XITS Math

PRÉVISION DE PRÉCIPITATIONS À PARTIR DE DONNÉES MÉTÉO

EA initiation à la recherche : Étude de cas

Janvier 2024 - Mars 2024

Léa BOHBOT, Mohammed-Yassine HABIBI, Kenta VERT

ÉCOLE POLYTECHNIQUE

IP PARIS

SOMMAIRE

1 Modèle LSTM appliqué à des données temporelles **3**
1.1 LSTM : Théorie . . . 3
1.2 LSTM : Application à nos données . . . 4
1.2.1 Adaptation du jeu de données . . . 4
1.2.2 Paramètres du modèle . . . 4
1.2.3 Choix de la taille de séquence n . . . 4

2 Application du modèle LSTM : comparaison et analyse des résultats **5**
2.1 Résultats . . . 5
2.2 Comparaison avec les précédents modèles . . . 5

3 Interprétation du modèle : Vers un modèle plus simple **7**
3.1 Variables principales . . . 7
3.2 Sélection des variables les plus explicatives . . . 7

4 Explicabilité du modèle **8**
4.1 Explication par approximation linéaire locale . . . 8
4.1.1 Principe de LIME . . . 8
4.1.2 Application à notre modèle et interprétation des résultats . . . 9

Bibliographie **10**

CONTEXTE

En première partie de projet, nous nous sommes familiarisé avec le jeu de données et avons pu tester des algorithmes de Machine Learning standards (régression logistique, Random Forest, XGB, Multi-layer Perceptron, ...) avec lesquels nous avons eu déjà des bons résultats.
Nous allons maintenant essayer d'appliquer une méthode plus avancée de DeepLearning adaptée aux séries temporelles, les réseaux Long-short term memory. Après une approche théorique puis pratique des LSTM, nous comparerons ses résultats a priori meilleurs à ceux des modèles précédemment étudiés. Puis nous nous intéresserons à l'interpretabilité et la simplification de notre modèle, qui est une étape importante pour s'assurer de la fiabilité des prédictions.

1
MODÈLE LSTM APPLIQUÉ À DES DONNÉES TEMPORELLES

1.1 LSTM : Théorie

LSTM signifie Long Short Term Memory. C'est un type de réseaux de neurones développé en 1997, inspiré des réseaux de neurones récurrents (RNN), et permettant de contourner la difficulté concernant l'explosion ou la disparition du gradient lors de l'entraînement des RNN lorsqu'on souhaite utiliser des longues séquences de données en entrée. Ce sont donc des modèles plutôt simples qui permettent de prendre en entrée une séquence d'informations et de réaliser une prédiction prenant en compte le contexte de chaque information. [1]

Soit $\mathbf{Y} \in \mathbb{R}^N$ la variable cible et $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N) \in \left(\mathbb{R}^V\right)^N$ les variables explicatives. N est le nombre de jours pour lesquels nous avons des données. Posons $n \leq N$ la taille d'une séquence qui sera utilisée comme entrée dans notre modèle LSTM.
Notons $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ une séquence de taille n réindexée de 1 à n. Soit $t \in [\![1, n]\!]$ désignant un temps dans la séquence de taille n.
Un réseau LSTM prenant en entrée une séquence de taille n peut être imaginé comme une suite de n cellules identiques, avec chacune 3 entrées et 2 sorties.

Voici schématiquement comment fonctionne la $t^{\text{ième}}$ cellule $\mathcal{C}_t$ d'un modèle LSTM.

Entrées : $\begin{cases} \boldsymbol{C}_{t-1} : \text{Mémoire long terme issue de } \mathcal{C}_{t-1} \\ \boldsymbol{h}_{t-1} : \text{Mémoire court terme issue de } \mathcal{C}_{t-1} \\ \boldsymbol{X}_t : \text{Données extérieures à l'instant } t \end{cases}$

- $\boldsymbol{C}_t$:Une proportion seulement de la mémoire long terme $\boldsymbol{C}_{t-1}$ est conservée ($\boldsymbol{f}_t$), puis on lui ajoute une contribution issue de $\boldsymbol{h}_{t-1}$ et de $\boldsymbol{X}_t$ pour avoir $\boldsymbol{C}_t$. ($\boldsymbol{i}_t$)
- $\boldsymbol{h}_t$: On le détermine en conservant une proportion de $\boldsymbol{C}_t$ préalablement normalisée. Cette proportion est établie à partir de $\boldsymbol{X}_t$ et de $\boldsymbol{h}_{t-1}$.

