UFR MATHÉMATIQUES

Master 2

Probabilités et Statistiques

Projet de Machine Learning

Prévision du nombre de vélos loués par la Méthode de la Régression Linéaire

---

Mars 2021

Rédigé Par

Afo Komlan Crépin AKAKPO

&

Nango FOFANA

Année Universitaire : 2020 - 2021

Table des matières

1 Introduction

La location de vélos où le processus d'obtention de l'adhésion, de location et de restitution du vélo est automatisé via un réseau de bornes réparties dans la ville. Grâce à ces systèmes, les gens peuvent louer un vélo à un endroit et le rendre à un autre endroit selon les besoins. Il existe actuellement plusieurs centaines de programmes de vélos en libre-service dans le monde (Vélib' à Paris lancé le 15 juillet 2007 ; Vélo Bleu à Nice lancé le 18 juillet 2009 ; Bicloo à Nantes lancé le 5 mai 2008 sont quelques exemples dans certaines villes de France).

Les données générées par ces systèmes les rendent intéressants pour les chercheurs car la durée du trajet, le lieu de départ, le lieu d'arrivée et le temps écoulé sont explicitement enregistrés. Les systèmes de partage de vélos fonctionnent donc comme un réseau de capteurs, qui peut être utilisé pour étudier la mobilité dans une ville. Ce document est le rapport d'une compétition sur kaggle dont nous avons combiner des modèles d'utilisation historiques avec des données météorologiques afin de prévoir la demande de location de vélos dans le programme Capital Bikeshare ou CaBi à Washington D.C..

Nous disposons de données horaires de location sur deux ans du 01/01/2011 au 31/12/2012. L'ensemble des données d'entraînement est composé des 19 premiers jours de chaque mois, tandis que l'ensemble des données test est composé du 20e jour jusqu'à la fin du mois. Nous devons prédire le nombre total de vélos loués pendant chaque heure couverte par l'ensemble de test, en utilisant uniquement les informations disponibles avant la période de location.

En cliquant sur le lien suivant vous accéderez à notre compétition sur le Kaggle : https://www.kaggle.com/nangofofana/ml-projet

2 Présentation de la base de données des vélos en libre-service

2.1 Lecture des données

Pour commencer nous allons importer notre dataset et afficher les données afin de voir les différentes variables. Ci-dessous une partie de notre dataset après l'importation.

Out[3]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
datetime											
2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

2.2 Champs de données

Dans cette partie nous allons donner l'interprétation de chaque variable utilisée.

datetime correspondant à la date horaire et l'horodatage

season correspondant aux différentes saison de l'année :
1 = printemps, 2 = été, 3 = automne, 4 = hiver

holiday si le jour est considéré comme un jour férié ou un week-end

workingday correspondant au cas où le jour de la semaine n'est ni un week-end ni un jour férié

weather correspondant aux différentes météo de l'année :
1 = Clair, Peu de nuages, Partiellement nuageux
2 = Brume + Nuageux, Brume + Nuages fragmentés, Brume + Peu de nuages, Brume
3 = Neige légère, Pluie légère + Orage + Nuages épars, Pluie légère + Nuages épars
4 = Forte pluie + Palettes de glace + Orage + Brume, Neige + Brouillard

temp correspondant à la température en degré Celsius

atemp correspondant à la température "ressentie" en degrés Celsius

humidity correspondant à l'humidité rélative

windspeed correspondant à la vitesse du vend

casual correspondant aux nombres de locations initiées par des utilisateurs non-enregistré

registered correspondant aux nombres de locations initiées par des utilisateurs enregistrés

count correspondant aux nombres total de locations de vélos.

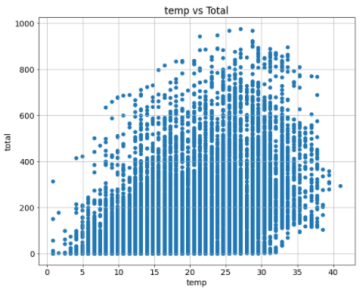
Pour éviter toute ambiguïté, la variable "count" sera remplacée par "total" car "count" en python correspond à une méthode.

```
# remplaçons "count" par total (comme le nombre total de vélos loués)
velo_partage.rename(columns={'count':'total'}, inplace=True)
```

