Challenge IMA205 : Détection de mélanomes.

# Introduction.

La détection de mélanomes est un enjeu majeur de machine Learning puisque qu’une détection suffisamment précoce permet d’éradiquer avec un taux très élevé de réussite la venue d’un cancer de la peau.

J’ai extrait des features et testé différentes méthodes de machine learning afin de déterminer laquelle était la meilleure pour la classification et pourquoi.

J’ai réalisé ce projet en 4 étapes, chaque étape est décrite dans un notebook. Ce document contient des explications rapides sur chaque méthode, des conclusions partielles sur chaque méthode et une conclusion générale.

1. Extraction des features et méthode de sélection des features.

*Notebook intitulé : I) Extraction,Sélection des features et premiers essais.ipynb*

1. Méthodes Bayésiennes et K-plus proches voisins

*Notebook intitulé : 2)K\_nearest\_neighboors\_Lda\_Qda.ipynb*

1. Méthodes ensemblistes : Decision Trees, Random Forest, Bagging

*Notebook intitulé : 3)Trees, bagging, random forest.ipynb*

1. Support Vector Machines.

*Notebook intitulé  : 4)Support Vectors Machines*

# Extraction des features et méthode de sélection des features.

Objectifs de la section :

\*expliquer les méthodes utilisées pour l’extraction des features

\*introduire l’utilisation d’une sélection de features

\*Discuter de l’influence des features sur la prédiction.

## Extraction des features.

Premièrement, une partie importante du projet a été d’extraire les features. Pour cela, j’ai notamment utilisé l'article *: Performance of a dermoscopy-based computer vision system for the diagnosis of pigmented skin lesions compared with visual evaluation by experienced dermatologists[[1]](#footnote-1)*.

L’idée est de s’inspirer du code ABCD déjà utilisé par les dermatologues pour en déduire des critères de classification (A (asymétrie), B (bords),C (couleurs),D (dimensions)).

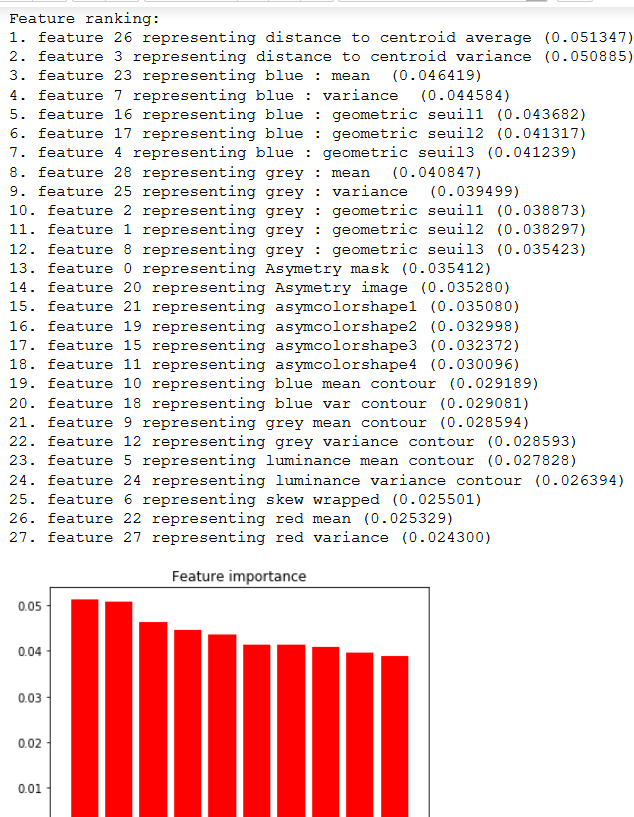
**Features d’Asymétrie :** 12,13, 24 (en utilisant scipy.stats.skew) (asymétrie de forme).

-features d’asymétrie de couleur et de forme : grâce à plusieurs seuils, on calcule la moyenne et la variance de la disposition des centroids et des moments des composantes des masques résultants (features 14,15,16,17).

**Features Bords :** Distance euclidienne au centroid des contours, (features 0 et 1).

Moyenne et variance de la couleur des contours, réalisé dans le canal bleu (features 18 et 19, dans le canal gris(20,21) et dans le canal de luminance des images masquée en lab (22,23).

**Features de Couleur** : features 2,3, 7,8, 25,26,25,28 qui représentent la moyenne et la variance des valeurs de pixels de l'image masquée pour les canaux (respectivement) bleu, gris, rouge et vert.

