Prédiction de la qualité du vin

Romane PERSCH

16 décembre 2014

Sommaire

Chapitre 1

Introduction

1.1 Présentation de la base de données

La base de données utilisée ici provient du site de l'UCI Machine Learning Repository. Initialement, 2 bases de données sont fournies : l'une contenant les variantes de vin rouge du vin portugais "Vinho Verde" (1599 individus), l'autre contenant ses variantes de vins blancs (4898 individus) [1]. **J'ai choisi de me concentrer uniquement sur la base contenant les vins blancs, espérant obtenir de meilleurs résultats avec plus d'invidus.**

Elle fournit **11 caractéristiques chimiques** pour chaque individu et une **note évaluant la qualité du vin allant de 1 (très mauvais) à 10 (excellent)**. Toutes les variables fournies sont quantitatives exceptée celle indiquant la qualité qui est qualitative ordinale. Aucune des variables disponibles n'indique le prix de la bouteille de vin, l'année, le cépage etc. Ceci va donc limiter les objectifs de la prédiction et conditionner les méthodes utilisées.

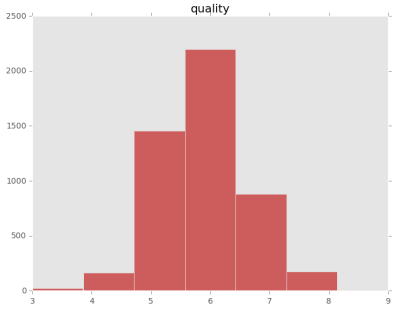

FIGURE 1.1 – Distribution des qualités dans la base de données vins blancs

Les qualités ne sont pas distribuées uniformément dans la base (Figure 1.1) : elle contient peu de vins de qualités "extrêmes", c'est-à-dire excellents ou très

1. Pour lire les informations fournies par l'UCI Machine Learning Repository sur la base de données, voir l'appendix

mauvais, et beaucoup de vins de qualités moyennes. Ceci est problématique pour la prédiction de l'appartenance à une classe de qualité, puisque certaines classes seront fortement sous-représentées, mais c'est en accord avec la réalité de la production de vin et ainsi les problèmes qui se poseraient avec des "vraies" données plus globales (par exemple, l'ensemble des vins blancs produits en France).

N.B : La base ne contenant aucune valeur manquante, il n'y a aucun travail préparatoire à effectuer.

1.2 Objectif(s)

L'objectif est de prédire la qualité du vin blanc à partir de ses caractéristiques chimiques. Néanmoins, comme ceci peut donner lieu à plusieurs interprétations, je vais essayer de répondre à un seul type de problème.

A noter que comme aucune variable prix n'est disponible, il n'est pas possible de s'intéresser à un problème de type prédiction du rapport qualité/prix du vin (bon ou mauvais), ce qui aurait peut-être été plus intéressant en pratique (application mobile ?).

Problème traité : Régression en considérant la variable "quality" (nombre entier entre 1 et 10) comme une variable quantitative. Il s'agit du problème le plus intéressant car donnant les résultats sur la qualité du vin les plus précis.

D'autres problèmes pourraient être traités :
- Classification en 2 classes : indiquer si le vin est mauvais (note 3 ou 4) ou non, de façon à détecter les mauvais vins
- Classification en 3 classes : indiquer si le vin est mauvais, moyen ou bon
- Sélection de caractéristiques : sélectionner un nombre restreint de variables pouvant permettre un jugement rapide sur la qualité du vin sans le goûter

La note attribuée à un vin étant un critère subjectif, on peut s'attendre à obtenir des performances relativement faibles, d'autant plus que peu de caractéristiques sont disponibles.

Dans tous les cas, même avec de très bonnes prédictions, l'utilisation en pratique des modèles obtenus reste limitée. Peu de productions viticoles sont suffisamment grandes pour être intéressées par une prédiction automatique de la qualité du vin sans avoir à les goûter. La seule utilité pratique pourrait éventuellement être une application mobile pour les consommateurs qui ne savent pas évaluer la qualité du vin en fonction de caractéristiques telles que le nom, l'année etc, mais le prix est déjà un indicateur très fort de qualité. [2]

2. Plus utile pour les producteurs de vin et pour les acheteurs serait la prédiction du prix du vin en fonction de ses caractéristiques chimiques, de l'année, du cépage etc. Ceci a été étudié par Orley Ashenfelter : http ://www.data-business.fr/statistiques-prediction-qualite-prix-vin-parker-ashenfelter/

1.3 Découpage en une base d'apprentissage et une base de test

La base de données est découpée en une **base d'apprentissage** (75% de la base initiale) et une **base de test** (25% de la base initiale). La base d'apprentissage est la base surlaquelle nous allons apprendre les modèles développés dans la partie 1. La base de test est la base sur laquelle nous allons appliquer les modèles de façon à prédire une qualité pour chaque vin. Ainsi, nous allons pouvoir tester la performance de chaque modèle en comparant les prédictions aux cibles (les qualités réelles).

Tous les modèles de régression développés ci-dessous sont appris et testés sur les mêmes bases, de façon à comparer leurs performances sur les **mêmes** individus.

La base de données est ainsi découpée en 4 parties :

— Une matrice X_train indiquant les caractéristiques chimiques des invididus séléctionnés dans la base d'apprentissage

— Un vecteur cible y_train indiquant la qualité réelle des vins de la base d'apprentissage

— Une matrice X_test indiquant les caractéristiques chimiques des vins sélectionnés dans la base de test

— Un vecteur cible y_test indiquant la qualité réelle des vins de la base de test

Ceci est effectué grâce à la commande train_test_split de scikit-learn qui effectue un tirage **au hasard** des individus de la base de test. Un random_state a été ajouté de façon à ce qu'on obtienne toujours le même découpage à chaque exécution du programme : ainsi les résultats présentés ci-dessous seront les mêmes que ceux obtenus si le programme est réexécuté.

Chapitre 2

Régression linéaire simple

2.1 Méthode : Régression linéaire

L'objectif est de prédire la note indiquant la qualité du vin (cible $t \in \mathbb{R}$) à partir du vecteur indiquant les caractéristiques chimiques du vin $x \in \mathbb{R}^{11}$. Le premier modèle que j'ai choisi de tester est un modèle de régression linéaire simple où la prédiction y s'écrit comme fonction linéaire de x avec $w = (w_1, \ldots, w_{11}) \in \mathbb{R}^{11}$ le vecteur des coefficients et b l'intercept :

$$y(x, w) = w^T x + b$$

Le vecteur $(b^\star, w^\star)$ des coefficients est obtenu en minimisant l'erreur quadratique moyenne :

$$w^\star = \text{Argmin} \sum_{i=1}^{n} (t_i - w^T x_i - b)^2$$

où n est le nombre de vins de la base d'apprentissage.

