Une méthode de Gradient Boosting d'Arbre de Décision explicative basée sur l'algorithme de Frank-Wolfe

Emilien Boizard, Gilles Chardon, Frédéric Pascal

Une méthode de Gradient Boosting d'Arbre de Décision explicative basée sur l'algorithme de Frank-Wolfe

Emilien BOIZARD[1,2] Gilles CHARDON[1] Frédéric PASCAL[1]

[1]Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes, 91190, Gif-sur-Yvette, France

[2]MeilleursAgents, 7 Boulevard Haussmann, Paris, France

Résumé – Les modèles de Gradient Boosting d'Arbre de Décision (GBDT) sont des algorithmes d'estimation très efficaces mais peu explicables. Pour améliorer cet aspect, nous proposons dans cet article une nouvelle méthode inspirée des GBDT permettant d'extraire les données du jeu d'entraînement sur lesquelles elle s'appuie pour faire une prédiction donnée. Pour s'assurer des performances de cette méthode et de son explicativité, elle sera testée sur trois différents jeux de données.

Abstract – Gradient Boosted Decision Trees (GBDT) are a highly effective learning method but suffers from a lack of explainability. To enhance the explainability of such algorithms, we propose in this article a method inspired by these models which allow to exhibit training data on which it relies the most to make a specific prediction. This method will be tested on three different datasets.

1 Introduction

Historiquement, les méthodes d'apprentissage machine ont été développées pour être efficaces selon des critères de performance d'estimation, sans se préoccuper de leur caractère non explicatif. Or, l'explicativité d'un modèle permet à l'utilisateur d'avoir plus confiance en celui-ci, ce qui peut l'inciter à l'utiliser en pratique. Pour certaines applications, il se peut même que l'on préfère utiliser un modèle basique - par exemple une régression linéaire ordinaire - plutôt que des modèles plus précis mais moins explicables. Aussi, l'interprétabilité permet de s'assurer qu'un modèle ait le bon comportement avant de l'utiliser en production. Cet article se focalise sur les modèles de Gradient Boosting d'Arbre de décision (GBDT) [5]. Ces modèles sont très utilisés en pratique, puisqu'ils permettent d'obtenir de très bon résultats sur un grand nombre de jeux de données. Cependant ils manquent d'interprétabilité, comme la plupart des autres méthodes d'ensemble. Des méthodes dans la littérature existent pour rendre ce type de modèle plus explicatif, comme les SHAP values [8], LIME [9], etc. Mais ces méthodes ne dépendent pas des modèles expliqués, et donc ne prennent pas en considération leurs particularités.

La contribution principale de cet article est l'introduction d'une méthode inspirée des GBDT qui a pour but d'améliorer l'explicativité de ce type de modèle, uniquement dans le cadre de régressions. Pour ce faire, notre méthode extrait des données d'entraînement similaires à une donnée de test, que ce soit en termes de réponse ou de caractéristiques. Cette façon d'expliquer des modèles peut être préférable aux méthodes classiques, surtout pour un public non expert.

La méthode proposée est basée sur l'observation qu'une prédiction d'un modèle GBDT peut être exprimée comme combinaison linéaire des variables cibles du jeu d'entraînement. Intuitivement, plus le coefficient d'une certaine donnée d'entraînement sera grand dans la décomposition de l'estimation d'une donnée test, plus ces deux données seront similaires. Cependant, dans le cadre des GBDT plusieurs limites sont observées : on peut calculer ces coefficients que pour certaines fonctions de perte, ils ne sont pas garantis d'être positifs et sont coûteux à calculer en pratique. Pour dépasser ces problèmes, nous proposons de contraindre notre méthode d'apprentissage à décomposer ses prédictions comme combinaisons linéaires des variables cibles du jeu d'entraînement, à coefficients positifs. Pour résoudre ce problème d'approximation contraint, nous introduisons une variante de l'algorithme du gradient boosting qui se base sur l'algorithme de Frank-Wolfe. Un avantage de cette méthode est que le vecteur de poids extrait sera parcimonieux (avec au maximum le nombre d'itérations de poids non nul). En plus d'une prédiction, notre méthode donne directement les poids qui permettent d'obtenir cette prédiction, ce qui permet de définir une notion de similarité entre les observations.

