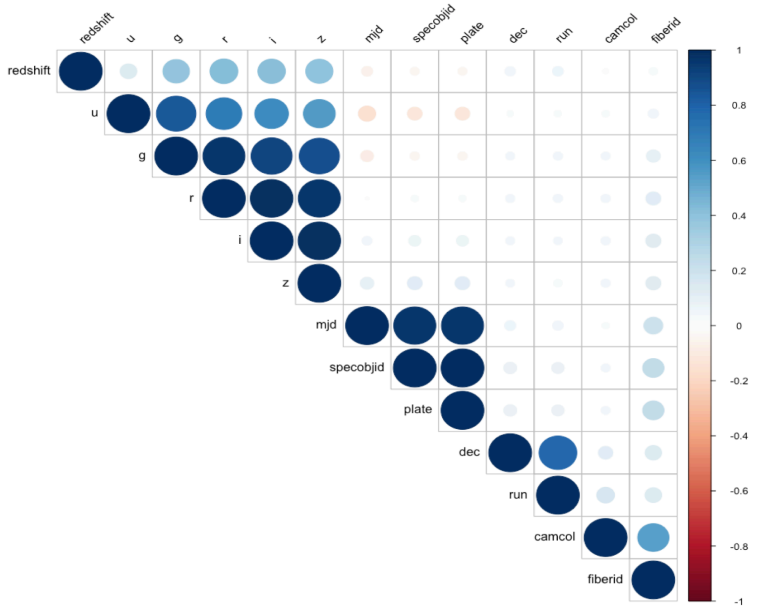
Astronomies

Extraction des données

Les données concernent la description de trois types d'objets astronomique. L'objectif est de prédire type d'astre entre étoile, galaxie, quasar. Il y a 5000 individus et 17 prédicteurs. On choisit dès à présent de supprimer les colonnes 1 et 10 contenant des identifiants unique n'apportant pas d'informations particulières.

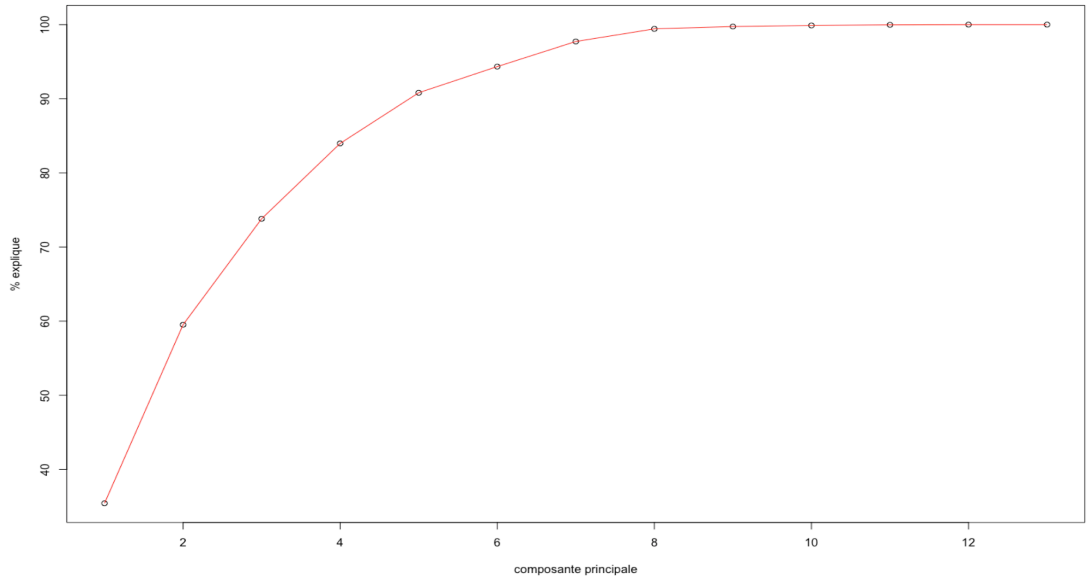
On sépare les données en 2/3 de données d'entraînements et 1/3 de données de test de manière aléatoire. Grâce à la fonction sample(). Par la suite nous évaluerons le MSE par validation croisée afin de pouvoir comparer les modèles en eux.

