Table des Matières

1 Introduction

Les différentes études fascinantes sur la planète Mars ont suscité depuis le 20^e siècle un vif intérêt scientifique, explorant des aspects tels que la présence d'eau passée, l'existence de méthane, la diversité minérale, et plus récemment, la découverte de nombreux volcans sur cette planète voisine. Dans le cadre de notre jeu de données composé de 1680 images, l'objectif était d'améliorer les algorithmes de classification pour détecter la présence ou l'absence de volcans sur ces images. Cependant, le défi majeur réside dans le fait que ces images présentent un certain niveau de bruit, conséquence de la suppression des images les plus bruitées.

Les questions cruciales soulevées par notre sujet de recherche sont les suivantes :

1 - Comment réaliser un pré traitement efficace dans notre jeu de données en utilisant des méthodes statistiques enseignées en cours ?
2 - Quel est le meilleur modèle permettant de détecter efficacement la présence de volcans sur Mars ?
3 - Comment interpréter les résultats obtenus et quelles perspectives de travail peut-on envisager ?

2 Analyse des données brutes

On utilise le langage de programmation Python et la bibliothèque pandas pour lire les données CSV (`mars_train.csv`). Les données sont stockées dans un dataframe `df` de 1680 lignes et 3601 colonnes, dont la dernière colonne contient les classes recherchées (présence (2) ou absence (1) de volcans).

2.1 Analyse exploratoire :

L'examen initial du jeu de données révèle l'absence de valeurs manquantes susceptibles de perturber le traitement des données, comme confirmé par `df.isnull().sum()` qui indique que toutes les valeurs sont disponibles. Cependant, l'observation des 800 colonnes pour toutes les images, avec une moyenne et un écart-type nuls (`X.describe`), indique la présence de pixels noirs sans impact significatif sur notre traitement. Ainsi, afin de réduire la dimension des images déjà bruitées, nous avons pris la décision de supprimer ces colonnes inutiles (entièrement nulle).

En poursuivant notre analyse, la visualisation du pourcentage de valeurs nulles dans toutes les colonnes révèle un taux élevé, comme illustré dans l'histogramme ci-dessous et sa distribution sur l'ensemble des images avec `imashow`. En réponse, nous avons opté pour deux approches d'imputation :

1. l'imputation biaisée, remplaçant les valeurs nulles par la moyenne
2. l'imputation non biaisée, utilisant la méthode des K plus proches voisins (KNN) pour remplacer les valeurs nulles.

Comme le montre ce diagramme, les données nulles dominent dans notre jeu de données et leur distribution sur l'ensemble des images est illustrée ci-dessous :