Les expériences empiriques sur de vrais jeux de données montrent que la méthode proposée donnent des performances équivalentes à celles obtenues par les meilleurs méthodes GBDT de l'état de l'art. Il est aussi montré que l'on peut obtenir des observations comparables à une certaine observation donnée en comparant leur décomposition. Les observations comparables sont très proches que ce soit en terme de caractéristiques ou de réponse.

2 Gradient Boosting d'Arbre

Dans les problèmes de regression, on cherche à apprendre une fonction F qui associe à $\mathbf{x} \in \mathbf{R}^d$ une réponse $y \in \mathbf{R}$, à l'aide de N échantillons d'entraînement $\{(\mathbf{x}_n, y_n)\}$, qu'on suppose distribués selon la même loi de probabilité $p(\mathbf{x}, y)$. F est entraînée de manière à ce qu'elle minimise l'espérance de la fonction de perte $\mathcal{L}(F) = E(L(y, F(\mathbf{x})))$ où L correspond à une fonction de perte et où E est l'espérance mathématique. Il est impossible en pratique d'estimer F sans introduire d'hypothèses supplémentaires. Souvent, on peut choisir de restreindre notre recherche à un modèle linéaire, faire un certain nombre d'hypothèses de régularité ou choisir un modèle paramétrique, où un petit nombre de paramètres permet de décrire entièrement F. Le gradient boosting d'arbre suit cette dernière approche : F est restreint à l'ensemble

$\left\{F, F(\mathbf{x}) = \sum_{t=1}^{T} \beta_t h_t(\mathbf{x}, \theta_t)\right\}$, où les h_t sont des fonctions paramétrées par des paramètres de petite dimension $\theta_t \in \Theta$, Θ étant un espace de petite dimension et β_t sont des poids réels. Puisque nous n'observons qu'un nombre fini de données, il est impossible de calculer le gradient de $\mathcal{L}$ par rapport à F. En revanche, on a accès à $\{g_t(\mathbf{x}_n)\} = -\frac{L\partial(y, F_t(\mathbf{x}_n)}{\partial F_t(\mathbf{x}_n)}$ en chacun des points, et qui correspond à la direction de plus grande pente dans l'espace de dimension N. Pour étendre cette quantité en dehors de l'espace d'entrainement, on choisit l'élement de la classe $\{h_t(\mathbf{x}, \theta_t), \theta_t \in \Theta\}$ qui produit des prédictions $h_t(\mathbf{x})$ les plus parallèles possible à $g_t(\mathbf{x})$. En fait, cela correspond à la solution de :

$$\text{argmin}_{\alpha \in \mathbf{R}, \theta_t \in \mathbf{R}^d} ||g_t(\mathbf{x}) - \alpha_t h_t(\mathbf{x}, \theta_t)||_2 \quad (1)$$

Algorithme 1 : Gradient boosting

Require: $\{(\mathbf{x}_n, y_n)\}_{n=1,\dots,N}$, T,
$F_0 = \text{argmin}_C \sum_{n=1}^{N} L(y_n, C)$
for t = 1 to T **do**
Compute $\{g_t(\mathbf{x}_n))\} = -\frac{L\partial(y_n, F_t(\mathbf{x}_n)}{\partial F_t(\mathbf{x}_n)}, n = 1 \dots N$
$\theta_t = \text{argmin}_\theta ||g_t(\mathbf{x}) - \alpha h_t(\mathbf{x}, \theta)||_2$
$\gamma_t = \text{argmin}_\gamma \sum_{n=1}^{N} L(y_n, F_{t-1}(\mathbf{x}_n) + \gamma h(\mathbf{x}_n, \theta_t)), n = 1 \dots N$
Update $F_t(\mathbf{x}) = F_{t-1}(\mathbf{x}) + \gamma_t h(\mathbf{x}, \theta_t))$
end for

Un choix populaire pour h_t est de prendre un arbre de régression peu profond [1], θ_t correspondant dans ce cas aux caractéristiques selon lesquelles on divise l'arbre, les valeurs auxquelles on divise, et aux valeurs des nœuds terminaux de l'arbre. En pratique, résoudre (1) correspond simplement à entrainer un arbre sur l'ensemble $\{(\mathbf{x}_n, g_t(\mathbf{x}_n))\}$ avec une fonction de perte quadratique.

