent utilisée pour importer des données à partir de fichiers au format CSV. Cette fonctionnalité permet de charger les données contenues dans un fichier CSV dans une structure de données appelée DataFrame, offrant ainsi une manipulation et une analyse aisées des données dans Python.

* **Répartition des données : Entraînement et Test**

Dans notre étude, nous avons effectué une étape cruciale de division des données en ensembles d'entraînement et de test. Nous avons attribué 80% des données à l'ensemble d'entraînement, tandis que les 20% restants ont été réservés à l'ensemble de test. Cette répartition nous permet de disposer d'un ensemble suffisamment large pour entraîner nos modèles tout en conservant un ensemble distinct pour évaluer leur performance. Cela garantit une évaluation fiable de la capacité de généralisation de nos modèles sur des données qu'ils n'ont pas encore vues.

* **Sélectionner les colonnes nécessaires**

Pour cette étude, j'ai sélectionné un ensemble de données pertinent en fonction de sa pertinence pour mon cas d'étude portant sur la prédiction des maladies basée sur les symptômes. Après avoir récupéré le jeu de données, j'ai utilisé la fonction **drop()** pour éliminer les paramètres (colonnes) non significatifs pour notre modèle, dans le but de sélectionner les symptômes les plus significatifs pour la prédiction des maladies. Ces symptômes ont été choisis avec soin en tenant compte de leur corrélation avec les maladies cibles et de leur importance dans la précision des diagnostics prédictifs. Cette étape a été réalisée en collaboration avec des experts médicaux et des médecins afin d'assurer la pertinence clinique et la validité des symptômes sélectionnés.

* **Suppression des documents contenant des valeurs nulles**

La tâche impliquait initialement de vérifier la présence de valeurs nulles dans un DataFrame Python. À cette fin, nous avons utilisé la méthode **isna().sum()** de la bibliothèque Pandas, permettant de calculer le nombre de valeurs nulles par colonne ou par ligne dans le DataFrame. Après avoir identifié les colonnes ou les lignes contenant des valeurs nulles, nous avons procédé à la suppression de ces entrées à l'aide de la méthode **dropna()** de Pandas. Cette opération a permis d'éliminer les lignes ou les colonnes contenant des valeurs nulles, assurant ainsi la qualité des données utilisées dans l'analyse subséquente.

* **compression des colonnes (la methode de ACP (pca en anglais))(optionnel)**

Dans ce projet, j'ai utilisé la méthode PCA (Analyse en Composantes Principales) de la bibliothèque **scikit-learn**, avec l'argument n\_components=2. Cette étape de réduction de dimensionnalité avec PCA(n\_components=2) avait pour objectif de comprimer les données vers un espace à deux dimensions. En fixant le nombre de composantes principales à 2, nous avons cherché à réduire la dimensionnalité du jeu de données tout en préservant au mieux sa structure et ses informations essentielles. Cette approche permet une représentation plus concise des données, ce qui facilite la visualisation, l'analyse et l'interprétation des résultats.

* **Encodage des caractéristiques**
* **encodage one-hot :** j'ai réalisé un encodage one-hot sur les variables relatives aux symptômes dans le jeu de données. Cela signifie que chaque symptôme présent dans les données a été transformé en une série de variables binaires, où chaque variable représente la présence ou l'absence d'un symptôme spécifique. Cette approche permet de traiter les symptômes comme vecteur des features binaire dans les modèles d'apprentissage profondeur**.**
* **encodage de label :** En ce qui concerne la variable cible, qui représente le pronostic de la maladie, j'ai utilisé un encodage de label en utilisant la fonction **LabelEncoder()** de la bibliothèque **scikit-learn**. Cela signifie que les différentes classes de pronostic de maladie ont été encodées avec des labels numériques uniques. Cette méthode est couramment utilisée pour traiter les variables cibles dans les problèmes de classification, où chaque classe est représentée par un nombre entier unique, facilitant ainsi l'entraînement des modèles d'apprentissage profondeur.

