06/01/2019

Rapport final

Projet inno : Mooveat

latitudes
exploring tech
for good_

mooveat
reconnectons-nous

Michel Richard, Othmane Jebbari, Alassane Diallo
CENTRALESUPELEC

Table des matières

1. Contexte

Le projet de Mooveat est de produire une plateforme qui permette à l'agriculteur de gérer sa production de manière raisonnée, en faisant correspondre offre et demande. Par ailleurs, il vise à donner l'ensemble des informations disponibles pour le client avec un mapping des différents points de vente disponibles et des informations sur ceux-ci. Pour que l'agriculteur puisse gérer sa production 3 fonctionnalités sont nécessaires : détermination de l'offre, détermination de la demande et simulation d'une potentielle nouvelle offre (comme par exemple l'achat d'une nouvelle parcelle).
Initialement, le cadre du projet était assez large avec 3 propositions d'algorithmes possibles : prévision de la demande, prévision de l'offre (c'est-à-dire de la production potentielle pour une parcelle donnée) et prévision des prix de vente optimaux.

2. Reformulation de notre besoin

Nous avons rapidement pris conscience du fait que, au vue de nos compétences en Machine Learning, produire les 3 algorithmes sur le temps imparti serait totalement impossible. Nous avons donc décidé de nous restreindre à un seul algorithme. Nos facteurs de décision était la présence potentielle ou non de Machine Learning dans le projet et la disponibilité de bases de données exploitables. La phase de d'exploration de 3 semaines nous a permis de décider de cela.

a. Phase d'exploration

Nous nous sommes attribués à chacun un algorithme sur lequel nous devions chacun comprendre au mieux le sujet et son cadre ainsi que rechercher des bases de données éventuellement disponibles.
Après recherches, nous avons pu élire en accord avec le client le sujet qui nous semblait le plus approprié à la fois pour notre montée en compétences et la présence de bases de données : l'algorithme de détermination de l'offre.

b. Facteurs de décision

Après la mise en commun de nos recherches, nous avons éliminé :

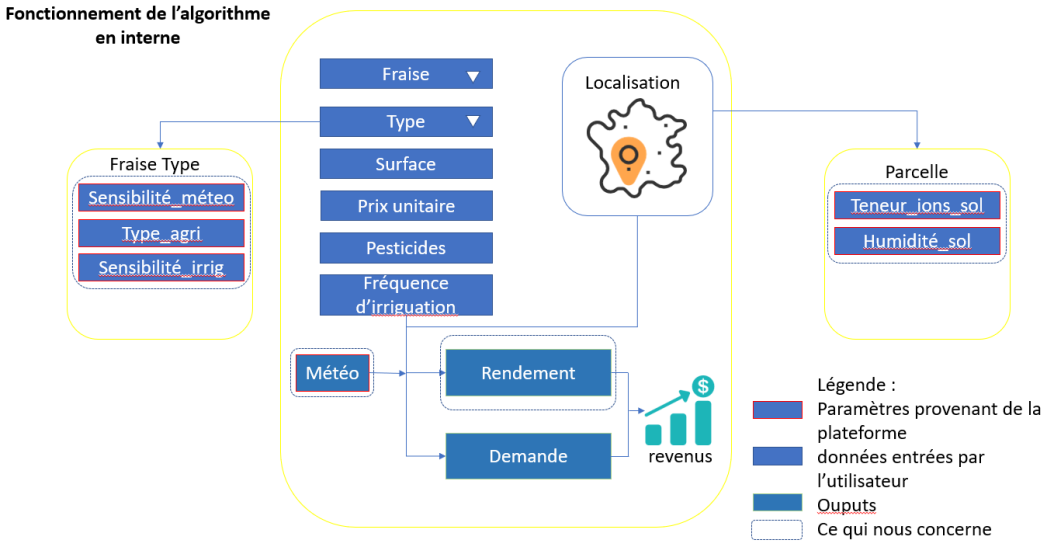
- L'algorithme de prédiction des prix de vente car il nous manquait des bases de données précises (notamment au niveau des prix moyens par département par exemple) et surtout que trouver l'ensemble des facteurs influençant serait extrêmement compliqué étant donné la complexité des modèles économiques.

- L'algorithme de prédiction de l'offre pour un point de vente donné. En effet, nous pouvions représenter l'offre créée par les ménages en une zone la carte du territoire grâce aux études de l'INSEE (recensement de la population en 2011) et de l'Anses (étude INCA 3, consommation moyenne par produit, tranche d'âge et région). Cependant, il n'y a aucun intérêt à utiliser du Machine Learning pour ce calcul. Là où le problème devient beaucoup plus intéressant est lorsque nous considérons que l'offre des ménages se répartit ensuite aux endroits où ses derniers achètent. Cela reviendrait donc à déterminer pour un point de vente l'offre réel qu'il attire en fonction de la concurrence, du temps de trajet pour les ménages à ce dernier, de la présence d'un axe de passage touristique, etc. Dans ce cas le Machine Learning serait clairement une solution pour résoudre ce problème. Cependant, nous n'avons pas réussi à trouver de base de données sur les ventes d'un point de vente précis. Nous n'aurions donc pas pu travailler sur l'aspect comportant du Machine Learning, composante que nous avions réellement envie de retrouver dans notre projet.

Ainsi, nous avons choisi de travailler sur le troisième algorithme de prédiction de l'offre agricole (c'est-à-dire détermination de la production sur une parcelle donnée). En effet, nous avions à notre disposition une base de données sur les données météorologiques ainsi que l'indice de végétation (surface recouverte par de la végétation/surface totale pour une parcelle agricole) qui à un comportement semblable à celui du rendement (voir section « Démarche ») qui nous permet de travailler l'aspect Machine Learning dans notre projet.

3. Objectifs

L'objectif fixé à la suite de la scope review est d'avoir produit un Back-end qui soit opérationnel, c'est-à-dire qui réponde au besoin du client.
Voici la vision que nous avons déterminé avec le client qui correspondrait à l'algorithme que nous devons fournir :



L'algorithme prend en input des paramètres comme le type de produit (espèce, variété) que l'agriculteur souhaite mettre en culture, la surface utilisée, le type de pesticide ou de fertilisant, la date de semi, la date de labour, la fréquence d'irrigation, localisation de la parcelle etc.). Le prix de vente est soit rentré par l'agriculteur, soit il est déterminé par l'algorithme de prédiction du prix de vente optimal. Ajoutés à des paramètres connus en interne par l'algorithme (typiquement la météo prévue, l'inflation, etc .), l'algorithme sort en output la prédiction de l'offre, la prédiction des ventes et les revenus associés. La partie à laquelle nous nous intéressons est de prévoir l'offre (ie. les rendements sur la parcelle).

Enfin, si le planning et notre vitesse d'exécution nous le permet deux autres parties pourront être ajoutées au livrable : Front-End et/ou réflexion sur la manière de réutiliser le code produit pour les autres fonctionnalités décrites dans la fiche projet de départ (voir planning).

4. Démarche

a) Bases de données trouvées pendant la phase d'exploration

Après les nombreuses séances passées à faire de la recherche documentaire pour à la fois comprendre le contexte du projet et ses notions clés et rechercher des données utiles à l'algorithme, nous avons trouvé 4 bases de données qui nous semblaient utiles pour l'algorithme.

Les 3 premières proviennent de la plateforme Agri4cast (open-source). Ce sont des données satellites avec un maillage de 25km sur 25km.

- Grids: (X_{min}, X_{max}, Y_{min}, Y_{max}, Grid code, Grid Id, hauteur). Les données qui nous semblent intéressantes sont les données de localisation (X et Y) et le Grid Id pour pouvoir localiser la parcelle et croiser les données avec les autres bases de données.
- Données météorologiques : (Grid_Id, Latitude, Longitude, Altitude, day T_{min}, T_{max}, T_{mean}, v_{vent}, Patm, précipitations, radiations, snow depth). Ici, ce sont toutes les données météorologiques qui nous semblent utiles (T_{min}, T_{max}, T_{mean}, v_{vent}, Patm, précipitations, radiations, snow depth). Il reste à déterminer quels paramètres sont vraiment essentiels dans la détermination des rendements.

