École Centrale de Nantes

STASC: Challenge Data

Wind Power forecasting for the day-ahead energy market par Compagnie Nationale du Rhône

Atys Panier

Info IA

27/01/2023

Table des matières

1 Introduction

CNR est le principal producteur français d'énergie renouvelable exclusive (eau, vent, soleil) et le concessionnaire du fleuve Rhône pour la production hydroélectrique, la navigation fluviale et l'irrigation pour l'utilisation agricole. Ce défi se concentre sur la prévision de la production d'énergie éolienne. CNR possède actuellement environ 50 fermes éoliennes (WF) pour une capacité installée totale de plus de 600 MW. Tous les jours, CNR vend sur le marché de l'énergie sa production d'énergie éolienne pour le lendemain. Afin de vendre la bonne quantité d'énergie, ainsi que pour les exigences légales envers le gestionnaire français du système de transmission (TSO) chargé de la stabilité du réseau électrique, CNR doit savoir à l'avance combien d'énergie les fermes éoliennes produiront le lendemain.

L'objectif de ce défi est de prévoir la production d'énergie de six WF appartenant à CNR. La production de chaque WF sera prédite individuellement, en utilisant les prévisions météorologiques comme entrée. Les prévisions se concentreront sur la production d'énergie à l'avance (prévisions de production horaire de la journée D+1 00h à la journée D+2 00h).

2 Analyse Des Données

2.1 Description des données

Les données fournies sont des variables météorologiques prévues à l'heure, fournies par divers modèles de prévision numérique du temps (NWP). Les modèles NWP sont des modèles météorologiques exploités par plusieurs services nationaux de prévision météorologique. Pour des raisons de confidentialité, le nom des modèles NWP n'apparaîtra pas. Ils seront désignés par les noms génériques NWP1,...,NWPn. Pour chaque ligne de données, nous avons le numéro de la "Wind Farm" correspondante ainsi que la date et l'heure correspondant à la production d'électricité observée. Le format de l'en-tête des fichiers csv pour les variables météorologiques est le suivant : NWPi-HourOfTheRun-DayOfTheRun-Variable.
Les variables influençant les valeurs météorologiques prévues par le NWP sont :
- les Composantes U et V du vent à 100m (ou 10m) de hauteur (m/s)
- la Température de l'air (°C)
- la Couverture nuageuse totale en pourcent abrégée CLCT

2.2 Analyse exploratoire des données

En me consacrant à la "Wind Farm" numéro 1, mon premier objectif était de comprendre du mieux possible les données afin de développer une première stratégie de modélisation.

J'ai d'abord analysé la forme de mes données. La variable "target" est la production qui atteint des valeurs entre 0 et 10.14 unités. Mon Data Frame X_WF1 obtenu par X_WF1 = X[X['WF']=='WF1'] contient 6 239 lignes et 104 colonnes. Il y a deux variables qualitatives (WF et Time) ainsi que 102 variables quantitatives. L'analyse des valeurs manquantes m'a indiqué qu'il y avait beaucoup Nan, avec plusieurs groupes de données réparties selon leur pourcentage de Nan. J'ai donc décidé d'éliminer les colonnes ayant plus de 90% de valeurs manquantes.

En observant la répartition des variables, j'ai pu comprendre qu'elles étaient continues, de loi normale centrée réduite pour U et V, de loi normale d'espérance 295 pour T (variable fournie seulement par NWP1 et NWP3), symétrique pour CLCT (variable fournie seulement par NWP4) qui semble majoritairement valoir son minimum et son maximum.

En créant 4 groupes de variables U, V, T et CLCT (X_WF1[liste_CLCT_columns] étant par exemple mon Data Frame réduit aux variables CLCT), j'ai observé comment était répartie la production par rapport à chaque variable. J'ai aussi pu me rendre compte qu'il y avait une forte corrélation entre les valeurs données par les différents modèles NWPi à la même horaire, le même jour.
Pour une variable comme CLCT, il n'y a aucune corrélation linéaire entre les valeurs espacées de 12h sur 2 jours alors qu'il y a une forte corrélation linéaire pour la température.

