Antoine

Moussu

Cnam Paris

Projet RCP 209 2019 /2020

I. INTRODUCTION

L'objectif du projet est de faire une analyse de données d'un jeu de données choisi et de réaliser une modélisation décisionnelle en fonction des objectifs donnés en consigne (classification ou régression).

Ici, on a choisis le jeu de données « Wine quality DataSet » , ce jeu de données contient des informations sur 4898 vins blancs du Portugal , parmi ces informations , on retrouve 11 variables explicatives qui sont des mesures physico-chimiques des vins en question (ph, concentration d'acide citrique etc) et une variable cible à expliquer qui est la note du vin sur 10.

L'objectif de notre projet est de construire un modèle soit de régression soit de classification afin de déterminer la note des vins en fonction des différentes variables. On peut réaliser une classification pour onze classes (de 0 à 10) ou une régression,

Bien évidement on explicitera plus tard dans notre projet le choix de la tâche à effectuer. Mais dans un premier temps, on va réaliser une analyse exploratoire des données afin de mieux comprendre comment elles sont structurées et quel type de modélisation est la plus adapté.

Puis dans un deuxième temps, on effectuera une modélisation prédictive des données.
On construira un jeu test et un jeu d'entrainement à partir desquels on entrainera et testera différents algorithmes de type machine learning et ainsi, on construira un modèle décisionnel efficace dans l'attribution des notes pour les vins.
Il convient de souligner qu'on se place dans le cadre d'un apprentissage supervisé puisque l'on possède l'information de supervision (la note sur 10) pour tous nos vins.

Dans cette étude, nous utiliserons l'outil python et les packages suivants :
numpy,pandas,sklearn,matplotlib.
Ainsi que l'application Jupyter Notebook

II. ANALYSE EXPLORATOIRE DES DONNEES

Répartition des données

L'une des premières actions à effectuer dans le cadre d'une analyse exploratoire est d'observer la distribution des notes attribuées à nos vins. Pour cela, on réalise un histogramme de la variable 'quality'.

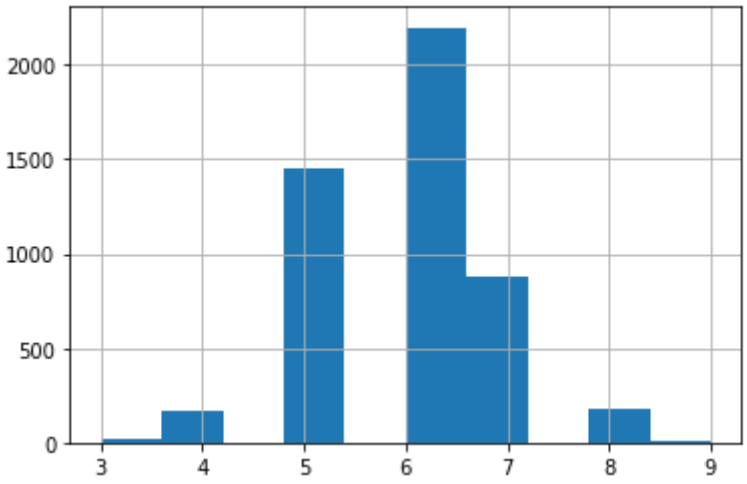

Figure1.Histogramme de la variable cible 'quality'

On remarque qu'il n'y a pas de note en dessous de 3 et il n'y a pas de 10 donc si on effectue une tâche de classification supervisée à partir de nos données , les prédicteurs que l'on entraînera seront dans l'incapacité d'attribuer les classes 0, 1, 2, 10 aux vins car ils ne seront pas entrainés à reconnaître des classes qui n'existent pas dans nos données.

On effectuera donc une classification supervisée à 7 classes.