Ce modèle repose sur l'hypothèse selon laquelle les différents échantillons (x_i, t_i) sont indépendants et identiquement distribués. Dans le cas présent, ceci peut être supposé de façon raisonnable.

Ce problème peut être résolu de façon analytique si $x^T x$ est inversible, autrement dit si l'hypothèse de non mutlicolinérité est vérifiée, ce qui est le cas ici puisqu'aucune variable n'est constante et qu'il n'y a aucune relation parfaitement linéaire entre les variables. [1] . Il n'y a donc pas de problème de convergence d'algorithme comme dans le cas de la régression logistique (qui utilise un algorithme de descente du gradient pour trouver la solution).

1. Si certaines variables sont néanmoins très fortement corrélées (sans être linéairement dépendantes), il en résulte une très grande variance des coefficients estimés, ce qui dégrade la qualité des prédictions. Or, dans cette base de données de nombreuses variables indiquent l'acidité du vin blanc (fixed acidity, volatile acidity, citric acid, pH). La variable pH est a priori susceptible d'être "quasiment" une combinaison linéaire des autres variables indiquant l'acidité. J'ai donc choisi ensuite de réapprendre le modèle sans tenir compte de la variable pH. Les performances ne semblaient néanmoins à première vue pas significativement différentes, on observait même une légère dégradation du taux de bonnes réponses. Je n'ai donc pas poursuivi mes recherches dans cette direction. De plus, comme expliqué par la suite, utiliser une régression Ridge permet de réduire ce problème de variance.

2.2 Recherche de bons indicateurs d'erreur et résultats

Analyse du R^2 Le R^2 obtenu, c'est-à-dire la fraction de la variation de t expliquée par le modèle, est très faible pour une prédiction : [2]

$$R^2 = 0.2821$$

Cela indique qu'il n'y a pas suffisamment d'information disponible dans la base de données pour effectuer une très bonne prédiction. A priori, seule une amélioration modérée des performances sera possible, même en changeant de modèle.

Par la suite, le modèle ainsi créé est appliqué à la base de test et les prédictions sont comparées aux qualités réelles.

Erreur quadratique totale [3] L'erreur quadratique totale est l'erreur qui est minimisée lors d'une régression linéaire. Ce n'est pas toujours cette erreur qui est retenue dans la minimisation, comme cela sera le cas lors d'une régression à vecteurs de support.

$$Erreur\_quadratique = 701.2$$

Erreur absolue totale et moyenne

$$Erreur\_absolue\_totale = 719.4$$

$$Erreur\_absolue\_moyenne = 0.587$$

L'erreur moyenne obtenue est très largement inférieure à 1, ce qui indique une bonne performance **globale**. En effet, une prédiction ayant un écart inférieur à 1 par rapport à la qualité réelle est une prédiction très précise.

Néanmoins, ces indicateurs valorisent des erreurs qui ne sont pas importantes en pratique. Ainsi, une prédiction de 7.2 au lieu de 7 n'est dans la réalité pas une erreur puisque les notes sont des entiers. Cependant,les 0.2 viennent augmenter ces mesures d'erreur globales. D'autres critères doivent donc être utilisés pour comparer les résultats entre les différents modèles.

Taux de bonnes réponses Une prédiction **exacte** dans la réalité est une prédiction dont l'arrondi à l'entier le plus proche est égal à la note réelle, les notes étant nécessairement des entiers. [4] Ce taux de bonnes réponses indique la proportion de prédictions exactes selon cette définition :

$$Taux = 51.79\%$$

2. Une bonne prédiction à l'aide d'une régression linéaire devrait avoir un R^2 proche de 1.

3. Je ne calcule par l'erreur moyenne car cela est inutile pour la comparaison par rapport aux modèles suivants : la même base de test est utilisée donc il y aura le même nombre d'individus

4. On pourrait être confronté à des prédictions inférieures à 0 ou supérieures à 10. Il faudrait à ce moment là donner la note 0 pour les prédictions négatives et donner la note 10 pour les prédictions supérieures à 10. Néanmoins, comme cela sera présenté par la suite, cela n'arrive jamais.

Il est relativement faible alors qu'on avait obtenu une erreur moyenne très faible. Ceci indique probablement que certains vins sont très mal prédits et d'autres en grande proportion très bien prédits.

L'inconvénient du taux de bonnes réponses est qu'il pondère de façon égale les très grandes erreurs (un vin de note réelle 3 prédit 7 par exemple) et les erreurs moins graves (un vin de note réelle 5 prédit 6.5 par exemple). J'ai donc choisi définir une mesure de l'erreur globale qui accorde des poids proportionnels aux écarts uniquement lorsque ceux-ci dépassent un certain seuil.

Erreur absolue totale sans prendre en compte les écarts inférieurs à 0.5
Cette erreur est définie par :

$$ErreurA = \sum_{i=1}^{m} |t_i - y_i| \mathbb{1}_{(|t_i - y_i| \geq 0.5)}$$

où m est le nombre de vins de la base de test, t la cible et y la prédiction selon le modèle appris.
Si l'écart en valeur absolue est inférieur à 0.5, alors l'arrondi à l'entier le plus proche de la prédiction est toujours égal à la qualité réelle, et réciproquement. Ainsi, toutes les prédictions exactes ont ici une erreur nulle et toutes les prédictions non exactes une erreur strictement positive proportionnelle à l'écart.
On obtient ici :

$$ErreurA = 557.7$$

Erreur absolue totale sans prendre en compte les écarts inférieurs à 1
Cette erreur est définie par :

$$ErreurB = \sum_{i=1}^{m} |t_i - y_i| \mathbb{1}_{(|t_i - y_i| \geq 1)}$$

où m est le nombre de vins de la base de test, t la cible et y la prédiction selon le modèle appris.
Cette mesure est retenue en considérant que faire une prédiction non exacte mais très proche de la note réelle n'est pas très grave. Par exemple, prédire 7.2 pour un vin noté 8 n'est pas très grave, donc ce n'est pas comptabilisé en tant qu'erreur alors que prédire un vin 3.4 alors que sa note est 8 est grave, ceci de façon proportionnelle à l'écart entre les deux notes.
On obtient ici :

$$ErreurB = 282.5$$

2.3 Analyse des erreurs effectuées

Comparaison des prédictions aux qualités réelles
En observant la figure 2.1 [5], on constate que globalement plus les qualités réelles sont élevées, plus les prédictions sont élevées, ce qui est rassurant. Cependant, la variance des prédictions autour de la qualité réelle est élevée, ce qui indique

5. Les prédictions ont été arrondies à 0.1 sur le graphique pour plus de précision au niveau des couleurs, et du bruit a été introduit autour de la qualité réelle pour éviter des empilements de points et mieux voir les couleurs. Cela ne change pas l'interprétation du graphique.

