Rapport de Projet : Bio-Informatique

Introduction :

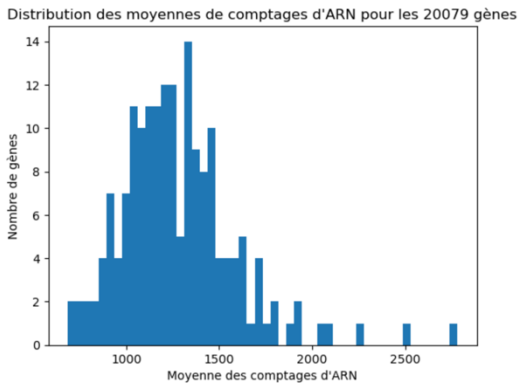
L'ensemble de données que nous avons à analyser se compose d'un fichier regroupant des échantillons pour lesquels nous avons la quantité de certains gènes, ainsi que d'un fichier regroupant les annotations sur ses échantillons ce qui nous permettra de mieux pouvoir les comprendre/ les analyser par la suite. Dans ses annotations nous pouvons différencier deux groupes principaux, les échantillons « ALS » (la maladie que nous essayons ici de comprendre) et ceux ne l'ayant pas.

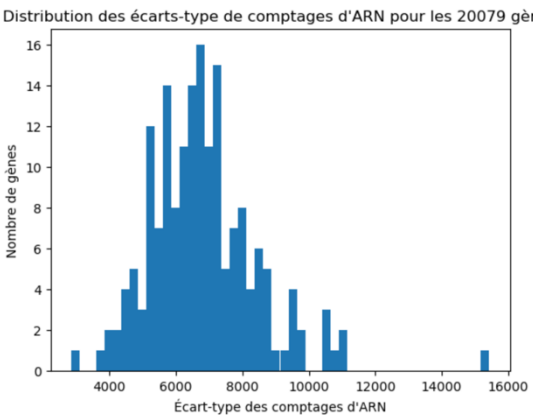
L'objectif de ce projet est donc de construire un modèle permettant de déterminer les gènes les plus liés à la maladie d'ALS. La classification des gènes en fonction de leur importance dans l'existence d'une maladie est une problématique importante en médecine, cela permettrait de faciliter le travail des médecins ainsi que de mieux pouvoir détecter la maladie en amont. Le jeu de données est supervisé, grâce aux annotations nous savons si un échantillon est ou n'est pas ALS. Nous sommes ici dans un problème de classification binaire des données.

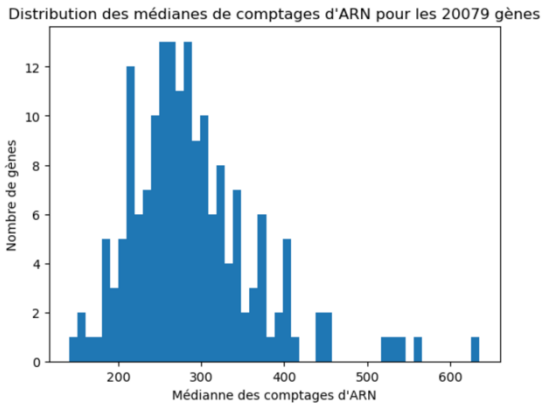
Il est important de noter ici que nos données sont brutes et qu'il faudra d'abord faire un pre-processing en enlevant les informations que nous pouvons tout de suite définir comme étant non-pertinentes. Un faible taux d'ARN par exemple, le corps peut survivre à des changements mineurs, on cherche donc à enlever le « bruit ». Par la suite nous essaierons de classifier ses données en utilisant la Régression Logistique et l'élastic-net. Finalement nous pourrons évaluer les performances de notre modèle en utilisant des métriques telles que la précision, le rappel, le F1-Score ainsi que l'aire sous la courbe ROC.

