TP Machine Learning

Groupe : Ewen PERON - Adrien LEBOUCHER - Eva GRATIUS - Alexandre RADIN - Maxime ROQUELLE - Thomas ROYER

- Coder en python (en utilisant le moins possible de package) les algorithmes suivants:
    - Arbres de décision
    - Forêts aléatoires
    - Régression ridge
    - Régression lasso
    - Support Vector Machine (SVM)
- Évaluer les performances sur les 2 jeux de données (un de classification binaire, l'autre de régression)
- Comparer (en temps et performance) avec scikit-learn.

Table des matières :

1. Decision Tree

1.1. Classification DT

Pour l'entraînement, nous effectuons un GridSearch sur une liste de paramètres. Cela va entraîner le modèle sur toutes les combinaisons possibles de paramètres et retourner les meilleurs paramètres trouvés suite à l'évaluation. On évalue les modèles avec le Matthews correlation coefficient, le F1-score et la précision.

SKLEARN

```
Meilleurs paramètres trouvés :  {'max_depth': 10, 'min_samples_split': 2}
Précision moyenne sur l'ensemble de train :  0.68125
Résultats sur ensemble de test avec les meilleurs paramètres
              precision    recall  f1-score   support

           0       0.84      0.76      0.80        50
           1       0.66      0.77      0.71        30

    accuracy                           0.76        80
   macro avg       0.75      0.76      0.75        80
weighted avg       0.77      0.76      0.77        80

Matthews correlation coefficient :  0.5139740384195608
F1-score :  0.7076923076923077
Précision : 76.2500%
Temps d'exécution SKLEARN : 2.6752655506134033 secondes
```

SCRATCH

```
Meilleurs paramètres trouvés :  {'max_depth': 10, 'min_samples_split': 2}
Précision moyenne sur l'ensemble de train :  0.76875
Résultats sur ensemble de test avec les meilleurs paramètres
              precision    recall  f1-score   support

           0       0.82      0.80      0.81        50
           1       0.68      0.70      0.69        30

    accuracy                           0.76        80
   macro avg       0.75      0.75      0.75        80
weighted avg       0.76      0.76      0.76        80

Matthews correlation coefficient :  0.4968631026001803
F1-score :  0.6885245901639343
Précision : 76.2500%
Temps d'exécution SCRATCH : 3.7253506183624268 secondes
```

SKLEARN est légèrement meilleur sur toutes les métriques.

1.2. Régression DT

Pour l'entraînement, une validation croisée est réalisée. On divise le training dataset en plusieurs folds utilisés pour l'entraînement et la validation. Pour les résultats, les valeurs RMSE du tableau correspondent donc aux résultats obtenus sur les différents folds pendant l'entraînement. Pour comparer les deux modèles, on va se pencher sur la RMSE et le R2 obtenus suite à l'évaluation sur le test set.

SCRATCH

```
Valeurs RMSE du tableau : [21.83373702 19.04454711 18.2004158  20.52872301 19.0008345  19.88893486
 19.60659018 22.14809288 19.05547881 21.20533381]
Moyenne : 20.05126879632792
Ecart-type : 1.2606323575017706
Valeur RMSE finale sur le test set :  21.384116340281082
Coefficient de determination sur le test set : 0.22
Temps d'exécution SCRATCH : 47.73008608818054 secondes
```

SKLEARN

```
Valeurs RMSE du tableau : [21.58300967 19.40876555 19.34858051 21.96010305 18.73400339 20.67816884
 19.3230106  23.04578287 19.09272113 21.33797298]
Moyenne : 20.451211858428813
Ecart-type : 1.3963166797784312
Valeur RMSE finale sur le test set :  21.29042747711453
Coefficient de determination sur le test set : 0.23
Temps d'exécution SKLEARN : 0.25776147842407227 secondes
```

SKLEARN est sensiblement meilleur sur les métriques. De plus, il est beaucoup plus optimisé car son temps d'exécution est deux fois plus rapide.

2. Random Forest

Ici, on applique les mêmes stratégies que celles expliquées pour le Decision Tree pour l'entraînement et l'évaluation.

2.1. Classification RF

SKLEARN

```
Meilleurs paramètres trouvés :  {'max_depth': 50, 'min_samples_split': 5, 'n_estimators': 30}
Précision moyenne sur l'ensemble de train :  0.784375
Résultats sur ensemble de test avec les meilleurs paramètres
              precision    recall  f1-score   support

           0       0.93      0.86      0.90        50
           1       0.79      0.90      0.84        30

    accuracy                           0.88        80
   macro avg       0.86      0.88      0.87        80
weighted avg       0.88      0.88      0.88        80

Matthews correlation coefficient :  0.7442877094063837
F1-score :  0.84375
Précision : 87.5000%
Temps d'exécution SKLEARN : 3.964258909225464 secondes
```

SCRATCH

```
Meilleurs paramètres trouvés :  {'max_depth': 50, 'min_samples_split': 5, 'n_estimators': 30}
Précision moyenne sur l'ensemble de train :  0.80625
Résultats sur ensemble de test avec les meilleurs paramètres
              precision    recall  f1-score   support

           0       0.90      0.90      0.90        50
           1       0.83      0.83      0.83        30

    accuracy                           0.88        80
   macro avg       0.87      0.87      0.87        80
weighted avg       0.88      0.88      0.88        80

Matthews correlation coefficient :  0.7333333333333333
F1-score :  0.8333333333333334
Précision : 87.5000%
Temps d'exécution SCRATCH : 104.06774759292603 secondes
```

Encore une fois, les résultats sont à peu près les mêmes mais SKLEARN est beaucoup plus rapide.

