Hes·so Master

MSE | MASTER OF SCIENCE IN ENGINEERING

Multimodal Processing, Recognition, and Interaction (MPRI)

Prévisions d'utilisation de vélos avec l'algorithme Random Forest

(Série 2)

Auteur :	Spinelli Isaia
Prof :	Stefano Carrino
Date :	17.10.2020
Salle :	A2 – Lausanne
Classe :	MPRI

Table des matières

Introduction ..... - 2 -
Partie 1 ..... - 2 -
Quels sont les coefficients de Gini les plus importants pour la station 0 ? ..... - 2 -
Combien d'arbres de décision sont utilisés pour l'algorithme Random Forest ? ..... - 3 -
Essayez d'expliquer le rôle des 3 caractéristiques les plus importantes dans le mécanisme de prédiction ..... - 3 -
Pouvez-vous expliquer pourquoi est-il important de ne pas mélanger les données lors de la division des données d'entrainement et de test ? ..... - 4 -
Partie 2 ..... - 5 -
Les différences d'erreur absolue moyenne ..... - 5 -
Comment montrer l'impact des nouvelles caractéristiques sur les calculs de l'algorithme ? ..... - 6 -
Une station a des résultats très différents des deux autres, pourriez-vous donner quelques hypothèses pour ces résultats ? ..... - 6 -
Question 3 : jeu de donnée univarié ..... - 7 -
Conclusion ..... - 8 -
Difficultés rencontrées ..... - 8 -
Compétences acquises ..... - 8 -
Résultats obtenus ..... - 8 -

Introduction

Durant ce laboratoire, nous utiliserons les données historiques des stations cyclables publiques pour prédire la quantité de vélos disponibles dans une station pour trois horizons différents (15, 30, 60 minutes) en utilisant un algorithme Random Forest.

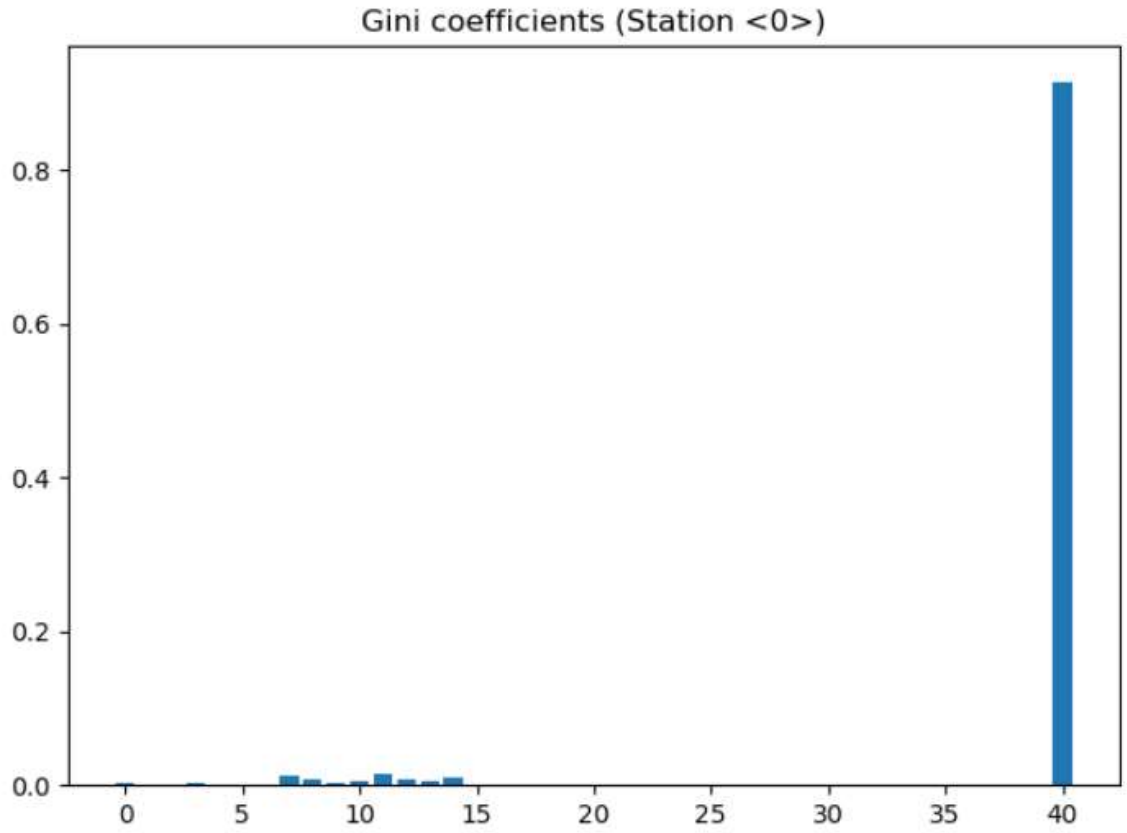
Les données fournies ont déjà été en grande partie prétraitées pour nous, il ne nous faudra donc que faire des petites modifications sur les données reçues. Nous devrons charger et préparer les données, puis former et évaluer l'algorithme d'apprentissage automatique.