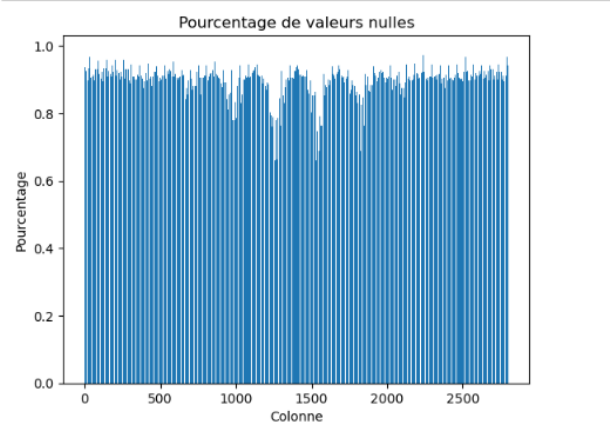

Figure 1: Histogramme des valeurs nulles

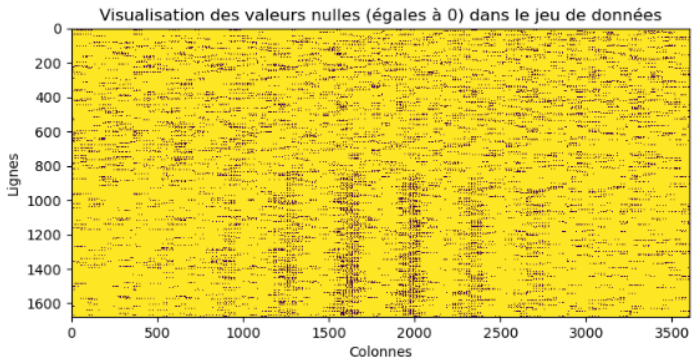

Figure 2: Distribution des valeurs nulles sur l'ensemble des images

3 Remarque générale

On n'a pas choisit de remplacer les valeurs nulles par la moyenne parce que comme le montre le graphique, la moyenne des valeurs nuls est la dominante (800 colonne) et le pourcentage des valeurs nuls dans les autres colonnes est aussi élevé.

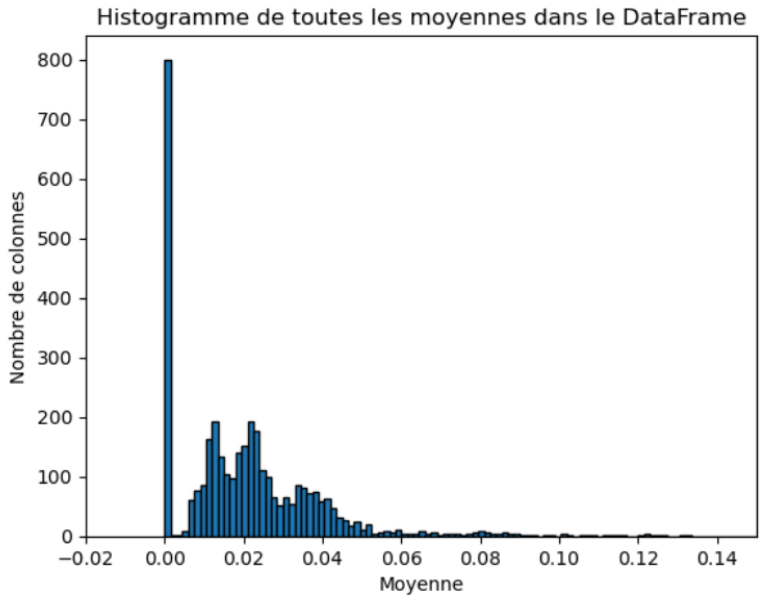

Figure 3: Histogramme de toute les moyenne dans df

3.1 Analyse des classes :

Ensuite, nous avons visualisé les classes pour étudier l'équilibre. En effet, on constate que les classes sont équilibrées, ce qui signifie généralement que le nombre d'échantillons dans chaque classe est relativement similaire. Ainsi, le modèle peut apprendre de manière plus équitable à partir de chaque catégorie,i.e la présence et l'absence des volcans sur Mars.

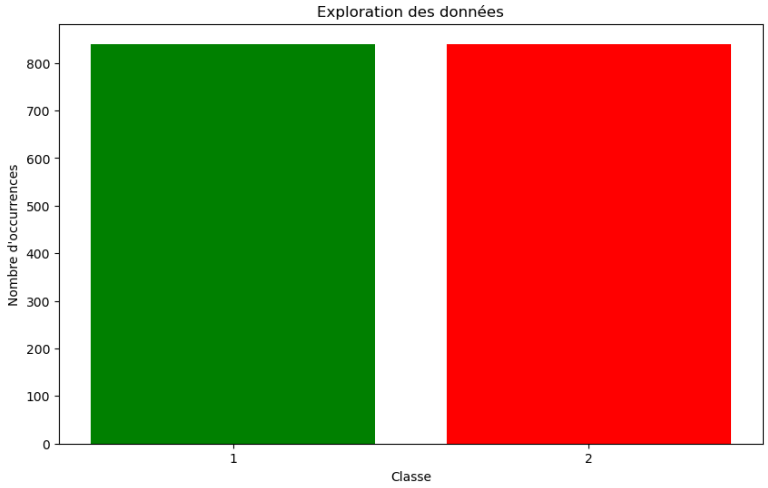

Figure 4: Classe équilibré

3.2 Préparation des données :

Comme mentionné dans le paragraphe précédent :
1 - Suppression des colonnes entièrement nulles.
2 - Utilisation de la méthode de KNN pour remplacer les autres valeurs nulles dans le jeu de données. Puisque la proportion de valeurs nulles est élevée, nous allons les remplacer par NaN, puis appliquer KNNImputer() sur notre jeu de données

