République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Alger 1 – BENYOUCEF Benkhedda

Faculté des Sciences
Département Informatique

Rapport

Module : Intelligence Artificielle

Réalisé par :

Moulay Abdellah Asma
Belabbas Rania

Section : M1 ISII groupe 3

Année universitaire : 2024/2025

1. Régression logistique :

La régression logistique est un modèle statistique permettant d'étudier les relations entre un ensemble de variables qualitatives Xi et une variable qualitative Y. Il s'agit d'un modèle linéaire généralisé utilisant une fonction logistique comme fonction de lien.

Un modèle de régression logistique permet aussi de prédire la probabilité qu'un événement arrive (valeur de 1) ou non (valeur de 0) à partir de l'optimisation des coefficients de régression. Ce résultat varie toujours entre 0 et 1. Lorsque la valeur prédite est supérieure à un seuil, l'événement est susceptible de se produire, alors que lorsque cette valeur est inférieure au même seuil, il ne l'est pas.

Nous disposons de deux fichiers CSV contenant les vecteurs de caractéristiques des images pour deux classes :

- 0 : Image Normal
- 1 : Image Pneumonia

Et 2 fichiers csv obtenu lors des premiers TP:

Premier fichier (4 colonnes de caractéristiques) :

- Une colonne contenant le vecteur des caractéristiques pour chaque méthode d'extraction (Gabor, DCT, Fourier, PHOG).
- Une colonne "Label" contenant les étiquettes des classes.

Deuxième fichier (1 colonne de caractéristiques) :

- Une colonne combinant les caractéristiques des quatre méthodes sous forme de liste.
- Une colonne "Label" contenant les étiquettes des classes.

L'objectif principal est de comparer les deux représentations des données pour déterminer laquelle est la plus efficace pour classifier les images en deux catégories : Normal (0) et Pneumonia (1), en utilisant un modèle de régression logistique

Les étapes :

1- Prétraitement :

Nous avons rencontré l'erreur suivante :

```
ValueError: could not convert string to float: 'IMG-001_NORMAL.jpeg'
```

Explication : indique le modèle tente de traiter une colonne contenant des chaînes de caractères comme des données numériques, car dans les anciens fichiers en plus du label on avait ajouté une colonne image qui contient le nom de l'image et Les algorithmes comme la régression logistique nécessitent que toutes les caractéristiques soient numériques

Solution : Consiste à supprimer cette colonne avant d'appliquer des transformations ou d'entraîner le modèle

Note : La colonne contenant le nom de l'image n'est pas nécessaire dans ce cas, donc elle peut être supprimée.

Nous avons rencontré aussi l'erreur suivante :

```
ValueError: could not convert string to float: '220.0976715200272,120.08968459165332,48.636990435772375,6.918581228980074,5.325
952946266243,4.814126935893408,240.77114199041003,1.1266433526815376,46.2802104954136,223.15785919932426,1.9243153877452217,3.1
34182199308821,220.10431034038763,120.0882865251003,48.628234557918425,6.06778472504777,4.809019576536998,4.574975149489341,24
0.7689968634278,1.1246578870112947,46.28199921714454,223.05490984286237,1.9302496381868655,3.1281959301816533'
```

Explication : L'erreur indique que certaines colonnes des fichiers CSV contiennent des valeurs non formatées comme des nombres. Par exemple, une cellule contient une chaîne de caractères ressemblant à une liste de nombres séparés par des virgules, au lieu d'un seul nombre.

Solution : Nous avons vérifié si une colonne contient des chaînes de caractères (dtype == object), ensuite convertir les chaînes de caractères en listes de nombres, puis calculez leur moyenne.

Note : dans la régression logistique il faut utiliser **StandardScaler** de la bibliothèque **sklearn** pour normaliser les données, elle permet de réduire l'échelle des données à une distribution avec une moyenne de 0 et un écart-type de 1. Cela améliore les performances des algorithmes d'apprentissage machine.

2- Fonction cout :

Calcule l'erreur moyenne (coût) entre les prédictions h et les vraies étiquettes y.

3- Division en ensembles d'entraînement et de test :

Nous avons divisé les données en un ensemble d'entraînement qui représente environ 70 % des données et un ensemble de test constitué des 30 % restants. Cette division permet d'entraîner le modèle sur une partie des données et de tester sa performance sur des données inconnues, ce qui simule une situation réelle où le modèle est utilisé sur de nouvelles données.