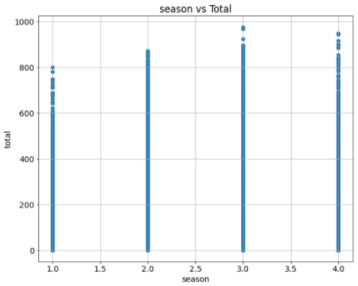
Le "total" est la somme de "casual" et de "registered"

2.3 Visualisation des données

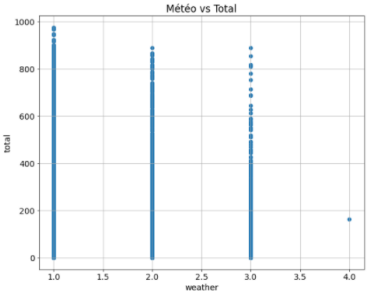
Pour la visualisation les variables telles que "temp", "humidity", "season", "weather" et "atemp" seront visualisées en fonctions du nombres total de vélos loués "total". Les résultats sont traduits par les graphiques suivants :



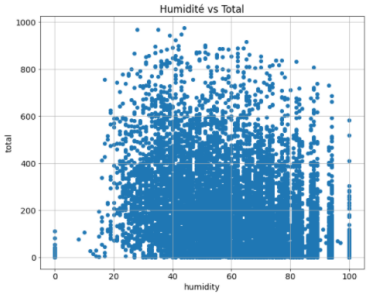
(a) Évolution du nombre de locations de vélos en fonction de la température.



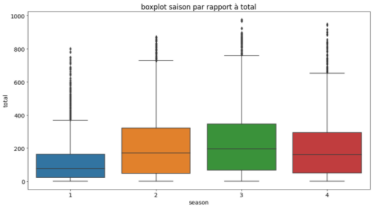
(b) Évolution du nombre de locations de vélos en fonction de la saison.



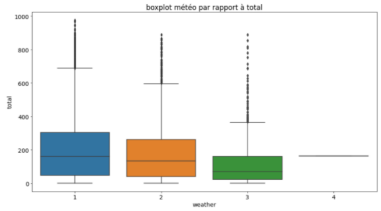
(a) Évolution du nombre de locations de vélos en fonction de la météo.



(b) Évolution du nombre de locations de vélos en fonction de l'humidité

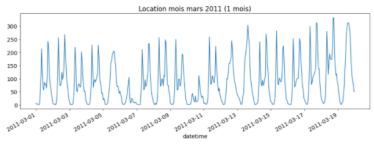


(a) Évolution du nombre de locations de vélos en fonction de la saison sous forme de boxplot.



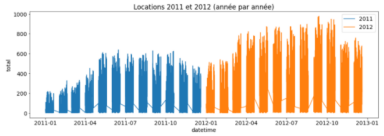
(b) Évolution du nombre de locations de vélos en fonction de la météo sous forme de boxplot.

(a) Évolution du nombre de locations de vélos en 24 heure.



(b) Évolution du nombre de locations de vélos sur 1 mois (mars 2011).

(a) Évolution du nombre de locations de vélos sur 3 mois (mai-juin-juillet 2012).



(b) Évolution du nombre de locations de vélos sur les deux années 2011 et 2012. On constate qu'il y a plus de locations en été qu'en hiver. De plus le système connaît une croissance générale.

2.4 Description statistique des données et la matrice de covariance

2.4.1 Description statistique des données

Le tableau suivant résume la description statistique (la moyenne, l'écart type,le minimum, le premier quartile, la médiane, le troisième quartile et le maximum) des variables de nos données.

Out[21]:

	count	mean	std	min	25%	50%	75%	max
season	10886.0	2.506614	1.116174	1.00	2.0000	3.000	4.0000	4.0000
holiday	10886.0	0.028569	0.166599	0.00	0.0000	0.000	0.0000	1.0000
workingday	10886.0	0.680875	0.466159	0.00	0.0000	1.000	1.0000	1.0000
weather	10886.0	1.418427	0.633839	1.00	1.0000	1.000	2.0000	4.0000
temp	10886.0	20.230860	7.791590	0.82	13.9400	20.500	26.2400	41.0000
atemp	10886.0	23.655084	8.474601	0.76	16.6650	24.240	31.0600	45.4550
humidity	10886.0	61.886460	19.245033	0.00	47.0000	62.000	77.0000	100.0000
windspeed	10886.0	12.799395	8.164537	0.00	7.0015	12.998	16.9979	56.9969
casual	10886.0	36.021955	49.960477	0.00	4.0000	17.000	49.0000	367.0000
registered	10886.0	155.552177	151.039033	0.00	36.0000	118.000	222.0000	886.0000
total	10886.0	191.574132	181.144454	1.00	42.0000	145.000	284.0000	977.0000

2.4.2 Matrice de covariance

La matrice de corrélation de nos jeux de données est la suivante :

Elle montre que le nombre de location de vélos augmente lorsque la température est élevée et diminue lorsque l'humidité est faible.