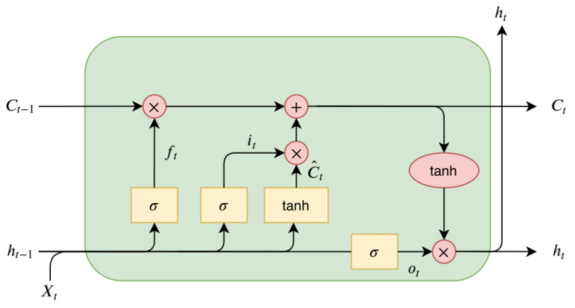


Figure 1 – Fonctionnement d'une cellule d'un réseau LSTM [5, 6]

La valeur de sortie $\boldsymbol{h}_n$ de la dernière cellule correspond à la prédiction du modèle LSTM. Dans notre cas, elle correspondra à une probabilité (dans $[0, 1]$) qu'il pleuve le lendemain du jour correspondant aux données $\boldsymbol{X}_n$.

Nous allons maintenant voir comment nous avons mis ça en place en pratique.

1.2 LSTM : Application à nos données

1.2.1 • Adaptation du jeu de données

Supposons dans un premier temps que la taille n des séquences que nous allons utiliser est fixée.

Il faut tout d'abord transformer notre ensemble $\boldsymbol{X}$ en ensemble de séquences de taille n.
La première difficulté est qu'il faut avoir assez de plages de temps continues pour pouvoir garder un ensemble d'entrainement assez grand. Pour cela nous avons conservé une plage de temps $[\boldsymbol{J}_{\text{début}}, \boldsymbol{J}_{\text{fin}}]$ sur laquelle il y a peu de données manquante puis nous les avons complétées par interpolation temporelle pour les données numériques et par remplissage vers l'avant pour les données catégorielles. Notons $\widetilde{\boldsymbol{N}} := \boldsymbol{J}_{\text{fin}} - \boldsymbol{J}_{\text{début}} + 1$.
Une fois que nous avons limité nos données à $\boldsymbol{X}_{\boldsymbol{J}_{\text{début}}}, \dots, \boldsymbol{X}_{\boldsymbol{J}_{\text{fin}}}$, nous créons donc notre nouvel ensemble de données correspondant aux séquences de taille n :

$$\boldsymbol{Z} = \left(\boldsymbol{Z}_1, \dots, \boldsymbol{Z}_{\widetilde{\boldsymbol{N}}-n}\right) \in \left(\mathbb{R}^{V \times n}\right)^{\widetilde{\boldsymbol{N}}-n} \text{ où pour tout } i,\ \boldsymbol{Z}_i := (\boldsymbol{X}_{J_{\text{début}}+i}, \dots, \boldsymbol{X}_{J_{\text{début}}+i+n-1})$$

1.2.2 • Paramètres du modèle

Nous allons utiliser l'architecture suivante, qui s'est avérée être efficace tout en restant simple :

- Une couche cachée LSTM avec 64 neurones de type LSTM
- Une couche Dense avec une fonction d'activation sigmoïde renvoyant notre prédiction sous forme d'une probabilité.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 64)	33792
dense (Dense)	(None, 1)	65

Total params: 33857 (132.25 KB)
Trainable params: 33857 (132.25 KB)
Non-trainable params: 0 (0.00 Byte)

Figure 2 – Architecture utilisée

1.2.3 • Choix de la taille de séquence n

On veut maintenant choisir la taille de séquence n qu'on utilisera de sorte à ce qu'elle soit ni trop petite, ni trop grande. En effet, un trop grand n donnera un modèle trop compliqué avec beaucoup de variables inutiles, là où un n trop petit serait sous-optimal. Nous utiliserons les résultats empiriques pour fixer un n qui nous permette de conserver un modèle explicable. Ici, $n = 10$ parait être un choix convenable.