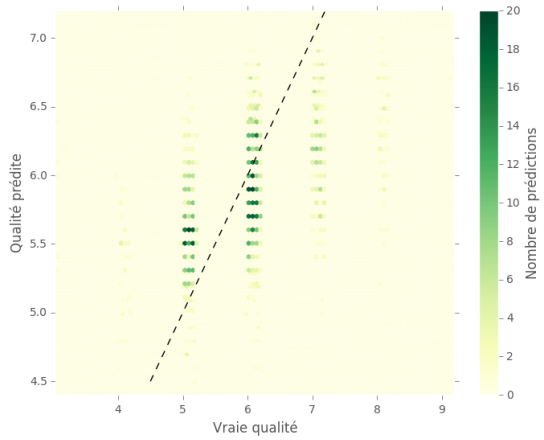

FIGURE 2.1 – Comparaison des prédictions aux qualités réelles

que les prédictions ne sont pas très cohérentes avec la qualité réelle. Des vins de la même qualité ne sont pas toujours prédits comme de qualité similaire. De plus, les très bons vins (notés 8 et 9) ne sont jamais prédits au-dessus de 7 et les très mauvais vins (notés 3 et 4) sont rarement prédits en dessous de 5.

Quelles sont les qualités mal prédites ?

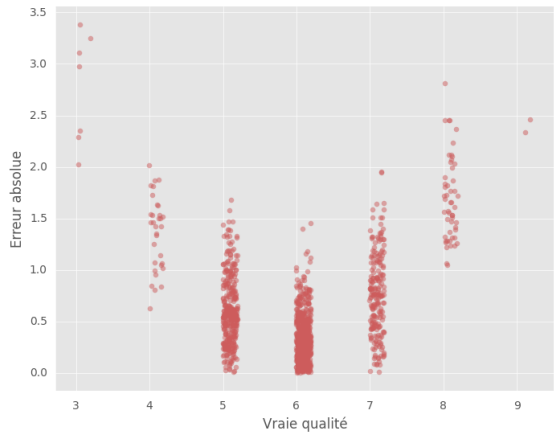

FIGURE 2.2 – Identification des qualités les plus mal prédites

La figure 2.2 confirme l'interprétation de la figure 2.1 : les qualités extrêmes sont très mal prédites par ce modèle alors que les qualités moyennes, les plus fréquentes, sont relativement bien prédites.

Ceci s'explique probablement en partie par le fait que les qualités extrêmes sont sous-représentées (Figure 1.1) : le peu d'individus dans ce cas ne permet

pas d'influencer suffisamment la minimisation de l'erreur quadratique totale. Les estimateurs des coefficients s'ajustent donc de façon précise à la majorité des vins : les vins moyens. Néanmoins, lorsqu'on enlève environ deux tiers des individus ayant une qualité moyenne, et qu'on refait une régression, on n'obtient aucune amélioration de la prédiction des qualités extrêmes. [6]

Une autre explication pourrait ainsi être que les vins très bien et très mal notés ne sont pas énormément différents chimiquement des vins de qualité moyenne. Ils sont peut-être seulement légèrement différents et c'est cette légère différence, rare, qui incite subjectivement à leur donner une très bonne note ou une très mauvaise note ou bien leur différence légère sur ces variables s'accompagne de grandes différences sur d'autres variables dont nous ne disposons pas. Cette hypothèse n'est pas absurde dans la mesure où le modèle prédit bien des qualités globalement inférieures pour les vins les plus mauvais et des qualités globalement meilleures pour les vins les meilleurs d'après la figure 2.1.

Quelles sont les caractéristiques chimiques des vins mal prédits ?
L'idée est ici de voir si les vins ayant certaines caractéristiques chimiques sont particulièrement mieux ou moins bien prédits que les autres. Ceci est analysé dans le but éventuel de créer plusieurs modèles complémentaires appliqués selon les caractéristiques chimiques du vin blanc dont la qualité est à prédire. [7] Néanmoins, comme l'indique la figure 2.3, il ne semble pas y avoir de caractéristiques liées à des prédictions particulièrement mauvaises (les graphes des autres variables sont similaires).

Si parfois on observe un pic sur le graphique qui croise la caractéristique chimique et l'erreur absolue, comme c'est le cas pour la variable sulphates sur la figure 2.3, cela ne signifie pas nécessairement que les vins ayant des valeurs proches de l'abscisse de ce pic sont particulièrement mal prédits. En effet, ce sont souvent des valeurs qu'une très grande partie des vins prend. Ainsi, en ce sens, les vins particulièrement mal prédits ont "plus de chances" de prendre également cette valeur en la variable considérée. Cette hypothèse est confirmée par les graphiques qui croisent chaque vin de la base de test avec leur erreur absolue et qui font apparaître les points de différentes couleurs selon la valeur qu'ils prennent en la variable indiquant la caractéristique chimique qui nous intéresse. On voit que les vins ayant une quantité de sulphates faibles sont tout simplement en très grande majorité dans la base et il ne semble pas y avoir de prédictions particlièrement plus mauvaises pour les vins ayant une quantité de sulphates faible.

On ne peut donc rien conclure de cette analyse.

6. Résultats non présentés ici mais observables en exécutant le code
7. ce qui ébauché fait à la fin de ce rapport avec 2 modèles de régression à vecteurs de support selon le taux d'alcool

FIGURE 2.3 – Analyse de l'erreur absolue en fonction des caractéristiques chimiques - en bleu/vert les valeurs faibles et en rouge/jaune les valeurs élevées

Chapitre 3

Régression Ridge polynomiale de degré 2

3.1 Méthode : Régression linéaire régularisée ou régression Ridge

Le modèle testé ici est une variante de la régression linéaire classique présentée ci-dessus : un terme de régularisation est introduit dans le problème de minimisation.