3 Modèle de la Régression linéaire

3.1 Forme de la régression linéaire

Considérons une régression linéaire de la forme :

$$y = \theta_n x_n + \theta_{n-1} x_{n-1} + \cdots + \theta_2 x_2 + \theta_1 x_1 + \theta_0. \tag{3.1}$$

Où y est la variable expliquée ou réponse, les θ_i sont les coefficients des variables explicatives x_i, $\forall \ \ 1 \leq i \leq n$.

Dans cette partie nous allons représenter les nuages ainsi que la droite de régression. Ensuite ont utilisera la méthode des moindres carrés pour estimer les coefficients θ et minimiser la sommes des carrés des erreurs qui nous permettront par la suite de prédire y.
Le schéma ci dessous nous donne la représentation graphique des nuages de points des valeurs observées y="total" en fonction de x="temp" et la droite de régression linéaire.

Out[24]:
```
<seaborn.axisgrid.FacetGrid at 0x7fcf1115da90>
```

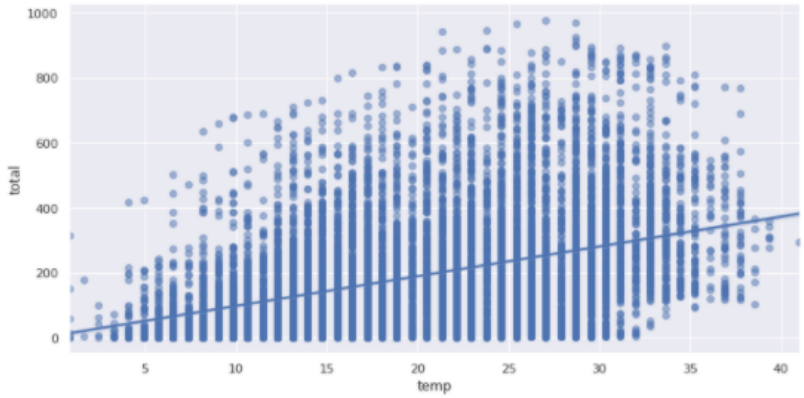


Figure : nuage des points

3.2 Exemple de prédiction de y avec notre modèle de régression linéaire

Pour la construction du modèle de régression nous allons créer les fonctionnalités x et y de la régression linéaire en utilisant le programme suivant :

In [25]:
```
# creation de features X et y de la droite de regression
feature_cols = ['temp']
X = velo_partage[feature_cols]
y = velo_partage.total
```

In [26]:
```
# importation de la librairie, instantiate, fit
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X, y)
```

Out[26]:
```
LinearRegression()
```

On constate que lorsque $x = 0$ nous avons $y = \theta_0$ ce qui correspond au nombre estimé de location de vélos lorsque la température est de 0 degré Celsius.
À 0 degré Celsius le nombres de location de vélos correspondant est de 6.04 soit 6 vélos.
En suite une augmentation de la température de 1 degré Celsius est associée à une augmentation de la location de 9,17 vélos (9 vélos).
Lorsque la température est de 37 degrés Celsius le nombre de locations de vélos est de 345.35 (345 vélos).

Plus la température augmente dans la mesure du normale, plus le nombre de location augmente.

3.3 Ajout d'autres variables au modèle

Dans cette partie nous allons augmenter le nombre de variables pour prédire y. Les variables utilisés sont : 'temp', 'season', 'weather', 'humidity', 'holiday', 'workingday'. Après prédiction ont obtient les résultats suivants :

Interprétation des coefficients :
▷ Toutes les autres caractéristiques étant fixes, une augmentation d'une unité de la température "temp" est associée à une augmentation de la location de 7,86 vélos.
▷ Toutes les autres caractéristiques étant fixes, une augmentation d'une unité de la saison "season" est associée à une augmentation de la location de 22,58 vélos.
▷ Toutes les autres caractéristiques étant fixes, une augmentation d'une unité de la météo "weather" est associée à une augmentation de la location de 6,67 vélos.
▷ Toutes les autres caractéristiques étant fixes, une augmentation de 1 unité de l'humidité est associée à une diminution de la location de 3,12 vélos.
▷ Toutes les autres caractéristiques étant fixes, une augmentation de 1 unité du jour férié "holiday" est associée à une diminution de la location de 10.70514989 vélos.
▷ Toutes les autres caractéristiques étant fixes, une augmentation de 1 unité des jours ouvrables "workingday" est associée à une diminution de la location de 1.66299834 vélos.