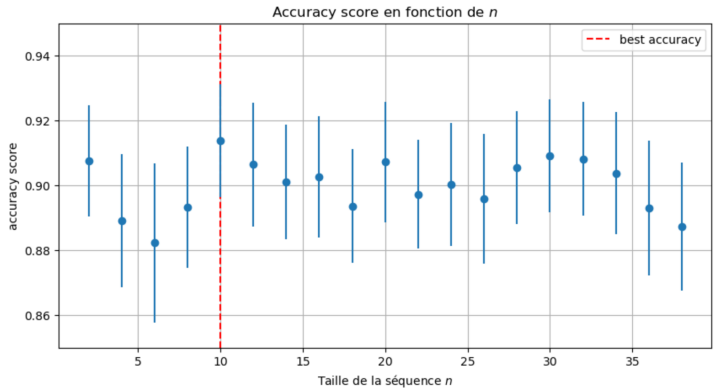


Figure 3 – Score du modèle LSTM en fonction de n

2
APPLICATION DU MODÈLE LSTM : COMPARAISON ET ANALYSE DES RÉSULTATS

2.1 Résultats

Comme dit précédemment, nous avons choisi un modèle très simple pour paramétrer notre modèle avec la taille de la séquence que l'on garde en mémoire. En effet, il nous paraissait intéressant dans ce cadre de prédiction météorologique de savoir combien de jours étaient important pour effectuer une prédiction. Ce nombre est bien évidemment amener à changer selon la complexité du modèle mais nous avons préféré nous concentrer sur cette question de taille de séquence plus facile à interpréter plutôt que de finetuner d'autres paramètres d'un modèle complexe.

Sans grande surprise les résulats obtenus étaient meilleurs, et nous avons trouvé comme optimum une période de **10 jours à garder en mémoire** afin d'être le plus précis (cf fig. 3).

2.2 Comparaison avec les précédents modèles

Commençons par rappeler les modèles que nous avions utilisés lors de la première partie du projet, ainsi que leur résultats. La métrique que nous avions utilisé était la précision obtenue en effectuant une validation croisée sur les 5 mêmes batchs de données.

- La régression logistique, avec une précision de **84,2 %**
- La random forest, avec une précision de **85 %**
- Le Extreme Gradient Boosting (XGB), avec une précision de **85,5 %**
- Le Adaptative Gradient Boosting (AdaBoost), avec une précision de **84,4 %**

Ainsi, en utilisant la même méthode d'évaluation, pour le meilleur modèle avec une séquence de 10 jours nous trouvons une précision légèrement meilleure de **87,98**

Voici le processus mis en place afin de comparer ces différents modèles dans des mêmes conditions :

- Nous séparons nos données en un ensemble d'entraînement et de test, avec une proportion de 80/20 entre les données d'entraînement et de test.
- Nous entraînons les modèles sur ces données d'entraînement
- Nous tirons aléatoirement 250 données de test au sein de l'ensemble défini préalablement, et nous évaluons notre modèles sur ces 250 jours.
- En répétant l'opération précédente de nombreuse fois, nous pouvons mieux comparer les modèles dans un même environnement.

C'est d'ailleurs ce protocole qui nous a permis de choisir le modèle avec une séquence de 10 jours et qui nous a permis de tracer les figures 3 et 4.

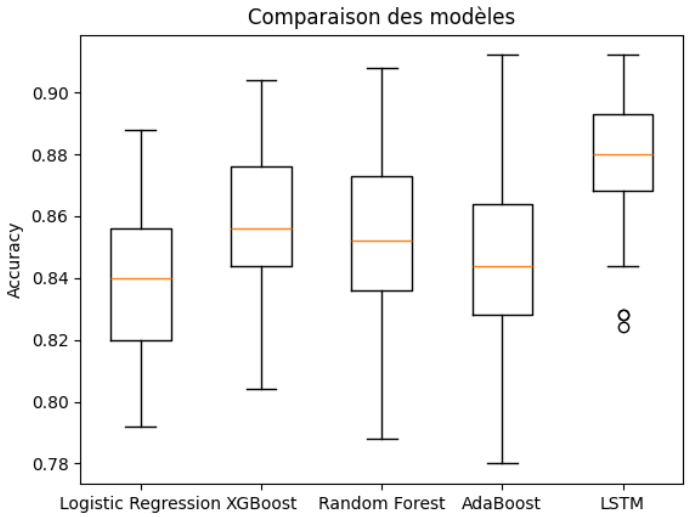


Figure 4 – Comparaison des différents modèles utilisés

Ce que nous pouvons noter graphiquement est la plus petite variance dans les résultats du LSTM sur cet ensemble de test. En effet comme la précision varie moins que les autres modèles, nous pouvons en déduire que le LSTM est **plus robuste** face aux données inconnue. De plus, en excluant les 2 valeurs aberrantes car trop faibles, les "pires" prédictions sont à peu près au niveau de la médiane des autres modèles. Ainsi, non seulement le LSTM est **plus précis** que les autres modèles, mais il est aussi **plus robuste et consistent** dans ses prédictions.