De tels arbres peuvent être décris à l'aide des sous ensemble $\{A_k^t\}$ recouvrant l'espace, ainsi qu'avec les valeurs b_k^t. On a :

$$h_t = \sum_{k=1}^{K} b_k^t \mathbf{1}_{A_k^t} \quad (2)$$

$$b_k^t = \frac{1}{\#\{n, \mathbf{x}_n \in A_k^t\}} \sum_{n \in \{n, \mathbf{x}_n \in A_k^t\}} g_t(x_n) \quad (3)$$

Le nombre de sous-ensembles K dépend de la profondeur des arbres. Après T itérations, le modèle de prédiction F_T est décrit par les coefficients γ_t, les ensembles A_k^t, et par les valeurs b_k^t, t variant de 1 à T.

2.1 Décomposition d'une prédiction de Gradient Boosting d'Arbre

En utilisant une fonction de perte ℓ_2 avec un algorithme GBDT, on peut décomposer la prédiction d'une réponse comme une moyenne pondérée des variables cibles y_n du jeu d'entraînement. Les poids induits ne dépendent que des ensembles A_k^t et des caractéristiques $\mathbf{x}$.

En choisissant $F_0 = \frac{1}{N} \sum_{n=1}^{N} y_n$, à partir des équations (2), (3) et de l'étape de mise à jour de l'algorithme 1, il est clair (par récurrence) qu'une prédiction $F_t(\mathbf{x})$ peut être exprimée comme une moyenne pondérée des valeurs d'entraînement y_n

$$F_t(\mathbf{x}) = \sum_{n=1}^{N} w_n^t(\mathbf{x}) y_n \quad (4)$$

et que les poids somment à 1. En effet, en supposant que $F_{t-1}(\mathbf{x}) = \sum_{n=0}^{N} w_n^{t-1}(\mathbf{x}) y_n$ avec $\sum_{n=0}^{N} w_n^{t-1}(\mathbf{x}) = 1$, et en observant que $g_t(\mathbf{x}_i) = y_i - F_{t-1}(\mathbf{x}_i)$, chaque $g_t(\mathbf{x}_i)$ peut être décomposé comme une combinaison linéaire des y_n, tout comme les b_k^t.

Durant l'étape de prédiction, on peut calculer - en plus des prédictions $F_t(\mathbf{x})$ - les poids $w_n^t(\mathbf{x})$ de la façon suivante :

$$w_n^t(\mathbf{x}) = w_n^{t-1}(\mathbf{x}) + \gamma_t \frac{1}{\#\{n', \mathbf{x}'_n \in A_k^t\}} \sum_{n' \in \{n', \mathbf{x}'_n \in A_k^t\}} \delta_{nn'} - w_n^{t-1}(\mathbf{x}'_n) \quad (5)$$

où k est tel que $\mathbf{x} \in A_k^t$. $\delta_{nn'}$ est défini par 1 si $n = n'$, 0 si $n \neq n'$.

Cependant, cette approche est limitée par trois aspects. Premièrement, le coût du calcul des poids en temps et en mémoire est très élevé. En effet, la mise à jour des poids à l'itération t dépend des b_k^t, qui dépendent eux-mêmes de chaque prédiction des $\mathbf{x}_n$ qui sont tombés dans la même feuille A_k^t que $\mathbf{x}$. Ainsi, les poids $w_n^t(\mathbf{x})$ dépendent des poids $w_{n'}^{t-1}(\mathbf{x}_{n'})$ de tous les vecteurs $\mathbf{x}_{n'}$ qui tombent dans la même feuille A_k^t. Ces poids $w_{n'}^{t-1}(\mathbf{x}_{n'})$ dépendent eux-mêmes des poids de toutes les données qui tombent dans la feuille A_k^t, etc. Au final, le calcul des poids $w_n^t(\mathbf{x})$ nécessite de calculer et de stocker les poids de chaque donnée à chaque itération, ce qui occasionne un coup en mémoire de l'ordre de N^2T. Deuxièmement, cette approche n'est valable que pour certaines fonctions de perte, et il est difficile de l'étendre à d'autres : les valeurs $g_t(\mathbf{x}_n)$ doivent être décomposables en combinaison linéaire des variables cibles. Enfin, les expériences numériques montrent qu'en pratique les poids $w_n^t(\mathbf{x})$ peuvent être négatifs, ce qui limite le caractère explicatif de cette méthode.