2.2. Régression RF

SCRATCH

```
Valeurs RMSE du tableau : [16.02650832 12.4543644  14.60043544 14.34589193 15.70555988 16.21790085
 16.0211274  17.00052815 14.06376655 16.06113074]
Moyenne : 15.249721365835219
Ecart-type : 1.285547521348104
Valeur RMSE finale sur le test set :  16.36301323837494
Coefficient de determination sur le test set : 0.54
Temps d'exécution SCRATCH : 178.74678921699524 secondes
```

SKLEARN

```
Valeurs RMSE du tableau : [15.43051184 12.31029208 14.24139982 13.83321386 15.78118766 14.99949548
 14.62885491 16.08782597 12.69429852 15.08959509]
Moyenne : 14.509667524112569
Ecart-type : 1.191978562594627
Valeur RMSE finale sur le test set :  15.873926108128728
Coefficient de determination sur le test set : 0.57
Temps d'exécution SKLEARN : 14.597980499267578 secondes
```

Ici encore, SKLEARN est plus performant au niveau du temps d'exécution.

Pour conclure, au niveau des Decision Trees et des Random Forests, les résultats sur le test set sont à peu près les mêmes pour les algorithmes from scratch et les algorithmes de SKLEARN, mais ceux de SKLEARN sont beaucoup plus performants au niveau du temps d'exécution.

3. Régression Ridge

On charge les données à partir du fichier texte ozone_complet.txt et on retire les colonnes inutiles ou redondantes, comme la colonne maxO3v. Ensuite, toutes les lignes contenant des valeurs manquantes sont éliminées. Pour garantir que les résultats ne soient pas faussés par des valeurs aberrantes, on applique la méthode de l'écart interquartile (IQR) à toutes les colonnes numériques. Ensuite, on normalise manuellement les données pour s'assurer que toutes les variables explicatives ont la même échelle. La prochaine étape est de réduire la dimensionnalité des variables explicatives en supprimant les caractéristiques fortement corrélées entre elles ainsi que les caractéristiques très peu corrélées à notre cible.

3.1. maxO3

3.1.1 From Scratch

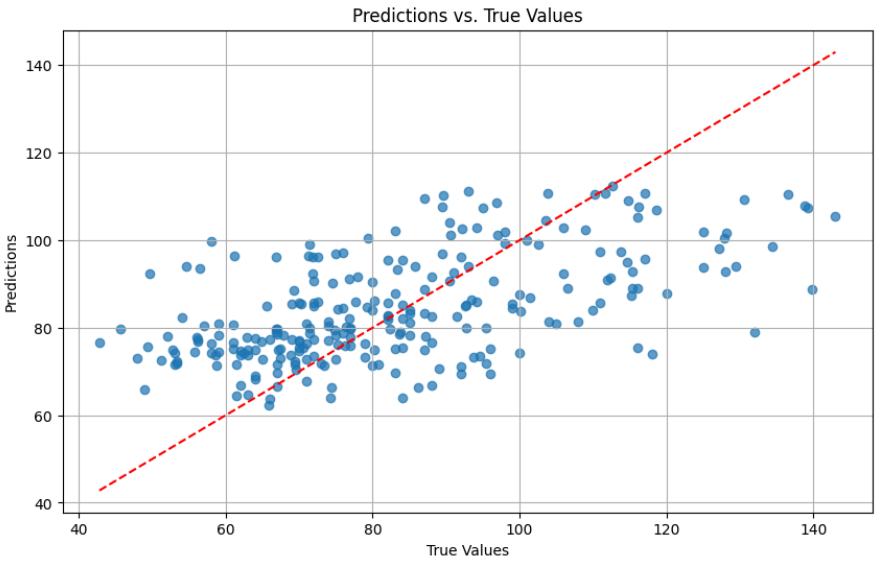
FROM SCRATCH for maxO3 - métriques standards

```
First 10 Predictions (from Scratch): [ 77.74368444 102.823234   107.48658271  72.31197122  97.02322812
  63.86478468  72.83846325  73.10511731 104.02083918  75.96820138]
First 10 True Values: [ 56.   94.2 139.2  53.4  76.   84.   71.   67.4  90.4  76.2]
Execution Time: 0.05217623710632324
Mean Squared Error (MSE): 299.54790083573
Root Mean Squared Error (RMSE): 17.307452176323643
Mean Absolute Error: 13.727536242088767
R2 Score (R²): 0.33695517916885886
```

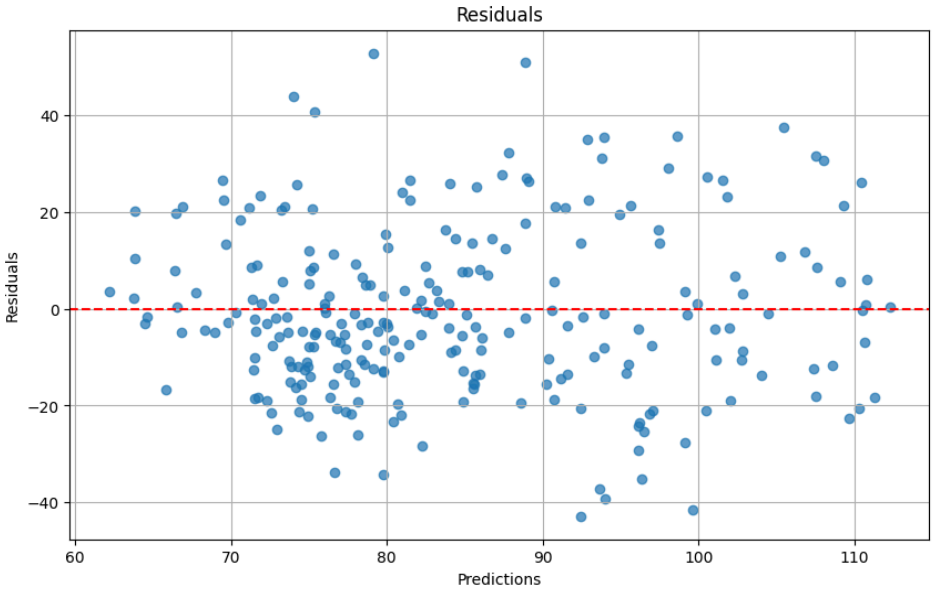
FROM SCRATCH for maxO3 - métriques de validation croisée

```
Cross-Validated Mean Squared Error (MSE): 299.0899
Cross-Validated Root Mean Squared Error (RMSE): 17.2942
Cross-Validated Mean Absolute Error: 13.7926
Cross-Validated R² Score (R²): 0.3198
```