Irrégularité de la distribution de couleur : en utilisant les features 'Geometric' décritent dans la publication. On seuille l'image aux 25è, 50è et 75è percentiles et on regarde le nombre de composantes connexes obtenues après seuillage. Plus il y en a, plus le grain de beauté a de chances d'être un mélanome. J'ai réalisé cela dans le canal bleu (features 4,5,6) et l'image en niveaux de gris (features 9, 10,11).

## Sélection des features :

Grâce à des forêts aléatoire, il possible de montrer l’importance relatives des différente features (l’importance d’une feature correspond à l’augmentation de l’erreur de prédiction qui est réalisée lorsque l’on retire cette feature du data set). J’ai donc affiché l’importance relative de celles-ci :

On constate que les features n’ont pas toutes la même importance.

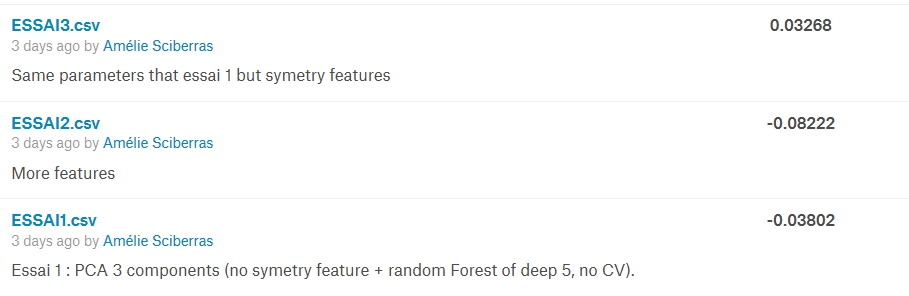
Il peut être dans ces cas là judicieux d’utiliser une méthode de sélection des features, afin d’être sûr que toutes les features utilisées dans l’algorithme ont un sens et donc de réduire le temps de calcul notamment.

J’utiliserai notamment la méthode de la principal component analysis qui consiste à projeter les features de plus grande variance (c’est-à-dire les plus importante au sens de la PCA) dans un espace de plus petite dimension.

## Tests avec un nombre de plus en plus important de features.

J’ai effectué une prédiction assez basique (pas de cross-validation ni de rééchantillonage) en utillisant une projection par PCA puis un classifieur de forêts aléatoires avec une profondeur de 5 dans les arbres utilisés.

J’ai notamment ajouté entre le test 1 et le test 3 une feature de symétrie voici les résultats :



On constate que l’ajout de cette feature permet d’améliorer grandement le score, bien qu’elle ne ressorte qu’en 12 et 13è positions dans le classement d’importance des features pour les forêts aléatoires.

Cependant, les tests sont assez bas, nous allons donc utiliser d’autre méthodes afin de tenter de les améliorer.

**Conclusion :**

-les features ont été extraites avec des critères de symétrie, des critères sur les bords des grains de beauté et des critères de couleur.

-Il semble que l’utilisation de la principale component analysis est judicieuse pour améliorer les prédictions.

-L’absence de cross-validation de rééchantillonnage ou la mauvaise efficacité de la méthode de Random forest peut être l’une des causes d’un score aussi bas.

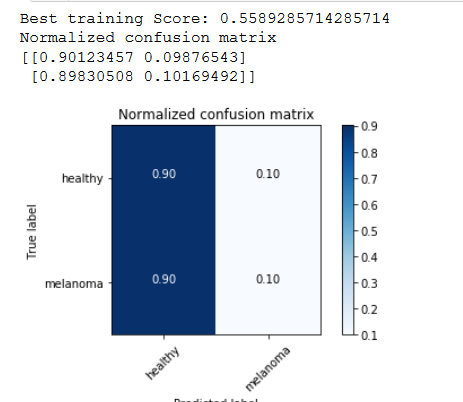
# Méthodes Bayésiennes et K-plus proches voisins

Objectifs de la section :

\*Comparer les méthodes de K-plus proches voisins, Linear Discriminant Analysis et Quadratic Discriminant Analysis