3.3 Remarque générale :

Dans un premier temps, nous avons envisagé d'utiliser la méthode de l'ACP pour regrouper la variance sur des axes précis, étant donné que le jeu de données est volumineux. Cependant, le choix des composantes principales n'était pas optimal, car en appliquant l'ACP sur les 3600 colonnes, nous avons constaté qu'elle ne sélectionnait que 1680 colonnes. Étant donné que l'ACP choisit les composantes principales en fonction de leur contribution à la variance totale, nous avons décidé d'utiliser la méthode de KNN pour éviter de réduire excessivement la dimension de notre jeu de données (de 1680x3600 à 1680x1680) et pour d'autres raisons que nous expliquerons lors de la conclusion.

3.4 Division des données :

Avant de choisir le modèle de classification, il fallait transformer les données provenant de KNN en dataframe. Pour ce faire, nous avons utilisé `pd.DataFrame(df_imputed, columns=df1.columns)`, puis nous avons divisé notre jeu de données en 20% de données de test et 80% de données d'entraînement en utilisant la méthode "train test split" de sklearn

4 Choix du modéle

Nous avons testé trois modèles de classification : SVM, Random Forest et Logistic Regression .

4.1 Pourquoi ?

:

- **SVM :** Efficace pour séparer les deux classes dans un espace de dimension 1630x2800. Puisque les images nécessitent une bonne séparation entre les deux classes, nous avons d'abord testé les noyaux linéaire puis gaussien avec une grille.
- **Random Forest :** Présente des avantages en termes de robustesse, capacité à traiter des caractéristiques importantes, adaptabilité à différentes structures de données, et réduction du risque de surajustement.
- **Logistic Regression :** Enfin, la régression logistique linéaire a été choisie en raison de sa simplicité et de sa facilité d'interprétation.

4.2 Evaluation ?

Pour chaque modèle, on a utilisé des paramètres spécifiques et ajusté le modèle avec les données d'entraînement.

Paramètres pour chaque modèle :

- SVM : On a d'abord essayé un noyau linéaire, mais grâce à GridSearchCV, on a trouvé que le noyau gaussien est le plus approprié avec gamma=0.1 et C=10
- Régression Logistique : On a testé le jeu de données avec 1000 itérations, mais nous avons conclu par la suite que 500 itérations suffisent pour notre jeu de données avec C=0.1.
- Forêt Aléatoire : Nous l'avons initialement testée avec 100 estimateurs, mais nous avons augmenté à 200 itérations avec GridSearch.

Ensuite, on a évalué la performance de chaque modèle en utilisant une validation croisée avec 10 plis (10-fold cross-validation) et en calculant la précision moyenne du modèle sur les données de test. Nous avons également visualisé des statistiques détaillées telles que la précision, le rappel et le F1-score pour chaque classe (1 et 2) afin d'évaluer le modèle de manière approfondie pour chaque classe.

On a également examiné la matrice de confusion pour évaluer comment le modèle se comporte vis-à-vis des classes positives et négatives. Ensuite, nous avons tracé la courbe ROC pour évaluer la capacité du modèle à discriminer entre les classes. Enfin, nous avons vérifié, pour chaque modèle, la possibilité de sur-ajustement (overfitting) en comparant les performances sur les ensembles d'entraînement et de test.