Le problème à résoudre est ici :

$$\text{Min} \sum_{i=1}^{n} (t_i - w^T x_i - b)^2 + \alpha \|w\|_2^2$$

où n est le nombre de vins de la base d'apprentissage.

L'hyperparamètre α du modèle est est choisi arbitrairement : plus il est élevé, plus la norme de w est pénalisée.

L'introduction d'un terme de régularisation, c'est-à-dire d'une contrainte sur la norme du vecteur w, a plusieurs bénéfices dans le cas d'une régression Ridge [1] :

1. Cela permet de même d'éviter le sur-apprentissage, c'est-à-dire l'apprentissage "par coeur" (cf Figure 3.1).

   Cette contrainte sur la norme de w permet ainsi dans l'exemple de la régression polynomiale où $x \in \mathbb{R}$ d'atténuer les oscillations du modèle appris.

2. Cela permet de faire face à d'éventuels problèmes de forte corrélation entre les variables en réduisant la variance des estimateurs des coefficients (au prix de l'ajout d'un biais).

1. Ce terme peut aussi avoir pour objectif d'accélérer la convergence du modèle lorsque le problème de minimisation ne peut pas être résolu analytiquement et qu'il est résolu numériquement à l'aide de l'algorithme de descente du gradient, comme dans le cas de la régression logistique.

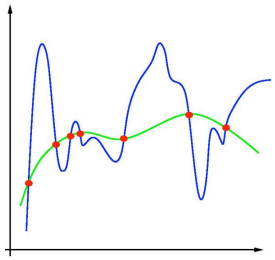

FIGURE 3.1 – Exemple de sur-apprentissage lors d'une régression polynomiale : en bleu le modèle appris et en vert le modèle qui a généré les données.

3.2 Application dans le cas polynomial de degré 2

Comme expliqué ci-dessus, la régularisation dans une régression Ridge a moins d'intérêt lorsqu'elle est appliquée à l'ensemble des variables à la puissance 1 que si elle est appliquée à ces variables élevées à différentes puissances ou transformées par une fonction de base, puisqu'elle se rapprocherait alors beaucoup d'une régression linéaire classique. [2] J'ai donc choisi d'effectuer une régression polynomiale régularisée. J'ai choisi le degré 2 car en passant au degré 3, il semblait y avoir une forte dégradation des performances malgré la régularisation. Pour ce faire j'ai dupliqué la base de données, j'ai elevé à la puissance 2 la copie et j'ai concaténé la base initiale et la base transformée.

3.3 Méthode : Leave-One-Out Cross-Validation pour choisir l'hyperparamètre α

Pour effectuer une régression Ridge et pour choisir en même temps l'hyperparamètre α donnant le moins d'erreur, j'ai utilisé la commande RidgeCV de scikit-learn qui procède à une Leave-One-Out Cross-Validation entre les différents hyperparamètres que l'on veut tester. [3]

La Leave-One-Out Cross-Validation est un cas particulier de la K-fold Cross-Validation où $K = n$, le nombre d'individus de la base de données.

Le principe d'une K-Fold Cross-Validation est de diviser la base d'apprentissage en K portions, et d'entraîner le modèle considéré en utilisant tour à tour chaque portion comme base de validation et l'union des autres portions comme base d'entraînement(Figure 3.2).

L'erreur du modèle est calculée à chaque tour pour l'ensemble des hyperparamètres à tester. Ainsi, pour chaque hyperparamètre, la moyenne de l'erreur sur les K tours est calculée, et l'hyperparamètre ayant la moyenne la plus basse est finalement sélectionné.

2. Sauf dans le cas d'une base de données très particulière, avec une très forte corrélation entre les variables ou un très grand nombre de variables par rapport au nombre d'individus

3. En réalité, il s'agit non pas d'une Leave-One-Out Cross Validation classique mais d'une forme efficiente qui nécessite moins d'opérations : la Generalized Cross-Validation. Néanmoins, comme le principe est le même et que je réutilise des K-Fold Cross-Validation par la suite, j'ai choisi d'expliquer le principe initial.

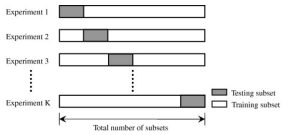

FIGURE 3.2 – 4-Fold Cross-Validation : la base est divisée en 4 portions, et chacune d'elle va être utilisée tour à tour comme base de validation

Ce type de validation croisée permet ainsi de faire de la sélection de modèle à partir uniquement de la base d'apprentissage. En effet, l'objectif est de sélectionner l'hyperparamètre donnant le moins d'erreur possible en général et non précisément sur la base de test choisie, puisqu'il s'agit à terme de faire de la prédiction sur de nouvelles données. La validation croisée sert donc à éviter un éventuel sur-apprentissage sur la base de test choisie. Cependant, son gros inconvénient est sa lenteur voire son inexécutabilité sur de grandes bases de données, puisqu'il s'agit d'un algorithme brute force. Cela ne pose néanmoins pas de problème ici puisque la base est relativement petite.

Résultats de la validation croisée :
Les hyperparamètres que j'ai testés sont : 0.01, 0.1, 0.2 et 0.3. [4] J'obtiens alors 0.1 comme meileur hyperparamètre.

Par la suite, les résultats présentés sont donc obtenus avec $\alpha = 0.1$.

3.4 Résultats

Erreur quadratique totale L'erreur quadratique totale est l'erreur qui est ici minimisée lors de l'apprentissage, comme avec une régression linéaire simple.

$$Erreur\_quadratique = 690.5$$

On observe une amélioration par rapport à la régression linéaire simple (différence d'environ 10).

Erreur absolue totale et moyenne

$$Erreur\_absolue\_totale = 713.2$$

$$Erreur\_absolue\_moyenne = 0.582$$

On observe également une légère amélioration de l'erreur absolue.

Taux de bonnes réponses

$$Taux = 52.45\%$$

Le taux de bonne réponse s'améliore également d'environ 0.7%.

