Quentin LEGRAND
Arthur MORVAN

Rapport IA

ESIR3

1) Introduction

Ce rapport présente le projet effectué dans le cadre de notre cours d'IA sur l'utilisation de transformers et de modèle de langage pré entraîné. Pour cela nous avons décidé de choisir pour sujet la classification de spam et de ham à partir d'un modèle XLM-Roberta réputé dans le domaine de la modélisation multilingue, ce qui dans notre cas peut-être pratique car un mail peut-être reçu dans toutes les langues du monde. L'idée de ce projet est à terme de créer un outil prédictif qui puisse assez bien estimer si un mail en entrée est un spam ou un mail légitime. Pour cela nous avons pris pour base le dataset d'un projet Kaggle déjà existant disponible à l'adresse suivante : https://www2.aueb.gr/users/ion/data/enron-spam/ .

Nous allons donc maintenant présenter en détails les différentes étapes que nous avons pu réaliser afin d'obtenir notre classifieur, à commencer par présenter le choix du modéle chosi : XLM-Roberta.

2) Présentation du Modèle XML-Roberta

Le modèle XLM-Roberta, acronyme pour "Cross-lingual Language Model - Roberta", constitue une avancée significative dans le domaine de la modélisation linguistique pré-entraînée. Conçu par Facebook AI, ce modèle repose sur l'architecture transformers et intègre des techniques d'entraînement robustes, combinant la pré-entraînement sur de vastes données multilingues avec une version optimisée de la méthode Roberta.

Celle-ci consiste à masquer dynamiquement des portions de texte lors de l'entraînement. Cette approche permet au modèle d'acquérir une compréhension contextuelle profonde et ainsi de surpasser les barrières linguistiques. C'est pourquoi la capacité de XLM-Roberta à apprendre des représentations linguistiques pour un large éventail de langues en simultané rend ce modèle particulièrement utile et adapté à la classification de spam.

3) Dataset et Prétraitement

a) Analyse des données

Nous avons décidé de réutiliser un dataset de mails en anglais d'un projet Kaggle réalisant également une classification de mails en spam ou non. Notre dataset comprend donc 3 colonnes, constituées des étiquettes suivantes:

- label : Permet d'indiquer si la donnée constitue un spam ou ham

- text : Comprends le mail intégral, y compris le corps du mail, le sujet ainsi que l'historique des mails précédents dans le cas d'une conversation sur plusieurs mails.
- target : Associé au label, la colonne target présente pour chaque mail une classification en 0 et 1 selon s'il s'agit d'un ham ou d'un spam.

Voici donc une idée de notre dataset:

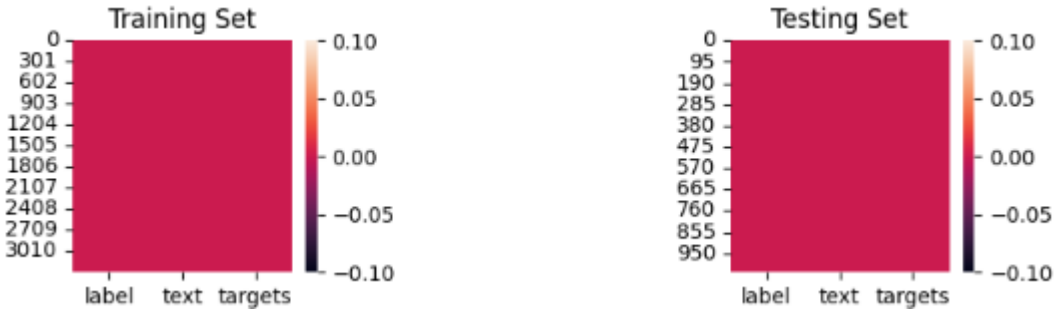
	label	text	targets
0	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	spam	Subject: photoshop , windows , office . cheap ...	1
4	ham	Subject: re : indian springs\r\nthis deal is t...	0
...	...	...	...
5166	ham	Subject: put the 10 on the ft\r\nthe transport...	0
5167	ham	Subject: 3 / 4 / 2000 and following noms\r\nhp...	0
5168	ham	Subject: calpine daily gas nomination\r\n>\r\n...	0
5169	ham	Subject: industrial worksheets for august 2000...	0
5170	spam	Subject: important online banking alert\r\ndea...	1