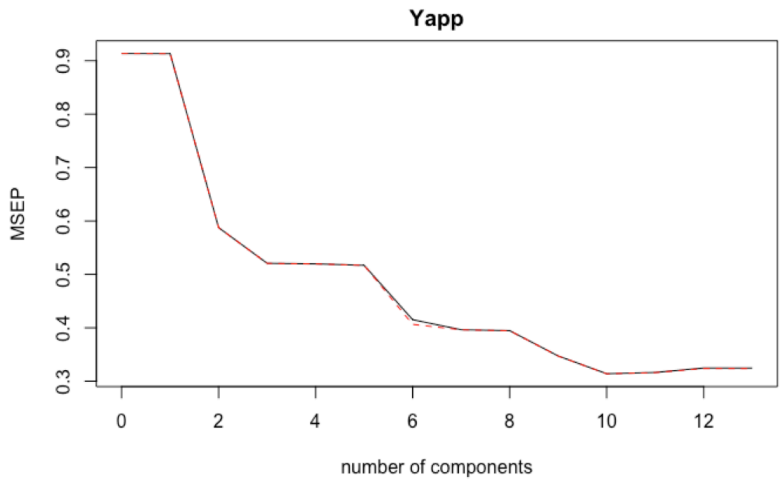
Matrice de corrélation et ACP



Dans un premier temps nous commencons par regarder notre dataset astronomy afin de se donner des idées de sa structure. Pour se faire nous réalisons sa matrice de correlation grâce à la function cor(). On Remarque alors que certaines variables sont fortement corrélées. Nous constatons sur le graphique que des variables sont fortement corrélées.

Ainsi nous réalisons une ACP sur notre dataset afin de réduire le nombre de variables. Après analyse de notre ACP nous réalisons que plus de 90% de l'information de notre dataset est contenu dans 6 à 9 variables.





Néanmoins une analyse en PCR prouve que nous aurons un MSE le plus faible avec l'intégralité de nos composantes. Ainsi, nous allons essayer d'effectuer un apprentissage sur l'ensemble de notre dataset. Nous allons comparer différents classifieurs que nous résumerons après.

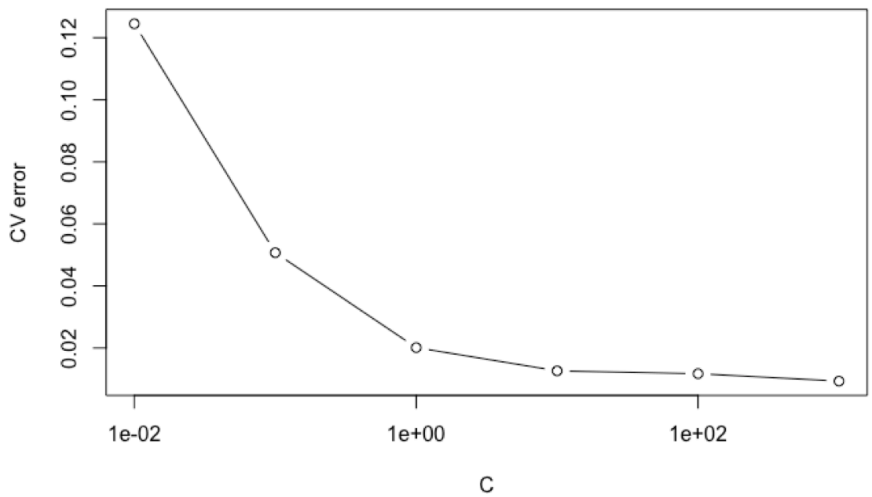
Normalisation des données

Dans certains cas, normaliser les données peut améliorer les performances de modèles. Les tests effectués précédemment ont également été faits sur les données normalisée mais n'a apporté aucune amélioration. Pour une question de lisibilité, on n'a pas considéré nécessaire de rajouter ces résultats.