Exploration préliminaire des données :

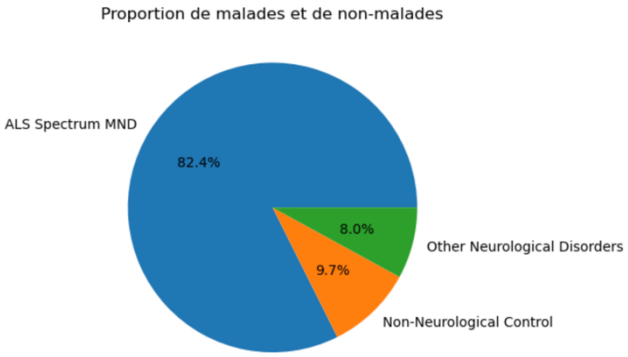
Avant de se lancer dans ce projet nous avons fait une première analyse qui nous a permis de mieux comprendre nos données. Nous avons tout d'abord cherché à voir la distribution de nos données, les moyennes de comptage d'ARN des différents gènes, les médianes ainsi que les écarts types. Cela nous a permis d'intuiter grossièrement le nombre de gènes importants.







Nous avons ensuite visualisé la proportion d'échantillons de personnes atteinte d'ALS vs ceux ne l'étant pas, nous avons observés que le set de donnée n'est pas balancé, ~82% d'échantillons ALS pour ~17% de non-ALS. Cela ne devrait pas poser des problèmes particuliers ici car il s'agit de comparer les échantillons malades et les contrôles, les contrôles n'ayant aucune maladie sont supposés être assez similaires les uns par rapport aux autres.

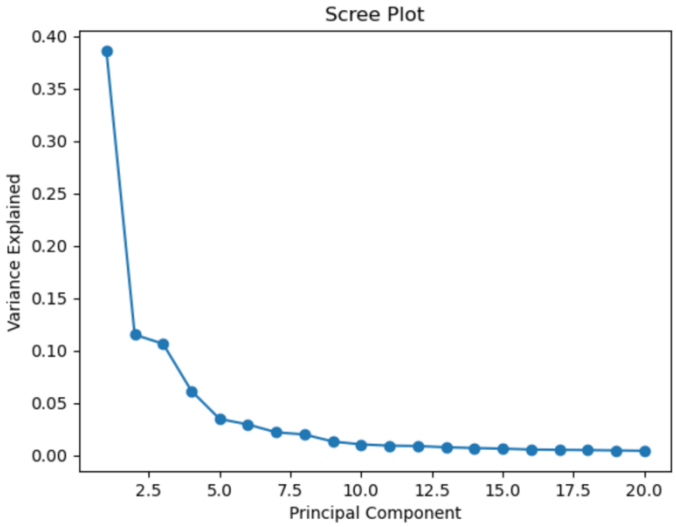


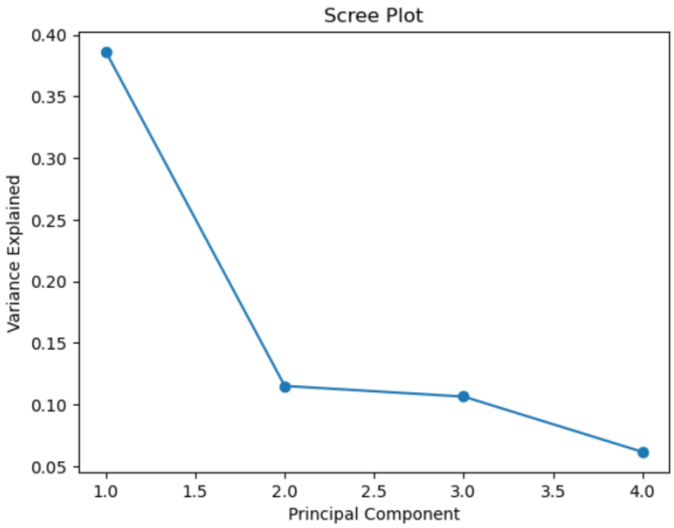
Pre-Processing :

Grâce aux informations que nous avons récoltés avec l'exploration préliminaire nous avons décidé d'exclure les gènes avec un faible comptage moyen du Dataset. Ces gènes ne seront en aucun cas significatif par la suite. Ensuite, ayant fusionné notre Dataset contenant les informations sur les gènes et celui des annotations nous avons pratiqué un encodage One-Hot dessus pour transformer nos données catégorielles en données numériques.