4- Descente de gradient :

Avant l'entraînement, les paramètres du modèle (theta) sont initialisés à zéro. Ces paramètres servent à pondérer les caractéristiques pour faire des prédictions. La descente de gradient est ensuite utilisée pour ajuster les valeurs de theta.

C'est une méthode d'optimisation itérative qui calcule l'erreur entre les prédictions et les vraies valeurs, puis met à jour les paramètres pour minimiser cette erreur. À chaque itération, le coût est recalculé pour suivre l'amélioration du modèle.

Résultat Fonction cout et descente de gradient pour les deux fichiers:

```
Iteration 0: Cost 0.691653140640382
Iteration 100: Cost 0.570753660871383
Iteration 200: Cost 0.4901295337630418
Iteration 300: Cost 0.4331077082894568
Iteration 400: Cost 0.3905331012624052
Iteration 0: Cost 0.6889362749720372
Iteration 100: Cost 0.43595427781656765
Iteration 200: Cost 0.3319113340941149
Iteration 300: Cost 0.2747859134108885
Iteration 400: Cost 0.23783239752700305
```

- Le coût diminue régulièrement au fil des itérations, ce qui montre que le modèle apprend efficacement.
- La descente de gradient atteint des coûts finaux bas pour les deux fichiers, ce qui indique une bonne minimisation de l'erreur.

Une fois le modèle entraîné, il est appliqué à l'ensemble de test pour prédire les étiquettes des échantillons. À chaque prédiction, la fonction sigmoïde (fonction d'activation de la régression logistique) est utilisée pour générer une probabilité, et une règle de décision (un seuil) est appliquée pour déterminer la classe de sortie.

5- Evaluation :

Les performances du modèle sont évaluées et la comparaison des deux représentations des données se fait à l'aide de métriques suivantes :

- **Précision**
  - Définit la proportion des prédictions positives correctes par rapport à toutes les prédictions positives.
  - Indique la capacité du modèle à ne pas classifier des images normales (classe 0) comme Pneumonia (classe 1).
- **Rappel (Recall)**
  - Définit la proportion des vraies instances positives correctement identifiées
  - Utile pour évaluer la capacité du modèle à détecter les cas de Pneumonia (classe 1).
- **F1-Score**
  - Combine la précision et le rappel en une métrique unique.
  - Fournit un équilibre entre les faux positifs et les faux négatifs.
- **Accuracy :**
  - Mesure globale de la proportion des prédictions correctes.
  - Fournit une vue d'ensemble des performances mais peut être biaisée si les classes sont désquilibrées.

Résultat dévaluations :

Résultats pour le fichier 1 :
Accuracy : 0.9545454545454546
Rapport de classification :

	precision	recall	f1-score	support
0	1.00	0.93	0.96	14
1	0.89	1.00	0.94	8
accuracy			0.95	22
macro avg	0.94	0.96	0.95	22
weighted avg	0.96	0.95	0.96	22

Résultats pour le fichier 2 :
Accuracy : 0.9545454545454546
Rapport de classification :

	precision	recall	f1-score	support
0	1.00	0.93	0.96	14
1	0.89	1.00	0.94	8
accuracy			0.95	22
macro avg	0.94	0.96	0.95	22
weighted avg	0.96	0.95	0.96	22

Note : Accuracy et les autres métriques dévaluation sont calculées grâce aux fonctions prédéfinis (accuracy_score …) de sklearn.

Explication :

Pour les deux fichiers : Accuracy élevée (95.45%), Le modèle est bien entraîné et offre des résultats cohérents, avec une capacité robuste à distinguer les classes 0 et 1.

Les caractéristiques extraites des deux formats de fichiers capturent les mêmes informations essentielles, bien que présentées différemment donc le modèle les utilisera de manière similaire, ce qui peut expliquer pourquoi les résultats (comme la précision …) sont identiques.

2. CNN (convolutional neural networks) :

Un réseau neuronal convolutif est une architecture réseau pour le Deep Learning qui apprend directement à partir des données. Ce type de réseau est particulièrement efficace lorsqu'il s'agit de trouver des patterns dans des images afin de reconnaître des objets, des classes et des catégories.

Objectif : Utiliser un CNN pour classer des images "pneumonia" ou "normal, et comparer ses performances avec celles d'une régression logistique appliquée aux fichiers CSV traités précédemment.

Pour la classification des images médicales, certaines architectures CNN sont plus couramment préférées en raison des exigences spécifiques des données médicales, telles qu'une grande précision, la capacité de généralisation à partir d'ensembles de données relativement petits et la robustesse : **ResNet (réseaux résiduels)**.