SVM

Dans cette partie nous détaillerons le choix de notre classifieur. Les autre seront listés dans un tableau sur le bilan des méthodes. Ici nous utilisons la fonction svm() du package e1071, en faisant varier divers paramètres. En particulier, on fixera les paramètres suivants par validation croisée en utilisant la fonction tune() du même package

L'hyperparamètre cost (correspondant au "C" dans la formule de Lagrange) qui précise le cost d'une violation de la marge. Lorsque cost est petit, les marges seront d'autant plus larges et de nombreux Support Vector chevaucheront les marges. Lorsque que cost est grand, les marges seront étroites et peu de Support Vector dépasseront les marges.



Le paramètre du noyau kernel, qui définit le type parmi : linear, polynomial, radialbasis et sigmoid.

Dans le cas du noyau polynomial, nous obtenons la courbe suivante pour déterminer la valeur de l'hyperparamètre C.

Après plusieurs essais nous obtenons un taux d'erreur extremement faible 0.8%. Cependant cette méthode reste difficilement interprétable.

Bilan des méthodes

Nom de la méthode	Erreur	Intervalle de confiance 95%
GMM	20%	18% < x < 22%
SVM	0.8%	0.% < x < 1.8%
Random Forest	1%	0.3% < x < 1%
Bagging	1.2%	0.7% < x < 1.8%

Conclusion

Nous réalisons ici que les arbres offrent un bon taux d'erreur de 1%. Alors une question se pose, vaut-il mieux privilégier dans ce cas l'interprétabilité par rapport au taux d'erreur ? Tout dépend de l'utilisation faite derrière grâce à ces données. Dans notre cas, avec 0.02% d'écart avec le SVM par rapport au arbres optimisés par du bagging, nous pouvons être tenté de prendre la méthode de bagging si l'utilisateur aura besoin de compréhension.

En revanche pour la finalité du TP nous allons conserver le classifieur SVM. Par ailleurs, le taux étant bas, nous n'avons pas besoin d'utiliser nos variables issus de l'ACP. Nous acceptons le trade-off biai-variance. En effet le nombre de variables reste acceptable (15).

Rendement du mais

Données

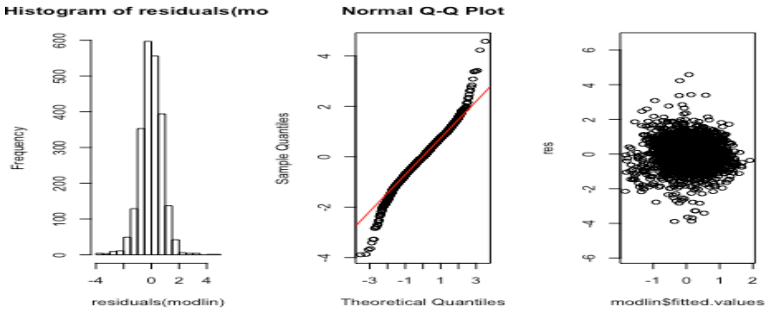
Extraction des données

Les données concernent le rendement du maïs en France, dans les différents département sur plusieurs années. L'objectif est de prédire le rendement à partir de données climatiques. Il y a 2300 individus et 58 variables : 57 prédicteurs. On choisit dès à présent de supprimer la colonne X contenant un identifiant unique n'apportant pas d'informations particulières.

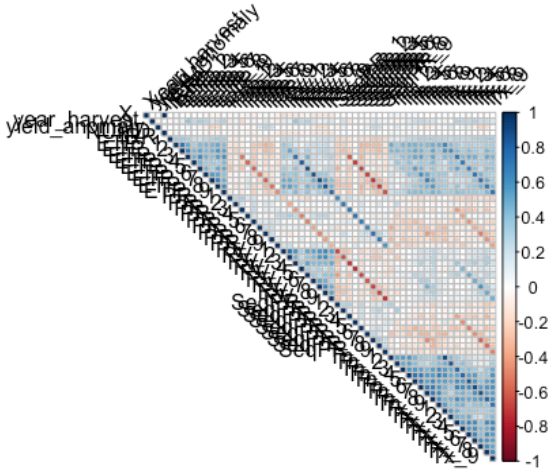
On sépare les données en 80% de données d'entraînements et 20% de données de test de manière aléatoire. Par la suite nous évaluerons le MSE par validation croisée afin de pouvoir comparer les modèles en eux.