4. D'autres essais ont montré que les hyperparamètres supérieurs donnaient de mauvais résultats

Erreur absolue totale sans prendre en compte les écarts inférieurs à 0.5

$$ErreurA = 549.2$$

Sans prendre en compte les écarts inférieurs à 0.5, l'erreur est également plus basse. Ceci indique que ce nouveau modèle n'a pas seulement réduit les erreurs négligeables en très grand nombre, il a également réduit les erreurs "importantes".

Erreur absolue totale sans prendre en compte les écarts inférieurs à 1

$$ErreurB = 272.7$$

La baisse de ce type d'erreur confirme l'interprétation ci-dessus : il y a une réelle réduction des erreurs "importantes", bien que cette réduction reste légère.

En conclusion, selon tous les indicateurs, ce nouveau modèle semble légèrement plus performant sur la base de test. Comme cela est conforté par tous les indicateurs, on peut raisonnablement supposer qu'il y a une réelle amélioration des performances du modèle. Avec plusieurs essais (différentes bases de test en enlevant le paramètre random_state de la commande train_test_split), le taux de bonnes réponses est également toujours très légèrement supérieur. [5] Néanmoins, cette amélioration est vraiment faible et invite à rechercher un autre modèle.

3.5 Comparaison des erreurs à celle obtenues avc le modèle de régression linéaire simple

Comparaison des prédictions aux qualités réelles
En observant la figure 3.3 et en la comparant à la figure 2.1, on remarque que la variance des prédictions autour de chaque qualité réelle est moins élevée, surtout pour les qualités moyennes. Ceci confirme une plus grande précision du modèle : des vins de même qualité réelle sont plus souvent prédits au même niveau. Les vins de qualité 6 semblent ainsi particulièrement bien prédits : il y a une très forte concentration de prédictions autour de 6. Les vins de qualité 5 sont très souvent prédits autour de 5.5.

Quelles sont les qualités mal prédites ? Comparaison avec le modèle précédent.
On remarque sur la figure 3.4 que les vins de qualités "extrêmes", c'est-à-dire très mauvais ou très bon, sont légèrement mieux prédits avec la régression Ridge polynomiale puisque l'erreur est légèrement moins grande. Néanmoins, ils restent particulièrement mal prédits comparé au reste des vins.

Quelles sont les caractéristiques chimiques des vins mal prédits ?
Les prédictions étant très similaires, on obtient quasiment les mêmes graphiques

5. Idéalement, pour conclure à une amélioration avec la régression Ridge polynomiale il faudrait effectuer une validation croisée qui compare les 2 modèles mais comme les prédictions sont vraiment similaires, j'ai plutôt choisi de commencer par essayer des modèles très différents comme la régression à vecteurs de support.

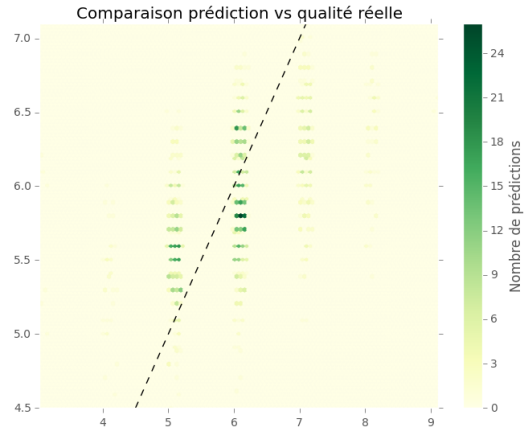

FIGURE 3.3 – Comparaison des prédictions aux qualités réelles

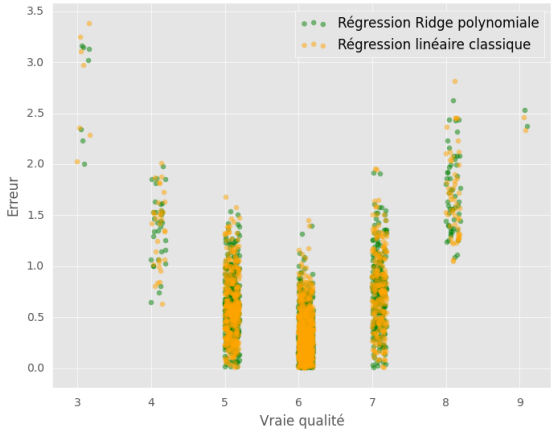

FIGURE 3.4 – Identification des qualités les pus mal prédites - Comparaison des 2 modèles

qu'avec le modèle de régression linéaire classique à la figure 2.3. De plus, en comparant les qualités réelles aux qualités prédites (Figure 3.5), on remarque que les vins à taux d'alcool élevé sont toujours prédits comme étant meilleurs, ce qui parfois n'est par pertinent comme le montrent sur le graphe de droite les vins de qualité réelle 6 et 7.

Ceci nous invite à essayer d'appliquer deux modèles différents selon le taux d'alcool. Mais avant cela, il est peut-être possible d'améliorer la performance globale avec un seul modèle plus adéquat : la régression à vecteurs de support.

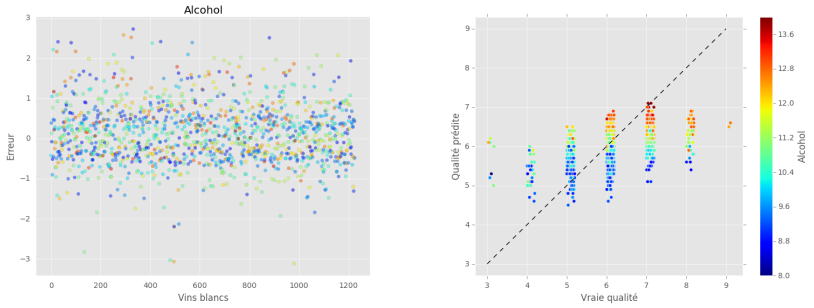

FIGURE 3.5 – Prédictions des vins selon le taux d'alcool par le modèle Ridge

Chapitre 4

Régression à vecteurs de support

4.1 Méthode : Régression à vecteurs de support

Dans le cas présent, les erreurs de régression inférieures en valeur absolue à un certain seuil sont peu importantes en pratique, car l'interprétation est la même en termes de qualité. Ce qui importe est le fait que la différence entre la qualité réelle et la prédiction ne soit pas trop grande. Ceci est également l'idée sous-jacente de la régression à vecteurs de support : il s'agit d'ignorer les différences inférieures à un certain seuil ϵ dans la régression. De même que dans la régression linéaire et la régression Ridge, il s'agit d'un modèle linéaire appliqué soit directement sur les données $x \in \mathbb{R}^{11}$ ou bien sur une transformation des données par une fonction de base $\phi(x) \in \mathbb{R}^{11}$:

$$y(x, w) = w^T \phi(x) + b$$

Ici encore, notamment lorsqu'une transformation non-linéaire des données est effectuée, on souhaite que les coefficients w soient le plus petits possible. Le problème devient alors :

$$Min \frac{1}{2} \|w\|^2$$

$$s.c : \forall i |t_i - w^T \phi(x_i) - b| \leq \epsilon$$

Néanmoins, une solution w^* n'existe pas nécessairement pour un tel ϵ choisi. Certaines erreurs sont donc autorisées mais pénalisées. Le problème devient donc :

$$Min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i$$

$$s.c : \forall i \ |t_i - w^T \phi(x_i) - b| \leq \epsilon + \xi i$$

$$\forall i \ \xi i \geq 0$$

où la pénalisation ξ_i vaut $max(0, |t_i - w^T \phi(x_i) - b| - \epsilon)$. Plus C est grand, plus la pénalisation est forte.

4.2 Application dans le cas d'un noyau gaussien

Après quelques essais, j'ai choisi d'utiliser une régression à vecteurs de support à noyau gaussien car elle semblait donner de bons résulats. Cela signifie que la fonction de base ϕ choisie est la densité d'une fonction gaussienne (on utilise la moyenne et la variance empirique de chaque variable). La commande SVR dans scikit-learn permet de mettre en oeuvre directement une régression à vecteurs de support à noyau gaussien en choisissant l'option 'kernel'=rbf.

4.3 5-Fold Cross Validation (Grid Search) pour obtenir les hyperparamètres C et ϵ selon le critère de calcul de l'erreur par défaut

Avant de réaliser une régression à vecteurs de support, il faut choisir arbitrairement 2 hyperparamètres : C et ϵ. Intuitivement, prendre $\epsilon = 0.5$ permettrait d'effectuer une régression qui minimise uniquement les erreurs supérieures à 0.5, ce qui nous intéresse dans ce problème. Néanmoins, on remarque que pour le même C, 0.5 semble donner de moins bons résultats en général que le paramètre par défaut 0.1. [1] Il est donc intéressant ici d'effecuer des tests de performance avec différentes valeurs de C mais aussi de ϵ.
La commande GridSearchCV de scikit-learn permet alors d'effectuer automatiquement une validation croisée de type K-Fold qui parcourt exhaustivement toutes les combinaisons possibles de paramètres pris dans des ensembles définis arbitrairement. Par défaut, elle compare le R^2 obtenu pour chaque régression sur chaque base d'apprentissage. J'ai choisi ici d'effectuer une 5-Fold Cross Validation et de comparer des régressions à vecteurs de support à noyau gaussien pour les hyperparamètres $C \in \{0.5, 1, 2, 5, 10, 50, 100\}$ et $\epsilon \in \{0.1, 0.2, 0.3, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.5\}$.
On obtient comme meilleurs hyperparamètres : $C = 1$ et $\epsilon = 0.3$.

4.4 5-Fold Cross Validation pour obtenir l'hyperparamètre C selon d'autres critères

Néanmoins, comme expliqué plus haut, le critère du R^2 n'est pas celui qui nous intéresse le plus. Nous souhaitons avant tout éviter les erreurs inférieures à un certain seuil (0.5 ou 1). J'ai donc choisi d'effectuer "manuellement" des validations croisées de type 5-Fold pour choisir le paramètre C (en fixant préalablement ϵ à 0.3 ou 0.1) qui comparent les erreurs de type A ou B(définies en 2.2).
Il faudrait idéalement compliquer le code pour effectuer un grid search (et non plus chercher C à ϵ fixé) selon ce critère d'erreur mais le résultat obtenu est plutôt cohérent avec les paramètres obtenus précedemment, je n'ai donc pas cherché à aller plus loin.
En effet, voici les résultats obtenus :

— En minimisant les erreurs supérieures à 0.5 : on trouve C=2 pour epsilon=0.1. L'erreur moyenne sur l'ensemble des bases de test pour cette

1. Ceci s'explique par le fait que dans la régression à vecteurs de support, on ne minimise pas que les erreurs supérieures à ϵ mais aussi la norme du vecteurs des coefficients pour éviter le sur-apprentissage.

valeur de C est alors 1625.3.
— En minimisant les erreurs supérieures à 0.5 : on trouve C=2 pour epsilon=0.3. L'erreur moyenne sur l'ensemble des bases de test pour cette valeur de C est alors 1558.3. [2]

J'ai alors choisi par la suite de fixer ϵ à 0.3 et C à 2.

4.5 Résultats dans le cas $C = 2$ et $\epsilon = 0.3$

Erreur quadratique totale L'erreur quadratique totale, contrairement aux régressions précédentes, n'est pas ici l'erreur qui est minimisée.

$$Erreur\_quadratique = 703.5$$

On observe une légère dégradration par rapport aux deux précédentes régressions. Néanmoins, comme les erreurs inférieures à 0.3 ont été "négligées" par le modèle, cela n'indique pas que le modèle n'a pas amélioré la prédiction de nombreux vins qui étaient prédits avec une erreur plus grande.

Erreur absolue totale et moyenne L'erreur absolue totale est déjà plus proche de l'erreur minimisée par la régression à vecteurs de support. On observe ici une amélioration par rapport aux modèles précédents, mais encore une fois cette amélioration n'est pas celle qui nous intéresse véritablement.

$$Erreur\_absolue\_totale = 698.6$$

$$Erreur\_absolue\_moyenne = 0.570$$

Taux de bonnes réponses

$$Taux = 60.77\%$$

Le taux de bonnes réponses s'améliore ainsi très sensiblement par rapport aux 2 précédentes méthodes, où il était d'environ 50%. **Avec ce modèle, bien plus de la moitié des vins ont une note prédite égale leur note réelle de qualité.**

Erreur absolue totale sans prendre en compte les écarts inférieurs à 0.5

$$ErreurA = 495.8$$

Sans prendre en compte les écarts inférieurs à 0.5, l'erreur est nettement plus basse que dans les modèles précédents, ce qu'indiquait déjà l'amélioration du taux de bonne réponse.