De plus, afin de confirmer cette hypothèse, nous pouvons effectuer un test statistique. Nous pouvons utiliser un test de Wilcoxon, qui teste si deux échantillons suivent la même distribution : ici nous allons voir si nos précisions obtenues sur le boxplot suivent la même distribution. L'hypothèse nulle est "les deux échantillons suivent la même loi" que nous opposons à l'hypothèse alternative "Le LSTM est plus précis que l'autre modèle". Ces test sont significatifs car les p-valeurs sont très petites, et nous gardons donc la deuxième hypothèse, "Le LSTM est plus petit que l'autre modèle" pour tous les autres modèles considérés, ce qui est en accord avec notre intuition.

Voici les p-valeurs :

— LSTM vs Logistic Regression, $p_{value} = 2.10 \cdot 10^{-16}$
— LSTM vs Random forest, $p_{value} = 1.83 \cdot 10^{-12}$
— LSTM vs XGBoost, $p_{value} = 3.19 \cdot 10^{-10}$
— LSTM vs AdaBoost, $p_{value} = 1.63 \cdot 10^{-14}$

3 INTERPRÉTATION DU MODÈLE : VERS UN MODÈLE PLUS SIMPLE

3.1 Variables principales

Un modèle comportant beaucoup de variables d'entrée est difficile à explorer et à interpréter. Souvent, seulement quelques entrées sont influentes.

L'objectif de cette partie est d'indentifier les entrées les plus influentes de notre jeu de données de façon empirique, en se basant sur des moyennes de performances en validation croisée.

Nous avons d'abord fait nos prédictions en ne conservant qu'une variable explicative. Cela nous a donné un premier classement des variables par ordre d'accuracy score, et donc d'importance en terme d'explicabilité.

Lorsque l'on affiche le résultat graphiquement, on se rend compte que, bien que la tendance globale soit l'augmentation de l'accuracy avec le nombre de variables, l'ajout de certaines variables a plutôt tendance à faire baisser l'accuracy. On observe également que celle-ci atteint un plateau à partir d'un certain nombre de variables.

3.2 Sélection des variables les plus explicatives

Nous avons donc essayé de partir de 5 variables explicatives - les plus influentes, et dont l'ajout conduit en moyenne à une augmentation de l'accuracy - puis d'ajouter les variables une à une selon la règle suivantes :

- si l'ajout de la variable fait augmenter en moyenne l'accuracy score par rapport à l'accuracy score du modèle jusqu'ici, la variable est ajoutée. - sinon, elle est rejetée

Précision concernant le calcul de l'accuracy : "en moyenne" signifie qu'on a fait la moyenne des accuracy score en validation croisée de 10 simulations (nombre choisi pour des raisons de complexité, de temps de calcul, et de convergence des résultats)

Bien sûr, cet algorithme ne prend pas en compte d'éventuelles interactions entre variables qui feraient que l'ordre dans lequel nous les ajoutons pourrait changer l'effet d'une variable. On pourrait par exemple imaginer que si nous ajontons la vitesse du vent à l'est, alors qu'il n'ya pas encore la vitesse du vent à l'ouest dans le jeu de données, l'accuracy serait moins importante que si c'est le cas.

Cependant, étant donnée qu'un feature engeeniring des données a déjà été effectué, notamment en combinant des variables liées entre elles et en supprimant des variables redondantes, nous avons noramlement minimisé cet effet en amont.

4
EXPLICABILITÉ DU MODÈLE

4.1 Explication par approximation linéaire locale

Dans cette partie, nous nous intéresserons à **l'explicabilité des prédictions du modèle LSTM** que nous avons entrainé. Le modèle sur lequel nous nous appuierons est un réseau de neurones LSTM qui classifie le jour suivant à partir des données des 10 jours précédents. Avec ses **33 857 paramètres**, il est *a priori* très difficile d'expliquer de manière intelligible les prédictions de ce modèle qui sont fortement non linéaires. Or l'explicabilité et la compréhension d'un modèle est très importante pour pouvoir se convaincre de la fiabilité de ses prédictions.