1. représentation des données

* **la représentation en vecteur de caractéristiques binaires :**

la représentation en vecteur de caractéristiques binaires est utilisée pour représenter les symptômes des patients. Chaque symptôme est représenté par une variable binaire, où 1 indique la présence du symptôme et 0 indique son absence. Par exemple, si nous avons un ensemble de symptômes comprenant {"fièvre", "toux", "maux de tête", "fatigue"}, et qu'un patient présente seulement la fièvre et la toux, sa représentation en vecteur de caractéristiques binaires serait [1, 1, 0, 0].

Cette représentation permet de traiter les symptômes comme des variables catégorielles dans nos modèles d'apprentissage automatique, facilitant ainsi l'analyse et la prédiction des diagnostics de maladies en fonction des symptômes présentés par les patients.

1. Processus général de notre système classification
2. Méthodologie de travail

Dans le cadre de notre étude, nous avons entrepris une évaluation comparative des performances de plusieurs modèles d'apprentissage profond, notamment des réseaux de neurones artificiels (ANN), avec et sans l'utilisation de l'Analyse en Composantes Principales (ACP), sur un ensemble de données composé de 5962 enregistrements. Notre objectif principal était de déterminer le modèle offrant les meilleures performances en termes de précision, de rappel, de score F1 et d'exactitude. Dans la phase de prétraitement des données, différentes techniques ont été appliquées. Pour coder les variables liées aux symptômes, nous avons opté pour la représentation en vecteur de caractéristiques binaires, tandis que les variables cibles ont été encodées en utilisant l'encodage de label. De plus, notre processus de prétraitement comprenait une phase de sélection des caractéristiques significatives et l'élimination des valeurs nulles, les données étant nettoyées selon cette méthode avant d'être utilisées pour l'entraînement et le test des modèles. Les résultats obtenus ont ensuite été analysés et comparés afin de déterminer le modèle présentant les performances les plus prometteuses pour la tâche de classification considérée.

1. Evaluation des modèle classification

Dans notre étude, nous avons sélectionné trois mesures de performance essentielles pour évaluer les modèles : la précision, le rappel et l'exactitude (Accuracy en anglais). Ces mesures ont été choisies pour fournir une évaluation holistique des performances des modèles dans la classification des données médicales. En plus de ces métriques, nous avons également pris en compte le score F1,

|  |  |  |  |
| --- | --- | --- | --- |
|  | | prédite | |
| Positive | Négative |
| Réelle | positive | VP | FP |
| Négative | FN | VN |

* **Rappel :**

Le rappel R est une métrique qui mesure à quelle fréquence un modèle identifie correctement les instances positives (vrais positifs) parmi tous les échantillons positifs réels dans l'ensemble de données.

R=

* **Précision :**

La précision (P) mesure la proportion d'échantillons correctement identifiés comme positifs parmi tous les échantillons identifiés comme positifs par le modèle.

P=

* **L'exactitude :**

L'exactitude (E) est une mesure qui évalue la proportion totale d'échantillons correctement classés parmi tous les échantillons dans l'ensemble de données. C'est-à-dire qu'elle mesure la capacité globale du modèle à prédire correctement les classes de tous les échantillons, qu'ils soient positifs ou négatifs.

E=

* **Score f1 :**

Le score F1 (F) est une métrique de performance qui combine à la fois la précision et le rappel d'un modèle en une seule valeur. Il offre une mesure globale de la capacité du modèle à classer correctement les échantillons positifs tout en minimisant à la fois les faux positifs et les faux négatifs.

F=

1. Optimisation des models de classification

**8-1 Adam (Estimation de Moment Adaptatif) :**

Ce sont les étapes principales de l'algorithme Adam pour mettre à jour les poids pendant le processus d'entraînement. L'algorithme Adam combine la prise en compte des moments 1 et 2 ainsi que le taux d'apprentissage pour mettre à jour les poids, ce qui lui permet d'améliorer la stabilité et la vitesse du processus d'entrainement dans de nombreux cas.

**Initialisation** : Les poids du modèle sont initialisés de manière aléatoire.

**Calcul des gradients** : Les gradients des poids du modèle par rapport à la fonction de coût sont calculés à l'aide de la technique de la rétropropagation du gradient.