PCA :

Après avoir visualisé nos données et les avoir trié une première fois intuitivement, nous avons procédé à l'analyse à l'aide d'outils mathématiques. Nous avons commencé par utiliser l'analyse en composantes principales (PCA) pour réduire le nombre de variables à utiliser par la suite. Après avoir standardisé nos données, nous avons cherché le nombre optimal de composantes à conserver tout en ne perdant pas trop d'informations. Pour cela, nous avons utilisé un Scree-plot, qui permet d'observer graphiquement les variables pouvant être considérées comme du "bruit" et les variables importantes. Cette analyse nous a permis de réduire le nombre de variables de ~28000 à 4, tout en conservant 67 % de l'information initiale.





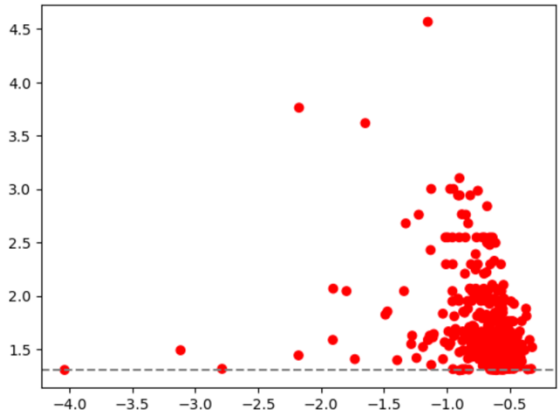
Analyse Unidimensionnelle :

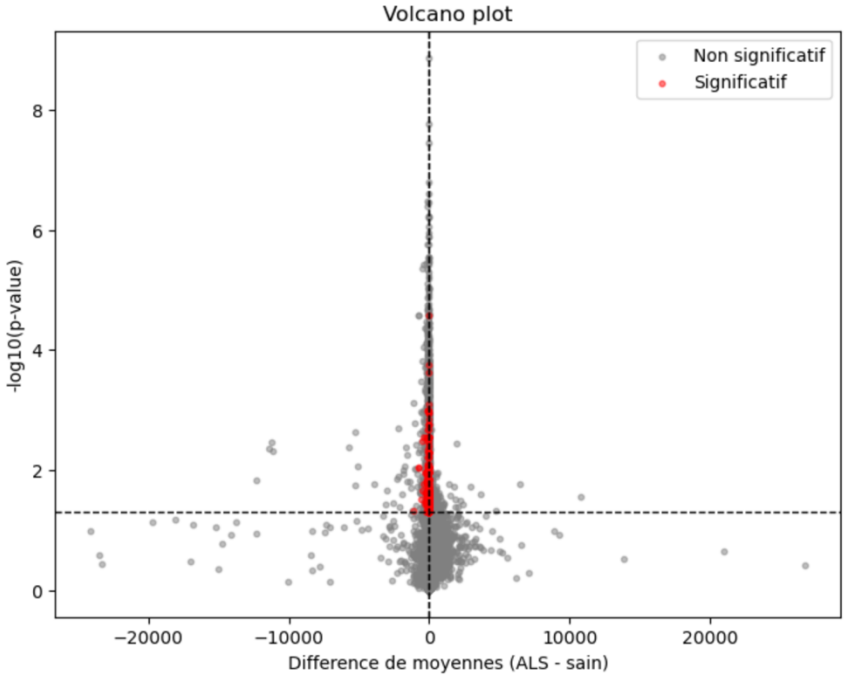
```
1 sum(pca.explained_variance_ratio_)
0.669250054678142
```

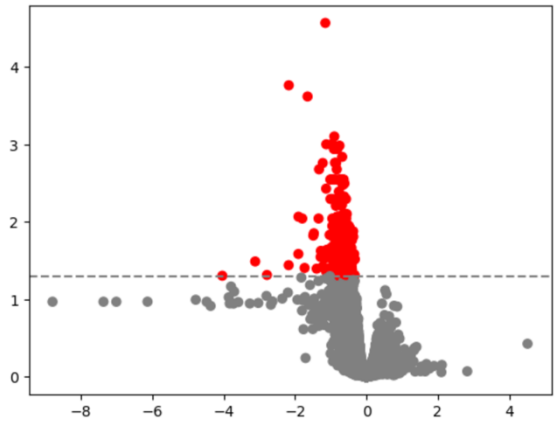
Nous avons ensuite poussé les analyses plus loin en utilisant un outil d'analyse plus puissant, le T-Test. Nous avons donc appliqué la fonction pour obtenir la valeur de la p-value de chaque gène. Le T-test est un moyen d'estimer l'importance de chaque gène, ici dans la différenciation des échantillons saint et ceux ALS, en regardant les gènes un par un.