5171 rows × 3 columns

Pour les besoins de l'étude et de l'efficacité des pré-traitements nous avons créé une petite fonction qui fixe la seed de génération. Cette fonction assure la reproductibilité des résultats grâce à la reproductibilité de tous les tirages des nombres aléatoires, les opérations de mélange et les calculs basés sur l'horloge.

La seed est ensuite utilisée lors du split du dataset en trois : le training set (ensemble d'entraînement) , le validation set (ensemble de validation) et le test set (ensemble de test). Pour rappel, l'ensemble d'entraînement est utilisé pour apprendre les paramètres du modèle, l'ensemble de validation est utilisé pour ajuster les hyperparamètres et prévenir le surajustement, et l'ensemble de test est utilisé pour évaluer les performances finales du modèle de manière objective. Cette division en ensembles distincts aide à garantir que le modèle est capable de généraliser correctement et de faire des prédictions précises sur de nouvelles données.

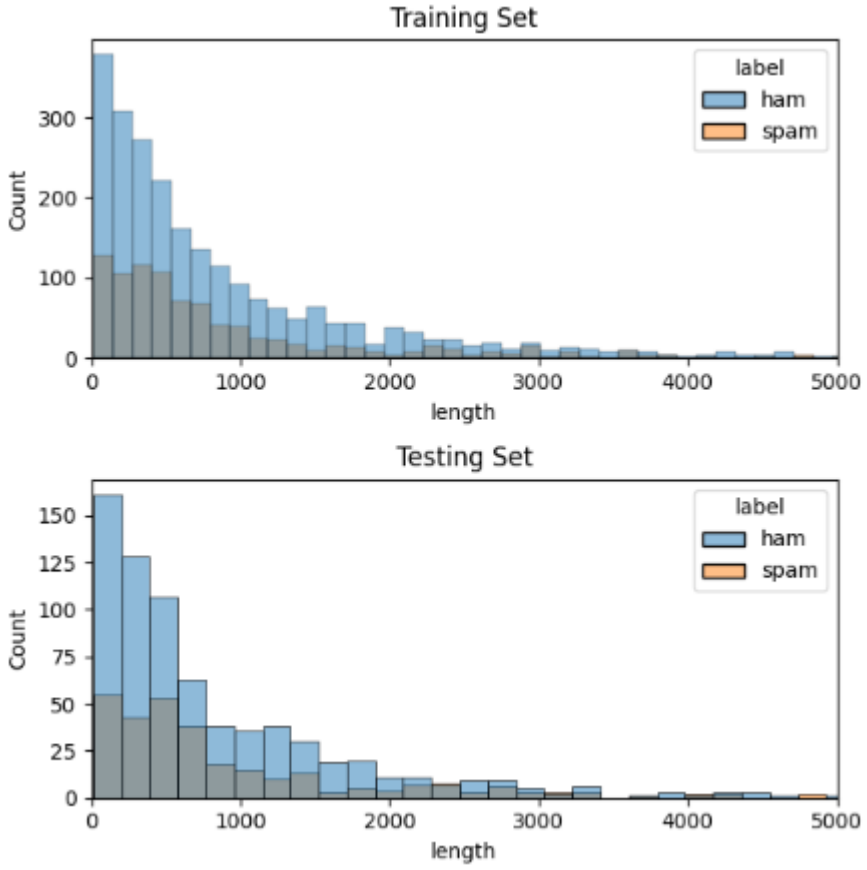
Après cette séparation, il est important de vérifier que les données sont adaptées pour notre futur entraînement de modèle, c'est pourquoi il est important de vérifier qu'aucune valeur nulle est présente dans nos datasets.



Ici en l'occurence, si une valeur était nulle, c'est à dire qu'aucune valeur n'aurait été constatée pour une des colonnes, un point blanc serait visbile sur nos graphiques.

