Rapport du projet Deep Learning. Réalisé par : HANINE Aymane

I - Introduction :

Ce travail consiste à modéliser deux cas différents. Le premier cas vise à prévoir le label d'un mail (spam ou non-spam) en utilisant des caractéristiques du type "bags of words". Le second cas consiste à prévoir la concentration en ozone à partir des conditions météorologiques et de la concentration d'ozone de la veille. L'objectif de ce travail est de développer une modélisation efficace pour ces deux cas en utilisant des techniques d'apprentissage automatique et profond.

Points techniques :

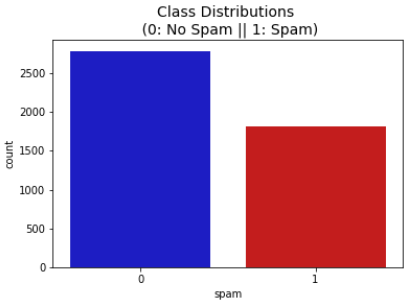
- *Pour ce rendu, j'ai utilisé le langage python avec des librairies comme : tensorflow, keras, pandas, sklearn…*
- *Pour la reproductibilité de ce travail j'ai défini la graine aléatoire pour numpy et TensorFlow (peu importe le moment ou l'environnement dans lequel ce travail est exécuté, les mêmes séquences aléatoires seront générées)*
- *J'ai réalisé un split de 0,7 et 0 ,3 pour le train et le test afin de garantir une référence standard pour la comparaison des modèles*
- *Chaque problème est accompagné de 3 fichiers : datasets (pour stocker les données test et train), history (stocker le callback des réseaux de neurones sur l'ensemble de validation) et finalement les modèles (pour les modèles entrainés)*

II - Problème 1 : Prévision du label d'un mail

Le problème consiste à prévoir le label d'un courrier électronique (spam 1 ou non-spam 0) en se basant sur des caractéristiques du type "bags of words". Le but est de classification supervisée dont l'objectif est de minimiser les erreurs de classification et de maximiser la précision de notre modèle. La méthode choisie pour aborder ce problème sera décrite en détail dans les sections suivantes.

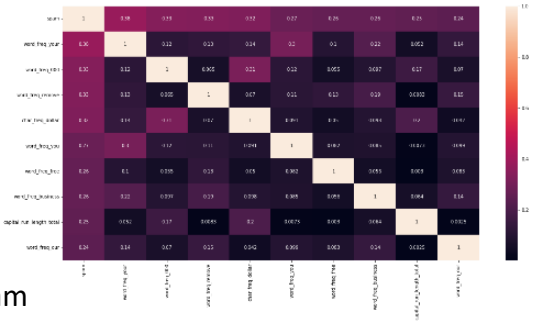
1 - Prétraitement :

La base de données "spam" contient des informations sur les courriels classés en spam ou non-spam avec 4601 échantillons. Elle comprend 56 variables qui représentent la fréquence de certains mots, des caractères spécifiques et des longueurs de séquences en capitales dans chaque courriel. Enfin, la dernière variable, "spam", représente le label de chaque courriel, soit "spam" ou "non-spam".



- On remarque que les données sont distribuées en 2788 non-spam et 1813 spam. On peut donc la considérer comme non déséquilibré et par conséquent aucun traitement supplémentaire, concernant l'équilibre, n'est pas requis.

- Ensuite, j'ai évalué la corrélation entre les variables dépendantes et indépendantes pour ressortir un peu les corrélations :



  - Positives : pourraient indiquer que certaines caractéristiques des mails telles que la fréquence d'utilisation de mots spécifiques ou le nombre de caractères en majuscules dans le corps du message sont associées à un plus grand risque de recevoir un mail de spam comme (word_freq_your, word_freq_000, word_freq_remove et char_freq_dollar)
  - Négatives : une corrélation négative pourrait signifier que ces mêmes caractéristiques sont associées à une probabilité moindre de recevoir un mail de spam, comme (word_freq_hp, word_freq_hpl, word_freq_george word_freq_1999)

- La standardisation : cette étape consiste à centrer les données autour de la moyenne et à les normaliser en divisant par l'écart-type. Cela permet de traiter les données de manière uniforme et de réduire l'influence de caractéristiques ayant une échelle différente sur les résultats de modélisation. Pour cela j'utilise StandardScaler de sklearn

2 - Modèles

Dans ce projet, je vais explorer différents modèles pour résoudre le problème de classification des emails. Les modèles comprennent les réseaux de neurones profonds (**DNN**), les réseaux de neurones à convolution (**CNN**), la régression logistique (**LR**), le **SVC**, qui est un classificateur à base de support vector machine, le classificateur polynomial (**PolyC**), la forêt aléatoire pour la classification (**RFC**) et le Gradient Boosting Classifier (**GBC**). Je vais évaluer les performances de ces modèles pour déterminer le meilleur pour résoudre le problème de détection de spam.