De plus on observe que les classes 5 et 6 sont bien plus fréquentes que les autres.
On peut penser que dans le cadre d'une classification à 7 classes , les modèles entrainés risquent d'attribuer plus facilement les classes 5 et 6 lors de la phase de test et de ne pas détecter les vins exceptionnellement bons ou mauvais appartenant aux autres classes on pourrait donc faire les choix de regrouper les vins appartenant aux classes 7 8 et 9 en une classe (note>6 ou vins exceptionnellement bons) et les autres vins ordinaires ou mauvais en une autre classe. Les vins exceptionnellement bons sont les plus importants à détecter dans le cadre d'une action commerciale d'où l'intérêt de faire une classification binaire qui pourrait s'avérer plus facile à réaliser et plus performante. Il convient de préciser qu'il n'existe pas de données manquantes dans notre dataset comme on peut le voir sur le descriptif suivant avec les données encadrées en rouge, nos variables possèdent toutes 4898 valeurs (une pour chaque vins).

	fixed acidity	volatile acidity	citric acid	residual sugar	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000
mean	6.854788	0.278241	0.334192	6.391415	35.308085	138.360657	0.994027	3.188267	0.489847	10.514267	5.877909
std	0.843868	0.100795	0.121020	5.072058	17.007137	42.498065	0.002991	0.151001	0.114126	1.230621	0.885639
min	3.800000	0.080000	0.000000	0.600000	2.000000	9.000000	0.987110	2.720000	0.220000	8.000000	3.000000
25%	6.300000	0.210000	0.270000	1.700000	23.000000	108.000000	0.991723	3.090000	0.410000	9.500000	5.000000
50%	6.800000	0.260000	0.320000	5.200000	34.000000	134.000000	0.993740	3.180000	0.470000	10.400000	6.000000
75%	7.300000	0.320000	0.390000	9.900000	46.000000	167.000000	0.996100	3.280000	0.550000	11.400000	6.000000
max	14.200000	1.100000	1.660000	65.800000	289.000000	440.000000	1.038980	3.820000	1.080000	14.200000	9.000000

Figure 2. Descriptif de notre jeu de données

Toutes nos variables sont des variables quantitatives, ce qui signifie que l'on aura pas besoin d'utiliser d'encodeur pour transformer d'éventuelles variables qualitatives en revanche nos variables quantitatives sont exprimées avec des unités différentes.
Par exemple : le ph n'a pas d'unité et la concentration d'acide s'exprime en mol.L^-1, or on sait que de nombreux algorithme de modélisation sont très dépendants des unités comme les SVM . Il est donc impératif de transformer nos données en les normalisant, on va utiliser la normalisation centrée-réduite de la classe StandardScaler du package sklearn de python, c'est-à-dire que l'on va centrer et réduire les données en soustrayant à chaque données la moyenne de la variable (c'est ce qui s'appelle un centrage). Cela consiste simplement en un changement d'origine, qui place la moyenne de la distribution au point 0 de l'axe définit par la variable. Puis on réduit chaque variable en divisant toutes ses valeurs par son écart type (on réduit les données).

Cette opération nous permet de négliger l'impact des unités et cela est très pratique puisque notre étude ne s'effectue qu'avec des variables quantitatives d'unités différentes.

Une démarche intéressante pour notre analyse serait d'observer les corrélations existantes entre nos différentes variables. Sans être expert en chimie, on sait que le ph qui mesure le caractère acide ou basique d'un solution va dépendre de la concentration d'acide présente dans cette solution , on peut donc réaliser une ACP dont l'intérêts est d'obtenir de nouvelle variable non corrélées et de réduire le nombre de dimensions de notre jeu de données.

La réduction de dimensionnalité d'un jeu de données est très pratique pour mieux visualiser le jeu de données auxquels on est confronté et ainsi pour émettre une meilleure hypothèse sur le modèle prédictif à sélectionner, enfin une autre utilité de cette méthode est de réduire le nombre de calcul pour notre machine, ce qui n'est pas négligeable dans le cas où l'on veut entrainer un modèle très couteux en temps de calcul comme les algorithme à noyau ou les SVM polynomiaux de degrés >2.

On réalise donc une analyse en composante principale ou ACP (on note qu'une opération de normalisation des données de type centrée-réduite est effectuée automatiquement lorsqu'on effectue une ACP avec la classe PCA de sklearn).