Pour pouvoir expliquer les prédictions, nous allons nous intéresser à la mise en place d'un « explicateur » qui serait en mesure de déterminer, pour une prédiction donnée, les paramètres jouant un rôle prédominant dans cette prédiction, et par conséquent la rationalité sous-jacente de notre classifieur. Plus particulièrement nous nous appuierons sur une méthode d'interprétation locale développée en 2016 appelée Local Interpretable Model-Agnostic Explanations (LIME) [4].

4.1.1 • Principe de LIME

- **Contexte théorique** :

Notons $f : \mathbb{R}^J \longrightarrow \{0, 1\}$ la fonction représentant notre modèle à expliquer.
Soit $x \in \boldsymbol{X}$ un point dont nous voulons expliquer la prédiction $f(x)$.
Définissons une mesure de proximité autour de x notée $\pi_x : \mathbb{R}^J \longrightarrow \mathbb{R}^+$. ($\pi_x(z)$ est d'autant plus grand que z est proche de x).
Fixons G une classe de modèles interprétables simples (régressions linéaires, arbres de décision,...) qui sera notre classe de modèles explicatifs pour notre modèle f. Pour $g \in G$, on définit $\Omega(g)$ comme étant la complexité du prédicteur g. (Cela peut être le nombre de coordonnées non nulles pour une régression linéaire, ou la profondeur pour un arbre de décision).
Finalement, on définit une fonction de perte $\mathcal{L}(f, g, \pi_x)$ qui est une mesure de l'écart entre le prédicteur f et le prédicteur g dans le voisinage définit par π_x.

Le principe de LIME est d'obtenir un modèle explicatif ξ_x pour la prédiction $f(x)$ en définissant

$$\xi_x := \underset{g \in G}{\operatorname{argmin}}\ \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

. Il y a donc un compromis à faire entre la proximité de notre modèle explicatif à notre modèle de départ, et sa simplicité/interprétabilité.

- **Détermination pratique de ξ_x** :

Dans notre cas, nous utilisons un noyau exponentiel $\pi_x(z) = \exp(-D(x, z)^2/\sigma^2)$ avec D la distance euclidienne et σ la largeur du noyau exponentiel qu'on a prise égale à $0,75\sqrt{N}$ en nous référant à l'usage habituel [2].
On utilisera comme classe de modèles interprétables simples G la classe des régresseurs linéaires, puis on utilisera une régression Ridge (moindres carrés avec une pénalisation L_2 pour notamment éviter le surapprentissage) pour trouver la fonction $g_{\theta,\theta_0}(\cdot) = \langle \theta, \cdot \rangle + \theta_0$ optimale.

On a donc ici

$$\xi_x = \underset{g_{\theta,\theta_0} \in G}{\operatorname{argmin}} \sum_{z \in \mathcal{V}(x)} \exp(-D(x,z)^2/\sigma^2) \left(f(z) - g_{\theta,\theta_0}(z)\right)^2 + ||\theta||_{L_2}^2 + \Omega(g_{\theta,\theta_0})$$

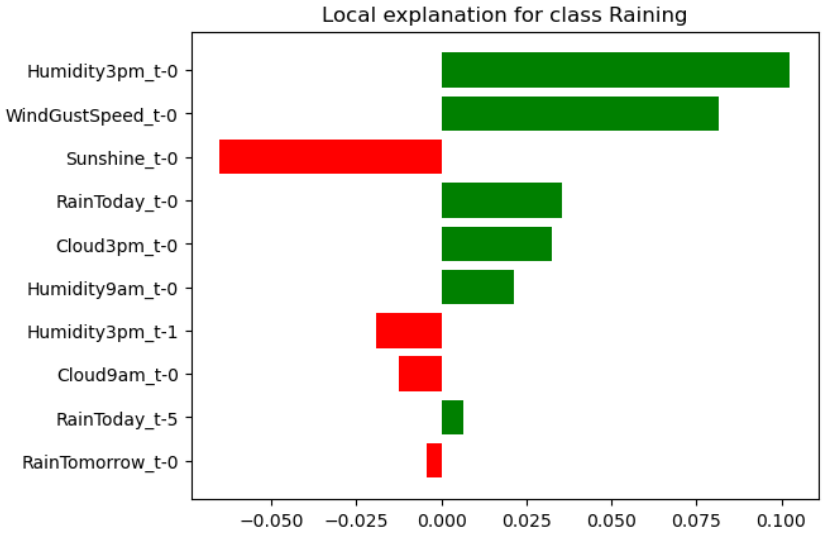
où $\mathcal{V}(x)$ est un ensemble fini de points échantillonnés selon une loi gaussienne réduite et centrée sur x.