Le modèle montre un certain niveau de capacité prédictive, mais les erreurs et le score R^2 indiquent qu'il pourrait être largement amélioré. On remarque cela sur le graphique comparant les valeurs réelles aux prédictions du modèle qui démontre une dispersion significative.



On retrouve un graphique avec des résidus dispersés, qui a donc bien appris la relation entre les variables et qu'il ne présente pas de problèmes majeurs tels que la multicolinéarité ou l'hétéroscédasticité:



3.1.2 Avec Sklearn

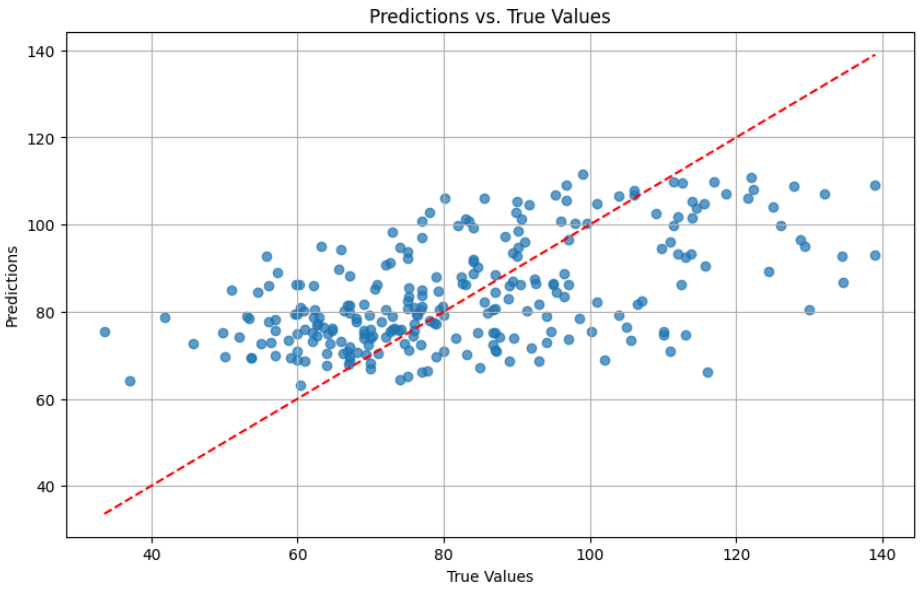
SKLEARN for maxO3 - métriques standards

```
First 10 Predictions (scikit-learn): [71.70339813 80.65733299 92.64214427 98.17106785 72.8629131  68.70815488
 80.31303474 94.88741945 81.71496969 78.51004643]
First 10 True Values: [ 92.   75.  134.4  73.   94.   93.   60.8  90.2  69.   98.6]
Execution Time: 0.1465463638305664
Mean Squared Error (MSE): 300.1319713910977
Root Mean Squared Error (RMSE): 17.32431734271506
Mean Absolute Error: 13.776288221428256
R2 Score (R²): 0.3053549898293877
```

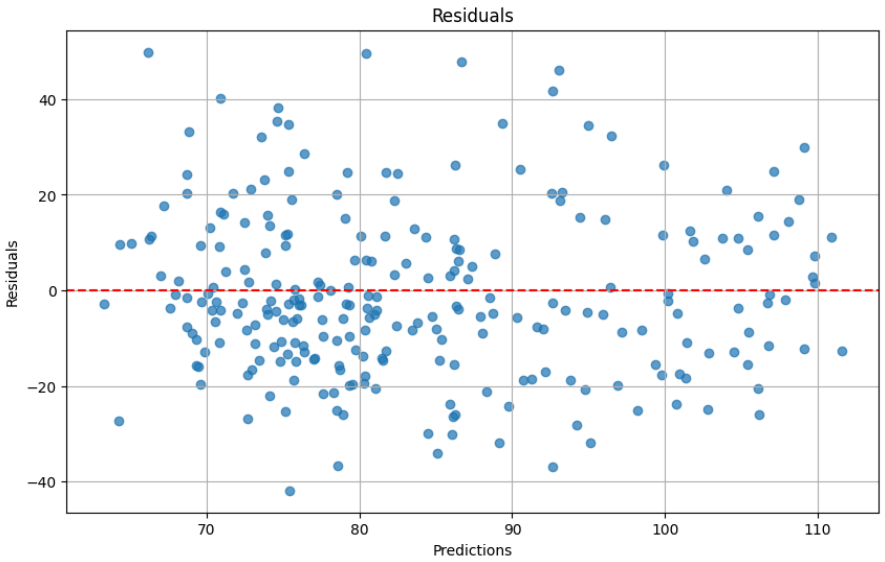
SKLEARN for maxO3 - métriques de validation croisée

```
Cross-Validated Mean Squared Error (MSE): 299.0050
Cross-Validated Root Mean Squared Error (RMSE): 17.2918
Cross-Validated Mean Absolute Error: 13.7957
Cross-Validated R² Score (R²): 0.3200
```

On retrouve également une dispersion significative :



Et des résidus également bien dispersés :



3.2. log(maxO3)