J'ai entrainé chaque algorithme selon ses spécificités et selon les hyperparamètres optimaux, comme suit :

- **DNN :** j'ai défini un réseau de neurone avec une architecture en séquence, qui comprend plusieurs couches de neurones avec différents nombres de neurones (en utilisant la fonction d'activation ReLU et a un nombre de neurones, couche de Dropout pour éviter l'overfitting, l'optimiseur 'adam', la fonction de coût 'binary_crossentropy' et les métriques 'accuracy'.) Finalement, j'ai utilisé un objet GridSearchCV est créé pour effectuer une recherche sur grille sur les meilleurs hyperparamètres définis dans un dictionnaire contenant les nombres de neurones à tester pour chaque couche et une validation croisée de 5 pour trouver les meilleurs hyperparamètres.
- **CNN** : Le modèle comporte 3 couches, notamment des couches Conv1D, MaxPool1D, Flatten et Dense. Le modèle est ensuite compilé en utilisant l'optimiseur Adam, une fonction de perte binary_crossentropy et la métrique 'accuracy'. Comme pour le DNN une grille de paramètres est définie pour explorer différentes configurations de filtres

Conv1D et de neurones Dense. La grille est passée à un objet GridSearchCV, qui effectue une validation croisée sur les données d'entraînement pour avoir le meilleur ensemble de hyperparamètres

- **LR** : j'ai entraîné un simple modèle de régression logistique, un algorithme populaire dans la classification.
- **SVC** : j'ai entrainé un SVC, qui fonctionne en trouvant une séparation linéaire entre les données de différentes classes. Il utilise un noyau pour transformer les données d'entrée dans un espace de plus grande dimension, où une séparation linéaire peut être trouvée.
- **PolyC** : la classification polynomiale a une particularité d'utiliser des fonctions polynomiales de degré élevé pour représenter les relations entre les caractéristiques et les classes.
- **RFC** : est un algorithme de classification basé sur les arbres de décision. Pour cela, j'ai défini d'abord un réseau de hyperparamètres (nombre d'estimateurs, profondeur maximale des arbres, nombre minimum d'échantillons pour faire une split et le nombre minimum de d'échantillons requis dans une feuille) qui peut être utilisé pour affiner les performances du modèle en testant différentes combinaisons d'hyperparamètres. Nous utilisons également GridSearchCV pour effectuer un ajustement de hyperparamètres afin de trouver la meilleure combinaison d'hyperparamètres pour notre modèle. Finalement, nous formons le modèle de forêt aléatoire en utilisant les hyperparamètres les plus performants et enregistrons le modèle formé pour une utilisation ultérieure.
- **GBC** : comme le RFC, c'est un algo de classification basé sur les arbres de décisions avec une particularité d'utiliser un algorithme d'optimisation pour ajouter des arbres de décision de manière séquentielle pour corriger les erreurs du modèle précédent, tandis que le random forest construit simultanément plusieurs arbres de décision indépendants et utilise la majorité pour faire des prédictions

3 - Résultats

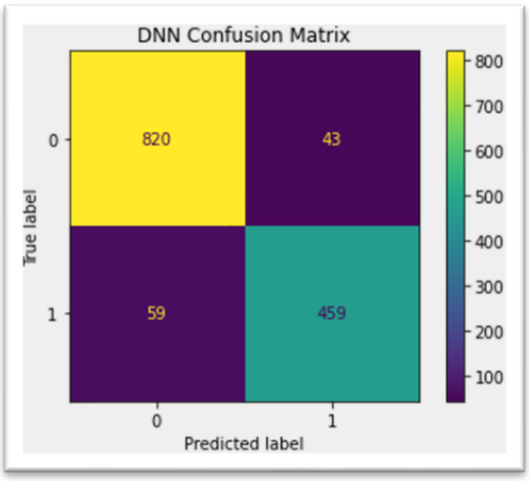
J'ai évalué les performances de ces algorithmes en utilisant deux métriques couramment utilisées en machine learning, à savoir le rappel (recall) et la F1. La recall mesure la capacité du modèle à détecter les courriels de spam réels, tandis que le F1 score est un indicateur de la précision globale du modèle, prenant en compte à la fois la recall et le taux de vrais négatifs. En utilisant ces deux métriques, j'ai pu obtenir une vue complète de la performance des différents algorithmes et identifier celui qui convient le mieux à la tâche de détection de spam dans les emails.