Erreur absolue totale sans prendre en compte les écarts inférieurs à 1

$$ErreurB = 294.4$$

Néanmoins, la somme des erreurs absolues supérieure à 1 augmente légèrement : il semble donc que l'amélioration du taux de bonne réponse est due à une

2. En minimisant les erreurs supérieures à 1 : on trouve C=2 pour epsilon=0.3. En minimisant les erreurs supérieures à 1 : on trouve C=1 pour epsilon=0.1. On obtient donc sensiblement les mêmes résultats que précédemment.

meilleure prédiction des vins dont la qualité réelle était assez proche de la qualité prédite mais dont l'arrondi de la prédiction n'était pas égal à la qualité réelle. Par contre, les vins plus éloignés sont moins bien prédits qu'avant.

4.6 Comparaison des erreurs à celles obtenues avec le modèle Ridge polynomial

Comparaison des prédictions aux qualités réelles

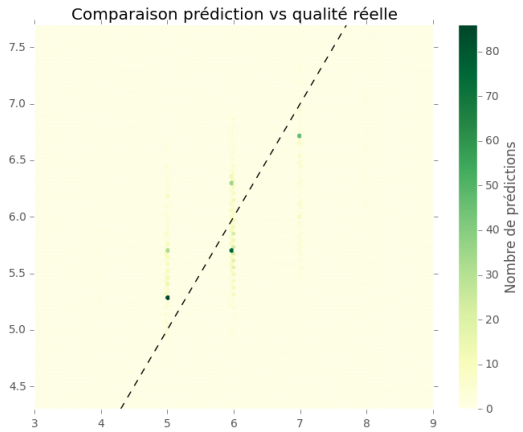

FIGURE 4.1 – Comparaison des prédictions aux qualités réelles

En observant la figure 4.1, on remarque que la particularité de ce modèle par rapport aux précédents est d'avoir une variance beaucoup moins forte des prédictions autour de la qualité réelle. Une grande partie des vins de qualités 5, 6 et 7 ont des qualités prédites quasiment identiques. (Pour le reste des qualités, leur faible effectif dans la base empêche de bien observer la variance de leurs prédictions.) En ce sens ce dernier modèle est beaucoup plus cohérent avec la réalité, du moins pour les qualités moyennes qui sont les plus répandues.

Quelles sont les qualités mal prédites ?Comparaison avec le modèle précédent.
On constate sur la figure 11 que, de même que dans les modèles précédents, les qualités "extrêmes" sont les plus mal prédites. Néanmoins, pour les vins de qualité 8, il semble y avoir une améliortion des prédictions. Au contraire, pour les vins de qualité 4 il y a une dégradation.

Ce modèle permet donc d'améliorer nettement les prédictions pour des vins de qualité moyenne, en offrant des prédictions précises et très cohérentes avec la réalité (mêmes prédictions pour vins de qualités réelles égales), mais ne permet pas de résoudre le problème des vins

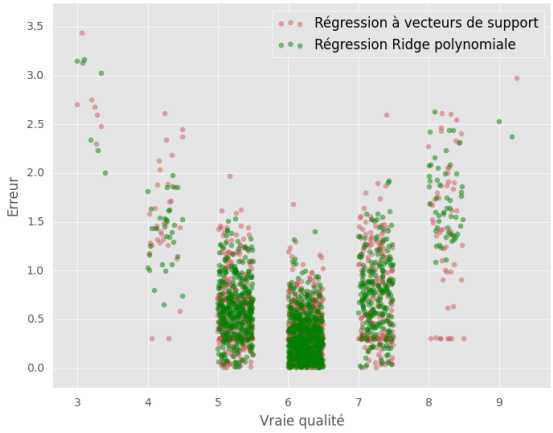

FIGURE 4.2 – Identification des qualités les plus mal prédites - Comparaison au modèle précédent

de qualité extrême.

Quelles sont les caractéristiques chimiques des vins mal prédits ?

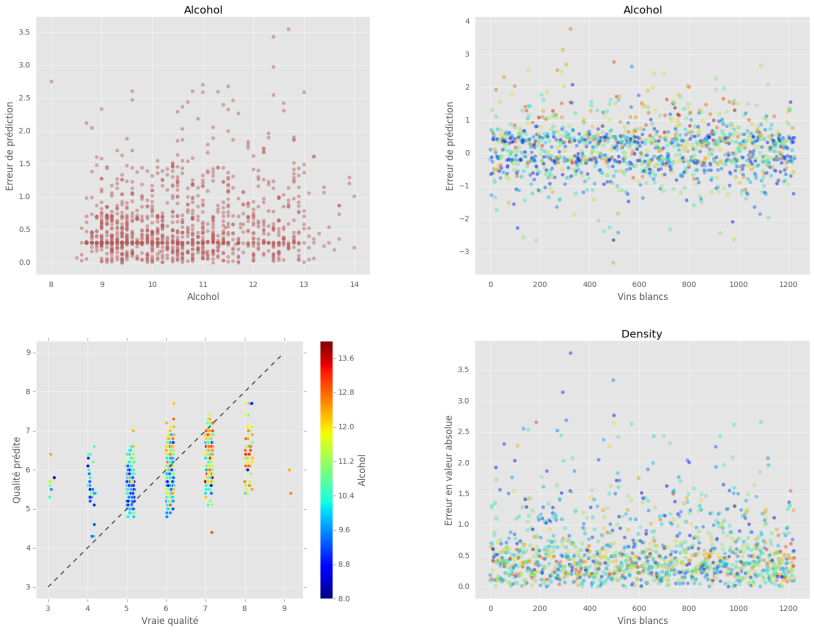

FIGURE 4.3 – Analyse de l'erreur en fonction des caractéristiques chimiques

Sur la figure 4.3, plus nettement avec la régression à vecteurs de support qu'avec les régressions précédentes, on observe une moins bonne prédiction des vins à taux d'alcool élevé et à densité faible. Les autres caractéristiques ne semblent

pas particulièrement discriminantes.

On remarque aussi que le taux d'alcool est moins nettement positivitement corrélé aux prédictions qu'avec les modèles précédents. Ceci pourrait être lié en un sens au fait qu'on obtient de meilleures prédictions avec ce modèle, et semble cohérent intuitivement : un vin avec un taux d'alcool plus élevé n'est pas nécessairement meilleur en pratique. Ainsi, il pourrait être intéressant de voir si la performance augmente lorsqu'on utilise deux modèles différents selon le taux d'alcool (haut ou bas) du vin inconnu.