5 Résultats

Après validation des trois modèles, nous avons obtenu les résultats suivants :

```
Rapport de classification Régression logistique :
              precision    recall  f1-score   support

           1       0.83      0.77      0.80       176
           2       0.77      0.82      0.80       160

    accuracy                           0.80       336
   macro avg       0.80      0.80      0.80       336
weighted avg       0.80      0.80      0.80       336

Rapport de classification SVM :
              precision    recall  f1-score   support

           1       0.79      0.87      0.83       176
           2       0.84      0.75      0.79       160

    accuracy                           0.81       336
   macro avg       0.82      0.81      0.81       336
weighted avg       0.81      0.81      0.81       336

Rapport de classification Random Forest :
              precision    recall  f1-score   support

           1       0.81      0.76      0.78       176
           2       0.75      0.80      0.78       160

    accuracy                           0.78       336
   macro avg       0.78      0.78      0.78       336
weighted avg       0.78      0.78      0.78       336
```

Figure 5: Comparaison des performances entre les trois modèles

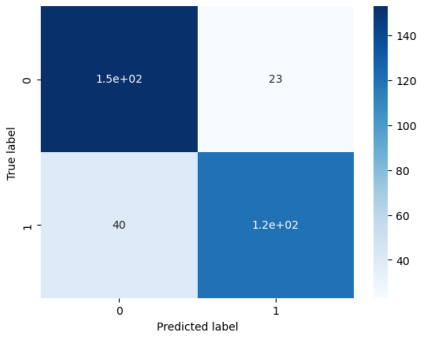

Figure 6: svm

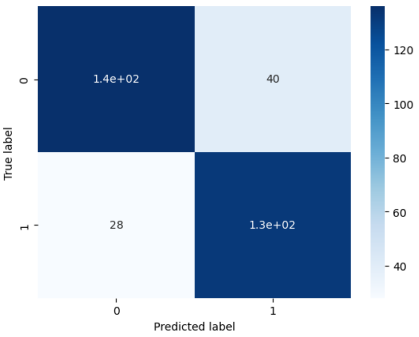

Figure 7: Régression logistique

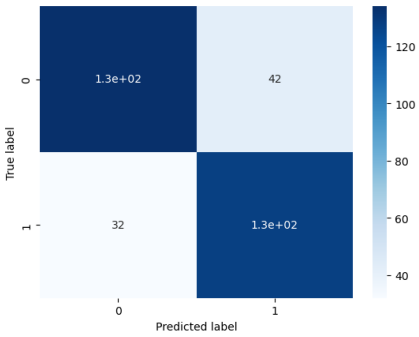

Figure 8: Forêt Aléatoire

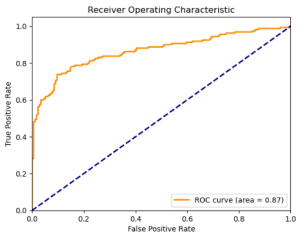

Figure 9: SVM

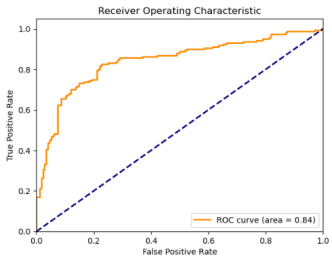

Figure 10: Logistique Linéaire

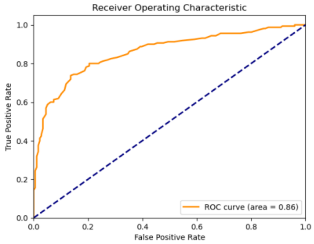

Figure 11: Forêt Aléatoire

5.1 Modèle choisi :

- En comparant les trois modèles, nous constatons que :
  - La précision sur l'ensemble de test est assez similaire pour les trois modèles, bien que le SVM semble légèrement meilleur.
  - Les courbes ROC indiquent que le SVM présente une meilleure capacité de discrimination que les deux autres modèles.

6 Interprétation

6.1 Analyse des Résultats de SVM :

L'évaluation du modèle SVM révèle des meilleurs résultats par rapport aux autres données . L'accuracy globale de 81.25% indique que le modèle a correctement identifié la présence ou l'absence de volcans.