Un autre point important à prendre en compte dans notre dataset est la répartition de chaque target: les ham et les spams. En effet, s'il y a beaucoup plus de ham ou de spam un dans le dataset, c'est un critère à prendre en compte. En affichant le nombre d'occurrences dans chaque catégorie on voit qu'en effet le nombre de spams est moindre: 950 par rapport au nombre de mails légitimes: 2358. Il faudra donc faire attention dans la suite de notre apprentissage de prendre en compte cette inégalité.

Enfin un des derniers critères que nous avons pu regarder, c'est la proportion du nombre de spams et de ham en fonction de la taille du mail. C'est pourquoi nous avons affiché la répartition en fonction de ce paramètre pour voir si oui ou non il s'agissait d'un paramètre significatif à prendre en compte.



Ici, que ce soit pour le training comme le testing set, aucune conclusion sur une corrélation entre la catégorie et la taille ne peut se faire.

b) Tokenisation et Prétraitement

Maintenant que nous avons pu analyser nos données, il est important de procéder à un prétraitement de celles-ci pour faciliter et améliorer l'apprentissage ensuite. Pour commencer il est important de tokeniser les données. Pour rappel, la tokenisation consiste à diviser un texte en unités plus petites appelées "tokens". Un token peut être aussi court qu'un caractère ou aussi long qu'un mot, mais le but principal est de fournir une représentation numérique structurée du texte. Cela nous permet de formater les textes en entrée pour que le modèle traite le langage naturel de manière plus efficace.

Dans notre cas, nous utilisons le modèle "alexandrainst" de transformers.XLMRobertaTokenizer qui est un modèle adapté à notre cas de figure pour tokenizer nos données correctement.

En utilisant en entrée le mail suivant :

```
Subject: re : spinnaker - supply verification 11958 su - october 2000
production
fyi .
bob
- - - - - - - - - - - - - - - - - - - - - - forwarded by robert cotten / hou / ect on 12 / 07 / 2000 10 : 12
am - - - - - - - - - - - - - - - - - - - - - - - - -
nick moshou @ enron
12 / 07 / 2000 10 : 13 am
to : christy sweeney / hou / ect @ ect , gary bryan / hou / ect @ ect
cc : vance l taylor / hou / ect @ ect , robert cotten / hou / ect @ ect
subject : re : spinnaker - supply verification 11958 su - october 2000
production
christy ,
per our conversation i have attached below the updated version of the
reallocation difference . texas general land received 21 . 8945 % of the total
volume which was 55 , 645 mmbtu . after updating the spreadsheet spinnaker will
owe hpl approximately $ 36 , 000 .
thanks ,
nick
```

On obtient la version tokenizer suivante:

```
 decoded message :
'<s> Subject: re : spinnaker - supply verification 11958 su - october 2000 product
0 10 : 12 am - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - nick moshou @ e
taylor / hou / ect @ ect, robert cotten / hou / ect @ ect subject : re : spinnaker
e updated version of the reallocation difference. texas general land received 21.
imately $ 36, 000. thanks, nick</s>'
```

On remarque que les retours à la ligne ont été supprimés.

Une dernière étape est nécessaire spécifiquement au modèle choisie, l'intégration des données dans un dataset spécifique: un BERTDATASET.

Ce BERTDataSet prépare les données dans un format compatible avec un modèle BERT. Il est conçue pour être utilisée avec l'objet DataLoader de PyTorch lors de l'entraînement d'un modèle basé sur BERT ou chaque élément renvoyé par cette classe contient les informations nécessaires pour l'entraînement du modèle, y compris les tokens encodés, les masques d'attention et les étiquettes associées.

Cela permet ensuite de faciliter l'intégration des données dans un dataloader, le notre prend un ensemble de données (généré par BERTDataSet), divise les données en mini-lots de taille spécifiée (batch_size), mélange les mini-lots (shuffle=True), utilise plusieurs travailleurs pour charger les données en parallèle (num_workers), et éventuellement utilise la mémoire épinglée pour accélérer le transfert des données vers le GPU (pin_memory=True). Nous réalisons ce processus pour nos trois set et nos données sont enfin prêtes pour le modèle.