Mise à jour des moments 1 et 2 : Les moments 1 et 2 sont mis à jour en utilisant les formules suivantes :

mt = β1 \* mt -1 + (1 - β1) \* gradient\_t

Vt = β2 \* Vt -1 + (1 - β2) \* gradient\_t2

( t est le temps actuel et β1 et β2 sont des paramètres qui contrôlent l'importance des moments.)

**Correction des biais 1 et 2** : Les biais 1 et 2 sont corrigés pour compenser les biais initiaux :

mt\_corr = mt (1 − β1t)

vt\_corr = vt (1 − β2t)

**Mise à jour des poids** : Les poids sont mis à jour en utilisant la formule suivante :

Wt+1 = Wt − (lr (𝑠𝑞𝑟𝑡(𝑣𝑡 \_𝑐𝑜𝑟𝑟)+𝑒𝑝𝑠𝑖𝑙𝑜𝑛))∗𝑚𝑡\_𝑐𝑜𝑟𝑟

( lr est le taux d'apprentissage et epsilon est une petite valeur pour éviter la division par zéro.)

**Répétition :** Les étapes 2 à 5 sont répétées jusqu'à ce qu'une condition d'arrêt spécifiée soit remplie.

**8-2 SGD (Descente de Gradient Stochastique) :**

Ce sont les principales étapes de l'algorithme SGD pour mettre à jour les poids pendant le processus d'entraînement. Les poids sont mis à jour en fonction du gradient calculé et du taux d'apprentissage. Les étapes sont appliquées à chaque échantillon ou lot de données dans le cas de l'entraînement par mini-lots (mini-batch training) ou à chaque donnée dans le cas de l'entraînement par lots complets (batch training). Le processus est répété jusqu'à ce que les conditions spécifiées, telles que le nombre de cycles ou l'atteinte de la précision souhaitée, soient remplies

**Initialisation**

Les poids du modèle sont initialisés de manière aléatoire.

**Calcul des gradients**

Les gradients des poids du modèle par rapport à la fonction de coût sont calculés en utilisant la technique de la rétro propagation du gradient.

**Mise à jour des poids**

Les poids sont mis à jour en utilisant la formule suivante :

Wt+1 = Wt - lr \* gradient(t)

(où Wt représente les poids actuels, Wt+1 est le poids mis à jour, et lr est le taux d'apprentissage.)

**Répétition :**

Les étapes 2 et 3 sont répétées pour chaque donnée d'entraînement ou pour un nombre spécifié de cycles (epochs) jusqu'à ce qu'un critère d'arrêt spécifié soit atteint

1. Pertes des modèle classification :

* **entropie croisée catégorielle éparses**

La fonction de perte "entropie croisée catégorielle éparses", est couramment utilisée dans les problèmes de classification où les étiquettes sont fournies sous forme d'entiers plutôt que sous forme de vecteurs one-hot. Elle mesure la différence entre les prédictions du modèle et les étiquettes réelles associées aux données. La formule mathématique de la fonction de perte entropie croisée catégorielle éparses est la suivante :

SCE(*y*,*t*)=−∑*i*​*ti*​⋅log(*yi*​)

- SCE représente l’entropie croisée catégorielle éparses.

*- y* est la sortie (prédiction) produite par le modèle.

*- t* est la valeur réelle cible (étiquette), où *ti*​ est soit 0 soit 1.

*-* log représente le logarithme naturel.

*-* La somme est effectuée sur toutes les classes *i*.

Cette fonction de perte mesure la dissimilarité entre la distribution de probabilité prédite par le modèle ( *y* ) et la distribution de probabilité réelle (*t*) pour chaque exemple dans l'ensemble de données. Elle est souvent utilisée dans les tâches de classification avec plusieurs classes.

10 – Conclusion

Dans ce chapitre, nous avons clarifié la stratégie générale de notre travail en exposant clairement nos objectifs. Nous avons décrit la méthodologie adoptée pour mener une étude comparative entre deux modèles de classification des symptômes, combinés à différentes méthodes d’optimisation été adopté. Nous avons expliqué que la mise en œuvre de cette comparaison sur un corpus de données réelles sera détaillée dans le prochain chapitre, où nous présenterons également les résultats obtenus, ainsi que leur analyse approfondie et leur discussion.