Points forts : les connexions de saut de ResNet aident à atténuer le problème de gradient de fuite, permettant la formation de réseaux très profonds, ce qui est utile pour capturer des caractéristiques complexes dans les images médicales.

Cas d'utilisation : couramment utilisé dans diverses tâches d'imagerie médicale telles que la détection de pneumonies, de tumeurs cérébrales, etc.

ResNet est un excellent choix pour le dataset pour plusieurs raisons :

- Capacité de réseau profond : l'architecture de ResNet, avec ses connexions résiduelles, permet des réseaux plus profonds, qui peuvent capturer des modèles plus complexes dans les données, améliorant potentiellement les performances de classification.
- Gestion de petits ensembles de données : il a été démontré que ResNet fonctionne bien même sur des ensembles de données relativement petits, ce qui est typique de l'imagerie médicale.
- Modèles pré-entraînés : vous pouvez utiliser un ResNet pré-entraîné (par exemple, ResNet-50 ou ResNet-101) sur ImageNet et l'affiner sur votre ensemble de données spécifique. Cette approche d'apprentissage par transfert est très efficace pour les tâches de classification d'images médicales, en particulier lorsque l'ensemble de données n'est pas très volumineux.
  Robustesse : les modèles ResNet sont connus pour leur robustesse et leur capacité de généralisation, ce qui les rend particulièrement adaptés à la classification d'images médicales où une grande précision est cruciale.

1. **<u>Importation des Bibliothèques Nécessaires :</u>**

tensorflow : C'est la bibliothèque principale pour le machine learning et l'apprentissage profond. Elle contient toutes les fonctions nécessaires pour créer, entraîner et évaluer des modèles de réseaux neuronaux.

2. **<u>Préparation des Données avec ImageDataGenerator :</u>**

Nous avons préparé les données d'entraînement et de validation en utilisant ImageDataGenerator pour normaliser les images et appliquer éventuellement des augmentations.

```
Found 59 images belonging to 2 classes.
Found 14 images belonging to 2 classes.
```

Ce résultat indique que ImageDataGenerator a trouvé :

59 images appartenant à 2 classes dans le sous-ensemble d'apprentissage. 14 images appartenant à 2 classes dans le sous-ensemble de validation. Ces chiffres représentent le nombre d'images trouvées et allouées à l'apprentissage et à la validation en fonction de l'ensemble de données total et du ratio de répartition que vous avez défini (dans votre cas, 80 % pour l'apprentissage et 20 % pour la validation).

Explication :

Nombre total d'images : l'ensemble de données contient un total de 73 images (59 pour l'apprentissage + 14 pour la validation).

Ratio de répartition : on a utilisé validation_split=0.2, ce qui signifie que 20 % du total des images sont utilisées pour la validation et les 80 % restants sont utilisés pour l'apprentissage (environ 80 % de 73 correspondent à 59 et 20 % à 14).

Nous avons rencontré les erreurs suivantes :

A. Invalid image file

Message : Invalid image file : IMG-001
Cause : Les fichiers d'image avaient des extensions incorrectes (.jpg.jpg au lieu de .jpg) ou étaient corrompus.
Solution : Vérification et correction des extensions des fichiers dans le répertoire.

B. Doublons dans les fichiers

Message : File already exists: IMG-001.jpg.jpg
Cause : Des doublons d'images existaient dans le répertoire.
Solution : Nettoyer le répertoire pour supprimer les doublons.

3. Chargement du Modèle Pré-entraîné ResNet50 :

Nous avons chargé le modèle ResNet50 pré-entraîné sans les couches supérieures (sans la tête de classification).