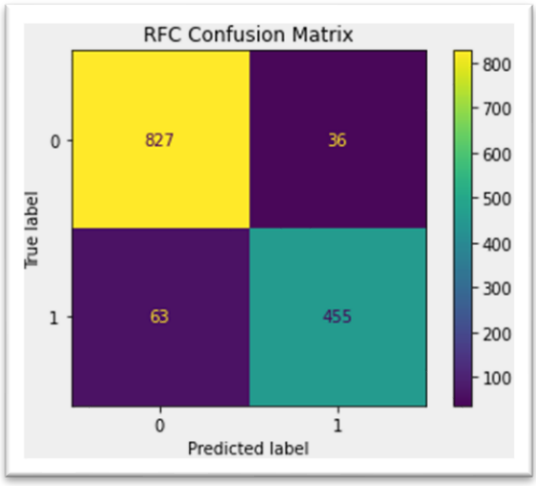
On peut voir les résultats complets dans ce tableau :

Metric	DNN	CNN	LR	SVC	PolyC	RFC	GBC
Recall	0.909266	0.818533	0.901544	0.889961	0.503861	0.8861	0.874517
F1	0.893738	0.855701	0.889524	0.891683	0.441624	0.906219	0.884766

Comme on peut le voir les DNN et le RFC ont donné des résultats assez proches avec un rappel légèrement élevé pour les DNN et une F1 légèrement élevée pour le RFC.

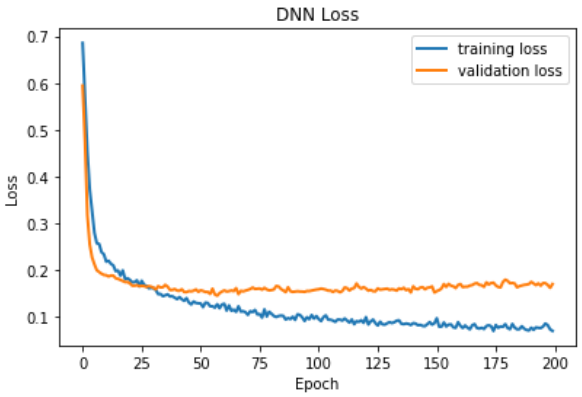
Visuellement on peut le voir dans la matrice de confusion.

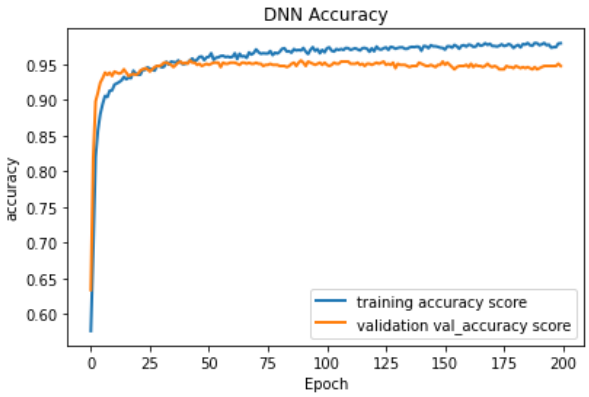




Dans ce cas, le choix entre les deux modèles (DNN et RFC) dépendra des critères de priorisation pour le modèle. Si la priorité est de maximiser la recall (c'est-à-dire de détecter le plus de courriels de spam réels possible), alors le modèle DNN serait le choix idéal car il a une recall légèrement supérieure. Cependant, si la priorité est de maximiser la précision globale du modèle (c'est-à-dire d'avoir un faible nombre de faux positifs)

Voici finalement à quoi ressemble la courbe d'apprentissage pour les réseaux de neurones :