Grâce à cette analyse, je me suis familiarisé avec les données fournies par la CNR tout en utilisant des fonctions de *Seaborn* comme 'heatmap', 'pairplot' et 'clustermap'. J'ai aussi remarqué qu'il fallait standardiser les données.

3 Premiers modèles

3.1 Régression Linéaire avec 3 variables

Suite à l'analyse des données, j'ai décidé d'estimer la production de WF1 en utilisant une régression linéaire sur seulement 3 variables :
1) NWP3_00h_D_U,V,T
2) NWP4_00h_D_U,V,CLCT
J'ai obtenu un score de 0,85 et 0,95 avec la métrique de la CNR. J'ai visualisé les différences entre ma production prédite et celle du y_test et, comme je m'en doutais, il fallait trouver une autre approche.

3.2 Test de plusieurs algorithmes de régression

En me référant à la "cheat-sheet" fournie par la documentation de *Scikit-Learn*, j'ai décidé de tester 5 algorithmes de régression sur mes données brutes : 'LinearRegression', 'Lasso', 'ElasticNet', 'Ridge' et 'SVR'.

Pour cela, j'ai d'abord créé une fonction de "preprocessing" à laquelle j'ai soumis mes données avant de tester les 5 modèles précédents. J'ai pu en apprendre davantage sur le principe de "feature engineering" ainsi que sur la création de pipeline. C'est l'algorithme de 'Support Vector Regression' (SVR) qui m'a fourni le meilleur résultat avec un score de 37,8.

Pour vérifier qu'il n'y avait pas de problématiques d'"overfitting" ou d'"underfitting", j'ai choisi de tracer la courbe d'apprentissage utilisant la méthode de cross validation.

3.3 Nouvelles tentatives de régression linéaire

J'ai tout de même essayé d'améliorer le score obtenu par régression linéaire. Pour ce faire, j'ai gardé pour chaque modèle NWP, les variables U, V et T/CLCT (quand c'est possible) à la même heure, le même jour.

Avec seulement 8 variables, le résultat de la régression linéaire était moins bon que celui fait sur l'ensemble du "dataset" puisque j'ai obtenu un score élevé de 87. En utilisant le "selector" 'SelectKBest', et en faisant une "GridSearchCV", j'ai quand même remarqué que ce sont les variables relatives à la vitesse du vent qui avait le plus de poids dans la régression. Cependant, quand on applique le même pipeline à l'ensemble des données, le meilleur estimateur donne un score de 80, avec plus d'importance sur les variables relatives à la température. J'ai aussi remarqué que la courbe d'apprentissage était très révélatrice de la limite du modèle de la régression linéaire puisque la courbe du score de validation et la courbe du score sur les données d'entrainement semble converger vers la même limite.

Ensuite, j'ai utilisé une technique d'apprentissage automatique pour prédire les valeurs manquantes des colonnes 'NWP2_00h_D_V' et 'NWP2_00h_D_U' en utilisant un algorithme de Random Forest, ce qui ne s'est pas montré concluant. J'ai donc finalement choisi de remplir les valeurs manquantes soit par des 0, soit par des calculs de médiane sur chaque colonne.