- Gridded remote sensing data (Id_Grid, Sensor_Id, Day, Value, Id_cover). “Value” correspond à l’indice de végétation, c’est-à-dire la surface couverte par la végétation sur la surface du grid. Nous pensions pouvoir remonter au rendement à partir de cette donnée éventuellement (en supposant que la relation entre indice de végétation et rendement existe).
- Yearly modeled crop area (Id_grid, Id_crop, year area, ratio). Le type de produit agricole (blé, riz, betterave, etc.) produit sur le grid correspondant est ce qui pouvait nous servir. Si le rendement était déterminable à partir de l’indice de végétation, il aurait aussi dépendu du type de produit. Nous pensions donc qu’en croisant ces deux données nous pouvions peut-être déterminer le rendement des parcelles et ainsi avoir les outputs test pour entraîner notre modèle.

Une seconde base de données trouvées est une base de données sur la composition des sols (les éléments minéraux présents dans les sols étant essentiels à la croissance des plantes).

- Gis sol : données sur plus de 26 millions d’échantillons de terre entre 2004 et 2009. On a les quantités chaque type de minéral dans les échantillons. Il nous reste à déterminer quels minéraux influent vraiment et voir s’il est possible de lier quantité de pesticides avec les minéraux présents dans le sol.

Enfin, nous avons trouvé une base de données de l’INRA correspondant exactement à ce que nous cherchons. Cette base de données -Aster-ix- contient les rendements par parcelles avec l’intégralité des paramètres dont nous avons besoin (hormis les données météorologiques qui nous sont données par la base de données Agri4Cast) comme l’itinéraire technique (ensemble des actions faites sur la parcelle avec date de début, date de fin, surface traitée, etc.), la variété produite, la surface de la parcelle, la localisation, etc. Cette base de données est une aubaine pour nous car elle présente la granularité nécessaire pour entraîner notre algorithme, ce que nous n’avions pas dans toutes les autres bases de données. Le problème est que cette base de données n’est pas encore en open-source et nous avons donc pris contact avec l’équipe sur place. Par ailleurs, le travail de l’équipe de l’INRA se base sur un site unique (Mirecourt) qui ne fait que de l’agriculture biologique. Cela s’inscrit bien dans la politique de Mooveat qui vise à promouvoir les circuits-courts dont les acteurs travaillent majoritairement autour de l’agriculture biologique, car se sont ce que cherchent majoritairement les clients.

En voici un schéma relationnel global mais simplifié de la base de données :

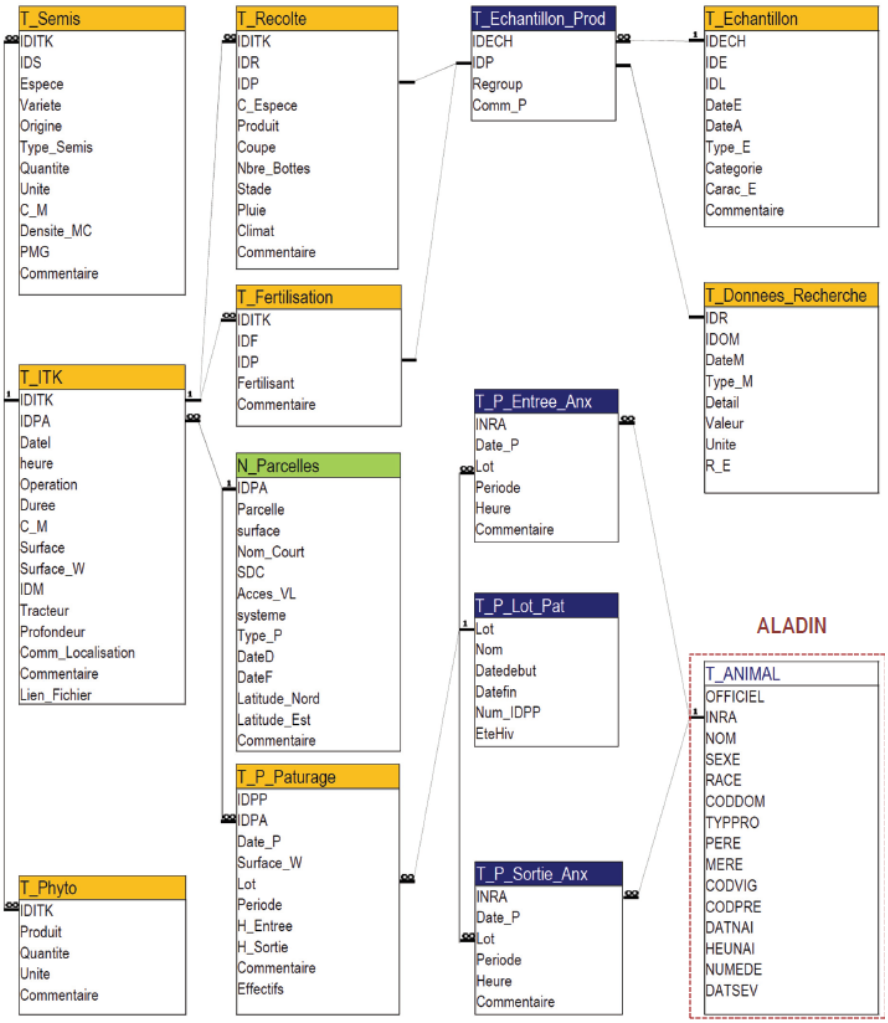


Schéma relationnel de la base de données Aster-ix

Les tables qui nous intéressent sont les suivantes : *T_Recolte* (pour récupérer les rendements surfaciques), *T_Parcelles* pour récupérer la localisation des parcelles ainsi que leur surface, *T_ITK* pour récupérer l'itinéraire technique, et les tables qui tournent autour de *T_ITK* pour pouvoir récupérer les données liées aux opérations (comme le type de fertilisant utilisé, la quantité, etc.).

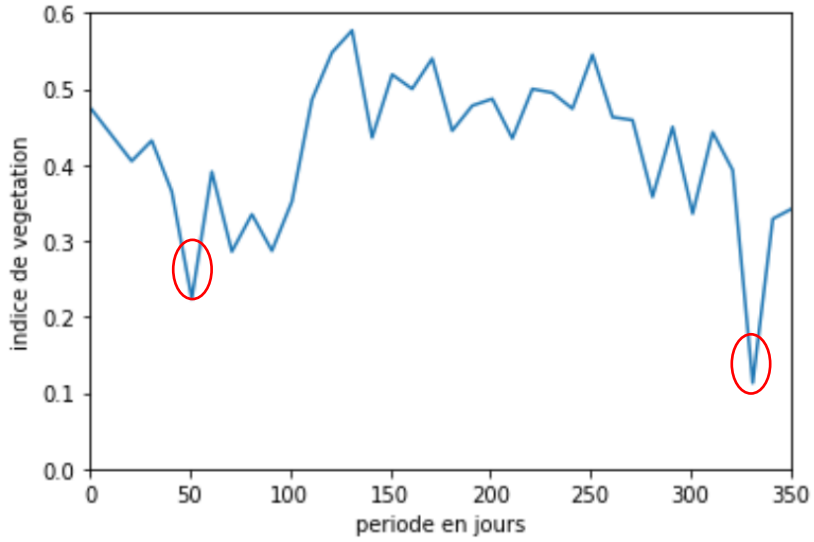
b) Utilisation de la base de données

Nous avons donc décidé de procéder de la manière suivante :

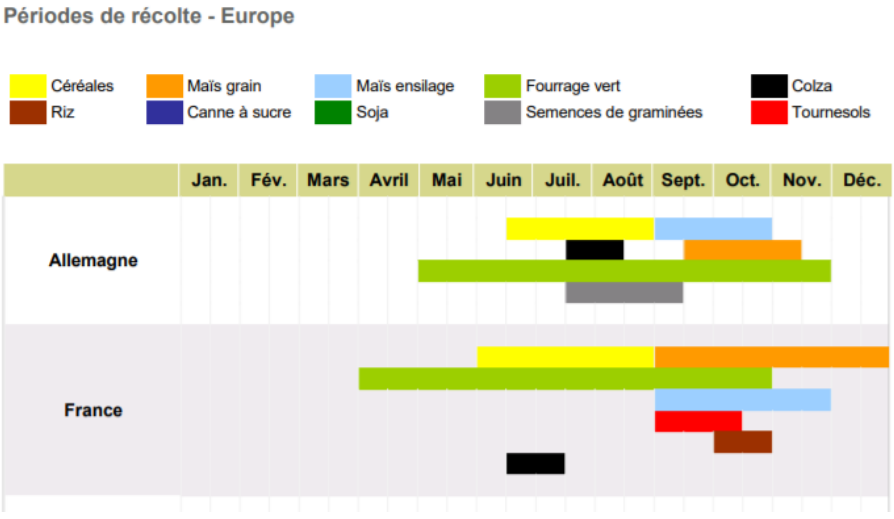
- Dans un premier temps, nous avons donc décidé de travailler sur la base de données Agri4Cast en attendant d'avoir accès à une extraction de la base de données de l'INRA. En faisant l'hypothèse que l'indice de végétation d'une parcelle (surface occupée par de la végétation / surface totale de la parcelle) se comporte de manière similaire au rendement sur une parcelle (pour plus de précisions voir ci-après), un algorithme de Machine Learning permettant de prédire l'indice de végétation permettrait d'avoir une structure similaire à celui pour les rendements. Nous avons donc commencé par travailler sur la mise en forme des données tout en essayant de trouver un lien entre indice de végétation et

rendement pour pouvoir produire un algorithme réutilisable pour la base de données Aster-ix.