On calcule l'inertie des 2 premiers axes de notre ACP (l'inertie est la quantité d'information conservée sur nos données en projetant les points de l'espace R^11 sur notre nouvelle composante).

On obtient une inertie de 98% pour nos 2 premiers axes, ce qui signifie qu'en projetant notre nuage de point dans l'espace R^11 le plan d'ACP former par nos 2 premières composantes principales, on a conservé 98% de l'information contenue dans notre jeu de données, autrement dit, on peut avoir une représentation fidèle de notre nuage de points dans notre plan d'acp.

Visualisation avec le plan d'acp :

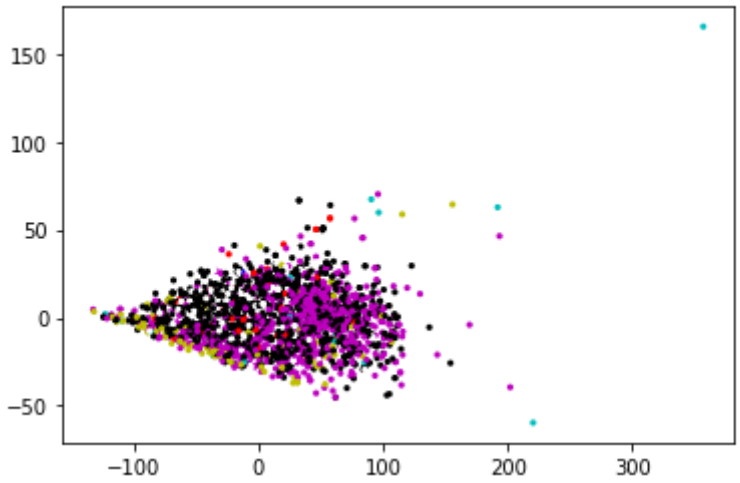

Figure 3. Nuage de point représentant notre jeu de données

Il est difficile de visualiser avec précision nos données car il y a beaucoup de vins et donc de nombreux chevauchements de point sur notre plan. Cependant on peut d'ores et déjà affirmer qu'une tache de régression ne pourra pas être possible pour bien modéliser notre problème.

On constate que les classes 5 (en violet) et 6 (en noir) dominent notre jeu de données comme on s'y attendait et on observe que la masse des 5 semble plutôt se placer sur la droite de notre plan tandis que les 6 semblent occuper la gauche de notre plan.

On se rappelle que l'un de nos principaux objectif est de repérer les vins exceptionnelle (de bonne qualité), ceci nous amène à observer comment sont répartis les très bon vins avec une note supérieur à 6 dans notre plan d'ACP.

On obtient le nuage de point suivant :

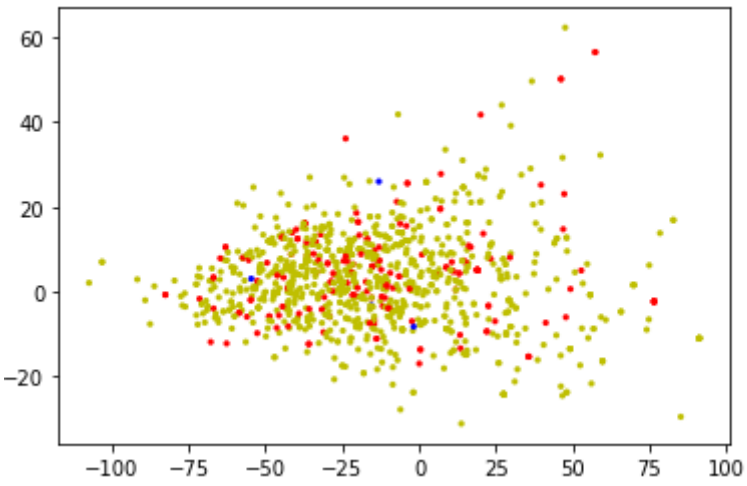

Figure 4. Nuage de point représentant les vins exceptionnels

Il y a peu de vins rares à la droite de notre plan d'ACP (avec une coordonnées sur le premier axe supérieure à 0).