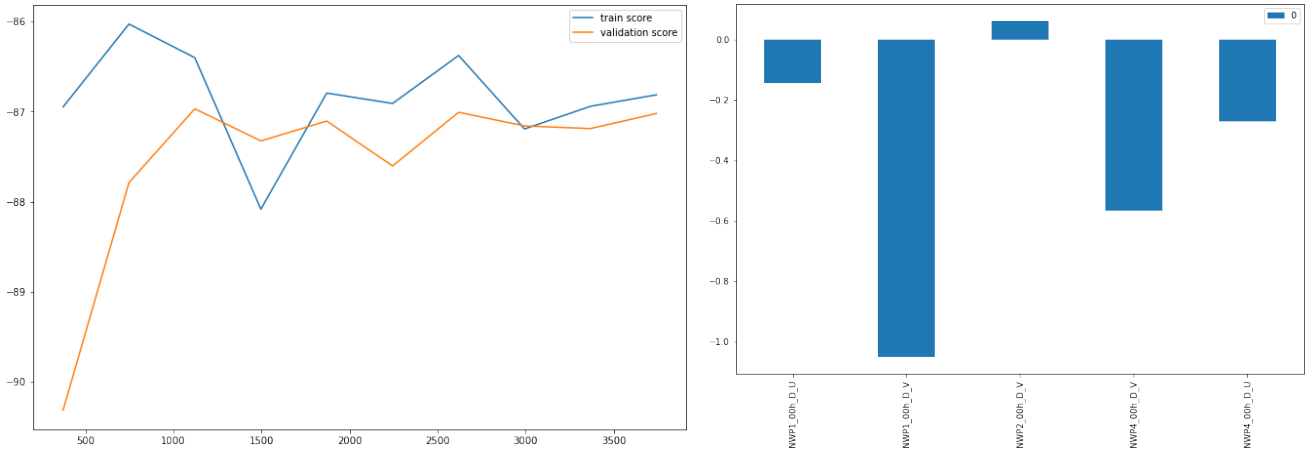


FIGURE 1 – Régression linéaire avec 5 sur 8 variables trouvées par recherche sur grille

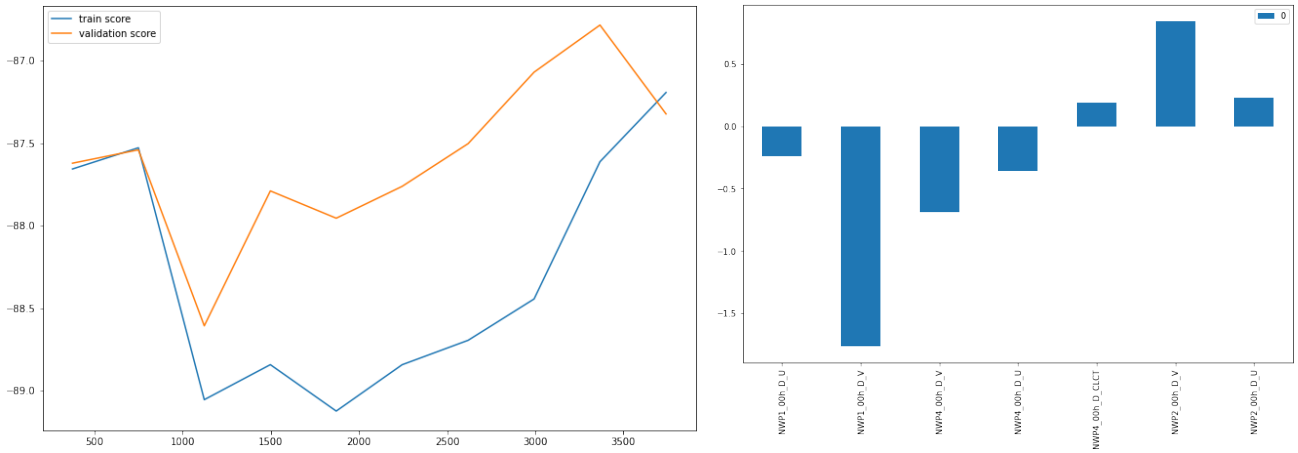


FIGURE 2 – Régression linéaire en ayant complétées les valeurs manquantes par un algo de RF

4 Support Vector Regression

4.1 Explication du script

J'ai utilisé l'algorithme de régression SVR pour prédire ma production. J'ai d'abord défini trois fonctions de prétraitement : 'imputation', 'feature_engineering' et 'preprocessing'. La fonction 'imputation' remplace les valeurs manquantes par la médiane des valeurs de chaque colonne, tandis que la fonction 'feature_engineering' supprime les colonnes qui ont plus de 90% de valeurs manquantes. La fonction 'preprocessing' utilise ces deux fonctions pour nettoyer les données.