4) Mise en place du Modèle

Dans cette partie, nous allons présenter la démarche effectuée pour la réalisation de notre modèle. Tout comme pour le tokenizer, nous utilisons la version alexandarinst du modèle XLMRoberta. Dans notre cas, il est instancié à partir du pré-entraînement "alexandrainst/da-ned-base" à l'aide de la bibliothèque Transformers de Hugging Face. Ce modèle offre une polyvalence linguistique étendue et est capable de traiter des tâches de classification binaire avec une seule étiquette de sortie. En utilisant l'option force_download=True, cela garantit la mise à jour constante des poids du modèle.

Afin de rendre plus efficace l'entraînement du modèle, on redirige en priorité les calculs vers le GPU pour utiliser la puissance du traitement en parallèle, puis on utilise la méthode train() sur le modèle pour signaler au modèle qu'il doit ajuster ses poids en réponse aux données d'entraînement.

Ensuite on définit un optimiseur avec détérioration des poids. Cet optimizer permet de mettre à jour les poids et paramètres du modèle avec la rétropropagation du modèle. L'ajout de détérioration des poids permet d'accélérer la convergence du modèle car on part du principe que le taux d'apprentissage est réduit au fur et à mesure des époques ce qui le stabilise plus rapidement.

Enfin, on instancie deux fonctions, tout d'abord la fonction d'entraînement (training) qui utilise le DataLoader train_dataloader pour itérer sur les mini-lots, calculer la perte, et effectuer la rétropropagation avec une gestion de la mise à l'échelle automatique du gradient pour optimiser le modèle. Puis, La fonction de validation (validating) qui évalue les performances du modèle sur un ensemble de validation. Ces fonctions sont cruciales et

utilisent l'optimiseur AdamW pour ajuster les poids du modèle en fonction de la perte calculée. La gestion de la mise à l'échelle automatique du gradient (scaler) est utilisée pour améliorer la stabilité de l'entraînement.

On définit également notre fonction de perte en utilisant BCEWithLogitsLoss qui est parfaitement adapté aux tâches de classification binaire. L'utilisation de cette fonction de perte, en combinaison avec la planification linéaire du taux d'apprentissage (Decay de l'optimiseur), permet de guider efficacement l'entraînement du modèle en facilitant la convergence du modèle. Il ne nous reste plus qu'à lancer l'apprentissage du modèle.