- Voici pourquoi nous pensons que cette démarche est valable :



Indice de végétation sur un grid, année 2015, maïs



Périodes de récoltes des différentes espèces(https://www.claas.fr)

Si on trace l'indice de végétation sur une année, on a toujours un ou deux pics nets qui peuvent correspondre à une récolte ou un labour. Le premier qui a lieu en décembre pourrait correspondre à la récolte du blé. Le deuxième pourrait être simplement lié à une diminution de l'indice de végétation en hiver (il apparait sur quasiment tous les grids, quelque-soit l'année). Ainsi, la variation brusque d'indice de végétation (Delta(I)) serait l'output de notre algorithme et pourrait avoir des variations similaires au rendement.

c) Planification concrète

Tout d'abord, nous nous sommes rajoutés une séance de travail par semaine le jeudi (14h-17h) car nous sommes tous libres et de ne se rejoindre qu'une fois par semaine ne nous permettait pas de mettre en commun nos informations assez souvent et endiguait notre avancée.
Par ailleurs, nous avons décidé de nous répartir les tâches en fonction de ce que chacun de nous voulait tirer de ce projet en termes de compétences. Tandis qu'Alassane et Othmane veulent vraiment monter en compétences en Machine Learning, Michel était plus intéressé par la compréhension du sujet de Mooveat dans sa globalité. Alassane et Othmane ont donc travaillé sur l'aspect informatique (mise en forme des données, implémentation de l'algorithme etc.) pendant que Michel a traité de trouver comment est-ce que l'ensemble des paramètres influant sur le rendement s'agençait et de faire un lien entre indice de végétation et rendement.

De plus, nous avons aussi mis en commun l'ensemble de nos impératifs (qu'ils soient scolaires ou personnels), nous avons donc connaissance de l'ensemble des séances de travail qu'il reste à notre disposition. Nous avons aussi déterminé l'ensemble des deadlines et jalons liés au projet pour pouvoir adapter notre planning en conséquence.

Par ailleurs, nous avons décidé de « faire d'une pierre deux coups » et d'utiliser notre projet inno et l'algorithme que nous devons produire pour notre projet de Machine Learning. Outre l'avantage de pouvoir faire un deux en un, cela nous rajoute un deadline au 6/01 (10 jours avant notre soutenance finale) ce qui nous permettra de nous concentrer ensuite sur des potentielles améliorations ou encore le Front-End.
Pour ce qui est de la formation en Machine Learning, nous la faisons en parallèle sur notre temps libre.

Voici le planning que nous avons établi :

06-nov	Définir ce que l'on fait une fois le code opérationnel Définition planning Définir étapes de l'algorithme Avoir l'ensemble des données dont on a besoin à disposition avant la fin de la semaine Mise en forme du Jalon 2
13-nov	Avoir un modèle précis de l'algo (étapes, input/output, modèle associé), Mise en forme du jalon 2
20-nov	Forum CentraleSupélec
22-nov	Mise en forme Jalon 2 et début du codage
27-nov	Jalon 2 Explication claire des objectifs

	Résumé de la recherche Explication de l'algo Mise en forme
28-nov	Exam Electif 2
29-nov	Exam Eco, Codage et recherche de contacts pour comprendre les paramètres sur le rendement
02-déc	Assignement 2 : Machine learning . Prise de contact avec l'INRA
07-déc	Exam Electif 3
14-déc	Exam d'Alassane, définition de la forme que prendrait le tableau issu de l'extraction de la base de données de l'INRA.
18-déc	Mi-projet : point sur ce qu'on a déjà fait avec le client, mise en forme des derniers objectifs
24-déc	Vacances de Noel : Codage, mise en forme du Jalon 3, recherche de contacts
06-janv	Deadline pour rendre le rapport et le code du projet en tant que projet ML

Nous avons décidé qu'à partir de la semaine où nous commençons le codage nous travaillerons sur des objectifs semaine à semaine, chaque semaine étant terminée par un debrief et une remise en forme des objectifs et éventuellement du modèle.
Pour les deux dernières semaines nous comptons réfléchir aux différentes ouvertures possibles et préparer la soutenance finale.

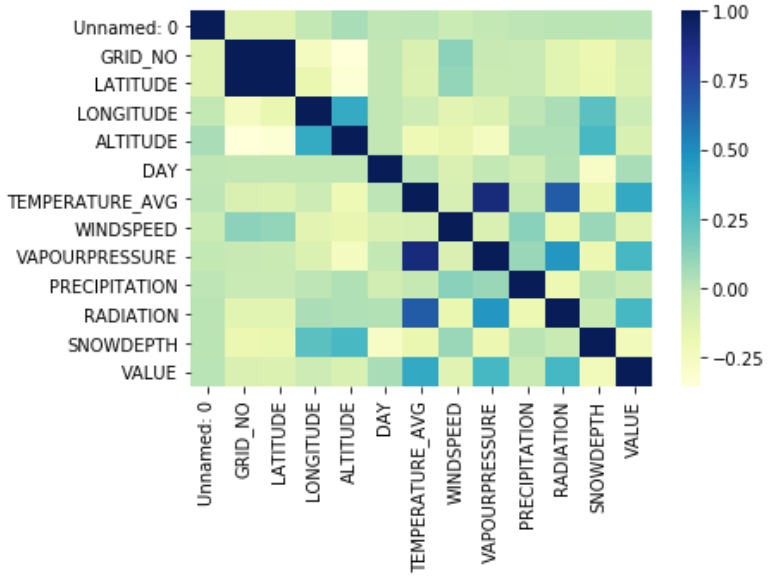
5. Livrable

Le livrable se compose d'un code pour déterminer l'indice de végétation en fonction de données météorologiques, d'une liste de contacts potentiels pour aider sur le sujet et d'un tableau représentant le format final théorique de l'extraction issue de la base de données de l'INRA

a) Code et explications

Les codes se trouvent en annexe. Chacun des codes est accompagné d'explications directement sur le fichier python. Chaque code peut être lancé via Pyzo ou Spyder. Le code « Tracé de l'indice de végétation permet de tracer l'indice de végétation pour un grid sur une année donnée, ce qui permet d'obtenir les courbes dans « Démarche » et « Limites ». Les fichiers « linear regression », « shallow neural network », « deep neural network (one hidden layer) », « deep neural network (two hidden layers) », correspondent chacun à une méthode spécifique de Machine Learning pour prédire les valeurs de l'indice de végétation. Nous avons un fichier « data.csv » correspondant à la data déjà traitée que nous ne pouvons mettre en annexe du fait de son volume (mais qui se trouve dans le dossier compressé rendu avec le rapport).
<u>Méthodologie</u>

Dans un premier temps, nous avions deux tables de données distinctes, une contenant les données météorologiques et une seconde contenant les indices de végétation. La base de données météorologiques était sur une base journalière tandis que celle contenant les indices de végétations était sur une base temporelle de 10 jours. Nous avons donc décidé de concaténer les données météorologiques en prenant la moyenne de chacun des paramètres sur les 10 jours précédents en tant que valeur pour le jour où l'indice de végétation est mesuré (exemple : si l'on a un indice de végétation le 21 janvier, on prend pour température moyenne du 21 janvier la moyenne de la température moyenne du 11 janvier au 21 janvier). Ainsi, on a obtenu une seule et unique table.
Pour le paramètre 'SNOWDEPTH' (épaisseur de neige), il y a des valeurs manquantes que nous avons remplacé par la médiane de celui-ci (soit approximativement 0).
Nous avons ensuite essayé de voir quel paramètre était corrélé avec l'indice de végétation à travers une heatmap des paramètres (corrélation de chacun des indices avec un autre sur toute la data) :