En se concentrant sur la classe 2, qui représente la présence d'un ou plusieurs volcans, la précision de 84% indique que le modèle a correctement identifié la grande majorité des cas positifs, montrant une aptitude à classifier de manière fiable la présence de volcans. Le recall de 87% suggère que le modèle a capturé la plupart des véritables cas de présence de volcans, et le F1-score élevé de 0.83 confirme une performance équilibrée entre précision et rappel pour cette classe .

En ce qui concerne la classe 1, représentant l'absence de volcans, la précision de 79% indique que le modèle a bien géré la classification des cas négatifs, mais le recall de 75% suggère qu'il a rencontré des défis dans l'identification de tous les cas d'absence de volcans. Le F1-score de 0.77 montre une performance raisonnablement équilibrée.

Sur la base de la matrice de confusion, nous pouvons faire les observations suivantes :

- Le modèle a correctement prédit la présence de 140 volcans et l'absence de 120 volcans.
- Le modèle a incorrectement prédit la présence de 23 volcans alors qu'ils étaient absents.
- Le modèle a incorrectement prédit l'absence de 40 volcans alors qu'ils étaient présents.

En fin, l'AUC de 0.87 confirme que le modèle SVM a une performance robuste dans la discrimination entre les deux classes, renforçant ainsi la confiance dans sa capacité à effectuer une classification précise.

6.2 Prédiction du SVM sur mars_unknown :

Après avoir choisi le modèle SVM, nous l'avons appliqué sur le fichier `mars_unknown.csv`. Cependant, nous avons commencé par appliquer le même principe de pré-traitement que celui utilisé pour les données d'entraînement, afin de garantir la cohérence et la fiabilité des résultats. Les étapes de pré-traitement incluses sont les suivantes :

- Suppression des colonnes entièrement nulles :

Avant d'appliquer le modèle SVM, nous avons exclu les colonnes qui étaient entièrement composées de valeurs nulles dans le fichier `mars_unknown.csv`, en utlisant : X = UDF.loc[:, (UDF != 0).any(axis=0)]

- Application de k-NN sur l'ensemble de données partiellement traité :

Nous avons appliqué l'algorithme k-NN sur l'ensemble de données pour conserver la cohérence dans le traitement des données, notamment en ce qui concerne la gestion des valeurs nuls dominés dans le fichier comme le montre le graphe en dessous .

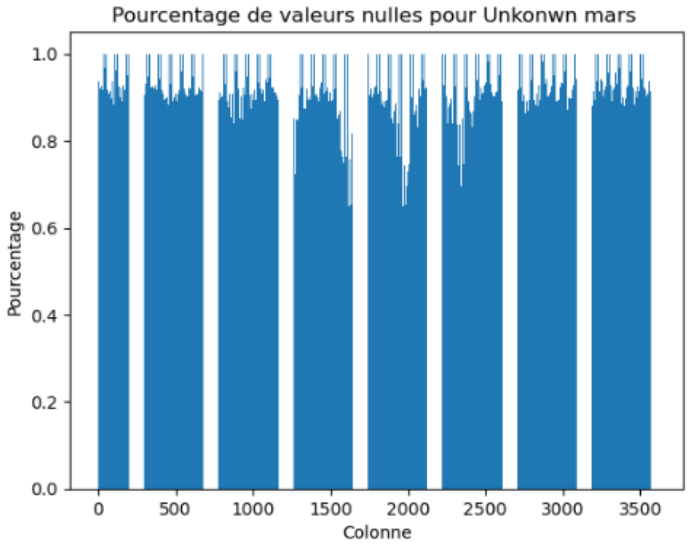

Figure 12: Pourcentage des valeurs nulles pour "unknown_mars"

7 Résultats intermédiaire :

En se basant sur les résultats de prédiction intermédiaire, tels que vous nous les avez communiqués, les résultats sont les suivants :

- Précision : 77,57% des détections de volcans effectuées par svm sont effectivement des volcans.
- Rappel : svm a réussi à détecter 71,19% de tous les volcans présents sur Mars.
- Exactitude : svm a correctement classé 71,19% de toutes les instances, qu'il s'agisse de volcans ou de non-volcans.