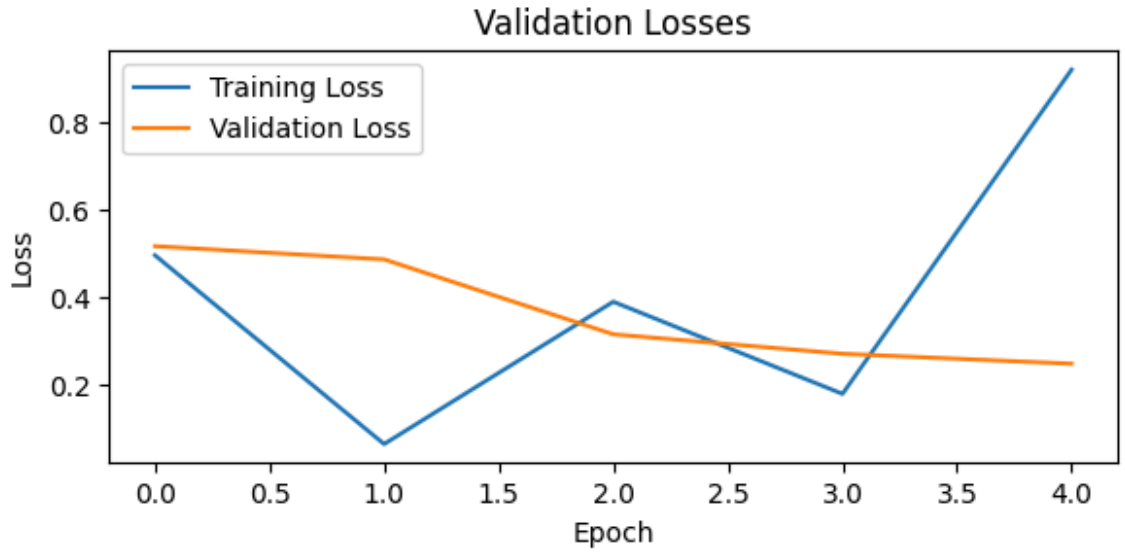
5) Résultat de l'entraînement

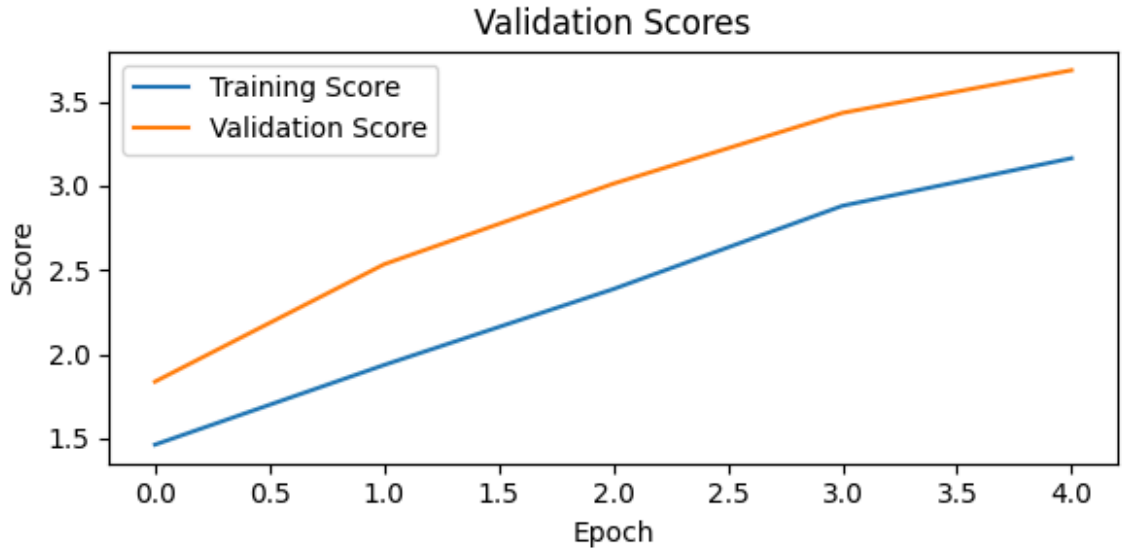
Suite à l'entraînement nous obtenons les résultats suivants:

```
train losses:  [0.7785849571228027, 0.3417717516422272, 0.05706420540809631 3, 0.3723258376121521, 0.021804381161928177]
val losses:  [0.3277888298034668, 0.05926472693681717, 0.011313550174236298, 0.00891048088669777, 0.006314849015325308]
train scores:  [1.4962028, 2.1974998, 2.7730682, 3.399895, 3.6486359]
valid scores:  [1.7635934, 2.6349225, 3.6231277, 3.8331997, 4.191798]
```