Grâce à cela nous pouvons appliquer du Multiple-testing de la librairie python « statmodels » avec le paramètre « fdr » (False Discovery Rate), le paramètre va permettre ici de réduire le nombre de faux positif au maximum et corrigera les p-values en fonction. Ici nous essayons d'obtenir 5% au maximum de faux positifs. Cette fonction nous donne aussi le nombre de paramètres importants. A l'inverse de la PCA, nous obtenons ici 290 gènes principaux.

Nous avons ensuite cherché à visualiser nos résultats avec un volcano plot, nous avons défini une limite en Y en dessous de laquelle les données ne sont pas importantes et de même en X, les données comprises entre les bornes sont celles qui nous intéressent le plus et celles que nous devrions retrouver par la suite comme étant les plus liées à la maladie d'ALS.







Analyse Multidimensionnelle :

Enfin nous avons utilisés l'analyse à plusieurs dimensions, cette manière d'analyser permet de prendre en compte les combinaisons de gènes. Pour cela nous allons utiliser le Classifier de régression logistique sur nos données pour essayer de prédire grâce au nombre d'ARN le groupe auquel l'échantillon appartient. Pour cela nous rajoutons en paramètre la méthode de régression : « elastic-net ». Cette méthode permet de gérer de grands data-set et est facile à paramétrer. Les meilleurs résultats que nous avons obtenus avec cette méthode nous ont donné une précision de ~64%. Le nombre de faux positif est assez faible mais les faux négatifs quant à eux nous ont semblé trop grand. Notre modèle avec les paramètres actuels semble donc prometteur et améliorable.

```
0.660377358490566

              precision    recall  f1-score   support

           0       0.17      0.20      0.18        10
           1       0.80      0.77      0.79        43

    accuracy                           0.66        53
   macro avg       0.49      0.48      0.48        53
weighted avg       0.68      0.66      0.67        53
```

Nous avons ensuite essayé de visualiser les 10 gènes étant les plus importants, grâce aux résultats de la régression nous avons obtenus un tableau de valeurs représentant les coefficients de régression. Ses coefficients une fois triés nous ont donnés la liste par importance des gènes permettant la différenciation entre un échantillon ALS et un échantillon de contrôle.

```
Gene n° 0 ['SLC2A14'] valeur de l importance : 0.02112915992143495
Gene n° 1 ['XAF1'] valeur de l importance : 0.01713894898045126
Gene n° 2 ['CYP1A1'] valeur de l importance : 0.01630316034869469
Gene n° 3 ['CHMP1B2P'] valeur de l importance : 0.01478026057013032
Gene n° 4 ['GBP4'] valeur de l importance : 0.01460767041080215
Gene n° 5 ['TNFSF10'] valeur de l importance : 0.014079352654184661
Gene n° 6 ['HSPB2-C11orf52'] valeur de l importance : 0.013681327256252583
Gene n° 7 ['TC2N'] valeur de l importance : 0.013044949329976395
Gene n° 8 ['MIR2861'] valeur de l importance : 0.013003389148636866
Gene n° 9 ['GBP1P1'] valeur de l importance : 0.012800438767299284
```

Nous avons ensuite effectué une cross-validation sur les données, les meilleurs résultats que nous avons obtenus nous ont donnés une « balanced-accuracy » d'environ 58%. Ainsi qu'une aire sous la courbe roc de 58% ce qui indique un modèle peu indicatif.

```
from sklearn.metrics import balanced_accuracy_score
balanced_accuracy_score(y_testF, model_prediction)
```

```
0.5883720930232558
```

```
from sklearn.metrics import roc_auc_score
sklearn.metrics.roc_auc_score(y_testF, model_prediction)
```

```
0.5883720930232558
```