Les statistiques descriptives de log_maxO3 montrent une légère amélioration par rapport à la distribution originale de maxO3, on l'applique alors pour les deux méthodes et obtenons les résultats suivants :

3.2.1 From Scratch

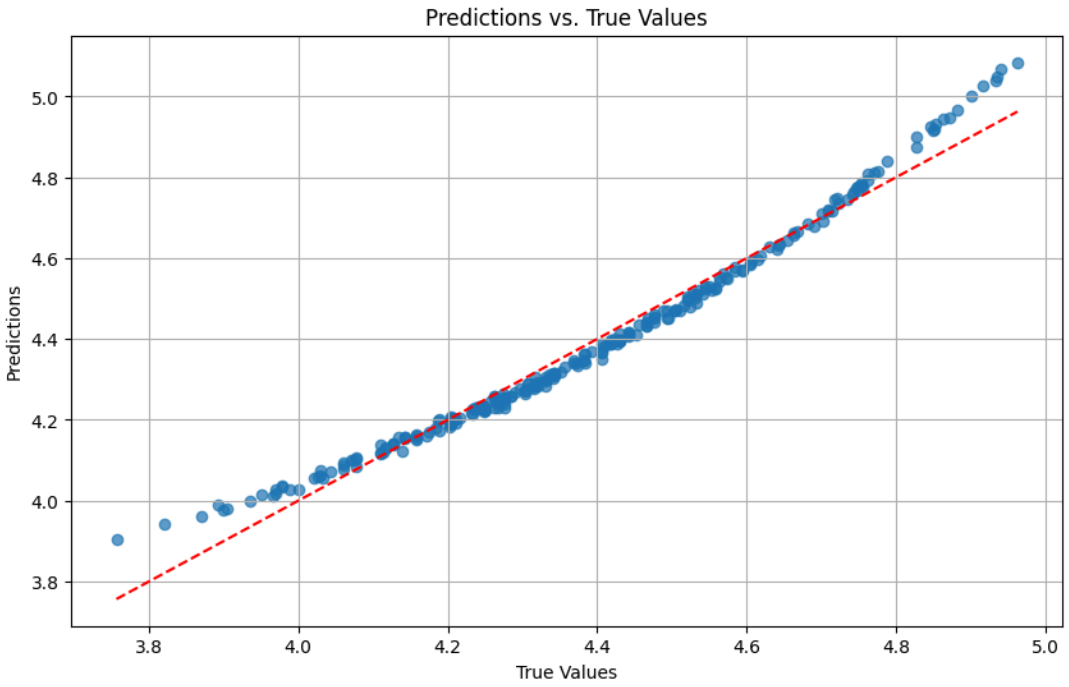
FROM SCRATCH for log(maxO3) - métriques standards

```
First 10 Predictions (from Scratch): [4.05804217 4.52922607 5.04924243 4.03761746 4.28404579 4.41438581
 4.2509958  4.20049785 4.4701539  4.30060194]
First 10 True Values: [4.02535169 4.54542018 4.93591175 3.97781075 4.33073334 4.4308168
 4.26267988 4.21064502 4.50424427 4.33336146]
Execution Time: 0.046822547912597656
Mean Squared Error (MSE): 0.0014346954783417227
Root Mean Squared Error (RMSE): 0.03787737422712566
Mean Absolute Error: 0.029650987219816922
R2 Score (R²): 0.9769683740184623
```

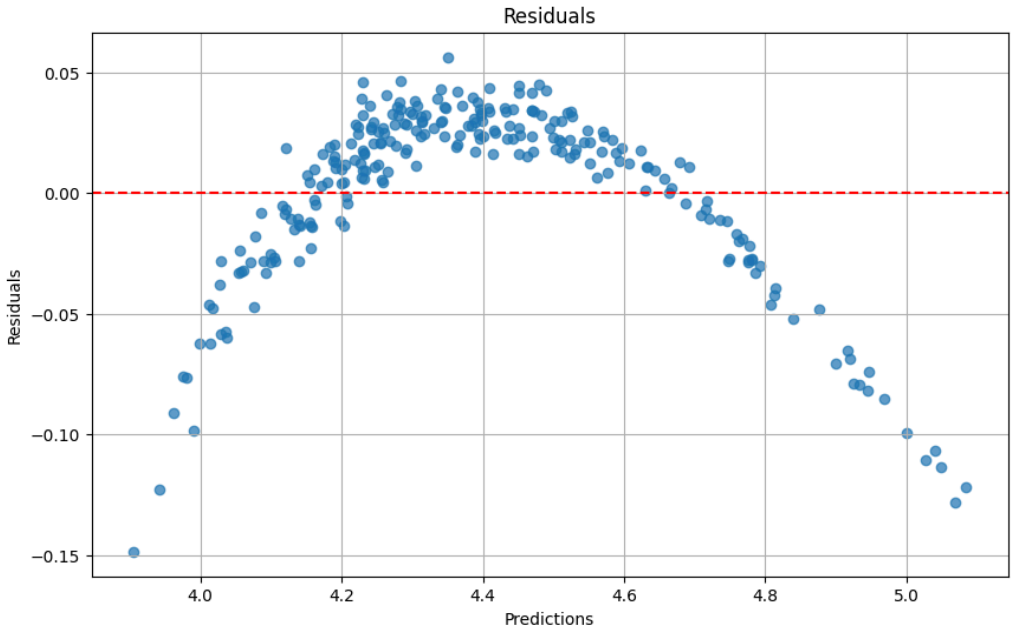
FROM SCRATCH for log(maxO3) - métriques de validation croisée

```
Cross-Validated Mean Squared Error (MSE): 0.0018
Cross-Validated Root Mean Squared Error (RMSE): 0.0429
Cross-Validated Mean Absolute Error: 0.0300
Cross-Validated R² Score (R²): 0.9715
```

Ces résultats sont visibles sur le graphique de dispersion qui compare les valeurs réelles aux prédictions du modèle :



Les points se rapprochent de la ligne de référence, ce qui signifie que les prédictions sont précises, cependant, il est important de noter que le graphique des résidus présente une forme en U. Cela indique que les résidus ne sont pas distribués de manière aléatoire, suggérant une possible non-linéarité dans le modèle.