Cette observation est intéressante et nous amène à envisager une modélisation telle qu'une classification binaire afin de repérer plus précisément ces vins rares qui représentent un intérêt plus important dans notre étude.

II. MODELISATION DECISIONNELLE

1) Classification multi-classes

La modélisation décisionnelle que nous allons effectuer s'opérera en plusieurs étapes dans un premier temps, on va tester différentes classes d'algorithme, puis dans un second temps on essayera d'affiner les hyperparamètres du modèle sélectionné pour améliorer la performance.

On précise que pour l'étape de modélisation on n'utilisera les données d'origine c'est à dires les 11 variables explicatives en les normalisant avec la méthode centrée-réduite et que l'on va segmenter nos données en un jeu d'entrainement (80% de nos données totales) et un jeu de test (20% de nos données totales)

On n'utilisera pas les nouvelles données obtenues avec l'ACP afin de conserver toute l'information possible.

On va donc tester différents algorithmes d'optimisation (descente de gradient, random forest, regression logistic, SVM) dans le cadre d'une classification multi-classes afin d'obtenir un modèle nous permettant de prédire le plus justement possible la classe d'un vin de notre jeu de données de 3 à 9

« Certains algorithmes (comme la descente de gradient stochastique, la classification par forêts aléatoires) sont capables de gérer des classes multiples en mode natif. D'autres (comme la régression logistique ou les machines à vecteurs de support) sont strictement des classificateurs binaires. Il

existe une stratégie nous permettant d'effectuer une classification multi-classes en utilisant plusieurs classificateurs binaires.Ainsi, pour répartir nos vins en 7 classes (note de 3 à 9), une des méthodes consiste à entraîner 10 classificateurs binaires, un pour chaque classe. Puis, pour classer une autre image, il suffit d'obtenir le score de décision pour cette image de chacun des classificateurs et de sélectionner la classe dont le classificateur a fourni le meilleur score. C'est ce qu'on appelle une stratégie un contre le reste (one-vs-the-rest, OvR). »

On va donc tester via une validation croisée k-folds avec k=4 , les 5 classifieurs multi-classes suivants :

RandomForest, regression logistique, SVM classifier

, on collecte les résultats dans le tableau suivant :

Nom du classifieur	Type de classifieur	Score moyen des validations croisées
Clf_def	Classifieur par défault	0 .449
Clf_rdmf	Forêt aléatoire	0.620
Clf2	SVM avec stratégie OvR	0.564
Clf3	Régression Logistique liblinear	0.543
Clf4	Régression Logistique lbfgs	0.546

Les meilleurs résultats sont obtenus avec la forêt aléatoire, néanmoins avec un taux de bonnes prédictions de 0,62. Le modèle n'est pas très fiable puisqu'on peut apercevoir qu'avec un prédicteur aléatoire qui ne prédit que des 6, on obtient un score de 45 % de bonne réponse.

Le meilleur modèle que l'on a évalué a donc un pourcentage de bonne prédiction 17% supérieur à ce prédicteur aléatoire qui ne prédit que des 6, ce qui n'est pas très satisfaisant

On veut observer de manière plus précise quel type d'erreur commet notre modèle clf_rdmf.

Visualisation des erreurs

On utilise la matrice de confusion pour visualiser quel type d'erreur commet notre classifieur clf_rdmf qui est un classifieur de type forêt aléatoire. On consulte donc la matrice de confusion :