Estimation du modèle et des graphes de résidus

Les hypothèses du modèle linéaire ne semblent pas être valable car les résidus ne sont pas gaussiens (test de Shapiro-Wilk). Cependant dans le cas qu'un échantillon de grande taille, ce modèle reste robuste et peux convenir. C'est pour cela que nous le testons par la suite.



Matrice de corrélation



Dans un premier temps, nous nous sommes intéressés à la corrélation entre les variables explicatives et yield_anomaly. Si la plupart des corrélations sont plutôt faibles, on remarque des coefficients relativement forts entre certaines variables explicatives.

Sélection de variables

Subset

On obtient donc les modèles suivant suite aux stepwise selection.

```
models <- c(
 yield_anomaly~.,
 yield_anomaly~IRR+ETP_2+ETP_3+ETP_6+ETP_7+PR_4+PR_5+PR_6+PR_7+PR_8+PR_9+RV_2+RV_6+SeqPR_1+SeqPR_2+SeqPR_4+SeqPR_9+Tn_1+Tn_3+Tn_4+Tn_5+Tn_8+Tx_2+Tx_3+Tx_7,
 yield_anomaly~ETP_2+ETP_3+ETP_6+ETP_7+ETP_8+PR_4+PR_5+PR_6+PR_7+PR_9+RV_2+RV_6+RV_7+RV_8+SeqPR_1+SeqPR_2+SeqPR_4+SeqPR_9+Tn_5+Tn_7+Tx_2+Tx_3+Tx_4+Tx_7
)
```

Régularisation

On va utiliser la régularisation pour tenter d'aider le modèle à mieux généraliser sur les données de test. Sélection de la pénalité par validation croisée.

Ridge, Lasso et ElasticNet

Les figures représentent l'évolution du MSE sur le jeu de données en fonction du paramètre α pour les 3 modèles. On se propose de tester 3 régularisations différentes : le ridge (α = 0), le lasso (α = 1) et elasticnet dont la valeur de alpha est déterminé par CV pour chacun des modèles. Cela donne respectivement pour les modèles 1, 2 et 3 : alpha = 0, 0.1 et 0.9.

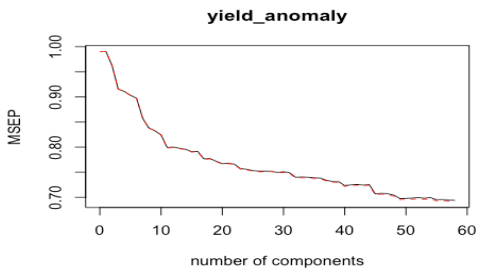
On utilise le package glmnet sur les trois modèles étudiés précédemment. Les performances des 9 modèles résultants seront mesurées par le MSE.

Le tableau récapitulatif des résultats est le suivant :

Model	Variables	Ridge (alpha = 0)	Lasso (alpha = 1)	Elastic Net (0,0.1,0.9)
1	57	0.7045976	0.7096448	0.7045976
2	25	0.7392032	0.7394032	0.7387808
3	24	0.759878	0.7585088	0.7585774

Réduction de dimension

PCR



On constate que, d'après l'analyse en composante principale, pour détenir le plus d'informations il faut 57 composantes principales.

Ainsi, on obtient :

PRC	Résultat
Composante	57
MSE	0.7116193

Normalisation des données

Dans certains cas, normaliser les données peut améliorer les performances de modèles. Les tests effectués précédemment ont également été faits sur les données normalisée mais n'a apporté aucune amélioration. Pour une question de lisibilité, on n'a pas considéré nécessaire de rajouter ces résultats.