Heatmap

Les coefficients les plus corrélés avec l'indice de végétation (ici le paramètre VALUE) sont la radiation journalière, la pression de l'air et la température moyenne (RADIATION, TEMPERATURE_AVG, VAPOUR_PRESSURE). Ces trois données sont des données constamment utilisées en agrométéorologie, ce résultat est donc logique. Ce qui est plus étonnant est la non corrélation avec les précipitations qui est aussi un paramètre essentiel en agrométéorologie. Cela peut être lié au fait que nous ne considérons qu'une période de 10j, qui pourrait être trop courte pour observer une influence.
Nous avons ensuite subdivisé pour chacun des modèles le dataset en un training set pour entraîner le modèle (valeurs pour les années de 2008 à 2014) et un test set pour valider le modèle (valeurs pour l'année 2015) et voir sa précision.
Enfin, nous avons normalisé chacune des données (c'est-à-dire que chacun des paramètres prennent uniquement des valeurs entre 0 et 1).

Résultats

Nous avons testé notre précision (nous avons appelé précision (1-coût)*100) pour différents jeux de paramètres sur le modèle de Régression Linéaire pour avoir une idée de la meilleure combinaison de paramètres :

Ensemble des paramètres : accuracy =86,69%.

‘TEMPERATURE_AVG’, ‘RADIATION’ : accuracy = 86,46%

Nous avons testé plusieurs autres solutions comme :

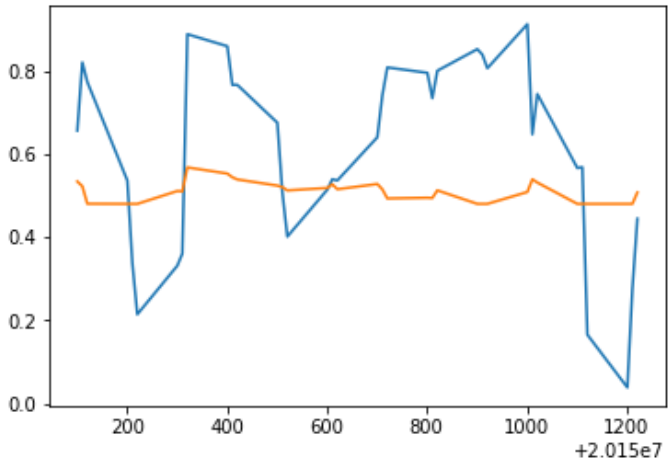
'GRID_NO', 'DAY', 'TEMPERATURE_AVG','WINDSPEED', 'VAPOURPRESSURE', 'PRECIPITATION', 'RADIATION' : accuracy = 84,32%

'GRID_NO', 'DAY', 'TEMPERATURE_AVG','WINDSPEED', 'PRECIPITATION', 'RADIATION' : accuracy = 86,25%

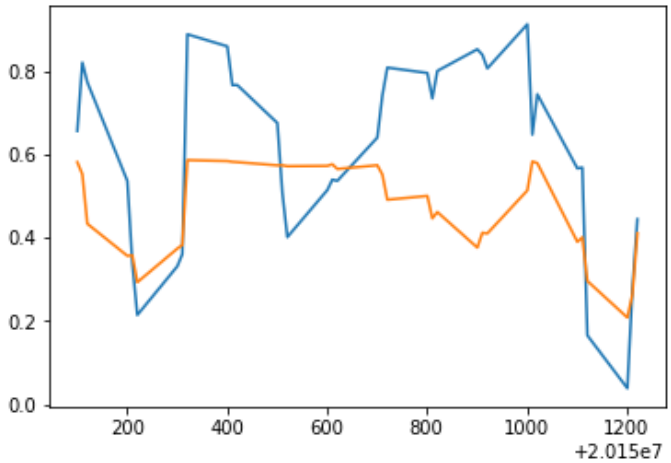
'GRID_NO', 'LATITUDE', 'LONGITUDE', 'ALTITUDE', 'DAY', 'TEMPERATURE_AVG','WINDSPEED', 'VAPOURPRESSURE', 'PRECIPITATION', 'RADIATION' : accuracy = 86,15%

'GRID_NO', 'DAY', 'TEMPERATURE_AVG', 'PRECIPITATION', 'RADIATION' : accuracy = 86,48%

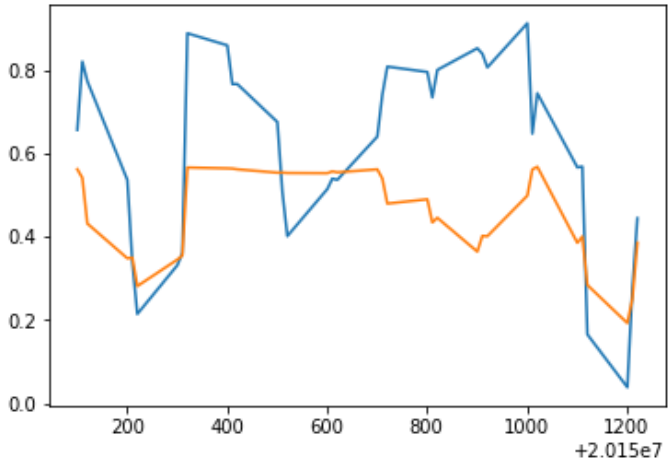
Nous obtenons les meilleurs résultats en gardant l’ensemble des paramètres mêmes si on obtient une précision assez proche en ne gardant que les paramètres fortement corrélés avec l’indice de végétation (cités ci-dessus). Nous avons aussi essayé de retirer le paramètre VAPOUR_PRESSURE qui est très fortement corrélé avec TEMPERATURE_AVG et qui pourrait donc être perçu comme redondant. Cependant, nous avons obtenu une légèrement meilleure précision en le conservant. Pour visualiser la précision de lu modèle entraîné pour les neural networks, voici leur performance sur le Grid n°93087 :



Shallow neural network (no hidden layer)



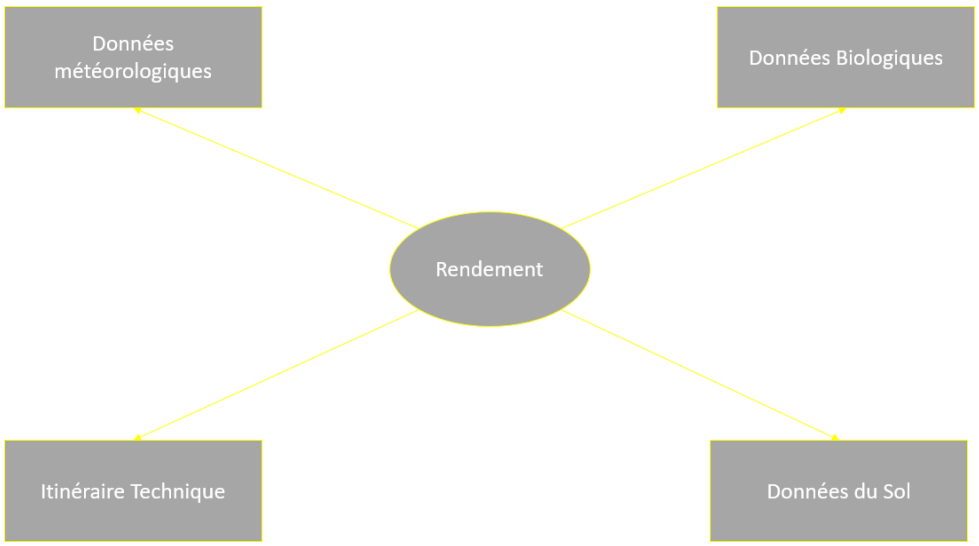
Neural network (one hidden layer)



Neural Network (two hidden layers)

On voit bien que le réseau de neurones sans couches-cachées est bien moins performant que les réseaux neuronaux avec des couches cachées. Le fait que l'on ne parvienne pas à prévoir les pics montre bien que nous n'avons pas l'ensemble des paramètres à notre disposition.

b) Mapping des paramètres



Mapping global

Ci-après, une présentation de chacun des paramètres dans les sous-catégories globales présentées ici.