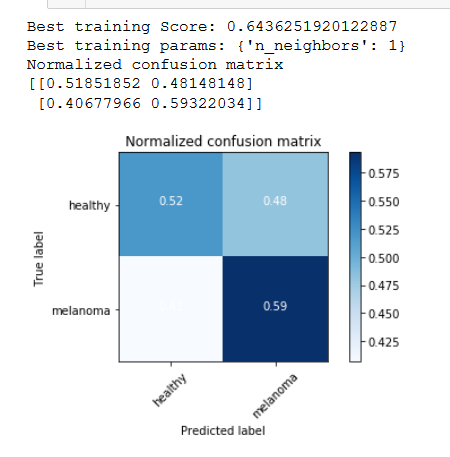
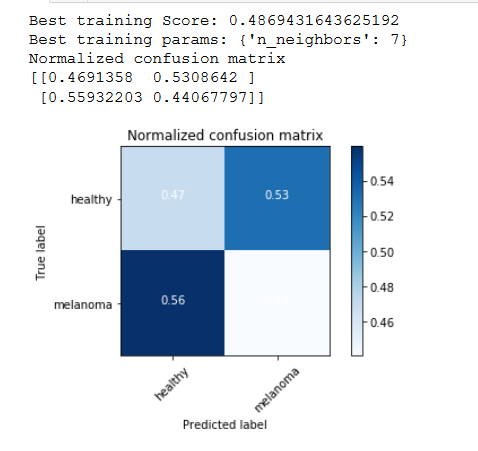
\*Montrer l’importance du rééchantillonnage et de la principale component analysis avec un nombre de composantes bien choisi.

## Importance du rééchantillonnage.

Sans équilibrer le dataset, mais avec une cross-validation sur le nombre de voisins,on obtient la matrice de confusion ci-contre. Tous les grains de beauté ou presque sont détectés dans la catégorie ‘en bonne santé’ car la majorité des grain de beauté de la base de donnée sont bénins.

Ainsi, pour la plupart des grains de beauté, les clusters de voisinages qui sont réalisés sont réalisés avec des grains de beauté sains. Ce test donne par ailleurs un très mauvais score sur Kaggle.

On utilise donc la méthode Adasyn qui consiste à générer de nouveaux points de la classe minoritaire en créant des lignes entre les points d’une même classe. Adasyn essaie de plus de rendre le modèle plus facile à apprendre lors du choix de ces points, c’est pourquoi j’ai choisi cette méthode plutôt qu’un random over-sampling.

En équilibrant la base de données : le score est meilleur sur Kaggle et sur le notebook.

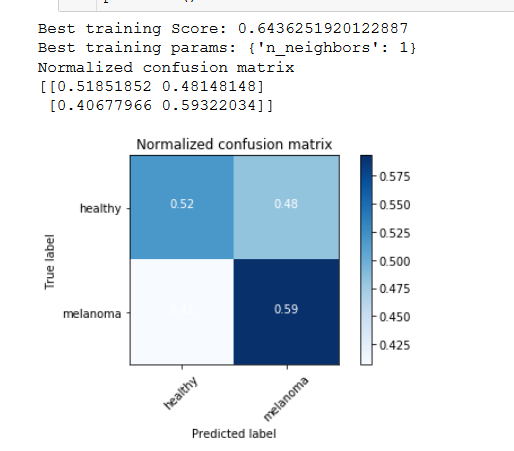
De plus la matrice de confusion est clairement meilleure.

On implémente ensuite un rééchantillonage et on compare avec un modèle avec PCA bien choisie.

## Méthode de sélection du nombre de composantes de la PCA et efficacité de la méthode.

Afin de choisir le meilleur nombre de composant, on calcule tous les scores réalisés avec PCA+KNN en itérant de 2 à Nombre de features-1.

On choisit ensuite de regarder le résultat de la PCA qui a donné le meilleur score :



On constate que l’efficacité est très bonne et la matrice de confusion est très belle.

On va maintenant choisir quelle méthode soumettre au Kaggle en comparant les K-plus proches voisins à la méthode de LDA et QDA.

## Efficacité comparée des trois méthodes

On utilise plus de features que précédemment (ce test a été effectué alors que plus de features avaient été calculées).

On utilise une PCA et un rééchantillonnage comme précédemment.

Résultats :

-La QDA a le meilleur score mais bien souvent, la matrice montre un trop grand nombre de faux positifs.

-La LDA a le moins bon score mais une bonne matrice

-Les KNN ont un bon score et une bonne matrice.

J’ai donc choisi de soumettre les KNN avec le meilleur critère de PCA à Kaggle. Le score était très mauvais, alors que dans l’ensemble d’entrainement il semblait assez bon ; avec un bon taux de vrais positifs et une accuracy plutôt grande.