Ensuite, la fonction train_test_split de scikitlearn va séparer les données en un ensemble d'entraînement et un ensemble de test. J'utilise ensuite les fonctions de prétraitement pour nettoyer ces deux ensembles de données. J'importe ensuite la classe 'SVR' de scikit-learn pour créer un modèle de régression ainsi que la classe 'StandardScaler' pour normaliser les données. J'utilise également la fonction 'make_pipeline' pour créer un pipeline qui effectue d'abord la normalisation des données, puis l'entraînement du modèle SVR.

J'importe également une fonction personnalisée appelée 'CAPE_CNR_function' pour évaluer les performances du modèle, ainsi que la fonction 'make_scorer' pour créer un score personnalisé basé sur cette fonction. Puis, mon code définit une fonction 'évaluation' qui prend un modèle en entrée, l'entraîne sur les données d'entraînement, effectue des prévisions sur les données de test et affiche un graphique des scores d'entraînement et de validation en fonction de la taille de l'ensemble d'entraînement.

4.2 SVR

Cet algorithme obtient un score de 37.85630767793122. On remarque que la courbe d'entrainement ainsi que la courbe sur les données de validation semble croitre vers un palier. Comme la courbe de validation est nettement en dessous de la courbe sur nos données d'entrainement, on peut desceller du surapprentissage. On va essayer d'améliorer les performances prédictives de notre algorithme.

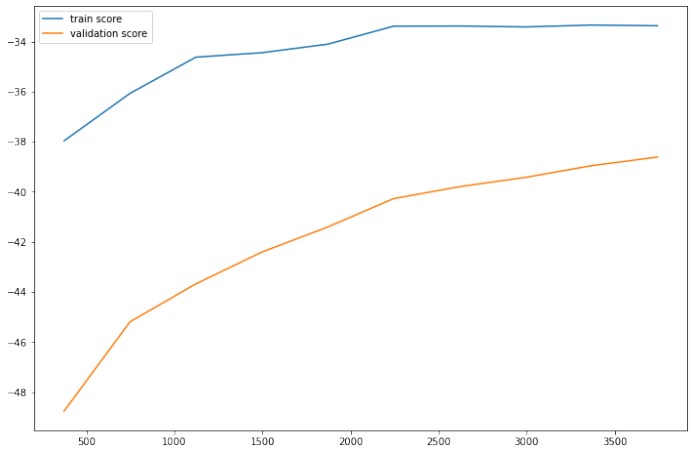


FIGURE 3 – Régression avec un algorithme de SVR

4.3 SVR optimisé par GridSearchCV

C et gamma sont deux des hyperparamètres du modèle de régression SVR de *scikit-learn*.

C représente le paramètre de régularisation. Un petit C permet une régularisation forte, ce qui signifie qu'une solution simple pourrait être préférée pour réduire l'erreur d'entraînement. Au contraire,

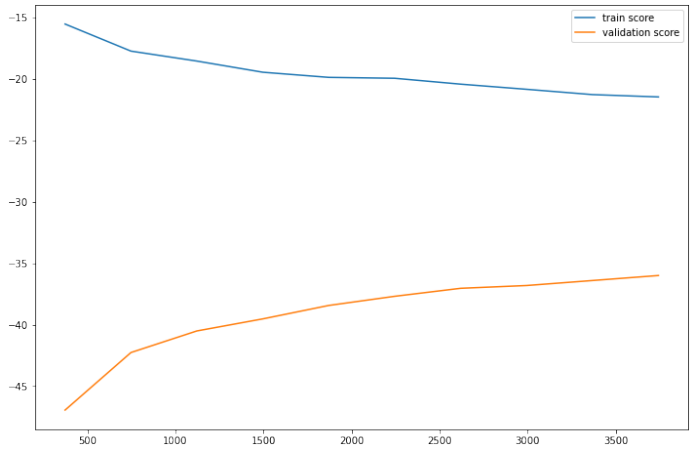


FIGURE 4 – Régression avec le meilleur estimateur (SVR) de notre grille GridSearchCV