II - Problème 2 : Prévision de la concentration en ozone

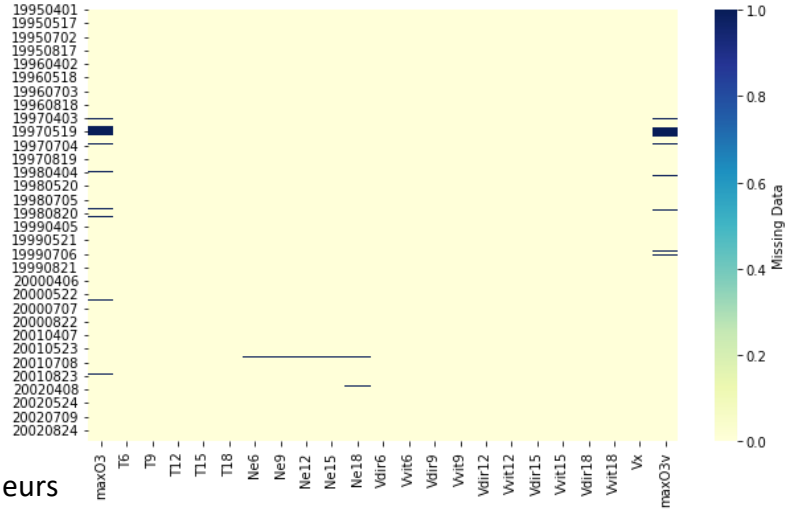
Le projet consiste à prédire la concentration en ozone (max03) à partir des conditions météorologiques et de la concentration en ozone de la veille. Il s'agit d'un problème de régression supervisée. Je vais évaluer différents algorithmes de prédiction pour déterminer le meilleur modèle pour prédire la concentration en ozone.

1 Prétraitement :

La base de données "ozone" contient des informations sur les conditions météorologiques et la concentration en ozone avec 1464 échantillons. Il contient 24 variables, y compris la date, la teneur maximale en ozone observée, la température à 9h, 12h et 15h, la nébulosité observée à 6h, 9h, 12h, 15h et 18h, la direction et la vitesse du vent observées à 6h, 9h, 12h, 15h et 18h, la composante est-ouest du vent observé et la teneur maximale en ozone observée la veille.

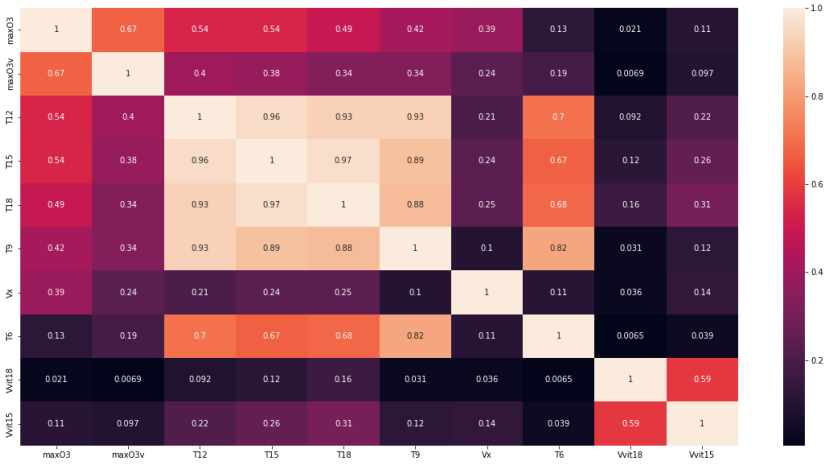
On remarque que les données présentent des valeurs manquantes et des outliers. Voici donc les étapes de prétraitement réalisé.

- En ce qui concerne les valeurs manquantes. J'ai commencé par visualiser la concentration de ces valeurs comme suit. Pour la variable cible (concentration en ozone max03), elle contient 73 valeurs manquantes que j'ai décidé de les supprimer au lieu d'imputer des valeurs, ce choix a été motivé par le fait que la feature maxO3v a généralement des valeurs manquantes dans les même jours que maxO3.



- Pour les autres quelques valeurs manquantes j'ai utilisé l'imputation itérative pour remplacer les valeurs manquantes dans les données. Cette méthode d'imputation utilise plusieurs algorithmes pour estimer les valeurs manquantes en utilisant les valeurs des autres variables dans le jeu de données.
- J'ai fait un test pour voir le pourcentage des outliers de chaque feature et ils représentent de très petites proportions (mois de 3%) donc j'ai décidé de les garder. Exemple (maxO outliers = 2.37%, T6 outliers = 1.29%, T9 outliers = 1.44%, T12 outliers = 1.01%...)