3.2.2 Avec Sklearn

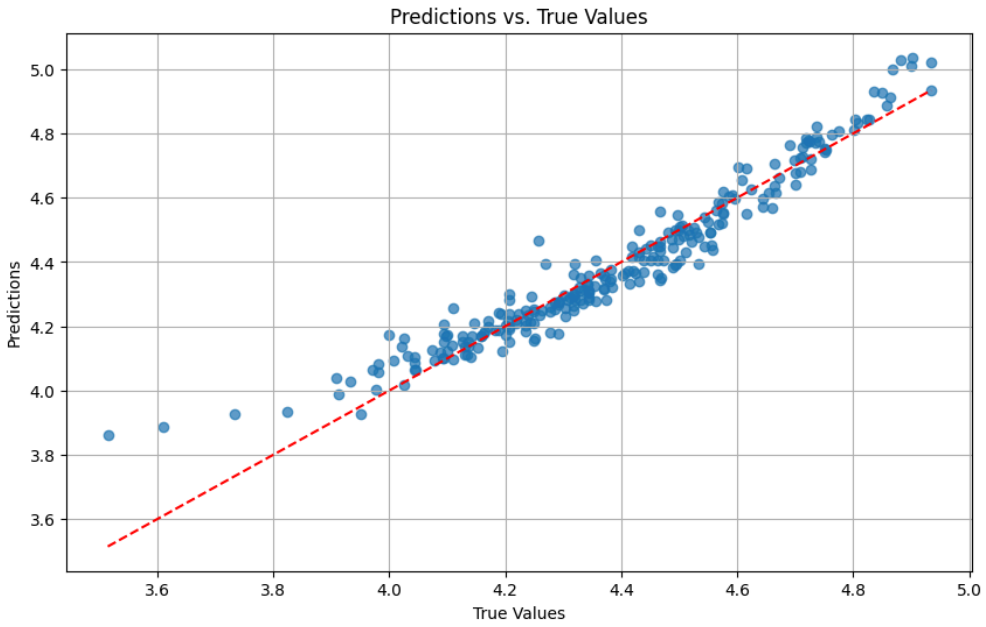
SKLEARN for log(maxO3) - métriques standards

```
First 10 Predictions (scikit-learn): [4.46374944 4.29439455 5.01130172 4.17662571 4.44715819 4.47666391
 4.13916263 4.40484597 4.18455173 4.60888423]
First 10 True Values: [4.52178858 4.31748811 4.90082043 4.29045944 4.54329478 4.53259949
 4.10758979 4.50202943 4.2341065  4.59107126]
Execution Time: 0.12021303176879883
Mean Squared Error (MSE): 0.004315769811942514
Root Mean Squared Error (RMSE): 0.06569451888812729
Mean Absolute Error: 0.04791375234067245
R2 Score (R²): 0.9317758236339182
```

SKLEARN for log(maxO3) - métriques de validation croisée

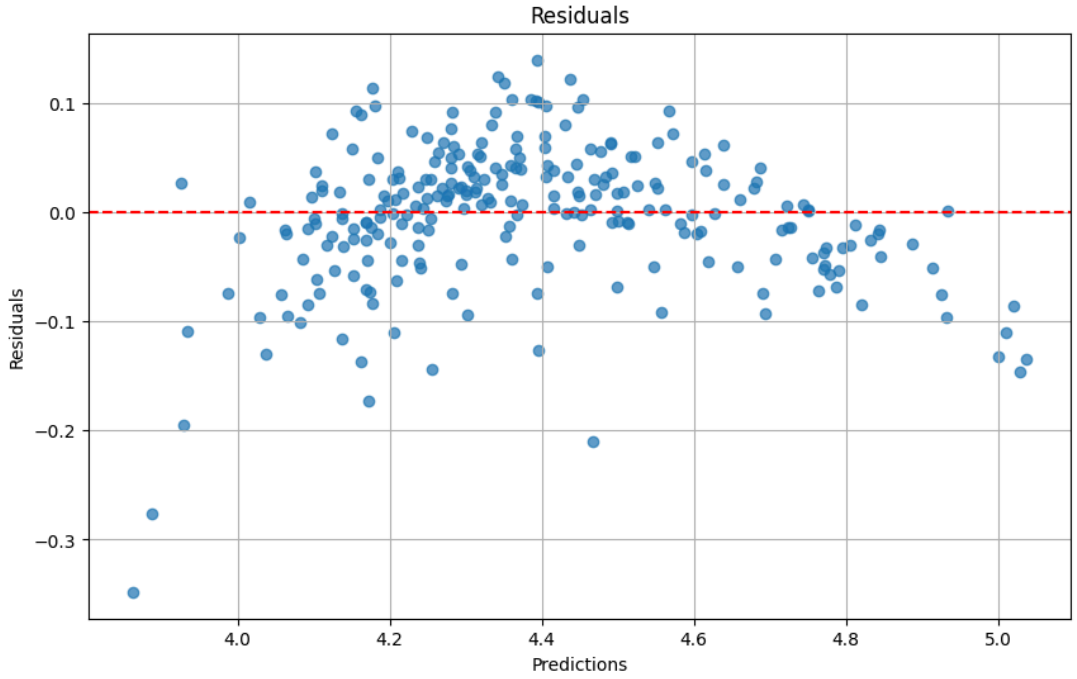
```
Cross-Validated Mean Squared Error (MSE): 0.0018
Cross-Validated Root Mean Squared Error (RMSE): 0.0429
Cross-Validated Mean Absolute Error: 0.0300
Cross-Validated R² Score (R²): 0.9715
```

Ces résultats sont visibles sur le graphique de dispersion qui compare les valeurs réelles aux prédictions du modèle :



On observe des résultats plus concentrés.