un grand C préfère une solution complexe qui peut bien correspondre aux données d'entraînement. Gamma est un hyperparamètre du noyau utilisé dans la méthode de 'support vector machine'. Il définit l'effet de la précision des fonctions noyau. Un grand gamma peut conduire à des fonctions noyau complexes avec des modèles complexes, tandis qu'un gamma petit peut rendre le modèle simple.

Lors de la recherche des hyperparamètres, nous testons différentes valeurs pour C et gamma par GridSearchCV pour trouver les meilleures valeurs qui minimisent l'erreur d'entraînement.

Avec une grille param_grid = 'svr___C' : [0.1, 1, 10], 'svr___gamma' : [0.01, 0.1, 1], on obtient comme meilleurs paramètres : 'svr___C' : 10, 'svr___gamma' : 0.01 et un score de 34.766314992124

4.4 SVR optimisé par RandomizedSearchCV

L'algorithme de recherche aléatoire (Randomized Search) est un algorithme de tuning de modèle de Machine Learning qui consiste à explorer un espace de recherche de manière aléatoire pour trouver les meilleurs hyperparamètres pour un modèle donné comme GridSearchCV.

L'idée ici était d'ajouter une recherche aléatoire itérée, pour ne pas explorer l'espace de recherche en une seule fois, mais plutôt de le faire en plusieurs étapes successives. À chaque étape, l'espace de recherche est réduit par deux et centré sur les meilleurs hyperparamètres trouvés à l'étape précédente.

Cela m'a été utile pour économiser du temps de calcul en ne parcourant pas tout l'espace de recherche. Ainsi, avec une grille param_grid = 'svr___C' : np.logspace(-3, 3, 64), 'svr___gamma' : np.logspace(-3, 3, 64), on obtient comme meilleurs paramètres : 'svr___C' : 200, 'svr___gamma' : 0.06 et un score de 36.88. On remarque que le resultat est proche de celui obtenu par recherche sur grille alors qu'on a exploré une plage plus grande de valeurs pour C et gamma.

5 Création des variables "WindSpeed" et "WindDir"

Comme nous l'indique la CNR, il peut être judicieux de transformer les composantes orthogonale et orthonormale (U et V) du vecteur vitesse afin d'obtenir une nouvelle caractérisation de la vitesse du vent, par WS et WR respectivement la norme et la direction du vecteur vitesse.

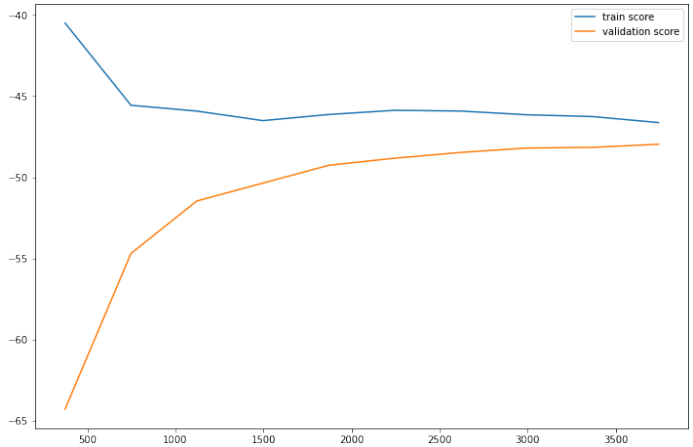
```
#Creation de la variable WindSpeed WS
WS = pd.DataFrame()
dico = dict(zip(liste_V_columns,liste_U_columns))
for key, value in dico.items():
    WS[key] = np.sqrt(df[key]**2+df[value]**2)
liste_WS_columns=[]
for elem in list(WS.columns):
    new_elem = elem.replace("_V","_WS")
    liste_WS_columns.append(new_elem)
WS.columns = liste_WS_columns

#Creation de la variable WindDir WD
WD = pd.DataFrame()
dico = dict(zip(liste_V_columns,liste_U_columns))
for key, value in dico.items():
    WD[key] = (270-np.arctan2(df[value],df[key])*180/np.pi)%360

liste_WD_columns=[]
for elem in list(WD.columns):
    new_elem = elem.replace("_V","_WD")
    liste_WD_columns.append(new_elem)
WD.columns = liste_WD_columns
```