Je me suis alors rendue compte qu’à chaque fois, la cross validation renvoyait un nombre de plus proche voisins égal à 1. Or, dans le cas des plus proches voisins cela peut très facilement correspondre à un cas de sur apprentissage.

J’ai donc souhaité comparé avec la Linear Discriminant Analysis, mais le score était encre moins bon. J’étais moins surprise car l’accuracy était plus faible.

**Conclusion :**

-La PCA et le Resampling permettent d’améliorer la prédiction dans les cas de K plus proches voisins notamment.

-Les K plus proches voisins peuvent sembler donner un bon score mais peuvent facilement donner des cas d’overfitting, surtout si le nombre de plus proches voisins est faible.

# Méthodes ensemblistes : Decision Trees, Random Forest, Bagging

Objectifs de la section :

\*Comparer les trois méthodes.

\*Essayer d’améliorer le score obtenu à la section 1, notamment en utilisant la cross-validation.

## Comparaison des trois méthodes.

Pour chaque méthode, j’utilise les techniques de PCA et de validation croisée, avec toujours la même méthode que précédemment pour choisir le meilleur nombre de composantes du PCA.

-Les arbres de décision donnent plutôt de bons résultats mais, sans surprises ceux-ci varient beaucoup. En effet, le moindre petit changement dans le dataset (par exemple lors d’un resampling) peut conduire à un arbre complètement différent et donc à un changement dans la prédiction.

-Les arbres de décisions et le bagging donnent un résultat faible mais meilleur que tout ce qui a été fait jusqu’à présent. Ces deux algorithmes sont plus stables car il pondère la variabilité des arbres en en implémentant plusieurs dans leur algorithme d’où les meilleurs résultats.

La cross validation utilisée et le plus grand nombre de features mettent en avant l’influence de ces deux paramètres pour donner une meilleure prédiction qu’à la question 1.

# Support Vector Machines.

Objectifs de la section :

\*tester une méthode qui peut s’avérer plus efficace que les méthodes ensemblistes testées plus haut.

\*Comparer SVM linéaire et SVM non linéaire.

## Test de la méthode.

Une fois de plus, j’utilise la PCA et je normalise mes variables avant d’utiliser cette méthode car dans le cas des SVM cela peut grandement fausser les résultats. J’utilise la cross-validation pour régler la régularité.

La PCA ne marchant pas toujours bien pour les SVM, j’ai également fait des tests sans cette méthode pour comparer.

## Résultats.

Pour les deux types de noyaux, la PCA semble bien fonctionner (elle améliore légèrement les résultats sur l’ensemble d’entrainement), mais l’accuracy peine à dépasser les 0.5. Cette méthode n’est donc pas très efficace dans ce cas, ce qui est confirmé par les scores rarement au-dessus de 0 que fournissent les essais sur Kaggle.

## Comparaisons des deux types de noyaux.

Le noyau linéaire est moins bon que le noyau non linéaire, aussi bien sur le Notebook que sur Kaggle. Il semble donc que le problème n’est pas linéairement séparable.

Je teste donc plusieurs noyaux parmi ceux proposés par SKlearn pour améliorer les résultats et c’est le noyau rbf qui fonctionne le mieux.

Cependant, bien qu’il me semble que cette méthode soit en général assez efficace, elle l’est moins que les classifieur faibles que sont les arbres de décision.

# CONCLUSION.

Il semble clair que des prétraitements comme la normalisation et le rééchantillonnages sont très utiles pour améliorer des résultats de prédictions. L’utilisation de la PCA permet également, bien souvent de beaucoup améliorer les résultats en générant des variables qui sont plus importantes. La validation croisée est aussi une méthode très efficace pour améliorer des résultats.

Le bagging est la méthode qui a donné le meilleur résultat, peut-être parce que les noyaux disponibles dans les SVMs sont peu nombreux et qu’aucun d’entre eux n’est adapté pour la classification dont ce projet fait l’objet. Les résultats finaux sont globalement assez décevants pour les méthodes qui ont été testées, peut être que le problème vient plus de l’extraction des features que des traitements que j’ai réalisé dessus par la suite. Une forward sélection, par exemple, aurait peut-être éliminé des features qui gênaient pour la prédiction.

1. de Maciel Zorteaa,Thomas R. Schopfb, Kevin Thonb, Marc Geilhufea, Kristian Hindberga, Herbert Kircheschc,Kajsa Møllersenb, Jörn Schulza,Stein Olav Skrøvsethb,Fred Godtliebsen. [↑](#footnote-ref-1)