Lorsque l'on observe le graphique des résidus, on observe un lègère tendant en U, mais moins forte que pour le modèle from scratch :

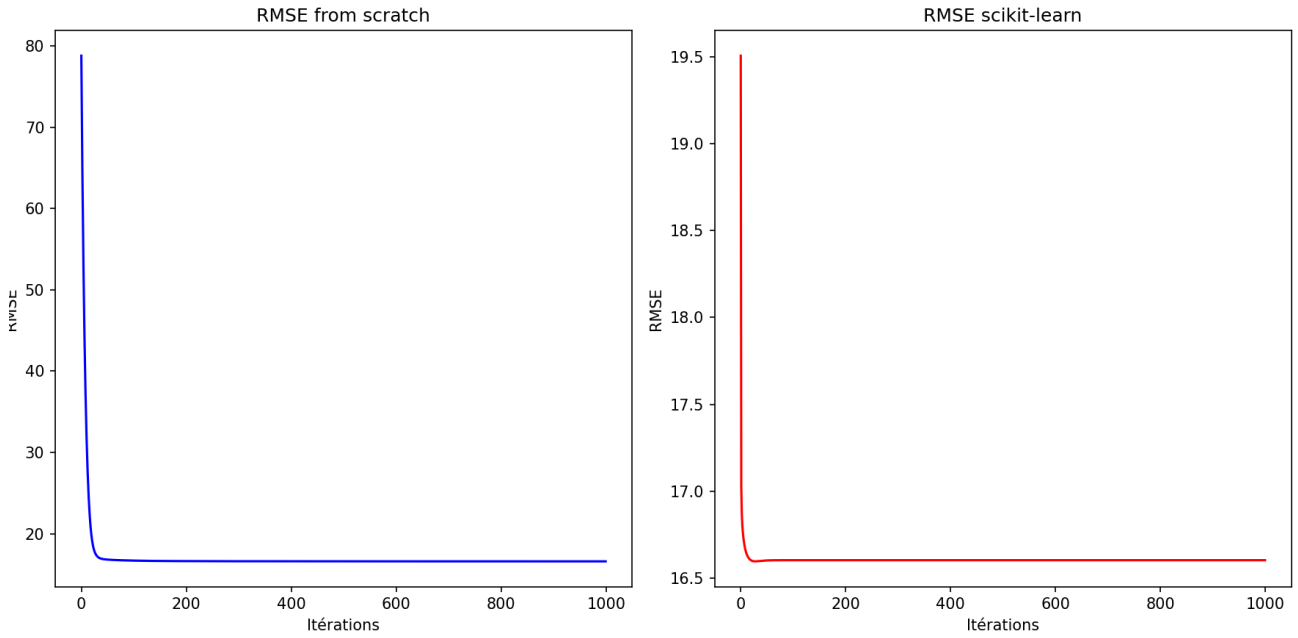


Conclusion:

Les résultats de la régression Ridge montrent que, bien que le modèle ait une capacité prédictive pour maxO3, les erreurs et le score R2 indiquent des possibilités d'amélioration. L'application de la transformation logarithmique à maxO3 a permis d'obtenir des prédictions beaucoup plus précises, mais la forme en U observée dans les résidus suggère une non-linéarité persistante. Il est donc nécessaire d'ajuster davantage le modèle pour mieux capturer la relation entre les variables. Un modèle plus adapté permettrait également d'améliorer les performances.

4. Régression Lasso

Dans un premier temps, on utilise les mêmes paramètres par défaut que la fonction lasso de scikit-learn (1000 itérations et alpha à 0.1). Pour évaluer la convergence, on utilise la métrique de la RMSE.



Dans les deux cas, la RMSE converge bien. Pour la régression lasso from scratch, elle a besoin d'un peu de plus d'itérations pour converger.

On peut également voir que la RMSE ne part pas de la même valeur au début. Cela est dû au fait que Lasso de scikit learn utilise une technique spécifique pour initialiser les coefficients, tandis qu'ils sont initialisés à 0 dans ma fonction from scratch.

Valeur final:

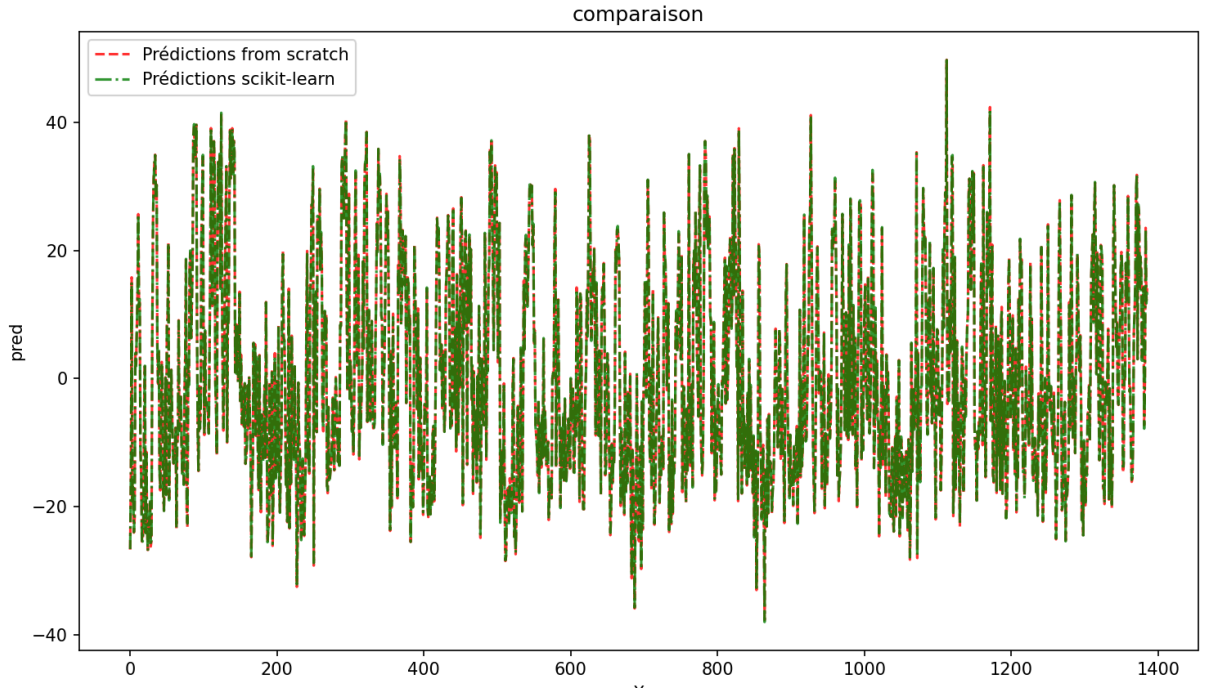
```
RMSE from scratch: 16.601498847661937
```

Temps d'exécution:

```
Time from scratch: 0.06502151489257812
Time from scikit-learn: 0.0030143260955810547
```

Pour ces mêmes paramètres, le temps pour ma fonction from scratch est environ 20 fois plus lent que la fonction de scikit learn.

En termes de résultats, en faisant des prédictions aléatoires, on peut voir que les prédictions de Lasso from scratch sont confondues avec les prédictions de scikit learn.



Pour l'optimisation, nous effectuons un GridSearch sur une liste de paramètres. Cela va entraîner le modèle sur toutes les combinaisons possibles de paramètres et retourner les meilleurs paramètres trouvés suite à l'évaluation. On évalue les modèles avec **le nombre d'itérations, le terme alpha pour L1 et le learning rate**. Parmi ces valeurs:

```
alphas = [0.001, 0.01, 0.1, 0.2, 0.5]
learning_rates = [0.001, 0.01, 0.1, 0.2, 0.5]
n_iterations_list = [10, 100, 500, 1000, 5000]
```

```
Best parameters for scratch model: {'alpha': 0.001, 'learning_rate': 0.001, 'n_iterations': 100}
Best parameters for scikit-learn model: {'alpha': 0.1, 'n_iterations': 100}
```

Le meilleur nombre d'itérations est 100 pour les deux. Alpha semble être plus intéressant s'il est petit dans la fonction from scratch. Enfin, on peut voir que la fonction Lasso de scikit learn n'utilise pas de learning rate. En effet, elle utilise une autre méthode que le gradient descent puisqu'elle utilise le coordinate descent qui n'utilise pas de learning rate.

5. SVM

5.1. SVR

SVR est une méthode de régression qui utilise la méthode de SVM pour prédire une valeur quantitative. Voici les résultats des évaluations modèles obtenus à partir du notebook python (SVM_reg.ipynb).

Scikit-Learn :

Temps d'execution scikit-learn : 0.0823s

- MAE scikit-learn : 13.0513 - MSE scikit-learn : 283.4976

- RMSE scikit-learn : 16.8374 - R^2 scikit-learn: 0.5039

Scratch :

Temps d'execution From Scratch : 5.2088s

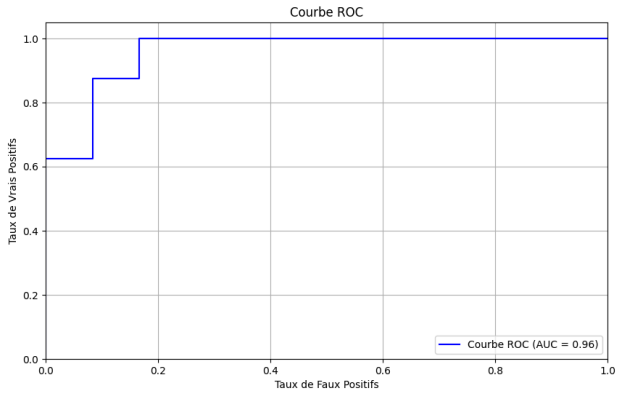
- MAE from Scratch: 14.3970 - MSE from Scratch: 356.1972

- RMSE from Scratch: 18.8732 - R^2 from Scratch: 0.3767

5.2. SVC

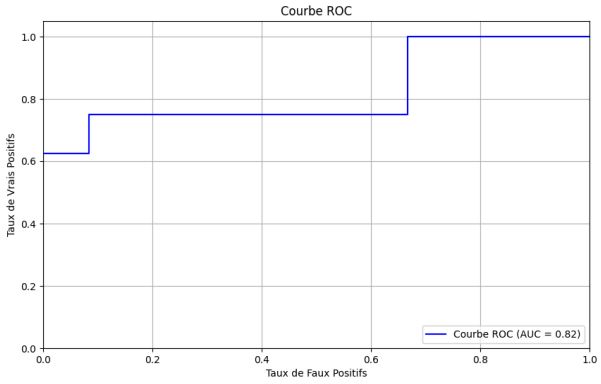
Scikit-learn:

```
Durée entrainement sklearn (s): 0.2283625602722168
Durée d'éxecution sklearn (µs): 1384.8
Accuracy SVC sklearn: 0.8
AUC SVC sklearn: 0.9583333333333334
```