Légende

- Donnée que l'on a déjà ou présentes dans la base de données Aster-ix
- Données que l'on pense pouvoir trouver
- Données que l'on ne pense pas trouver mais dont on peut se passer
- Données que l'on ne pense pas pouvoir trouver

Données Météorologiques

- Vitesse du vent
- Précipitations
- Température
- Eléments exceptionnels (neige, grêle)
- Rayonnement solaire global (radiations)
- Evapo-transpiration : Apparaît dans le bilan énergétique donné par la formule de Penman:
  Rayonnement global = flux de chaleur dans le sol + evapotranspiration + chaleur sensible.
  C'est un paramètre important mais qui peut probablement être traduit via les radiations.

Données Biologiques

- Espèce, variété
- Résistance aux maladies
- Présence de parasites
- Mélange de cultures

Remarque : La résistance aux maladies n'est pas forcément un input nécessaire, il se traduira à travers un plus grand ou non rendement dans les données pour la variété donnée. De même, la présence de parasites

Itinéraire technique

- Epandage (utilisation de pesticides)
- Fertilisation (par exemple mettre du fumier)
- Drainage (Aération du sol pour favoriser l'écoulement)
- Fréquence d'irrigation
- Types de machines utilisées
- Ecart de temps entre les différentes étapes de l'itinéraire technique
- Cultures précédentes : typiquement, une plantation hivernale de légumineux permet d'apporter 75% de l'azote présent dans le sol au début de la deuxième.

Données liées au sol

- Présence d'ions (cycles de fermeture (K,N,C,Ca,P,Mg))
- Humidité
- Ruissèlement
- Présence de nappes phréatiques
- Drainage (retirement des minéraux, régulation de l'humidité du sol)

c) Liste de contacts

INRA

- Amandine Durpoix : amandine.durpoix@inra.fr (ingénieure dans la gestion de la base de données Aster-ix). Nous sommes actuellement en contact pour l'extraction de la base de données de l'INRA.
- Fabienne Barataud : Directice du projet Aster-ix : fabienne.barataud@inra.fr
- Margot Tysebaert : Ingénieur chez INRA Transfert. Contactée sur LinkedIn (pas de réponse). Connaissance d'une amie de Michaël Medioni (oriane@frenchbureau.com qui a fait AgroParisTech).

Agro-paristech

- Christine Martin : christine.martin@agroparistech.fr (contacté via Céline Hudelot)
- Marc dufumier : marc.dufumier@agroparistech.fr (contacté via Pascal Da Costa, professeur d'economie et de développement durable à centraleSupélec).
- Safia Mediene : safia.mediene@agroparistech.fr (contactée via Oriane Chu oriane@frenchbureau.com) experte pour tout ce qui est des paramètres influant sur le rendement (hors sol).
- Claire Chenu : claire.chenu@agroparistech.fr experte pour tout ce qui est influence de la composition du sol sur les rendements. Contactée via Oriane Chu oriane@frenchbureau.com
- http://siafee.agroparistech.fr/cv-theque?trie=service ensemble des contacts du département siafee d'AgroParisTech regroupant les spécialistes de l'agronomie.

Latitudes

- Vincent Liagre : vincent.liagre@eleven-strategy.com. Expert de la communauté Latitudes autour du Machine Learning et du Deep Learning. Très réactif, on a pu faire un appel téléphonique avec lui.
- Oriane Chenu oriane@frenchbureau.com : nous sommes actuellement en contact avec elle, c'est elle qui nous a donné la quasi-totalité des contacts à AgroParisTech.

CentraleSupélec :

- Jean-Guillaume Messmer : jgmessmer@gmail.com. A travaillé sur les circuits-courts lors de son stage de fin d'études. Les rapports sont disponibles sur son LinkedIn. Contacté via Céline Hudelot.

d) Extraction théorique de la base de données de l'INRA

Voici la proposition que nous avons fait à l'INRA pour la forme de que l'extraction de la base de données prendra.

	Quantité récoltée	Parcelle	Latitude Est	Latitude Nord	Surface	Variété	Récolte précédente	Date de labour	Date de récolte	Opération 1	Opération 2	Opération 3	Opération 4	Opération 5
Récolte 1														
Récolte 2														
Récolte 3														
Récolte 4														
Récolte 5														
Récolte 6														
Récolte 7														
Récolte 8														
Récolte 9														
Récolte 10														
Récolte 11														

Les récoltes seraient triées par parcelle et par année. La quantité récoltée est ce qui nous intéresse, c'est l'output de notre algorithme. Les données Latitudes Est et Nord permettront de localiser la parcelle pour pouvoir coupler cette extraction avec la base de données météorologiques (Agri4Cast). La surface permettra d'avoir le rendement (en divisant quantité récoltée par surface. La variété correspond à l'espèce récoltée. La date de labour permet de trier les récoltes temporellement et nous donne le début de l'itinéraire technique. Les opérations correspondent à l'ensemble des opérations qui influent sur le rendement. Ces colonnes seront divisées en plusieurs données qu'il reste à définir avec l'INRA. En effet, si nous l'on considère par exemple la fertilisation pour l'opération 1, on subdiviserait en 3 données différentes comme la date, l'intrant utilisé et la quantité utilisée. Ces opérations restent à être définies avec l'INRA et nous sommes toujours en attente de réponse de leur part. Nous leur avons par ailleurs demandé une extraction préliminaire pour pouvoir nous recentrer sur quelques espèce(s)/variété(s) précise. :

	Date de récolte	Parcelle	Produit	Variété
Récolte 1				
Récolte 2				
Récolte 3				
Récolte 4				
Récolte 5				
Récolte 6				

Cette extraction nous permettra de déterminer quelle produits (espèces) ou variétés sont les plus représentées et ainsi centrer notre travail sur celles-ci dans un premier temps.

6. Ouvertures possibles

Pour la suite du projet, il reste plusieurs points à faire notamment autour de l'adaptation de l'algorithme à l'extraction de la base de données de l'INRA :

- Il faudra tout d'abord formater les données issues de l'INRA pour en faire des données traitables pour l'algorithme. En effet, le type d'intrant par exemple devra être traduit sous forme numérique pour pouvoir appliquer la régression linaire.
- Il faudra fusionner les deux tables d'Agri4Cast et de l'extraction d'Aster-ix pour pouvoir considérer l'ensemble des paramètres.
- Il faudra aussi envisager de contacter des experts en dehors de la France car visiblement les réponses sont très dures à obtenir dans le monde agricole. Peut-être que ce sera différent dans d'autres pays.

La suite du projet pourra ensuite être faite à travers plusieurs ouvertures possibles :

- Travailler sur le Front-End de cet algorithme pour pouvoir intégrer la fonctionnalité à la WebApp de Mooveat.

- Travailler sur un des deux autres algorithmes proposés initialement : détermination du prix de vente optimal ou détermination de la demande disponible pour un point de vente donné. Cela reviendrait à refaire le même travail fait pendant ce semestre mais dans un autre domaine.

7. Limites et problèmes rencontrés

a) Problèmes rencontrés :

Le premier problème que nous avons rencontré est celui de la définition et de la compréhension du sujet. Le sujet étant très vaste et peu cadré au départ nous avons perdu beaucoup de temps à nous décider sur ce que nous allions viser comme objectifs ainsi que sur la compréhension du sujet. Notre mentor de Latitudes nous a beaucoup aidé pour sur ce point en nous faisant nous poser les bonnes questions et poser les bonnes questions au client.
Le second problème rencontré est celui de la recherche de données. En effet, pour parvenir à nos fins nous avions besoin d'une base de données de grande granularité dans le but d'entrainer notre modèle. Or, une telle base de données n'existe pas au niveau national, il n'y a que des données moyennées au niveau des régions et regroupées par espèce (comme sur le site API-agro par exemple https://plateforme.api-agro.fr/pages/plateforme/). Le client cherchant à obtenir des rendements précis dépendant de la variété proposée et de la parcelle, nous ne pouvions pas nous permettre de nous arrêter à ces bases de données et devions aller chercher plus loin. C'est ainsi que nous avons fini par trouver la base de données Aster-ix. Celle-ci présente précisément la granularité dont nous avons besoin et tout les paramètres que pourrait prendre en compte notre algorithme (sauf les données météorologiques que nous avions déjà). Le problème de celle-ci est qu'elle n'est pas encore en open source car le projet s'étend jusqu'en 2020, il nous a donc fallu entrer en contact avec eux pour expliquer précisément ce dont nous avons besoin, ce qui nous ralenti dans notre travail.