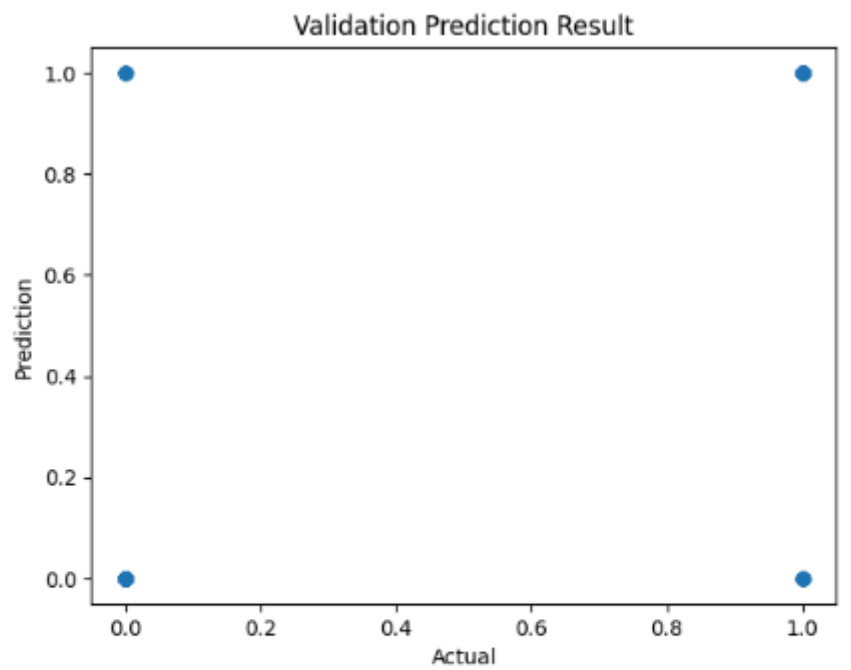
Les "train losses" représentent les valeurs de la fonction de perte (loss) sur l'ensemble d'entraînement à chaque époque, indiquant l'erreur du modèle pendant la phase d'entraînement. Les "val losses" représentent les valeurs de la fonction de perte sur l'ensemble de validation, évaluant l'erreur du modèle sur des données non vues. Une valeur basse dans ces deux cas indique une meilleure performance, car cela signifie que le modèle parvient à minimiser l'erreur pendant l'entraînement et généralise bien sur des données inconnues.

Les "train scores" et "valid scores" représentent des mesures de performance du modèle, des scores d'évaluation spécifiques à la tâche de classification de séquences. Une valeur élevée dans ces scores indique une meilleure performance.





En observant ces graphiques on remarque qu'au fur et à mesure des époques, la précision de l'entraînement augmente drastiquement et qu'il serait cependant encore plus intéressant d'augmenter le nombre d'époques pour observer une stagnation dans la précision de l'entraînement ou non. Voyons maintenant les résultats en termes d'accuracy, de rappel et de F1-score.



	precision	recall	f1-score	support
ham	0.9039	0.9328	0.9181	595
spam	0.8131	0.7468	0.7785	233
accuracy			0.8804	828
macro avg	0.8585	0.8398	0.8483	828
weighted avg	0.8784	0.8804	0.8788	828

La précision mesure la proportion des prédictions positives qui sont correctes par rapport à l'ensemble des prédictions positives (vrais positifs + faux positifs). Dans le contexte de ces résultats, la précision pour la classe "ham" est de 90.39%, ce qui signifie que 90.39% des messages classés comme "non-spam" le sont effectivement. Pour la classe "spam", la précision est de 81.31%, indiquant que 81.31% des messages classés comme "spam" le sont réellement. Ces résultats peuvent être largement améliorés et c'est ce que nous verrons dans un second temps par la suite.

Le rappel mesure la proportion des exemples positifs qui ont été correctement identifiés par rapport à l'ensemble réel des exemples positifs (vrais positifs + faux négatifs). Une valeur de 93.28% pour la classe "ham" indique que le modèle a réussi à identifier 93.28% des vrais messages "non-spam", tandis que pour la classe "spam", le rappel est de 74.68%, ce qui signifie que le modèle n'a identifié que 74.68% des vrais messages "spam". On peut se demander quelle proportion est la plus intéressante à garder, est-ce qu'un spam considéré comme étant un courrier légitime ou un un courrier légitime considéré en spam est le plus contraignant?

Le F1-score est une métrique qui combine la précision et le rappel en une seule valeur. Il s'agit de la moyenne harmonique entre ces deux métriques, fournissant un équilibre entre les faux positifs et les faux négatifs. Un F1-score élevé indique un équilibre optimal entre précision et rappel. Les valeurs de 91.81% pour "ham" et 77.85 % pour "spam" montrent la performance globale du modèle.

La colonne "support" indique le nombre d'exemples réels de chaque classe dans le jeu de données, elle permet de voir les nombre de spam et de ham dans le set de validation.