3.4 Évaluation métriques pour les problèmes de régression linéaire

Pour le problème de régression linéaire nous étudierons les trois (3) évaluations métriques qui sont : L'erreur absolue moyenne notée (MAE), l'erreur quadratique moyenne notée (MSE) et la racine de l'erreur quadratique moyenne notée (RMSE).

L'erreur absolue moyenne notée (MAE) est la moyenne de la valeur absolue des erreurs qui est la plus facile à comprendre. donnée par la formule suivante :

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.2}$$

L'erreur quadratique moyenne notée (MSE) est la moyenne des erreurs au carrée qui est la plus populaire que l'erreur moyenne MAE car MSE "punit" les erreurs plus importantes, ce qui tend à être utile dans le monde réel

donnée par la formule suivante :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.3}$$

Et la racine de l'erreur quadratique moyenne notée (RMSE) est la racine carrée de la moyenne des erreurs au carré qui est encore plus populaire que MSE, car RMSE est interprétable dans les unités "y" donnée par la formule suivante :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.4}$$

Ces trois indicateurs servent surtout à comparer plusieurs modèles ou prévisions par rapport à une série d'observations, ou encore plusieurs méthodes entre elles.

Pour évaluer ses métriques qui sont des fonctions de perte car nous voulons minimiser ces fonctions nous allons utiliser des valeurs réelles et des valeurs prédites. On a :

In [39]:
```
# example et les valeurs de réponse prédites
valeurs_réelles = [8, 3, 25, 10, 15, 4]
valeurs_predites = [7, 2, 8, 4, 6, 5]
```

Pour les valeurs métriques ont obtient le résultat suivant :

In [40]:
```
# calculer des metriques
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(valeurs_réelles, valeurs_predites))
print('MSE:', metrics.mean_squared_error(valeurs_réelles, valeurs_predites))
print('RMSE:', np.sqrt(metrics.mean_squared_error(valeurs_réelles, valeurs_predites)))
```

```
MAE: 5.833333333333333
MSE: 68.16666666666667
RMSE: 8.256310717667224
```

Faisons un autre exemple d'évaluations métriques en conservant les mêmes valeurs réelles de départs et en choisissant de nouvelles valeurs prédites qui sont totalement différentes des valeurs précédentes. Pour les nouvelles valeurs prédites nous avons : [8, 3, 9, 18, 2, 5]. Nous obtenons des nouvelle valeurs métriques qui sont :

$$MAE = 6.33$$

$$MSE = 81.66$$

$$RMSE = 9.03$$

3.5 Comparaison des modèles avec la division train/test et RMSE

Définissons à partir du programme une fonction qui accepte une liste de caractéristiques et renvoie le RMSE de test.

les caractéristiques utilisée sont : "temp", "season", "weather" et "humidity"

In [45]:
```
from sklearn.model_selection import train_test_split

# définir une fonction qui accepte une liste de variables et retourne des tests RMSE
def train_test_rmse(feature_cols):
    X = velo_partage[feature_cols]
    y = velo_partage.total
    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=123)
    linreg = LinearRegression()
    linreg.fit(X_train, y_train)
    y_pred = linreg.predict(X_test)
    return np.sqrt(metrics.mean_squared_error(y_test, y_pred))
```

En comparant les différents ensembles de caractéristiques ont obtient les valeurs du train-test-rmse qui sont :

In [46]:
```
# comparaison des différents ensembles de variables
print(train_test_rmse(['temp', 'season', 'weather', 'humidity']))
print(train_test_rmse(['temp', 'season', 'weather']))
print(train_test_rmse(['temp', 'season', 'humidity']))
print(train_test_rmse(['temp', 'weather', 'humidity']))
print(train_test_rmse(['season', 'weather', 'humidity']))
```

```
155.64945913079674
164.1653997629182
155.59818936691417
157.25029518102825
164.97494126307595
```