Finalement, la pénalisation $\Omega(g_{\theta,\theta_0})$ sera choisie de sorte à se limiter aux régresseurs 10-sparse (soit avec au plus 10 coordonnées non nulles). En définissant pour $\theta \in \mathbb{R}^J$, $||\theta||_0 = |\{i \in [\![1, J]\!], \theta_i \neq 0\}|$, on pose donc

$$\Omega(g_{\theta,\theta_0}) = \begin{cases} +\infty & \text{si } ||\theta||_0 > 10 \\ 0 & \text{sinon} \end{cases}$$

4.1.2 • Application à notre modèle et interprétation des résultats

En appliquant le modèle explicatif LIME [3] aux prédictions de notre modèle sur le sous-ensemble de test $\boldsymbol{X}_{test} \subset \boldsymbol{X}$, nous pouvons déterminer pour chaque point $x \in \boldsymbol{X}_{test}$, un classifieur linéaire ξ_x approximant f au voisinage de x et caractérisé par un vecteur θ_x. Les poids θ_x nous informent sur l'importance, localement, des variables/colonnes de $\boldsymbol{X}$ dans la prédiction $f(x)$.



Pour obtenir la figure 5, nous avons fixé un point x singulier tel que $f(x) = 0$ (ie tel qu'on prédit qu'il ne pleuvra pas le lendemain). Les variables i telles que $\theta_x^i > 0$ apparaissent en vert et celles telles que $\theta_x^i < 0$ en rouge. On lit par exemple à la première ligne que la variable avec le plus grand poids en valeur absolue dans le régresseur ξ_x est celui lié à la variable « Humidity3pm_t-0 » qui correspond à l'humidité de l'air mesurée à 15h le jour précédent celui de la prédiction.

Figure 5 – Poids du vecteur θ_x pour les 10 coordonnées non-nulles.

La valeur de $\theta_x^{\text{Humidity3pm\_t-0}}$ est ici prépondérante car $\dfrac{|\theta_x^{\text{Humidity3pm\_t-0}}|}{|\theta_x|_1} \approx 27\%$. Cette variable influe donc de façon importante sur la prédiction qu'il ne pleuvra pas. Notons que cette comparaison est valable car toutes nos variables ont été centrées-réduites et ont en particulier la même variance après normalisation.
En observant les 4 paramètres ci-dessus avec un poids absolu plus grand que 0.5, on voit les grandeurs liés à l'humidité, l'ensoleillement, et sur le fait qu'il n'a pas plu ce jour-là. La prédiction peut donc nous sembler très fiable. On peut confirmer ça avec notre modèle initial qui nous donne l'information d'une confiance à 99% pour la prédiction $f(x)$.

Avec plus de temps, nous aurions pu généraliser cette approche, et essayer de trouver les paramètres prédominants à l'échelle du modèle global en nous interessant aux paramètres $i \in [\![1, J]\!]$ tels que $|\{x \in \boldsymbol{X}_{test} / i = \underset{j \in [\![1,J]\!]}{\operatorname{argmax}} |\theta_x^j|\}|$ est grand. Ainsi, nous aurions pu confirmer ou infirmer certains résultats donnés par le modèle XGB.

BIBLIOGRAPHIE

[1] Sepp Hochreiter. “Long Short-term memory”. In : (1997).

[2] *Lime Package Documentation*. https://lime-ml.readthedocs.io/en/latest/lime.html. Lime Package.

[3] Marco Tulio Correia Ribeiro. *LIME*. https://github.com/marcotcr/lime?tab=readme-ov-file.

[4] Carlos Guestrin Marco Tulio Ribeiro Sameer Singh. “”Why Should I Trust You ?” : Explaining the Predictions of Any Classifier”. In : (2016).

[5] Saba Hesaraki. *Long Short-Term Memory (LSTM)*. https://medium.com/@saba99/long-short-term-memory-lstm-fffc5eaebfdc.

[6] Josh Starmer. *Long Short-Term Memory (LSTM), Clearly Explained*. https://www.youtube.com/watch?v=YCzL96nL7j0&t=360s.