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 64s 1us/step
```

Ce message indique que les pondérations pré-entraînées pour ResNet50 sont en cours de téléchargement depuis le stockage de TensorFlow. Plus précisément :

URL : les pondérations sont téléchargées depuis l'emplacement hébergé par TensorFlow.

Fichier : resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5 contient les pondérations pré-entraînées pour ResNet50 sans les couches supérieures (entièrement connectées).

Taille : le fichier fait environ 94,76 Mo.

Durée : le téléchargement a duré environ 64 secondes.

C. Compilation du Modèle :

Nous avons compilé le modèle avec une fonction de perte adaptée et un optimiseur, ce qui implique la configuration de la formation, y compris la fonction de perte, l'optimiseur et les mesures d'évaluation.

1-Gel (Freeze) des couches du modèle de base :

Geler les couches du modèle de base ResNet50 pour éviter qu'elles ne soient mises à jour pendant la phase de formation initiale. Cela garantit que les poids pré-entraînés d'ImageNet ne sont pas modifiés. Le gel s'effectue en définissant layer.trainable = False pour chaque couche du modèle de base.

2-Ajout de la fonction de perte

Spécifier l'entropie croisée binaire (binary_crossentropy) comme fonction de perte, appropriée pour les tâches de classification binaire telles que la distinction entre les cas NORMAL et PNEUMONIA. L'entropie croisée binaire mesure la différence entre la probabilité prédite et l'étiquette binaire réelle (0 ou 1).

3-Choix d'un optimiseur :

L’optimiseur Adam, un choix populaire pour la formation de modèles d'apprentissage profond en raison de son efficacité et de ses capacités de taux d'apprentissage adaptatif. Le taux d'apprentissage est fixé à 0,001 pour la phase de formation initiale.

4-Définition des mesures d'évaluation :

Spécifier la précision comme mesure d'évaluation, qui suivra la proportion d'images correctement classées pendant la formation et la validation.

D. Entraînement du Modèle :

Nous avons entraîné le modèle sur les données d'entraînement et de validation sur 10 époques.

```
Epoch 1/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 6s 3s/step - accuracy: 0.4988 - loss: 0.7359 - val_accuracy: 0.5714 - val_loss: 0.6871
Epoch 2/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 5s 3s/step - accuracy: 0.5925 - loss: 0.6495 - val_accuracy: 0.5714 - val_loss: 0.8096
Epoch 3/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 5s 3s/step - accuracy: 0.5817 - loss: 0.6816 - val_accuracy: 0.5714 - val_loss: 0.6767
Epoch 4/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 5s 3s/step - accuracy: 0.5960 - loss: 0.6672 - val_accuracy: 0.3571 - val_loss: 0.7123
Epoch 5/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 6s 3s/step - accuracy: 0.5934 - loss: 0.6488 - val_accuracy: 0.6429 - val_loss: 0.6910
Epoch 6/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 6s 3s/step - accuracy: 0.7769 - loss: 0.6134 - val_accuracy: 0.5714 - val_loss: 0.7202
Epoch 7/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 6s 3s/step - accuracy: 0.5404 - loss: 0.6164 - val_accuracy: 0.5714 - val_loss: 0.7397
Epoch 8/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.5821 - loss: 0.6079 - val_accuracy: 0.6429 - val_loss: 0.7036
Epoch 9/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 6s 3s/step - accuracy: 0.6187 - loss: 0.5851 - val_accuracy: 0.7143 - val_loss: 0.7132
Epoch 10/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 4s/step - accuracy: 0.7350 - loss: 0.6017 - val_accuracy: 0.4286 - val_loss: 0.7901
```

Nous pouvons voir les résultats de l'entraînement et de la validation après chaque époque. Voici une répartition des informations clés :

- **Mesures d'entraînement :**
  - **Précision** : la précision de l'entraînement s'améliore progressivement de 49,88 % à l'époque 1 à 73,50 % à l'époque 10.
  - **Perte** : la perte d'entraînement diminue au fil du temps, commençant à 0,7359 à l'époque 1 et tombant à 0,6017 à l'époque 10.
- **Mesures de validation :**
  - **Précision** : la précision de validation fluctue au fil des époques, allant de 35,71 % à 71,43 %.
  - **Perte** : la perte de validation fluctue également, atteignant un pic à 0,8096 dans l'époque 2 et se terminant à 0,7901 dans l'époque 10.

Observations :

- **Sur-ajustement** : le modèle montre des signes de sur-ajustement, car la précision de l'apprentissage augmente considérablement, tandis que la précision de la validation reste relativement plus faible et plus instable. Cela suggère que le modèle apprend à bien s'adapter aux données d'apprentissage, mais ne se généralise pas efficacement aux données de validation.
- **Précision de validation fluctuante** : la précision de validation ne montre pas d'amélioration constante, ce qui indique que le modèle n'apprend peut-être pas efficacement à partir des données de validation.

Suggestions d'amélioration :

- **Augmentation des données** : envisagez d'appliquer une augmentation des données plus forte ou plus diversifiée (par exemple, en augmentant la rotation, le zoom ou le décalage) pour aider le modèle à mieux généraliser.
- **Régularisation** : introduisez des techniques de régularisation telles que la régularisation Dropout ou L2 pour éviter le sur-ajustement.
- **Ajustement du taux d'apprentissage** : vous devrez peut-être affiner le taux d'apprentissage ou utiliser un planificateur de taux d'apprentissage.
- **Augmenter la taille de l'ensemble de données** : si possible, essayez de collecter davantage de données, en particulier pour la classe PNEUMONIE, qui pourrait être sous-représentée.

E. Evaluation du modèle :

Après l'entraînement, nous avons évalué le modèle sur le jeu de données de validation pour obtenir la précision et la perte.

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 0.6429 - loss: 0.6728
Validation accuracy: 64.29%
```

Le modèle a atteint une précision de 64,29 % sur l'ensemble de validation, ce qui signifie que pour les images qu'il a évaluées, il a correctement classé environ 64,29 % d'entre elles.