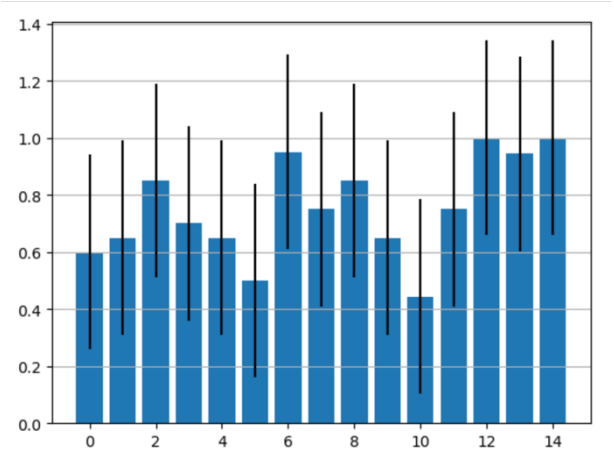
```
from sklearn.metrics import matthews_corrcoef
matthews_corrcoef(y_testF, model_prediction)
```

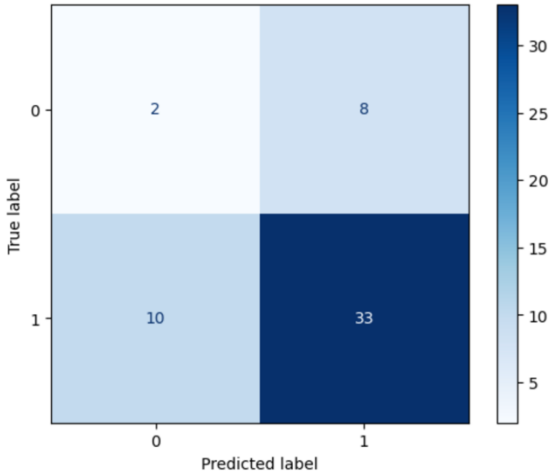
```
0.2992497077641
```

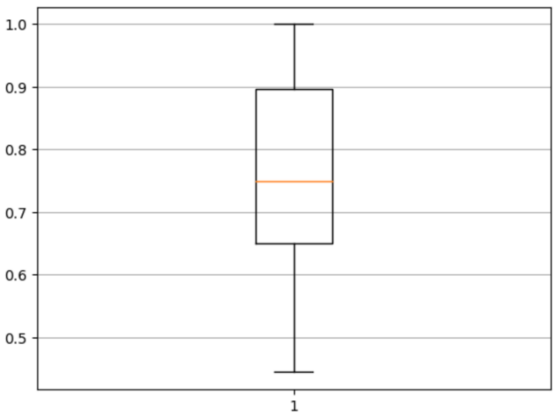
```
unique, frequency = np.unique(y_trainF, return_counts = True)
print("Unique Values:", unique)
print("Frequency Values:", frequency)
```

```
Unique Values: [0 1]
Frequency Values: [ 21 102]
```

D’après la matrice de confusion sur les résultats que nous obtenons les résultats ont l’air plutôt bons dans l’ensemble, peu de faux positifs sont détectés ce qui est rassurant, on ne détectera pas quelqu’un de non-malade comme étant malade dans la plupart des cas, le nombre de faux négatifs quant à lui est plutôt élevé ce qui montre que notre modèle n’est pas encore parfait et qu’il faudrait pousser les analyses avant d’affirmer quelque chose. De plus lorsque l’on observe l’histogramme de la précision de notre modèle sur différents découpage des échantillons nous pouvons observer qu’il est extrêmement performant sur certains sets de données et beaucoup moins sur d’autres (on varie de 40% à 90% d’accuracy).








```
1 sklearn.metrics.roc_auc_score(y_test, model_prediction)
```

```
0.48372093023255813
```

Conclusion :

Nous avons testé de nouvelles méthodes de machine learning dans l’objectif de déterminer l’importance de certains gènes dans la maladie d’ALS. Nous avons pu observer qu’ici la meilleure méthode de classification est la Régression Logistique. Les résultats sont encore largement améliorables avec une meilleur sélection préliminaire des gênes à garder.