3.6 Comparaison entre la RMSE test et la RMSE nulle

On remarque que la RMSE nulle est la RMSE qui pourrait être obtenue en prédisant toujours la valeur moyenne de la réponse. Il s'agit d'une référence par rapport à laquelle nous pouvons vouloir mesurer notre modèle de régression linéaire. Nous allons diviser x et y en ensembles d'entraînement et de test. Pour cela nous utiliserons les fonctions de programme suivant :

In [48]:
```
# diviser X et y en ensembles d'entrainement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=123)

# création d'un tableau NumPy ayant la même forme que y_test
y_null = np.zeros_like(y_test, dtype=float)

# remplissage du tableau avec la valeur moyenne de y_test
y_null.fill(y_test.mean())
y_null
```

On obtient le résultat suivant :

Out[48]:
```
array([192.26451139, 192.26451139, 192.26451139, ..., 192.26451139,
       192.26451139, 192.26451139])
```

Pour la valeur RMSE nulle ont obtient 179.57

3.7 Traitement des caractéristiques catégorielles

Les variables catégorielles du dataset utilisées sont : "season", "holiday", "workingday" qui sont des variables non ordonnée, et la variables "weather" qui est une variable ordonnées. Pour utiliser ces variables nous allons utiliser un encodages et un codage fictifs.
Pour "weather" nous utiliserons l'encodage **0** ou **1**.
Pour "holiday" et "workingday" nous utilisons un codage fictif
Pour la "season" nous ne pouvons pas simplement laisser l'encodage précédent c'est-à-dire **1 = printemps**, **2 = été**, **3 = automne** et **4 = hiver**, parce que cela impliquerait une relation ordonnée. Au lieu de cela, nous allons créer plusieurs variables fictives. Pour créer ces variables fictives et afficher par exemples 10 dates aléatoire des 4 saison nous avons utilisé la fonction suivante :

In [50]:
```
# création des variables nulles (dummies).
season_dummies = pd.get_dummies(velo_partage.season, prefix='season')

#Afficher aléatoirement 10 lignes des 4 saison
season_dummies.sample(n=10, random_state=1)
```

Exemple de variables fictive :

Out[50]:

	season_1	season_2	season_3	season_4
datetime				
2011-09-05 11:00:00	0	0	1	0
2012-03-18 04:00:00	1	0	0	0
2012-10-14 17:00:00	0	0	0	1
2011-04-04 15:00:00	0	1	0	0
2012-12-11 02:00:00	0	0	0	1
2011-07-13 21:00:00	0	0	1	0
2011-04-05 14:00:00	0	1	0	0
2012-09-03 18:00:00	0	0	1	0
2012-05-09 15:00:00	0	1	0	0
2011-11-01 13:00:00	0	0	0	1

Contentons nous d'utiliser trois (3) variables entre les 4 car ces 3 variables possèdent les informations nécessaires sur la fonctionnalité "season", en choisissant ainsi "season3" = Automne comme étant la base de référence. On obtient le tableau suivant

Out[51]:

	season_1	season_2	season_4
datetime			
2011-09-05 11:00:00	0	0	0
2012-03-18 04:00:00	1	0	0
2012-10-14 17:00:00	0	0	1
2011-04-04 15:00:00	0	1	0
2012-12-11 02:00:00	0	0	1
2011-07-13 21:00:00	0	0	0
2011-04-05 14:00:00	0	1	0
2012-09-03 18:00:00	0	0	0
2012-05-09 15:00:00	0	1	0
2011-11-01 13:00:00	0	0	1

Remarque : *En général, si nous avons une caractéristique catégorielle avec n valeurs possibles, nous ne pouvons créer que n-1 variables nominales.*

Maintenant nous allons associer ses trois variables fictives "season1", "season2", "season4" à nos jeux de données et observé les résultats. En incluant ses trois variables fictives dans notre modèle de régression ont obtient les résultats suivant :
temp = 11.186405863575775
season1 = 41.73686071317084
season2 = 38.346317613452946
season4 = 106.15282218141292
humidity = -2.8194816362596526

Interprétation : Nous allons donner une interprétation de ses résultats. Pour une interprétation de sas résultat

Etant données que toutes les autres caractéristiques sont fixes, ont remarque dans le printemps "season1" une augmentation de location de 41.73 vélos par rapport a l'automne "season3". Ensuite si toutes les autres caractéristiques sont fixés, ont constate une augmentation des locations de 38.34 vélos en été "season2" par rapport à l'automne "season3". Enfin si toutes les autres caractéristiques sont fixes, l'hiver "season4" est associé à une augmentation des locations de 106.15 vélos par rapport à l'automne "season3". 857 6580 1470 En effet, en cas de changement de base de référence(season), ce qui changerait serait l'interprétation des coefficients.