Modèle

Régression linéaire

Model	LM
1	0.7137491
2	0.7394938
3	0.7586552

KNN - Plus proche voisin

Model	KNN (k=10)
1	1.033251
2	0.8369283
3	0.830511

Régression Multinomiale

La régression multinomiale est assez limitée quant aux modèles mis en place, on se focalisera plus sur les splines, en particulier les modèles additifs. De plus le nombre de combinaison est trop grand, ce qui nous empêche de tester.

Splines

Dans le cas d'un modèle multidimensionnel, les splines basiques, naturelles et smooth ne s'appliquent pas. On peut faire appel aux tensors mais la dimension croit exponentiellement et on perd cruellement en interprétabilité. Pour pallier à ce problème, on utilise les GAM (Generalized Additive Models). Ce modèle est bon pour expliquer mais reste souvent moins performant pour prédire.

Ainsi, on obtient :

Model	GAM	Résultat
	DF	5
1	MSE	0.6822596
2	MSE	0.7300008
3	MSE	0.752631

Cependant si on enlève les outliers, les autres modèles sont moins précis sauf les GAMs :

Model	GAM	Résultat
	DF	5
1	MSE	0.6498522
2	MSE	0.7221726
3	MSE	0.7463895

SVR

On détermine par CV la valeur de C et on l'applique sur notre modèle afin de connaitre le MSE :

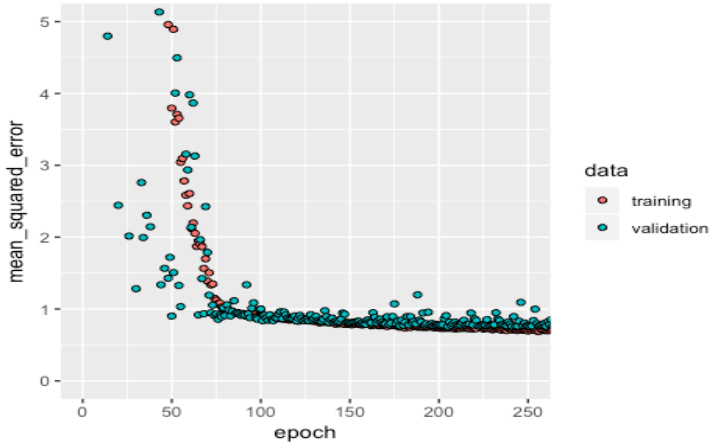
Model	SVR	Résultat
	C	1
1	MSE	0. 5495835
	C	1
2	MSE	0.5594924
	C	1
3	MSE	0.5833493

Neural network

Dans un premier temps, nous avons utilisé le package NNET en optimisant le nombre de neurones cachés et en sélectionnant le lambda optimal par cross-validation.

Model	NNET	Résultat
	size	5
1	MSE	0.8002449
	lambda	1
2	MSE	0.6861912
	lambda	10
3	MSE	0.7372022
	lambda	6

Nous allons utiliser à présent le package Keras qui offre plus de possibilités.



Dans un premier temps nous avons considéré un modèle de réseau de neurone à 3 couches cachées. Par la suite nous avons voulu adapter les différents paramètres : Ce modèle à tendance à overfit les données d'entrainements. C'est pourquoi on a mis en place kernel_regularizer = regularizer_l2(l=0.1)) qui permet de régulariser les données. Dans cette optique on utilise également la méthode d'early stop pour éviter au maximum les problèmes d'overfit.

Concernant le nombre d'units, nous avons testé plusieurs valeurs, de 1 å 70. De plus, nous avons fait varié le nombre de couche, cependant, lorsque celui-ci est trop grand devient moins pertinent.

Deep NN	Résultat
MSE	0.8984264
couche	(64,64,1)
Regularisé	oui
MSE	0.7208582
couche	(64,64,1)
Regularisé	non

Conclusion

Après une étude de différents modèles en fonction de la structure de nos données, il se trouve que le modèle basé sur le SVR donne un MSE minimum de 0.5495835 pour le modèle sans sélection de colonnes.