5.1 Régression Linéaire avec WS et WR

En procédant toujours de la même manière, la régression linéaire est plus convaincante avec ces nouvelles variables qui remplacent U et V, car le score obtenu est de 48.26.



5.2 SVR avec WR et WS

On peut aussi faire tourner un algorithme SVR sur nos données transformées. Le résultat n'est pas si différent de ce que l'on avait avant.

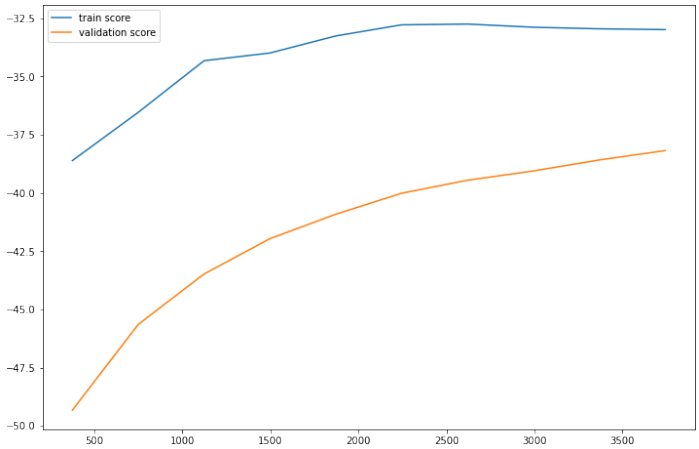


FIGURE 5 – SVR avec WR et WS

6 Soumission des résultats et conclusion

Avant de pouvoir passer à la soumission, il fallait construire un prédicteur pour chaque WF numéroter de 2 jusqu'à 6. J'ai donc entrainé 6 algorithmes SVR avec différents hyperparamètres pour qu'ils aient les meilleurs scores selon la méthode vue au paragraphe 4. Cependant, lorsque j'ai effectué mes prédictions sur le fichier X_test.csv, je me suis rendu compte que ces algorithmes de machine learning étaient mauvais pour s'adapter à ce nouveau fichier, n'ayant eu un score que de 80... (problème de surapprentissage). Je ne m'attendais pas à cela et j'ai donc seulement eu le temps d'entrainer un autre modèle basé sur la régression linéaire en ajoutant les variables WS et WR qui m'a donné un meilleur score de 73,4559.

Ainsi, même si j'ai eu plusieurs difficultés lors de ce data challenge, notamment à la fin lors de la soumission, j'ai quand même pu en apprendre beaucoup plus sur la data science utilisant python. Ayant travaillé seul puisque autre puisqu'aucun autre binôme n'a travaillé sur ce challenge (d'ailleurs lorsque j'ai cliqué sur le lien sur hippocampus environ fin décembre c'est sur celui-là que j'ai été redirigé), je me suis beaucoup attardé sur les premières parties de ce projet pour essayer de comprendre des données pas toujours faciles à analyser et je pense que c'était une bonne expérience.

7 Annexe