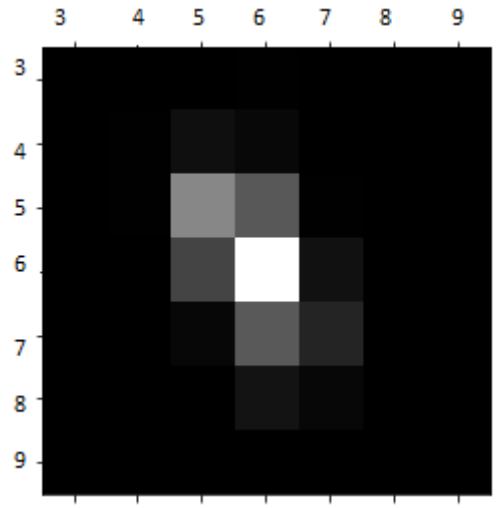

Figure5.Matrice de confusion

On remarque que la diagonale de notre matrice de confusion est sombre pour les classes 3,4,7 ,8,9 ce qui signifie que nos résultats ne sont pas satisfaisants et que notre prédicteur ne fait jamais les bonnes prédictions pour les classes 3,4,7,8 et 9 en revanche le carré blanc au milieu de la matrice de confusion nous indique que l'on prédit relativement bien les 6

On divise chaque valeur de la matrice de confusion par le nombre d'instances dans la classe correspondante, afin de pouvoir comparer des taux d'erreur et non le nombre d'erreurs.

On obtient la matrice suivante :

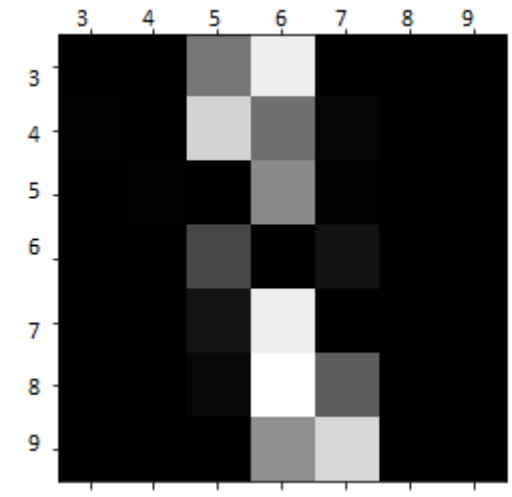

Figure 6 .Matrice de confusion pondérée

On remarque que les vins ayant des notes supérieur à 6 appartenant aux classes 7,8,9 ont tendance à se voir attribuer la classe 6 par notre classifieur, de même que ceux appartenant à la classe 3 en revanche les vins de la classe 4 sont souvent assimilés à des vins de la classe 5 par notre prédicteur.

On en déduit que les caractéristiques exceptionnelles de certains vins ne sont pas pris en compte et que la prédominance des classes 5 et 6 influe sur l'entrainement de nos algorithme de classification et que la classification multi-classes s'avère infructueuse pour bien modéliser notre jeu de données,

une classification plus judicieuse mais moins ambitieuse serait de classifier les vins en 2 catégorie les vins exceptionnellement bon (tous les vins avec une note >6) et les vins moyens-mauvais (vins avec une note <7).

2) classification binaire

Restructuration des données

On a observé qu'il y avait une prédominance des classes 5 et 6 dans notre jeu de données, ces classes présentent peu d'intérêts pour notre modélisation, car il ne s'agit que de vins de moyenne qualité, or l'intérêt de notre modélisation serait de détecter les vins rares de bonnes qualité.

On va donc restructurer nos données en 2 classes : les vins rares (avec une note >6) et les vins standard ou mauvais (avec une note <7)

On va entrainer différents classifieurs parmi lesquels un SVM à noyau gaussien et une forêts aléatoire (modèle qui a le mieux fonctionné dans la classification multi-classes).

Choix d'une famille de modèle

Ici on va utiliser un nouvel indice de performance qui est l'aire sous la courbe ROC , comme nos données sont distribuées de manière très déséquilibré , les vins rares étant nettement moins fréquent que les vins ordinaires (qualité 5 ou 6) , un score de précision (accuracy) n'aurait pas un grand intérêt puisqu'on pourrait facilement avoir un bon taux de prédiction en ne prédisant que des vins ordinaire.

Maintenant que nous sommes dans une configuration binaire , nous pouvons utiliser l'AUC de la courbe ROC en rappelant que la courbe ROC représente de taux de Vrai Positif (vins rares bien classés par le prédicteur) en fonction du taux de Faux Positif (vin classé à tort parmi les vins rares alors qu'il est ordinaire).