- Ensuite j'ai étudié la corrélation des features avec la variable cible pour avoir une idée sur les feature les plus corrélés



2 Modèles

Pour prédire la concentration en ozone, j'ai utilisé différents modèles de machine learning. Les modèles utilisés sont les suivants : les réseaux de neurones définis manuellement (**DNN**), les réseaux de neurones convolutionnels (**CNN**), la régression linéaire (**LR**), la régression par vecteurs de support (**SVR**), la régression polynomiale (**PR**), la forêt aléatoire (**RF**) et la régression gradient boosting (**GBR**). Chacun de ces modèles a ses propres avantages et inconvénients et convient à différents types de données. L'objectif de ce travail est de comparer les performances de ces modèles pour prédire la concentration en ozone à partir des conditions météorologiques.

J'ai entrainé chaque algorithme selon ses spécificités et selon les hyperparamètres optimaux, comme suit :

- **DNN :** j'ai défini un réseau de neurone, comme dans le premier cas) avec une architecture en séquence, qui comprend plusieurs couches de neurones avec différents nombres de neurones (en utilisant la fonction d'activation ReLU et a un nombre de neurones, couche de Dropout pour éviter l'overfitting, l'optimiseur 'adam', la fonction de coût 'mean_squared_erroret 'les métriques 'mae' (mean absolute error). Finalement, j'ai utilisé un objet GridSearchCV est créé pour effectuer une recherche sur grille sur les meilleurs hyperparamètres définis dans un dictionnaire contenant les nombres de neurones à tester pour chaque couche et une validation croisée de 5 pour trouver les meilleurs hyperparamètres.
- **CNN** : Le modèle comporte 3 couches, notamment des couches Conv1D, MaxPool1D, Flatten et Dense. Le modèle est ensuite compilé en utilisant l'optimiseur Adam, une fonction de perte mean_squared_error et la métrique mae. Comme pour le DNN une grille de paramètres est définie pour explorer différentes configurations de filtres Conv1D et de neurones Dense. La grille est passée à un objet GridSearchCV, qui effectue une validation croisée sur les données d'entraînement pour avoir le meilleur ensemble de hyperparamètres
- **LR** : j'ai entraîné un simple modèle de régression linéaire, un algorithme populaire dans la régression.
- **SVR**

- **PolyR**
- **RFR** : est un algorithme de régression basé sur les arbres de décision. Pour cela, j'ai défini d'abord un réseau de hyperparamètres (nombre d'estimateurs, profondeur maximale des arbres, nombre minimum d'échantillons pour faire une split et le nombre minimum de d'échantillons requis dans une feuille) qui peut être utilisé pour affiner les performances du modèle en testant différentes combinaisons d'hyperparamètres. Nous utilisons également GridSearchCV pour effectuer un ajustement de hyperparamètres afin de trouver la meilleure combinaison d'hyperparamètres pour notre modèle.
- **GBR**

3 *Résultats*

Pour évaluer la performance de mes prédictions, j'ai utilisé deux métriques couramment utilisées en régression, la RMSE (Erreur quadratique moyenne) et R2 (coefficient de détermination). La RMSE mesure la différence moyenne entre les valeurs réelles et prédites, tandis que le R2 mesure la proportion de la variance totale de la cible qui est expliquée par les prédictions du modèle. J'ai utilisé ces deux métriques pour comparer les performances des différents algorithmes que j'ai utilisés pour prédire la concentration en ozone.

On peut voir les résultats complets dans ce tableau :

Metric	DNN	CNN	LR	SVR	PolyR	RFR	GBR
RMSE	24.3172	13.7763	13.8679	13.8246	18.0674	12.4448	16.8959
R2	-0.21958	0.60857	0.60335	0.60583	0.32676	0.68058	0.41123

On peut voir clairement le random forest regressor qui a 12.44 en RMSE ce qui signifie que, en moyenne, la différence entre les valeurs prédites et les valeurs réelles est de 12.44 unités. La R2 quant à elle, mesure l'ajustement du modèle. Un R2 de 0.68 signifie que 68% de la variance de la variable cible peut être expliquée par les variables explicatives. Cela signifie que le modèle explique bien une partie de la variabilité de la variable cible, mais il reste encore des variations non expliquées.