3 Explicabilité des prédictions de Gradient Boosting

Nous proposons d'utiliser l'algorithme de Franck-Wolfe [4, 7] donné dans l'algorithme 2 et de contraindre les estimateurs F_t à appartenir à l'ensemble Ω des fonctions pour lesquelles, pour tout $\mathbf{x}$, les poids $w_n^t(\mathbf{x})$ appartiennent au simplexe $\Delta = \{(w_1, \dots, w_N), w_n \geq 0, \sum_{n=1}^{N} w_n = 1\}$. Nous montrons que cette approche permet de résoudre les trois problèmes mentionnés précédemment : réduction significative des temps de calcul et de stockage, élimination des contraintes sur les fonctions de perte, et obtention de poids positifs. De plus, les performances prédictives de cette approche restent comparables à celles des autres algorithme de gradient boosting.

A l'itération t l'arbre h_t joue le rôle du gradient dans l'algorithme 2 et il nous faut donc résoudre :

$$s_t = \text{argmin}_{s \in \Omega_m} \langle s, h_t \rangle \quad (6)$$

Ici Ω_m est l'ensemble des fonctions de Ω, telles que les poids $w_n^t(\mathbf{x})$ pour $\mathbf{x}$ dans A_k^t ne font intervenir que les observations dans A_k^t. Ce choix est fait afin de s'assurer que les

Algorithme 2 : Algorithme Frank-Wolfe

$x^0 \in \Omega$
for t = 1 to T **do**
  Calculer $s := \text{argmin}_{s\in\Omega}\langle s, \nabla f(x^t)\rangle$
  $\gamma := \frac{2}{t+2}$
  Mise à jour $x^{t+1} = (1-\gamma)x^t + \gamma s$
end for

poids $w_n^t(\mathbf{x})$ ne vont augmenter que pour les observations $\mathbf{x}_n$ tombant dans les mêmes feuilles que l'observation testée.

Le produit scalaire $\langle\cdot,\cdot\rangle$ qu'on considère est défini par :

$$\langle s, h_t\rangle = \int_{\mathbf{R}^d} s(\mathbf{x})h_t(\mathbf{x})p(\mathbf{x})d\mathbf{x} \tag{7}$$

où p est la densité de probabilité de $\mathbf{x}$. En explicitant h_t et en écrivant $s(\mathbf{x}) = \sum_{n=1}^N w_n(\mathbf{x})y_n$, on obtient :

$$\langle s, h_t\rangle = \int_{\mathbf{R}^d} \sum_n w_n(\mathbf{x})y_n \sum_{k=1}^K b_k^t \mathbf{1}_{A_k^t}(\mathbf{x})p(\mathbf{x})d\mathbf{x} \tag{8}$$

Pour un $\mathbf{x}$ donné tombant dans la feuille $A_{k^\star}^t$, l'intégrande

$$\sum_{n=1}^N w_n(\mathbf{x})y_n b_{k^\star}^t \tag{9}$$

est maximisée pour $w_n(\mathbf{x}) = 1$ pour l'indice n^+ ou n^- dépendamment du signe de b_k^t, avec n^+ et n^- les indices du plus grand, resp. plus petit y_n dans A_k^t.

L'algorithme complet est donné dans l'algorithme 3. Remarquons que la prise en compte la contrainte d'appartenance à Δ ne nécessite pas de projection.

On peut noter que pour un $\mathbf{x}$, les poids sont garantis d'être positifs et de sommer à un. De plus, à chaque itération on a au plus un poids non nul ajouté dans la combinaison convexe et donc il suffit de stocker l'index n_k^t, (soit n^+ ou n^-) pour chaque ensemble A_k^t à chaque itération pour calculer la décomposition en poids. L'algorithme de prédiction est donné dans l'algorithme 4. Notons que l'algorithme décrit ne correspond pas strictement à celui de Frank-Wolfe car les gradients ont été remplacés par des arbres de décision et l'espace de recherche pour s à été restreint à un sous-ensemble de Ω.

Algorithme 3 : ExpGB, apprentissage

Require: $\{(\mathbf{x}_n, y_n)\}_{n=1,\dots,N}$, T,
  initialisation : $F_0(\mathbf{x}_n) = \frac{1}{N}\sum_{n=1}^N y_n, n = 1\dots N$
  for t = 1 to T **do**
    Calculer $\{g_t(\mathbf{x}_n))\} = -\frac{L\partial(y, F_t(\mathbf{x}_n)}{\partial F_t(\mathbf{x}_n)}, n = 1\dots N$
    Apprendre un arbre h_t sur les données $\{(\mathbf{x}_n, g_t)\}, n = 1\dots N$ avec feuilles A_k^t et valeurs b_k^t
    Définir $n_k^t = \begin{cases} \text{argmax}_{\{n,\mathbf{x}_n\in A_k^t\}} y_n & \text{if } b_k^t > 0 \\ \text{argmin}_{\{n,\mathbf{x}_n\in A_k^t\}} y_n & \text{else} \end{cases}$
    $\gamma := \frac{2}{t+2}$
    $F_t(\mathbf{x}_n) = (1-\gamma)F_{t-1}(\mathbf{x}_n) + \gamma y_{n_k^t}$, où $\mathbf{x}_n \in A_k^t$
  end for
  return $h_t, t = 1\dots T$