Plus l'AUC est élevée, plus notre prédicteur est performant.

Voici les résultats que l'on obtient pour nos 2 classifieurs :

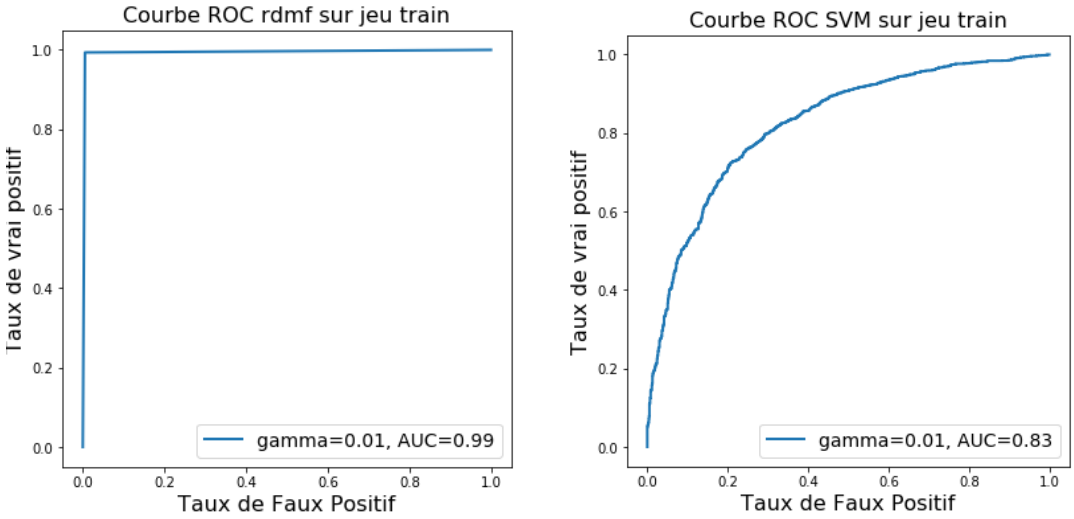

Figure 7.Comparaison courbe ROC entrainement

On constate que la forêt aléatoire Clf_rdmf a une AUC=1 , ce qui signifie que aucun vins n'est mal classés. Bien sûr, on comprend aisément que le classifieur est surentrainé (overfitting). Il a appris par cœur le jeu d'entrainement d'où le fait qu'il ne se trompe jamais sur celui-ci.

On se permet donc d'utiliser le jeu de test afin de révéler l'imposture et effectivement les résultats ne sont plus vraiment en faveur de clf_rdmf :