On en vient au troisième et dernier problème récurrent que nous avons rencontré : la non joignabilité générale des personnes dans le monde agricole. Malgré les nombreux contacts que nous avons pu obtenir nous n'avons reçu réponse (malgré les relances) pour ce qui est de comprendre les différents paramètres qui influent sur le rendement. Cette partie est importante pour nous car elle nous permettrait d'aller beaucoup plus vite dans l'entraînement de notre modèle car nous pourrions remodeler/éliminer des paramètres en fonction de ce que nous aurions appris pour pouvoir augmenter l'accuracy de notre algorithme. Par ailleurs, nous aurions aimé pouvoir savoir si la démarche que nous prenions vis-à-vis l'indice de végétation était viable ou non. Même si nous avions l'intuition que cela pouvait fonctionner il y a plusieurs points où nous doutions quelque peu et nous n'avons pas pu lever ses doutes.

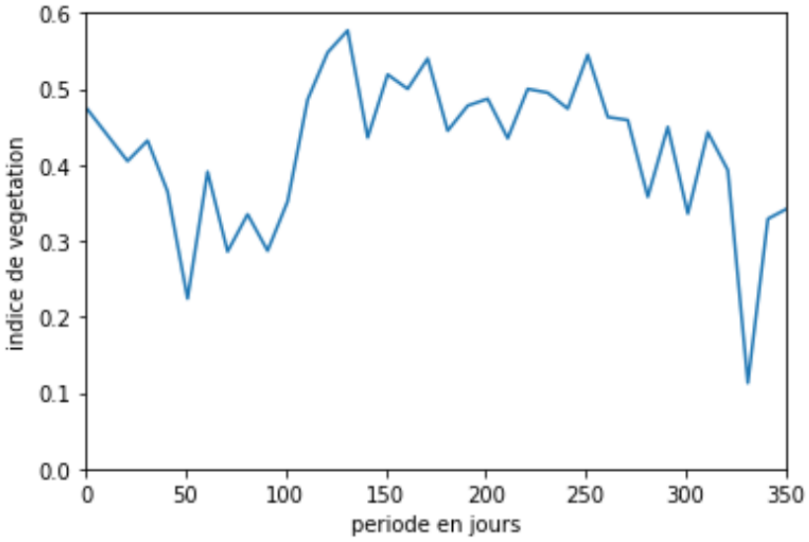
b) Limites de la démarche et du livrable

Le premier point noir de notre démarche est qu'elle repose entièrement sur notre capacité à obtenir une extraction de la base de données d'Aster-ix. Nous sommes

quasiment sûr que ce sera possible mais nous ne savons pas le temps qu'il faudra pour obtenir cette extraction ce qui nous limite dans notre vitesse d'exécution. Par ailleurs, si jamais nous n'avons pas compris réellement comment la base de données s'organise nous nous exposons au risque de ne pas avoir les données nécessaires (par exemple s'il y a trop de changements de variété, nous n'obtiendrions pas la granularité nécessaire pour notre algorithme et nous devrions regrouper plusieurs variétés entre elles pour créer des sous-groupes).

La deuxième limite est liée au fait que notre algorithme travaillant sur l'indice de végétation ne puisse potentiellement pas être facilement adaptable à l'algorithme pour calculer les rendements. En effet, il est possible que l'indice de végétation se révèle très peu corrélé avec le rendement ce qui fausserait notre démarche. Ainsi, plusieurs points nous inquiètent par rapport à cela. Tout d'abord, il y a plusieurs pics de variations de l'indice de végétation sur une année dont nous ne comprenons pas l'origine. L'idéal pour nous aurait été qu'il y ait un seul pic (ou éventuellement 2 pour chaque année s'il y a une récolte et une culture de rotation avec des légumineux pour apporter de l'azote au sol). Cependant, nous retrouvons bien une chute brutale pour chaque année qui pourrait correspondre à une récolte. Le deuxième problème est d'assimiler chaque récolte à une espèce, ce que nous ne pouvons pas faire pour l'instant. Enfin, les grids étant de 25km/25km, il est possible de se trouver avec plusieurs parcelles différentes sur le même Grid. Ceux-ci correspondent principalement à de la culture intensive et donc généralement deux parcelles adjacentes on la même culture donc on peut espérer que les résultats ne seront que peux faussés par la présence de deux parcelles avec des cultures différentes sur certains grids.

Par ailleurs, nous avons finalement appliqué un algorithme qui prévoit l'indice de végétation en fonction des paramètres moyennés sur les 10 jours précédent contrairement à ce que nous avions prévu initialement. Cela est lié au fait que nous n'avons pas pu avoir d'information sur le fonctionnement global de l'agriculture.
En effet, si on reprend le tracé des indices de végétation :



Il est difficile pour nous de comprendre pourquoi est-ce qu'il y a autant de variations d'indice de végétation. Même si nous pouvons faire des hypothèses, nous n'avons rien pour les confirmer. Par ailleurs, il nous semble particulièrement étonnant qu'après le pic

en décembre, il y ait une remontée immédiate d'indice de végétation, cela signifierait qu'il y a immédiatement une repousse (par exemple de mauvaises herbes) ou une plantation de culture hivernale (comme des légumineux) en l'espace de 10 jours. Mais nous ne pouvons savoir si c'est normal ou non car nous n'avons toujours pas de contact dans le domaine.

Annexes

Codes

Tracé de l'indice de végétation

```
#Ce code permet de tracer l'indice de végétation sur une année donnée sur un GRID donné (25km/25km)

##importation de l'ensemble des bibliothèques nécessaires
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

## Mise en forme de la data
meteo_data = pd.read_csv('data.csv')
print(max(meteo_data['GRID_NO']), min(meteo_data['GRID_NO'])) # max et min des valeurs que peut prendre le numéro de GRID
meteo_data0 = meteo_data.loc[meteo_data['GRID_NO']==93088] #valeur à changer pour modifier le GRID dont on trace l'indice de végétation

## Création de la variable à tracer en abscisse (c'est à dire jour de 0 à 3)
def year(day):
return day//10000 # les jours sont mis sous la forme 20080301 pour 01/03/2008

meteo_data1 = meteo_data0
meteo_data1['YEAR'] = meteo_data1['DAY'].apply(year)
meteo_data1=meteo_data1.loc[meteo_data1['YEAR']==2011] # valeur à changer pour changer l'année du tracé
meteo_data=meteo_data1
meteo_data
#conversion de la date en jours
meteo_data['DAYCONVERT'] = meteo_data0['DAY']%100
meteo_data['MONTH']= (meteo_data['DAY']-meteo_data['DAY']%100 )//100
meteo_data['MONTH']= ((meteo_data['MONTH']%100)-1)*30
meteo_data['FINALE']= meteo_data['MONTH']+meteo_data['DAYCONVERT']
meteo_data['FINALE']

## Tracé des points bruts non-reliés
plt.figure(1)
axes = plt.gca()
axes.set_xlim([0,365])
axes.set_ylim([0,0.6])
plt.plot(np.array(meteo_data['FINALE']),np.array(meteo_data['VALUE']),'bo')
plt.xlabel('periode en jours')
plt.ylabel('indice de vegetation')

##Tracé de la courbe d'indice de végétation sur une année
plt.figure()
```

```
axes = plt.gca()
axes.set_xlim([0,365])
axes.set_ylim([0,1])
plt.plot(np.array(meteo_data['FINALE']),np.array(meteo_data['VALUE']))
plt.xlabel('periode en jours')
plt.ylabel('indice de vegetation')
```