Partie 1

Premièrement, le code de la partie 1 a été réalisé. Le programme python complet est en annexe dans le même dossier ZIP (bike_prediction.py).

Quels sont les coefficients de Gini les plus importants pour la station 0 ?

Le graphique ci-dessous montre les coefficients de Gini pour toutes les caractéristiques :



Voici la liste des quatre caractéristiques les plus importantes selon les coefficient de Gini pour ce modèle :

Index 7	(Valeur = 0.01225)	: Température courante
Index 11	(Valeur = 0.01451)	: Température dans 3 heures
Index 14	(Valeur = 0.01059)	: Vent dans 3 heures
Index 40	(Valeur = 0.91519)	: Quantité de vélos disponibles actuellement à la station 0

Combien d'arbres de décision sont utilisés pour l'algorithme Random Forest ?

Étant donné que les paramètres du Random Forest a été laissés à leur valeur par défaut, le nombre d'arbres de décision est sûrement de 100 comme indiqué dans la documentation de la classe « `RandomForestRegressor` » :

Parameters: **n_estimators : *int, default=100***
The number of trees in the forest.

Afin de confirmer ceci, j'ai utilisé la fonction apply() qui retourne le nombre d'arbres de décision (n_estimators) :

`apply(X)` [source]

Apply trees in the forest to X, return leaf indices.

Parameters: **X : *{array-like, sparse matrix} of shape (n_samples, n_features)***
The input samples. Internally, its dtype will be converted to `dtype=np.float32`. If a sparse matrix is provided, it will be converted into a sparse `csr_matrix`.

Returns: **X_leaves : *ndarray of shape (n_samples, n_estimators)***
For each datapoint x in X and for each tree in the forest, return the index of the leaf x ends up in.

Voici la ligne de code qui exécute la fonction :

```
test = RF.apply(X_train_set)
```

On peut facilement voir en mode debug le nombre d'arbres de décision à 100 qui correspond à la documentation.

Essayez d'expliquer le rôle des 3 caractéristiques les plus importantes dans le mécanisme de prédiction

1. Température (now / 3h forecasts) : On peut facilement concevoir que la température joue un grand rôle sur la décision de se déplacer à vélo ou pas.

2. Le vent (3h forecasts) : Le vent, tout comme la température, peut naturellement influencer le choix d'un utilisateur à se déplacer à vélo.

3. La quantité de vélos disponible actuellement : C'est de loin le facteur le plus important dans ce mécanisme de prédiction. En effet, il est primordial de savoir la quantité de vélos actuellement disponible.

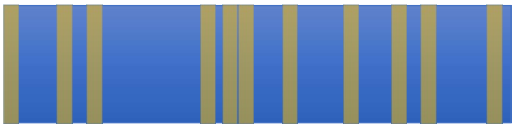
Pouvez-vous expliquer pourquoi est-il important de ne pas mélanger les données lors de la division des données d'entrainement et de test ?

Je pense qu'il est important de ne pas mélanger les données lors de la division car il y a une saisonnalité. En effet, ce cas d'utilisation est particulièrement influencé par les trajets hebdomadaires des utilisateurs qui se répète. Si on mélangeait toutes les données il y aurait des chances que le modèle crée/apprend des mauvais paramètres.