Algorithme 4 : ExpGB, prédiction

Require: $\mathbf{x}$, arbres h^t, indices n_k^t, valeurs y_n
  $F_0(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^N y_n$
  $w_n^t(\mathbf{x}) = 1/N$
  for t = 1 to T **do**
    $\gamma := \frac{2}{t+2}$
    Avec k tel que $\mathbf{x} \in A_k^t$
    $F_t(\mathbf{x}) = (1-\gamma)F_{t-1}(\mathbf{x}) + \gamma y_{n_k^t}$
    $w_n^t(\mathbf{x}) = (1-\gamma)w_n^{t-1}(\mathbf{x}) + \gamma\delta_{nn_k^t}$
  end for

3.1 Mesure de similarité

Nous introduisons maintenant une mesure de similarité entre les observations basée sur la décomposition en poids de leur prédiction par le modèle. Cette mesure de similarité permet de trouver des observations du jeu d'entraînement similaires à une observation test donnée : si elles ont des poids proches alors elles sont souvent tombées dans les mêmes feuilles. La similarité entre deux points $\mathbf{x}$ et $\mathbf{z}$ est évaluée en comparant les décompositions en poids $\mathbf{w}^T(\mathbf{x})$ et $\mathbf{w}^T(\mathbf{z})$, par exemple en calculant la distance ℓ_1 entre leur poids. On définit la distance :

$$d(\mathbf{x}, \mathbf{z}) = \|\mathbf{w}^T(\mathbf{x}) - \mathbf{w}^T(\mathbf{z})\|_1, \tag{10}$$

avec $\mathbf{w}^T(\mathbf{x}) = (w_1^T(\mathbf{x}), \dots, w_N^T(\mathbf{x}))$.

Ce choix est justifié par les observations suivantes :

- une petite distance $d(\mathbf{x}, \mathbf{z})$ implique que les réponses estimées sont proches. En effet, $|F_T(\mathbf{x}) - F_T(\mathbf{z})| \leq d(\mathbf{x}, \mathbf{z}) \max|y_n|$
- les échantillons tombant souvent dans les mêmes feuilles auront des poids similaires et donc une distance petite.

Un ensemble d'observations du jeu d'entraînement similaires à une observation test est choisi en sélectionnant les L (avec L choisi par l'utilisateur) données les plus proches. Pour évaluer la qualité de cette distance, nous montrerons dans la section 4 que les L observations similaires à une observation test en sont proches à la fois en termes de caractéristiques et de réponse.

4 Résultats

Pour évaluer les performances de l'approche proposée, nous la comparons avec d'autre méthodes de gradient boosting - Catboost, XGBoost et l'implémentation scikit-learn du gradient boosting - sur différents jeux de données de régression publiques.

Le premier jeu de données qu'on utilise est le 'bike sharing dataset' [2, 3]. La tâche consiste à estimer le nombre de vélos qui seront loués par jour/par heure à Washington à l'aide de variables catégorielles et de variables continues.

Le deuxième jeu de données consiste à évaluer la consommation électrique de la ville de Tetouan [10]. Il contient 5 variables et environ 50000 observations.

Le troisième jeu de données consiste à estimer la température critique de superconductivité [6]. Il contient 81 variables consistant en des propriétés physico-chimiques des matériaux.