Linear Regression

```
#Part 1 : importing the necessary models
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn import cross_validation
from sklearn import preprocessing,model_selection,tree,ensemble
import seaborn as sns
#generating and plotting the heatmap
meteo_data = pd.read_csv('data.csv')
meteo_data0 = meteo_data.loc[meteo_data['GRID_NO']==93087]
corr = meteo_data.corr()
print(corr)
sns.heatmap(corr,cmap = 'YlGnBu')
#
meteo_data0.ix[:,'SNOWDEPTH'] =
meteo_data0.SNOWDEPTH.fillna(meteo_data0.SNOWDEPTH.median())
train_data = meteo_data0[['Unnamed: 0','GRID_NO', 'LATITUDE', 'LONGITUDE', 'ALTITUDE',
'DAY'
, 'TEMPERATURE_AVG','WINDSPEED', 'VAPOURPRESSURE', 'PRECIPITATION',
'RADIATION',
'SNOWDEPTH']]
train_target = meteo_data0[['VALUE']]
print(meteo_data0)
def classify(trainSet, trainLabels, testSet):
lr = LinearRegression()
predictedLabels = np.zeros(testSet.shape[0])
lr.fit(trainSet,trainLabels)
predictedLabels=lr.predict(testSet)
return predictedLabels
#using cross validation to calculate accuracy for linear regression
X = np.array(train_data)
y = np.array(train_target)
# Initialize cross validation
kf = cross_validation.KFold(X.shape[0], n_folds=10)

totalCost = 0 # Variable that will store the correctly predicted intances

for trainIndex, testIndex in kf:
trainSet = X[trainIndex]
testSet = X[testIndex]
trainLabels = y[trainIndex]
testLabels = y[testIndex]
```

```
predictedLabels = classify(trainSet, trainLabels, testSet)

cost = 0
for i in range(testSet.shape[0]):
cost += (predictedLabels[i]-testLabels[i])**2
print ('cost: ' + str(np.sqrt(float(cost)/(testLabels.size))))
totalCost += np.sqrt(float(cost)/(testLabels.size))
print ('Total Cost: ' + str(totalCost/10))
print('Total Accuracy:' + str(1-totalCost/10))
```

Shallow Neural Network

```
#part 1 : importing the necessary models
from keras.callbacks import ModelCheckpoint
from sklearn import cross_validation
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from matplotlib import pyplot as plt
import seaborn as sb
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import warnings
import seaborn as sns
from sklearn import preprocessing
from keras.layers import Dropout
warnings.filterwarnings('ignore')
warnings.filterwarnings('ignore', category=DeprecationWarning)
#Part 2 :getting the data
meteo_data = pd.read_csv('data.csv')
meteo_data['SNOWDEPTH'] =
meteo_data.SNOWDEPTH.fillna(meteo_data.SNOWDEPTH.median())
meteo_data=meteo_data.sort_values(['DAY','GRID_NO'])
def year(day):
return day//10000
meteo_data1 = meteo_data
meteo_data1['YEAR'] = meteo_data1['DAY']
meteo_data1['YEAR'] = meteo_data1['DAY'].apply(year)
train_target=meteo_data1.loc[meteo_data1['YEAR']<2015].ix[:,'VALUE']
test_target =meteo_data1.loc[meteo_data1['YEAR']==2015].ix[:,'VALUE']
meteo_data1=meteo_data1[['Unnamed: 0','GRID_NO', 'LATITUDE', 'LONGITUDE', 'ALTITUDE',
'DAY'
, 'TEMPERATURE_AVG','WINDSPEED', 'VAPOURPRESSURE', 'PRECIPITATION',
'RADIATION',
'SNOWDEPTH','YEAR']]

#Part 3 : separating the trainset and the testset
train = meteo_data1.loc[meteo_data1['YEAR']<2015]
test = meteo_data1.loc[meteo_data1['YEAR']==2015]
```

```
#Part 4 : scaling the training and the test data
x = train.values
min_max_scaler = preprocessing.MinMaxScaler()
x=min_max_scaler.fit_transform(train)
train = pd.DataFrame(x)

y = test.values
min_max_scaler = preprocessing.MinMaxScaler()
y=min_max_scaler.fit_transform(test)
test = pd.DataFrame(y)

#Part 5 : Build the neural network
NN_model = Sequential()

#Part 6 : add The Input Layer :
NN_model.add(Dense(train.shape[1], kernel_initializer='random_uniform',input_dim = train.shape[1],
activation='relu'))

#Part 7 : Use Dropout to avoid overfitting
NN_model.add(Dropout(0.2, input_shape=(train.shape[1],)))

#Part 8 : add The Hidden Layers :
#NN_model.add(Dense(train.shape[1]//2, kernel_initializer='normal',activation='relu'))
#NN_model.add(Dense(train.shape[1]//4, kernel_initializer='normal',activation='relu'))

#Part 9 : add The Output Layer :
NN_model.add(Dense(1, kernel_initializer='normal',activation='linear'))

#Part 10 : Compile the network :
NN_model.compile(loss='mean_squared_error', optimizer='adam',metrics=['accuracy'])
NN_model.summary()

#Part 11 : Define a checkpoint callback
checkpoint_name = 'Weights-{epoch:03d}--{val_loss:.5f}.hdf5'
checkpoint = ModelCheckpoint(checkpoint_name, monitor='val_loss', verbose = 1, save_best_only =
True, mode ='auto')
callbacks_list = [checkpoint]

#Part 12 : Train the neural network
NN_model.fit(train, train_target, epochs=5, batch_size=32, validation_split = 0.2,
callbacks=callbacks_list)

#Part 13 : Test the model and calculate accuracy
predictedLabels = NN_model.predict(test)
cost = 0
for i in range(test_target.shape[0]):
cost += (float(predictedLabels[i])-test_target.values[i])**2
cost = np.sqrt(float(cost)/(test_target.size))
print ('cost: ' + str(cost))
accuracy = 1-cost
print(accuracy)

#Part 14 : Plot the real values and the predicted values for GRID_NO = 93087
test['VALUE']=pd.DataFrame(test_target.values)
```

```
test['PREDICTEDVALUE']=pd.DataFrame(predictedLabels)
def year(day):
return day//10000
test_one_grid=test.loc[meteo_data1['GRID_NO']==93087]
plt.plot(np.array(test_one_grid['DAY']),np.array(test_one_grid['VALUE']))
plt.plot(np.array(test_one_grid['DAY']),np.array(test_one_grid['PREDICTEDVALUE']))
```

Neural Network (One hidden layer)

```
#part 1 : importing the necessary models
from keras.callbacks import ModelCheckpoint
from sklearn import cross_validation
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from matplotlib import pyplot as plt
import seaborn as sb
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import warnings
import seaborn as sns
from sklearn import preprocessing
from keras.layers import Dropout
warnings.filterwarnings('ignore')
warnings.filterwarnings('ignore', category=DeprecationWarning)
#Part 2 :getting the data
meteo_data = pd.read_csv('data.csv')
meteo_data['SNOWDEPTH'] =
meteo_data.SNOWDEPTH.fillna(meteo_data.SNOWDEPTH.median())
meteo_data=meteo_data.sort_values(['DAY','GRID_NO'])
def year(day):
return day//10000
meteo_data1 = meteo_data
meteo_data1['YEAR'] = meteo_data1['DAY']
meteo_data1['YEAR'] = meteo_data1['DAY'].apply(year)
train_target=meteo_data1.loc[meteo_data1['YEAR']<2015].ix[:,'VALUE']
test_target =meteo_data1.loc[meteo_data1['YEAR']==2015].ix[:,'VALUE']
meteo_data1=meteo_data1[['Unnamed: 0','GRID_NO', 'LATITUDE', 'LONGITUDE', 'ALTITUDE',
'DAY'
, 'TEMPERATURE_AVG','WINDSPEED', 'VAPOURPRESSURE', 'PRECIPITATION',
'RADIATION',
'SNOWDEPTH','YEAR']]

#Part 3 : separating the trainset and the testset
train = meteo_data1.loc[meteo_data1['YEAR']<2015]
test = meteo_data1.loc[meteo_data1['YEAR']==2015]

#Part 4 : scaling the training and the test data
x = train.values
```

```
min_max_scaler = preprocessing.MinMaxScaler()
x=min_max_scaler.fit_transform(train)
train = pd.DataFrame(x)

y = test.values
min_max_scaler = preprocessing.MinMaxScaler()
y=min_max_scaler.fit_transform(test)
test = pd.DataFrame(y)

#Part 5 : Build the neural network
NN_model = Sequential()

#Part 6 : add The Input Layer :
NN_model.add(Dense(train.shape[1], kernel_initializer='random_uniform',input_dim = train.shape[1],
activation='relu'))

#Part 7 : Use Dropout to avoid overfitting
NN_model.add(Dropout(0.2, input_shape=(train.shape[1],)))

#Part 8 : add The Hidden Layers :
NN_model.add(Dense(train.shape[1]//2, kernel_initializer='normal',activation='relu'))
NN_model.add(Dense(train.shape[1]//4, kernel_initializer='normal',activation='relu'))

#Part 9 : add The Output Layer :
NN_model.add(Dense(1, kernel_initializer='normal',activation='linear'))

#Part 10 : Compile the network :
NN_model.compile(loss='mean_squared_error', optimizer='adam',metrics=['accuracy'])
NN_model.summary()

#Part 11 : Define a checkpoint callback
checkpoint_name = 'Weights-{epoch:03d}--{val_loss:.5f}.hdf5'
checkpoint = ModelCheckpoint(checkpoint_name, monitor='val_loss', verbose = 1, save_best_only =
True, mode ='auto')
callbacks_list = [checkpoint]

#Part 12 : Train the neural network
NN_model.fit(train, train_target, epochs=5, batch_size=32, validation_split = 0.2,
callbacks=callbacks_list)

#Part 13 : Test the model and calculate accuracy
predictedLabels = NN_model.predict(test)
cost = 0
for i in range(test_target.shape[0]):
cost += (float(predictedLabels[i])-test_target.values[i])**2
cost = np.sqrt(float(cost)/(test_target.size))
print ('cost: ' + str(cost))
accuracy = 1-cost
print(accuracy)

#Part 14 : Plot the real values and the predicted values for GRID_NO = 93087
train = meteo_data1.loc[meteo_data1['YEAR']<2015]
test = meteo_data1.loc[meteo_data1['YEAR']==2015]
test['VALUE']=pd.DataFrame(test_target.values)
test['PREDICTEDVALUE']=pd.DataFrame(predictedLabels)
```

```
def year(day):
return day//10000
test_one_grid=test.loc[meteo_data1['GRID_NO']==93087]
plt.plot(np.array(test_one_grid['DAY']),np.array(test_one_grid['VALUE']))
plt.plot(np.array(test_one_grid['DAY']),np.array(test_one_grid['PREDICTEDVALUE']))
```