Lors de cette première partie, j'ai divisé les données par semaine mais simplement en deux blocs, alors qu'il serait sûrement plus efficace de séparer en plusieurs blocs aléatoires (Weekly cross-validation).

Deux blocs (75% / 25%)

Weekly cross-validation (75% / 25%)



Partie 2

Les différences d'erreur absolue moyenne

Afin de comparer au mieux les erreurs absolues moyennes, voici les tableaux des trois stations pour chaque horizon :

Prévisions dans 15 minutes :

Station ID	MAE sans les informations de temps	MAE avec les informations de temps	Amélioration
0	0.114844	0.11255	2%
1	0.28543	0.251527	12%
2	0.73228	0.69273	5%
Moyenne	0.378	0.352	7%

Prévisions dans 30 minutes :

Station ID	MAE sans les informations de temps	MAE avec les informations de temps	Amélioration
0	0.305131	0.25335	17%
1	0.551313	0.445021	19%
2	1.27046	1.165699	8%
Moyenne	0.709	0.621	12%

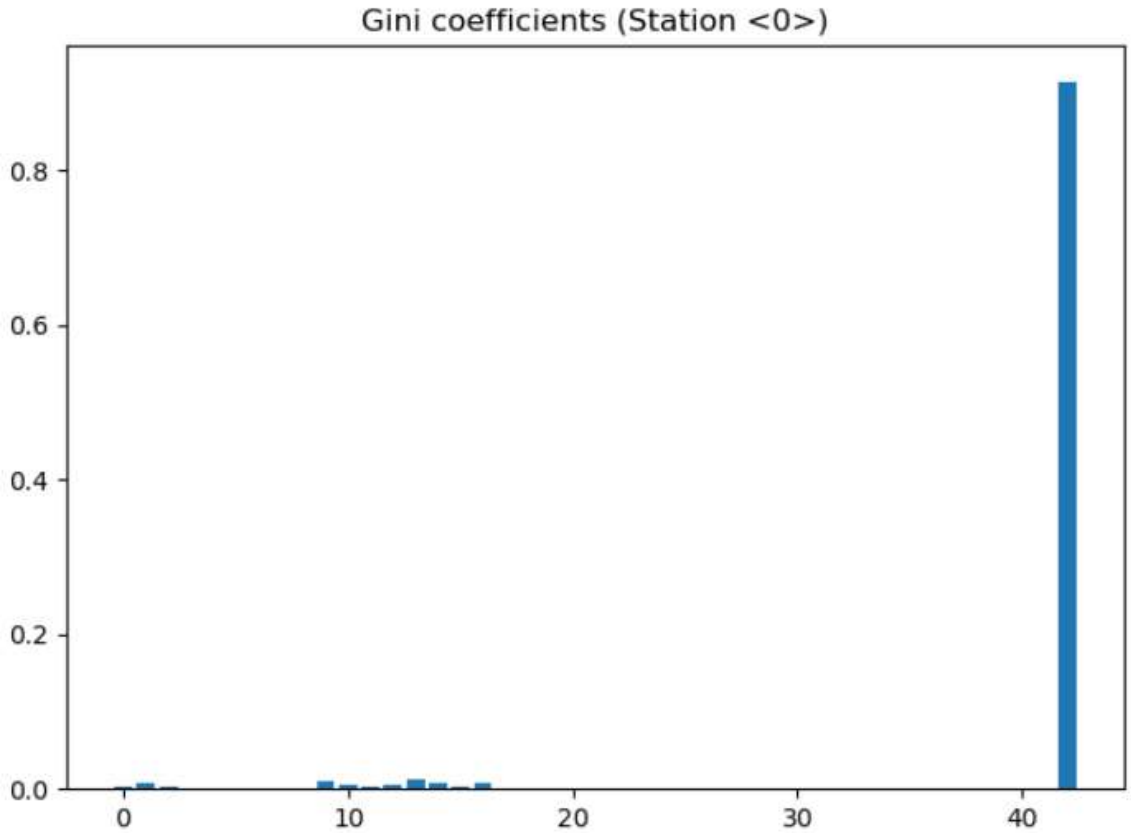
Prévisions dans 60 minutes :