3.8 Ingénierie des fonctionnalités

Dans cette partie nous allons utiliser l'heure comme une caractéristique catégorielle c'est à dire nous allons créer 23 variables fictives et comme une caractéristique numérique unique noté de 0 à 23. On utilise le programme suivant :

In [55]:
```
# Choix de l'heure comme features numérique
velo_partage['hour'] = velo_partage.index.hour
```

In [56]:
```
# Choix de l'heure comme features catégoriel
hour_dummies = pd.get_dummies(velo_partage.hour, prefix='hour')
hour_dummies.drop(hour_dummies.columns[0], axis=1, inplace=True)
velo_partage = pd.concat([velo_partage, hour_dummies], axis=1)
```

Aussi nous utiliserons l'heure du jour (daytime) comme une caractéris-

tique catégorielle unique c'est à dire.

$$daytime = \begin{cases} 1 \quad de \quad 6H00 \quad à \quad 19H59 \\ 0 \quad de \quad 20H00 \quad à \quad 5H59 \end{cases}$$

On a le programme suivant :

In [57]:
```
# Choix de la journée comme variable catégoriel
velo_partage['daytime'] = ((velo_partage.hour > 5) & (velo_partage.hour < 20)).astype(int)
```

Ensuite, essayons d'utiliser chacune de ces trois caractéristiques avec train-test-rmse pour voir laquelle est la plus performante ! Pour cela nous allons utiliser le programme suivant pour obtenir les valeur du train-test-rmse :

In [58]:
```
print(train_test_rmse(['hour']))
print(train_test_rmse(velo_partage.columns[velo_partage.columns.str.startswith('hour_')]))
print(train_test_rmse(['daytime']))
```

```
165.67174264111398
128.3112050281119
152.56495224553757
```

On constate que la caractéristique la plus importantes est celle dont le train-test-rmse est plus grand.

4 Quelques avantages de la régression linéaire

Le modèle de la régression linéaire est un modèle qui permet de prédire mais aussi d'expliquer les données. Il est également :
◇ Simple à expliquer
◇ Hautement interprétable
◇ L'apprentissage du modèle et la prédiction sont rapides
◇ Aucun réglage n'est nécessaire (à l'exception de la régularisation)
◇ Les caractéristiques n'ont pas besoin d'être mises à l'échelle
◇ Peut être performant avec un petit nombre d'observations

5 Conclusion

Nous avons exploré l'ensemble des données et les avons divisées en données journalières, mensuelles, trimestrielles et annuelles, ce qui nous a permis

de constater que la demande de location a une tendance qui change avec le jour, le mois, le trimestre et l'année. La demande de location de vélos est plus importante en semaine que le week-end. Nous constatons que la demande de vélos est plus importante de 6h à 10h et de 16h à 20h les jours ouvrables "workingday", ce qui s'explique par le fait que la demande est plus importante pendant les heures de bureau. Les jours non ouvrables "holiday, la demande est plus importante de 10h à 20h, ce qui est évident étant donné que les gens aiment se lever un peu tard le week-end. Nous pouvons voir que la demande de vélos a augmenté de 2011 à 2012, la tendance est très similaire pour les deux années. Cette augmentation est due aussi aux témoignages des abonnés qui ont profité des service du vélopartage. On peut voir que la demande augmente de Mai à Novembre, puis commence à diminuer, car la période Mai-Novembre correspond à mi-printemps, été et mi-automne qui correspond également aux grandes vacances caractérisées par la présence des touristes. La matrice de corrélation montre que la demande de vélos augmente lorsque la température est élevée et diminue lorsque l'humidité est faible. Nous avons utilisé la régression linéaire pour prédire les locations. L'erreur (RMSE) des valeurs d'entraînemens et de tests varie faiblement. Mais nous pouvons encore améliorer ce modèle par l'ingénierie des caractéristiques. Nous avons essayé l'ingénierie des caractéristiques en effectuant une transformation de l'heure du jour (daytime) comme une caractéristiques catégorielles, ce qui nous a permit d'obtenir des nouvelles valeurs de notre RMSE.

Le QR CODE ci dessous permet d'accéder à tous les résultats de notre compétition kaggle.