Jeu de données	Cat	XGB	Sklearn	ExpGB
bike hourly	2.80e2	2.72e2	3.88e2	2.57e2
bike daily	1.35e5	2.30e5	1.61e5	1.02e5
power consumption	1.15e7	7.60e6	9.05e6	1.83e7
superconductors	2.37e1	2.18e1	2.01e1	3.13e1

TABLE 1 : Erreurs (MSE) sur le jeu d'entrainement

Jeu de données	Cat	XGB	Sklearn	ExpGB
bike hourly	8.69e2	9.79e2	9.92e2	9.86e2
bike daily	3.82e5	4.10e5	4.12e5	3.20e5
power consumption	2.44e7	2.47e7	2.49e7	2.59e7
superconductors	8.33e1	8.79e1	8.90e1	8.56e1

TABLE 2 : Erreurs (MSE) sur le jeu de test

4.1 Analyse des performances

Les MSE sur le jeu d'entraînement et de test sont données dans les tables 1 et 2. On peut voir que ExpGB atteint un niveau de performance similaire à ceux des autres algorithmes.

4.2 Analyse des comparables

Pour s'assurer de la pertinence des comparables obtenus par la méthode, nous introduisons un algorithme des k plus proche voisins KNN basé sur la distance de similarité proposée. Cet algorithme n'a pas vocation à être utilisé en pratique, mais permet simplement de mesurer la capacité de la méthode à extraire des observations similaires. La réponse d'une observation test est estimée en moyennant les réponses des L (ici L= 10) plus proches données d'entraînement. La MSE de cet estimateur est comparée avec celle de ExpGB dans la Table 3 et avec la variance des réponses (c'est à dire en estimant les réponses par la moyenne). La MSE du KNN reste comparable à celle de ExpGB, ce qui montre que les L plus proches voisins ont une réponse proche de celle du test.

Nous avons aussi mesuré la similarité des comparables en terme de caractéristiques, les résultats sont dans la table 4. Le ratio entre la MSE faite par le KNN et la variance pour chacune des variables est représentée. Nous pouvons voir que les variables des plus proches voisins se concentrent près de la variable du test.

5 Conclusion

Dans cet article, nous avons introduit un nouvel algorithme de gradient boosting, dont les performances sont comparables à celles des autres implémentations de gradient boosting dans l'état de l'art. En utilisant une version modifiée de l'algorithme de Frank-Wolfe à la place d'une descente de gradient classique,

Jeu de données	ExpGB	KNN estimator	Variance
bike hourly	9.86e2	1.26e3	2.42e4
bike daily	3.20e5	3.93e5	3.41e6
power consumption	2.59e7	2.44e7	5.04e7
superconductors	8.56e1	9.64e1	1.18e3

TABLE 3 : MSE de l'estimateur KNN comparé à ExpGB et à la variance de la réponse.

Jeu de données	Caract. 1	Caract. 2	Caract. 3	Caract. 4
bike hourly	5.14e-3	1.67e-2	5.59e-2	5.33e-2
bike daily	6.26e-3	3.73e-2	3.79e-2	1.64e-1
power consumption	3.73e-3	1.50e-2	1.23e-2	1.47e-2
superconductors	2.11e-2	1.25e-2	1.13e-2	1.25e-2

TABLE 4 : Ratio entre la MSE de l'estimation par KNN et la variance des variables.

cette approche permet de décomposer une prédiction comme une combinaison convexe des réponses des données du jeu d'entraînement. Nous avons montré que les observations comparables du jeu d'entraînement extraites par notre méthode sont proches à la fois en termes de caractéristiques et de réponse.

Références

[1] Leo BREIMAN, J. H. FRIEDMAN, R. A. OLSHEN et C. J. STONE : *Classification and regression trees*. Chapman & Hall, New York, 1993.

[2] Dheeru DUA et Casey GRAFF : UCI machine learning repository, 2017.

[3] Hadi FANAEE-T et Joao GAMA : Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, pages 1–15, 2013.

[4] Marguerite FRANK et Philip WOLFE : An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

[5] Jerome H. FRIEDMAN : Greedy function approximation : A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001.

[6] Kam HAMIDIEH : A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154:346–354, 2018.

[7] Martin JAGGI : Revisiting Frank-Wolfe : Projection-free sparse convex optimization. *In Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*.

[8] Scott M. LUNDBERG et Su-In LEE : A unified approach to interpreting model predictions. *In Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc.

[9] Marco Tulio RIBEIRO, Sameer SINGH et Carlos GUESTRIN : "Why should i trust you ?" : Explaining the predictions of any classifier. *In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 1135–1144, New York, NY, USA, 2016.

[10] Abdulwahed SALAM et Abdelaaziz EL HIBAOUI : Comparison of machine learning algorithms for the power consumption prediction :-case study of tetouan city–. *In 2018 6th International Renewable and Sustainable Energy Conference (IRSEC)*, 2018.