Station ID	MAE sans les informations de temps	MAE avec les informations de temps	Amélioration
0	0.481215	0.4187538	13%
1	0.745571	0.627672	16%
2	1.9393	1.814447	6%
Moyenne	1.055	0.954	10%

On peut constater que les deux nouvelles caractéristiques ont un peu amélioré l'erreur absolue moyenne pour tous les horizons et toutes les stations. De manière générale, l'amélioration de la station 2 est inférieure aux autres stations. C'est le deuxième horizon (prévision dans 30 minutes) qui a toujours la plus grande amélioration.

Comment montrer l'impact des nouvelles caractéristiques sur les calculs de l'algorithme ?

Je pense que les coefficients de Gini correspondent bien à l'impact des caractéristiques sur l'algorithme.



Les deux nouvelles caractéristiques (l'heure de la journée et le jour de la semaine) ont été ajoutées au premier index de la dataframe. Dans le graphique ci-dessus, on peut constater que les coefficients de Gini à l'index 0 (le jour de la semaine) est supérieur à la majorité des autres indexes qui est quasiment nul. L'index 1, qui correspond à l'heure de la journée, a un impact encore plus fort.

Une station a des résultats très différents des deux autres, pourriez-vous donner quelques hypothèses pour ces résultats ?

On remarque que la station 2 a une moyenne d'erreur absolue beaucoup plus élevée que les deux autres stations. On peut conclure que cette station est plus complexe à prédire. Ceci est sûrement dû au fait que cette station doit être plus utilisée que les deux autres et surtout de manière moins régulière. On pourrait typiquement penser que c'est la station de la gare de la ville, la station la plus fréquentée et la moins stable.

Question 3 : jeu de donnée univarié

L'objectif est de former un algorithme d'apprentissage automatique avec un jeu de donnée univarié. Jusqu'à maintenant, nous avons toujours pratiqué avec une méthode supervisée. Cette dernière consiste à fourni à l'algorithme un jeu de donné d'entraînement annotés (avec les résultats). Dans ce cas, nous devons ajouter les annotations (labels / résultats) dans le jeu de donnée fourni.

Voici un exemple de jeu de donnée si nous souhaitons prévoir dans la prochaine minute :

t	Valeur(t)	Valeur(t+1)
0	1	2
1	2	3
2	3	3
3	3	..
…	…	6
98	6	7
99	7	-

Voici un petit exemple de code qui appliquerait ce principe avec trois horizons (prévision pour +1min / +2min / +3min) :

```
df = pandas.DataFrame({'value(t)': range(100)})

df['t+1'] = df['value(t)'].shift(-1)
df['t+2'] = df['value(t)'].shift(-2)
df['t+3'] = df['value(t)'].shift(-3)
```

Voici le résultat du jeu de donnée :

	value(t)	t+1	t+2	t+3
0	0	1.00000	2.00000	3.00000
1	1	2.00000	3.00000	4.00000
2	2	3.00000	4.00000	5.00000
3	3	4.00000	5.00000	6.00000

…

96	96	97.00000	98.00000	99.00000
97	97	98.00000	99.00000	nan
98	98	99.00000	nan	nan
99	99	nan	nan	nan

On peut constater que maintenant notre jeu de donnée est complet et prêt à former un algorithme d'apprentissage automatique.

Conclusion

Difficultés rencontrées

- Manipulation du langage Python
- Recherche des différentes fonctions à utiliser

Compétences acquises

- Familiarisation des librairies
- Amélioration de la maîtrise du langage python
- Algorithme Random Forest / Coefficient de Gini
- Meilleure compréhension des horizons

Résultats obtenus

Toutes les étapes demandées du laboratoire ont été réalisées et les questions répondues. J'ai trouvé ce laboratoire particulièrement instructif car il était bien dirigé, ce qui m'a aidé à le réaliser pas à pas. De plus, le fait de travailler sur un projet concret et réel est particulièrement intéressant.

Date : 17.10.20

Nom de l'étudiant : Spinelli Isaia