8 Conclusion

Pourquoi ne pas traiter les données telles qu'elles :

Avant de choisir le modèle SVM, nous avons tenté de traiter les données sans supprimer les colonnes entièrement nulles et d'utiliser la méthode k-NN. Cependant, nous avons également essayé de visualiser les données, comme la matrice de corrélation, mais nous avons trouvé des valeurs NaN partout, car il y a 800 colonnes entièrement nulles. Après l'entraînement des données non traitées, nous avons conclu que le modèle de régression logistique est le mieux adapté à ce jeu de données, avec une précision de 0.84, surpassant ainsi SVM et Random Forest.

Pourquoi ne pas utiliser l'ACP :

De plus, pour l'ACP, nous avons quand même essayé d'entraîner le modèle et de choisir un meilleur classifieur. Nous avons constaté que le SVM est le plus performant, avec une précision de 0.85, un F1-score de 0.87 pour la classe 1 et 0.83 pour la classe 2, ainsi qu'un rappel de 0.95 pour la classe 1 et 0.74 pour la classe 2. Ces résultats sont légèrement meilleurs que ce que nous avions trouvé précédemment, mais le problème persiste lors de la prédiction car l'ensemble de données du nouveau jeu avait une dimension de 420x2800 après la suppression des colonnes entièrement nulles. Ensuite, après l'ACP, l'ensemble a été automatiquement réduit à 420x420 sans spécifier le nombre de composants principaux, car l'ACP de scikit-learn choisit le minimum entre le nombre de lignes et de colonnes. Cependant, la transformation avec fit ne fonctionne pas, et nous ne voulions pas par la suite réduire la dimension de l'apprentissage pour éviter des prédictions plus erronées.

Philosophie de k-NN :

En se basant sur le contexte, où les images sont déjà bruitées et plusieurs méthodes ont été utilisées pour les nettoyer, il reste néanmoins du bruit pour détecter la présence de volcans sur l'image ou non. Nous avons conclu que le traitement que nous avons effectué est le plus raisonnable, car il est difficile de trouver des informations sur un pixel noir et de prédire de manière précise l'information sur des images remplies de pixels noirs. Ainsi, il est préférable de réduire la dimension de l'image si elle est trop bruitée et d'essayer de conclure la couleur des pixels en utilisant k-NN.

9 Perspective

On peut envisager l'exploration de plusieurs différentes approches afin qu'on améliore la précision de notre classification. Tout d'abord, il serait intéressant d'explorer les méthodes de transfert de connaissances. C'est à dire utiliser les leçons apprises dans des situations de classification similaires pour aider notre modèle à mieux reconnaître les exemples positifs et négatifs. En comblant les espaces entre ces exemples, cela aide à repérer plus précisément les exemples positifs.

De plus, il est pertinent d'explorer autres approches pour traiter les valeurs nulles dans nos données. Parmi ces méthodes d'imputation, on peut considérer l'utilisation de la médiane conditionnelle comme une alternative à KNN pour estimer ces valeurs manquantes. Cette approche consiste à remplacer les valeurs manquantes en utilisant la médiane des données disponibles basée sur d'autres variables connexes. Par exemple, si une variable est manquante, elle pourrait être estimée en utilisant la médiane des observations qui partagent des caractéristiques similaires dans d'autres variables du jeu de données. Cela permet de remplir les valeurs manquantes en se basant sur des relations conditionnelles entre les différentes caractéristiques, offrant ainsi une approche robuste pour l'imputation des données.

Pour élargir davantage notre spectre d'algorithmes de classification, le modèle XGBoost pourrait être une option intéressante à étudier. Ce type des modèles combinent plusieurs arbres de décision plus faibles pour former un modèle robuste et performant. Cette approche pourrait être particulièrement bénéfique pour améliorer la précision de la prédiction dans notre cas, en prenant en considération les valeurs nulles tout en préservant la qualité de la classification binaire.