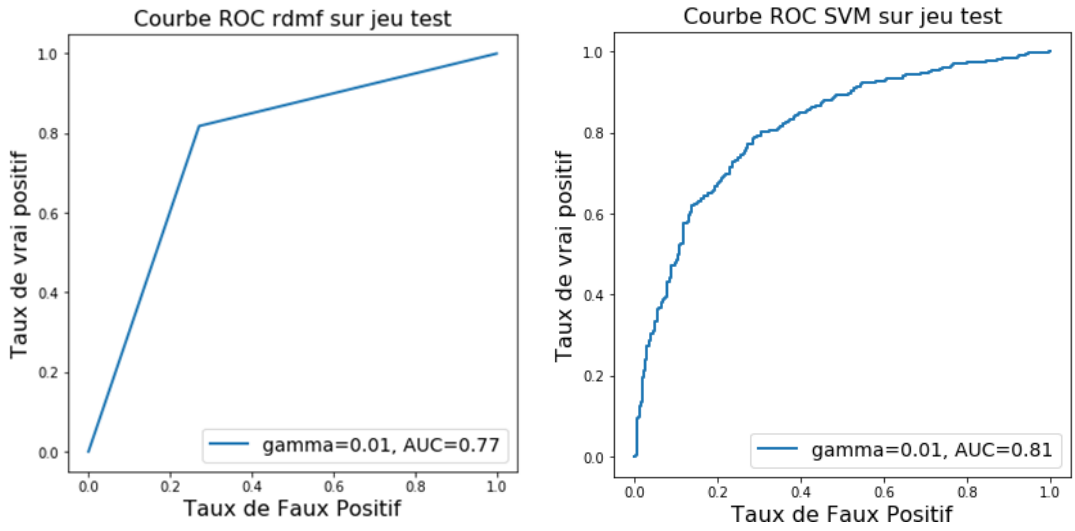

Figure 8.Comparaison courbe ROC test

AUC est passée à 0,77 pour notre classifieur rdmf alors que notre SVM à noyau gaussien garde lui une AUC élevée y compris sur des données qu'il ne connait pas (AUC=0,81).

On va donc privilégié le classifieur SVM à noyau pour la suite de notre étude.

On va maintenant régler les hyperparamètres afin d'obtenir le modèle le plus optimisé possible.

Choix des hyperparamètres

On va s'intéresser aux 2 principaux hyperparamètres des SVM à noyau gaussien :

« C » , qui est le paramètre de régularisation qui influe sur les empiètements de marge , c'est-à-dire que pour une grande valeur de C, une marge plus étroite est acceptée si la fonction de décision classifie le mieux possible nos observations.

Le paramètre « gamma » définit la portée de l'influence d'un seul exemple d'entraînement, les valeurs faibles de gamma signifiant «loin» et les valeurs élevées signifiant «proche».

La méthode la plus courante pour sélectionner avec précision nos hyperparamètres est la recherche en grille ou GridSearch c'est-à-dire qu'on définit pour chaque hyperparamètre un intervalle et un pas de variation, on obtient ainsi une grille (de dimension m) dont les nœuds correspondent aux valeurs à tester. Pour chaque nœud de la grille on appliquera une validation croisée afin d'obtenir une estimation de l'erreur de généralisation des modèles correspondants. Et on sélectionnera le meilleur modèle afin de l'entrainer sur le jeu d'entrainement total.

On trouve que le meilleur modèle a pour l'hyperparamètre gamma=1 et C=1 avec une AUC=0 ,85

On est donc plus performant de 4% par rapport au SVM par défaut avec gamma =0,01 , ce qui est une mesure très satisfaisante.

Test final de notre modèle

Il ne nous reste plus qu'à tester notre nouveau modèle en utilisant le jeu de test , qui n'a pas été utilisé ni pour l'entrainement ni pour la validation de notre modèle.

On espère pouvoir observer une AUC proche de celle obtenue avec la validation croisée cela signifierait que notre modèle généralise bien et s'avère efficace pour prédire la rareté d'un vin.

On obtient AUC_final= 0,83 , ce qui est proche des 0,85 qu'on avait obtenu avec la validation croisée .

On conclut donc que notre modèle généralise plutôt bien et peut servir de prédicteur à vins rares et excellents.

IV. CONCLUSION

On a donc bien réussi à bien modéliser notre jeu de données, il a néanmoins fallu faire différents choix :

On a restructuré nos données car les classes prédominantes empêchaient une vision multi-classes de notre modèle, or dans le cas de notre étude, ce sont ces vins-là qui nous intéressaient le plus.
Suite à cette restructuration, on a effectué une classification binaire à partir de différentes familles de modèle . Les SVM à noyau gaussien se sont avéré les plus efficaces pour détecter les meilleurs vins.On a ensuite affiné notre modèle pour avoir la meilleure performance possible à l'aide d'un GridSearch.

La classification multi-classes s'est avérée inefficace à cause de la dominance des notes 5 et 6 nous empêchant ainsi de détecter les meilleurs vins

Enfin la visualisation des données nous a permis d'observer qu'une régression était impossible pour notre jeu de données.

Source et liens internet :

https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning

https://cedric.cnam.fr/vertigo/Cours/ml2/preambule.html

Aurélien Géron , *Machine learning avec scikit-learn*