L'exactitude mesure la proportion totale de prédictions correctes par rapport à l'ensemble des prédictions. Avec une exactitude globale de 88.04 %, le modèle réussit à classer correctement 88.04 % de l'ensemble des messages.

Enfin, les Moyenne pondérée et Moyenne Macro fournissent des indicateurs globaux de performance. La moyenne pondérée donne plus de poids aux classes qui ont un plus grand nombre d'exemples, tandis que la moyenne macro donne le même poids à toutes les classes. Les valeurs moyennes montrent une performance équilibrée entre les deux classes.

D’une manière générale notre détection de spam est moins efficace que notre détection de courrier légitime, on peut donc se demander comment optimiser notre modèle pour prendre davantage en compte la différence entre la proportion de chaque classe et améliorer ainsi la fiabilité de notre future prédiction.

6) Optimisation du modèle

Au vu des précédents résultats nous allons améliorer les performances de notre modèle. Une des pistes d'amélioration est d'augmenter le volume de données d'entraînement, cependant il en existe peu à disposition, une idée est donc de procéder à une cross validation. Il s’agit d’une technique essentielle en apprentissage automatique qui permet d'évaluer la performance d'un modèle de manière robuste et fiable. Elle vise à estimer comment un modèle généralise à des données non vues en simulant plusieurs scénarios d'apprentissage et de test à partir d'un seul ensemble de données.

L'ensemble de données est divisé en k sous-ensembles, généralement appelés "plis" (folds). Le modèle est entraîné k fois, chacune avec un pli différent comme ensemble de test, et les autres plis combinés comme ensemble d'entraînement. Cela permet de tester la capacité du modèle à généraliser à différentes portions des données. Les performances du modèle sont évaluées k fois (une fois pour chaque pli de test), et les résultats sont souvent moyennés pour obtenir une mesure globale de la performance du modèle. Utiliser cette technique permet d’éviter les problèmes de surajustement (overfitting).

Dans notre cas pratique nous avons implémenté une cross validation à 5 plis.

	precision	recall	f1-score	support
ham	0.9270	0.9628	0.9445	725
spam	0.9043	0.8226	0.8615	310
accuracy			0.9208	1035
macro avg	0.9156	0.8927	0.9030	1035
weighted avg	0.9202	0.9208	0.9196	1035

On se rend bien compte de l'amélioration des performances suite à l'utilisation de la cross validation. En effet, on a pu augmenter la précision globale à environ 91% et diminuer le nombre de faux positifs à 10% ce qui est deux fois mieux que précédemment.

7) Prédiction

Une fois notre modèle final implémenté, l'idée est de l'utiliser afin de prédire si oui ou non un mail que nous aurions pu recevoir est un spam. C'est pourquoi nous avons essayé avec deux mails différents un spam et un ham et avec la précision de notre modèle, celle-ci se fait correctement, notre outil de classification est donc terminé, cependant la précision n'est pas à 100% donc à prendre avec un certain recul, mais reste tout de même fiable selon les résultats donnés précédemment.

8) Conclusion et Ouverture

En conclusion, à travers ce rapport nous avons pu présenter les différentes étapes pour la réalisation d'un classifieur de mails en spam ou en courrier légitime, nous avons tout d'abord retracé le pré traitement des données, puis présenté en détail le modèle XML Roberta utilisé. Après entraînement nous avons décidé d'améliorer les résultats en utilisant de la cross validation afin d'utiliser le même nombre de données. Nous avons pu voir que les résultats ont pu affiner notre modèle pour permettre de concevoir un prédicteur de spam ou non.

Ce prédicteur peut être un outil pratique pour les services de messagerie pour permettre de classifier avec plus de précisions encore les mails reçus pour les placer ou non dans le dossier de spam. On pourrait également imaginer utiliser ce classifieur pour permettre de reprendre les conclusions du modèle déterminants ce qu'est un spam ou un courrier légitime, pour créer un producteur de spam qui puisse se fondre parmi les courriers légitimes.