La perte de validation de 0,6728 indique à quel point les prédictions du modèle étaient éloignées des valeurs réelles, une valeur inférieure étant généralement meilleure.

4. Comparaison entre les deux méthodes :

1.Régression Logistique (95% de Précision) :

- **Nature du modèle** : La régression logistique est un modèle **linéaire** qui est souvent utilisé pour des problèmes simples de classification binaire. Dans notre cas, il semble que les deux classes (NORMAL et PNEUMONIA) soient relativement faciles à séparer.
- **Précision** : **95%**.
    - Cela signifie que **95% des prédictions sont correctes**, ce qui est une excellente performance pour un modèle aussi simple. Cela suggère que la séparation entre les classes est bien définie et que le modèle n'a pas besoin de structures complexes pour bien prédire.
- **Avantages** :
    - **Rapidité d'entraînement** : La régression logistique est rapide à entraîner, surtout avec un jeu de données relativement petit.
    - **Simplicité et interprétabilité** : Le modèle est facile à comprendre et à interpréter.
    - **Moins de risques de surajustement (overfitting)**, surtout avec des petits jeux de données.
- **Limites** :
    - **Manque de flexibilité** : La régression logistique est limitée par sa capacité à modéliser des relations non linéaires. Si les classes ne sont pas linéairement séparables, les performances peuvent chuter.

2. ResNet (64.29% de Précision) :

- **Nature du modèle** : ResNet est un modèle d'apprentissage profond avec des couches convolutives (CNN), conçu pour apprendre des caractéristiques complexes dans les images. Il est puissant pour traiter des données volumineuses et des tâches de classification d'images plus complexes.
- **Précision** : **64.29%**.
    - Cela signifie que **64.29% des prédictions sont correctes**. Bien que ce soit une performance correcte, elle est bien inférieure à celle de la régression logistique dans ce cas spécifique. Cela peut être dû à plusieurs raisons, comme un manque de données suffisantes, des ajustements des hyperparamètres nécessaires, ou simplement la difficulté de la tâche pour un modèle profond sur un petit jeu de données.
- **Avantages** :
    - **Capacité à apprendre des caractéristiques complexes** : ResNet est capable d'extraire des caractéristiques très complexes des images, ce qui est utile pour des tâches plus difficiles.
    - **Adapté à de grandes quantités de données** : Si le jeu de données était plus grand, la performance de ResNet aurait pu s'améliorer.
- **Limites** :
    - **Besoin de plus de données** : ResNet fonctionne mieux avec de grandes quantités de données, et un jeu de données plus petit peut entraîner un **surajustement (overfitting)**, ce qui expliquerait les performances relativement faibles.
    - **Temps d'entraînement élevé** : Le modèle prend plus de temps à entraîner et nécessite souvent une plus grande puissance de calcul.
    - **Nécessite un ajustement fin** : ResNet a plusieurs hyperparamètres à ajuster pour optimiser les performances, et cela peut nécessiter plus d'expérimentation.

Conclusion :

- **La régression logistique** a montré une **très bonne performance** avec **95% de précision**, ce qui signifie que le problème était relativement simple à modéliser pour un modèle linéaire. Cela peut indiquer que les classes dans les images (NORMAL et PNEUMONIA) sont suffisamment séparables avec une approche linéaire.
- **ResNet**, bien que très puissant, a montré **64.29% de précision**, ce qui est **inférieur** à la régression logistique dans ce cas. Cela peut être dû au fait que **ResNet a besoin de plus de données** pour être vraiment efficace et pour apprendre les bonnes caractéristiques. Le modèle pourrait également avoir besoin de réglages plus fins (comme l'ajustement des hyperparamètres, de l'architecture, ou de l'augmentation des données).
- Si le **jeu de données est relativement petit** et bien séparé, il semble que **la régression logistique** soit un excellent choix.
- Si **les données sont plus complexes** ou si on ajoute plus de données, **ResNet** pourrait potentiellement surpasser la régression logistique après un meilleur ajustement et plus de ressources de calcul.