Neural Nework (two layers)

```
#part 1 : importing the necessary models
from keras.callbacks import ModelCheckpoint
from sklearn import cross_validation
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from matplotlib import pyplot as plt
import seaborn as sb
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import warnings
import seaborn as sns
from sklearn import preprocessing
from keras.layers import Dropout
warnings.filterwarnings('ignore')
warnings.filterwarnings('ignore', category=DeprecationWarning)
#Part 2 :getting the data
meteo_data = pd.read_csv('data.csv')
meteo_data['SNOWDEPTH'] =
meteo_data.SNOWDEPTH.fillna(meteo_data.SNOWDEPTH.median())
meteo_data=meteo_data.sort_values(['DAY','GRID_NO'])
def year(day):
return day//10000
meteo_data1 = meteo_data
meteo_data1['YEAR'] = meteo_data1['DAY']
meteo_data1['YEAR'] = meteo_data1['DAY'].apply(year)
train_target=meteo_data1.loc[meteo_data1['YEAR']<2015].ix[:,'VALUE']
test_target =meteo_data1.loc[meteo_data1['YEAR']==2015].ix[:,'VALUE']
meteo_data1=meteo_data1[['Unnamed: 0','GRID_NO', 'LATITUDE', 'LONGITUDE', 'ALTITUDE',
'DAY'
, 'TEMPERATURE_AVG','WINDSPEED', 'VAPOURPRESSURE', 'PRECIPITATION',
'RADIATION',
'SNOWDEPTH','YEAR']]

#Part 3 : separating the trainset and the testset
train = meteo_data1.loc[meteo_data1['YEAR']<2015]
test = meteo_data1.loc[meteo_data1['YEAR']==2015]

#Part 4 : scaling the training and the test data
x = train.values
min_max_scaler = preprocessing.MinMaxScaler()
x=min_max_scaler.fit_transform(train)
train = pd.DataFrame(x)
```

```
y = test.values
min_max_scaler = preprocessing.MinMaxScaler()
y=min_max_scaler.fit_transform(test)
test = pd.DataFrame(y)

#Part 5 : Build the neural network
NN_model = Sequential()

#Part 6 : add The Input Layer :
NN_model.add(Dense(train.shape[1], kernel_initializer='random_uniform',input_dim = train.shape[1],
activation='relu'))

#Part 7 : Use Dropout to avoid overfitting
NN_model.add(Dropout(0.2, input_shape=(train.shape[1],)))

#Part 8 : add The Hidden Layers :
#NN_model.add(Dense(train.shape[1]//2, kernel_initializer='normal',activation='relu'))
NN_model.add(Dense(train.shape[1]//4, kernel_initializer='normal',activation='relu'))

#Part 9 : add The Output Layer :
NN_model.add(Dense(1, kernel_initializer='normal',activation='linear'))

#Part 10 : Compile the network :
NN_model.compile(loss='mean_squared_error', optimizer='adam',metrics=['accuracy'])
NN_model.summary()

#Part 11 : Define a checkpoint callback
checkpoint_name = 'Weights-{epoch:03d}--{val_loss:.5f}.hdf5'
checkpoint = ModelCheckpoint(checkpoint_name, monitor='val_loss', verbose = 1, save_best_only =
True, mode ='auto')
callbacks_list = [checkpoint]

#Part 12 : Train the neural network
NN_model.fit(train, train_target, epochs=5, batch_size=32, validation_split = 0.2,
callbacks=callbacks_list)

#Part 13 : Test the model and calculate accuracy
predictedLabels = NN_model.predict(test)
cost = 0
for i in range(test_target.shape[0]):
cost += (float(predictedLabels[i])-test_target.values[i])**2
cost = np.sqrt(float(cost)/(test_target.size))
print ('cost: ' + str(cost))
accuracy = 1-cost
print(accuracy)

#Part 14 : Plot the real values and the predicted values for GRID_NO = 93087
test['VALUE']=pd.DataFrame(test_target.values)
test['PREDICTEDVALUE']=pd.DataFrame(predictedLabels)
def year(day):
return day//10000
test_one_grid=test.loc[meteo_data1['GRID_NO']==93087]
plt.plot(np.array(test_one_grid['DAY']),np.array(test_one_grid['VALUE']))
plt.plot(np.array(test_one_grid['DAY']),np.array(test_one_grid['PREDICTEDVALUE']
```

Bibliographie

https://www.claas.fr/blueprint/servlet/blob/158506/f3739f59dde5cc434228c6c7890567ac/eu-fr-data.pdf

Liens vers le rapport de la base de données Aster-ix de l'INRA http://www.sad.inra.fr/Toutes-les-actualites/La-ferme-experimentale-de-Mirecourt-lieu-d-echanges-de-savoirs-et-d-experiences

Base de données Gissol, Groupement d'intérêt scientifique, INRA https://www.gissol.fr/

Bases de données de la plateforme API-AGRO https://plateforme.api-agro.fr/pages/plateforme/

Etude INCA3 , Anses, https://www.anses.fr/fr/content/inca-3-evolution-des-habitudes-et-modes-de-consommation-de-nouveaux-enjeux-en-mati%C3%A8re-de

Rapport méthodologique de l'outil Agribalyse de l'Ademe, https://www.ademe.fr/sites/default/files/assets/documents/agribalyse-rapport-methodologique-v1_2.pdf

TEF 2017 (INSEE), Budget des ménages en 2017, https://www.insee.fr/fr/statistiques/2569364?sommaire=2587886

Au menu : confiance et innovation, PwC, https://www.pwc.fr/fr/assets/files/pdf/2018/09/pwc-au-menu-confiance-et-innovation-2018.pdf

Autonomie alimentaire, Utopies, http://www.utopies.com/fr/publications/autonomie-alimentaire-des-villes

Predictive ability of machine learning methods for massive crop yield prediction, Alberto Gonzalez-Sanchez, June 2014, https://www.researchgate.net/publication/263368516_Predictive_ability_of_machine_learning_methods_for_massive_crop_yield_prediction

Base de données Agri4Cast http://agri4cast.jrc.ec.europa.eu/DataPortal/Index.aspx

Données autour de l'agriculture biologique, http://www.agencebio.org