4.7 Pour aller plus loin : ébauche d'un double modèle selon le taux d'alcool

J'ai séparé la base initiale entre une base contenant les vins ayant un taux d'alcool supérieur à 12 (points jaunes, oranges et rouges sur les graphes précédents) et les vins ayant un taux d'alcool inférieur.

Pour pouvoir comparer correctement les résultats, j'ai concaténé les 2 bases d'apprentissages obtenues en une seule etles bases de tests obtenues en une seule, de façon à entraîner un modèle de régression à vecteurs de support sur cette nouvelle grande base d'apprentissage et obtenir des résultats sur cette nouvelle grande base de test. Ainsi, ce seront les prédictions des mêmes vins qui seront comparées pour voir s'il y a une amélioration lorsqu'on utilise deux modèles distincts selon le taux d'alcool.

Pour chacune des deux nouvelles bases, j'ai entraîné une régression à vecteurs de support après recherches des hyperparamètres par validation croisée (Grid Search comme décrite plus haut). On obtient bien des hyperparamètres optimaux différents pour les vins à taux d'alcool élevé et les vins à taux d'alcool bas.

Critère d'erreur	**Double modèle SVR**	**Modèle simple SVR**
Erreur quadratique	671.5	709.6
Erreur absolue	682.7	711.2
Erreur A	504.5	514.5
Erreur B	284.6	295.1
Taux de bonnes réponses	57.55%	58.64%

On remarque que ce double modèle n'apporte pas d'amélioration claire. Certes les erreurs semblent plus faibles mais le taux de bonnes réponses diminue légèrement. Il ne semble donc pas très intéressant de creuser dans cette direction.

Chapitre 5

Conclusion

Pour conclure, les qualités les plus rares, c'est-à-dire les plus basses et les plus hautes, semblent particulièrement difficiles à prédire. On observe bien des prédictions en général plus hautes ou plus basses pour ces qualités que pour les qualités moyennes mais jamais "suffisamment" et de nombreuses prédictions sont vraiment très éloignées. Cela peut s'expliquer de plusieurs façons. La première serait qu'il n'y a pas suffisamment de vins de ce type dans la base de données, et donc qu'on ne réussit pas à leur trouver des caractéristiques chimiques véritablement communes qui les différencient des autres vins. La seconde interprétation serait qu'une note attribuée à la qualité du vin est une donnée trop subjective, et ce particulièrement pour les vins plus mauvais ou meilleurs que la moyenne. Il pourrait très bien y avoir une tendance, parfaitement subjective, à sur-noter ou sous-noter les vins plus particuliers. La troisième serait qu'il existe certaines caractéristiques du vin sur lesquelles nous n'avons aucune information qui gonflent ou font plonger les notes de certains vins alors que les caractéristiques chimiques sur lesquelles nous avons des informations ici sont plutôt similaires à celles des vins de qualité moyenne.

Néanmoins, on arrive avec le modèle de régression à vecteurs de support à obtenir des prédictions relativement précises sur les vins les plus communs, et surtout très cohérentes avec la réalité au sens où des vins de qualité réelle identique sont souvent prédits au même niveau. Il est même étonnant d'arriver à une telle précision de résultats pour la prédiction d'une variable aussi subjective que la qualité d'un vin notée sur 10. Si la base de données contient des vins dont les disparités en termes de qualité sont proches de la réalité dans un supermarché, le modèle pourrait être exploitable dans une application mobile pour consommateurs puisqu'il n'est pas tellement mauvais pour les vins les plus courants. Encore faut-il que les caractéristiques chimiques soient affichées sur la bouteille... ce qui n'est pas le cas en France.

Annexe A

Informations sur la base de données

Citation Request : This dataset is public available for research. The details are described in [Cortez et al., 2009].

Please include this citation if you plan to use this database :
P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
Modeling wine preferences by data mining from physicochemical properties.
In Decision Support Systems, Elsevier, 47(4) :547-553. ISSN : 0167-9236.

Available at : [@Elsevier]
http ://dx.doi.org/10.1016/j.dss.2009.05.016 [Pre-press (pdf)]
http ://www3.dsi.uminho.pt/pcortez/winequality09.pdf [bib]
http ://www3.dsi.uminho.pt/pcortez/dss09.bib

1. Title : Wine Quality

2. Sources
Created by : Paulo Cortez (Univ. Minho), Antonio Cerdeira, Fernando Almeida, Telmo Matos and Jose Reis (CVRVV) @ 2009

3. Past Usage :
P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
Modeling wine preferences by data mining from physicochemical properties.
In Decision Support Systems, Elsevier, 47(4) :547-553. ISSN : 0167-9236.

In the above reference, two datasets were created, using red and white wine samples.
The inputs include objective tests (e.g. PH values) and the output is based on sensory data (median of at least 3 evaluations made by wine experts). Each expert graded the wine quality between 0 (very bad) and 10 (very excellent). Several data mining methods were applied to model these datasets under a regression approach. The support vector machine model achieved the best results. Several metrics were computed : MAD, confusion matrix for a fixed error tole-

rance (T), etc. Also, we plot the relative importances of the input variables (as measured by a sensitivity analysis procedure).

4. Relevant Information :
The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine.
For more details, consult : http ://www.vinhoverde.pt/en/ or the reference [Cortez et al., 2009].
Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

These datasets can be viewed as classification or regression tasks.
The classes are ordered and not balanced (e.g. there are munch more normal wines than excellent or poor ones). Outlier detection algorithms could be used to detect the few excellent or poor wines. Also, we are not sure if all input variables are relevant. So it could be interesting to test feature selection methods.

5. Number of Instances : red wine - 1599 ; white wine - 4898.

6. Number of Attributes : 11 + output attribute

Note : several of the attributes may be correlated, thus it makes sense to apply some sort of feature selection.

7. Attribute information : For more information, read [Cortez et al., 2009].

Input variables (based on physicochemical tests) :
1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 - alcohol
Output variable (based on sensory data) :
12 - quality (score between 0 and 10)

8. Missing Attribute Values : None

Annexe B

Code Python

Voir le notebook.

Un certain nombre d'essais annexes ne sont pas présentés dans le rapport et donc légèrement expliqués dans le notebook. Notamment, j'ai commencé une réflexion sur un modèle de détection des mauvais vins en utilisant des classifications à 2 classes.