SCRATCH

```
Durée entrainement from scratch (s): 3.8680896759033203
Durée d'éxecution from scratch (µs): 62.1
Accuracy SVC from scratch: 0.85
AUC SVC from scratch: 0.8229166666666666
```



6. Bonus Prédiction Salaire

6.1 Decision Tree et Random Forest

Régression

Decision Tree Scratch

```
Valeurs RMSE du tableau : [504.71022109 259.43854213 480.38039933 327.63267238 224.78870387
 534.11545932 377.47897609 371.47275551 354.24086062 391.85417469]
Moyenne : 382.6112765042032
Ecart-type : 95.68872646096841
Valeur RMSE finale sur le test set :  385.80974339638357
Coefficient de determination sur le test set : 0.50
Temps d'exécution SCRATCH : 12.723039150238037 secondes
```

Random Forest Scratch

```
Valeurs RMSE du tableau : [438.40634316 185.71830184 519.75725145 194.6475786  216.81716083
 239.7423857  205.63963423 197.55372268 242.34259355 367.86869549]
Moyenne : 280.84936675373876
Ecart-type : 112.17020843688847
Valeur RMSE finale sur le test set :  345.89478342960706
Coefficient de determination sur le test set : 0.60
Temps d'exécution SCRATCH : 58.52687311172485 secondes
```

Classification

Decision Tree Scratch

```
Meilleurs paramètres trouvés :  {'max_depth': 50, 'min_samples_split': 2}
Précision moyenne sur l'ensemble de train :  0.8035365853658536
Résultats sur ensemble de test avec les meilleurs paramètres
              precision    recall  f1-score   support

           0       0.88      0.79      0.83        28
           1       0.83      0.91      0.87        32

    accuracy                           0.85        60
   macro avg       0.85      0.85      0.85        60
weighted avg       0.85      0.85      0.85        60

Matthews correlation coefficient :  0.7002186247515697
F1-score :  0.8656716417910447
Précision : 85.0000%
Temps d'exécution SCRATCH : 5.299955129623413 secondes
```

Random Forest Scratch

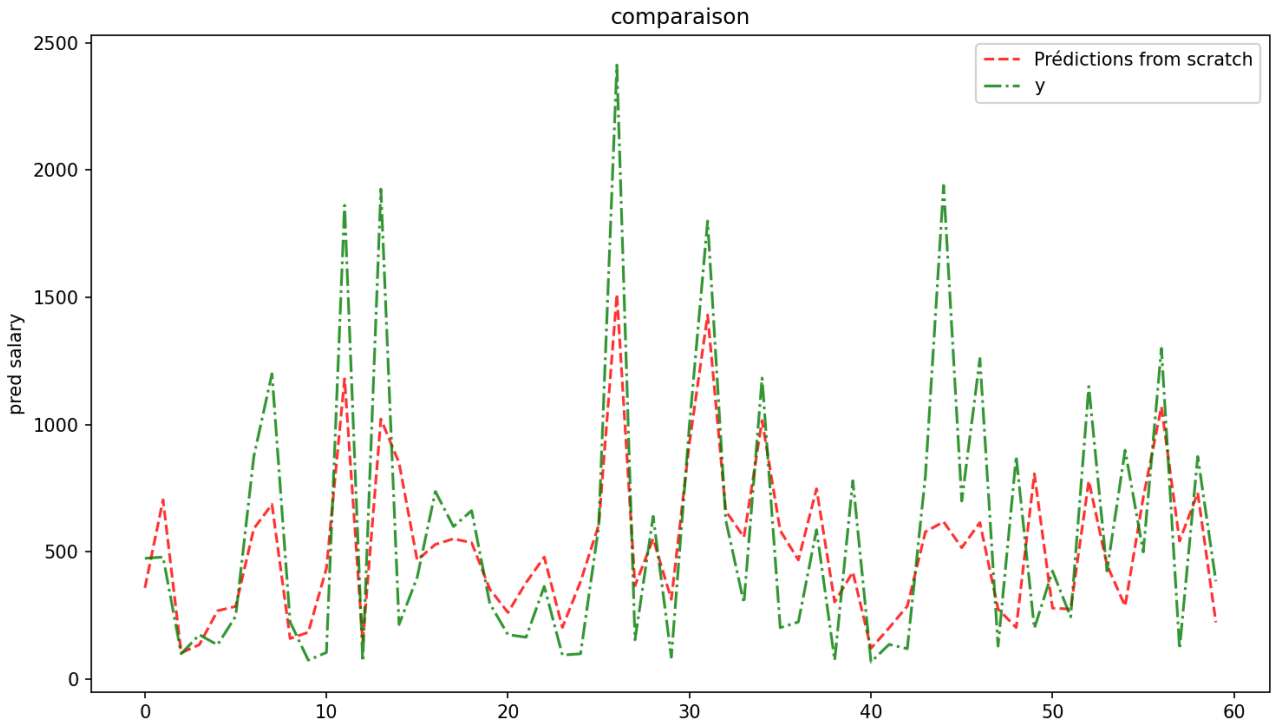
```
Meilleurs paramètres trouvés :  {'max_depth': 500, 'min_samples_split': 5}
Précision moyenne sur l'ensemble de train :  0.8576829268292684
Résultats sur ensemble de test avec les meilleurs paramètres
              precision    recall  f1-score   support

           0       0.83      0.89      0.86        28
           1       0.90      0.84      0.87        32

    accuracy                           0.87        60
   macro avg       0.87      0.87      0.87        60
weighted avg       0.87      0.87      0.87        60

Matthews correlation coefficient :  0.734968415259167
F1-score :  0.870967741935484
Précision : 86.6667%
Temps d'exécution SCRATCH : 22.19308876991272 secondes
```

6.1 Régression Lasso




```
RMSE from scratch: 292.76555406927105
```

```
Time for scratch: 0.05557036399841308 6
```