Samuel Dussault : SADUS7 : 111 041 109 (1er cycle)
Thibault Meuret : THMEU2 : 111 166 203 (1er cycle)

Livrable 4 – Rapport Final
Bike Sharing Demand

Travail présenté à
Richard Khoury

Dans le cadre du cours :
GLO-4027
Analyse et traitement de données massives

Le 12 avril 2017

I – Analyse de problème, construction d’une solution et résultats :

I-1) Présentation du problème :

Le vélo-partage (ou « Bike Sharing ») est un service offert par plus de 500 villes dans le monde, telles que Montréal et Vancouver, visant à mettre à la disposition de ses citoyens des bicyclettes, généralement à très faible coût. Ce service requérant l’achat et l’entretien, d’un nombre considérable de bicyclettes et d’installations, il va sans dire qu’une bonne analyse de la demande est nécessaire afin d’offrir un service qui soit aussi optimisé que possible. Une approche très efficace dans ce type de situation est l’emploi de l’apprentissage machine, qui permet d’analyser les données recueillies par le passé afin d’estimer la demande future pour le service.

Dans le cadre de ce projet de session, c’est cette approche que notre équipe a employée afin de développer un algorithme capable d’estimer le nombre total de locations à un moment donné, en tenant compte des paramètres fournis, pour la ville de Washington, DC, aux États-Unis. Le challenge Kaggle correspondant à ce travail peut être consultés à l’adresse suivante : https://www.kaggle.com/c/bike-sharing-demand.

I-2) Analyse des données et pré-traitement :

I-2-a) Prise de connaissance des données :

Avant de débuter le développement de l'algorithme de traitement, il était impératif de prendre connaissance du type de données fournies. Dans notre cas, cela consistait en deux fichiers au format csv – « comma-separated values », ou, valeurs séparées par des virgules – contenant les données d'entraînements (train.csv), et les données de tests (test.csv).

Les paramètres en entrée de notre problème étaient les suivants :

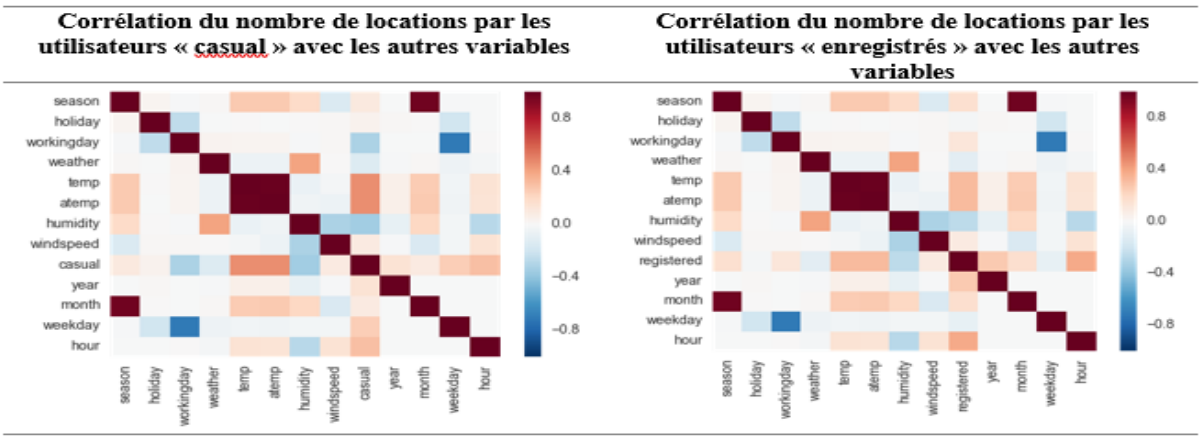
- **Datetime** : l'horodatage de la donnée, au format année-mois-jour heure:minutes:secondes
- **Season** : saison pendant laquelle la donnée a été récoltée, représentée par un chiffre (1 = printemps, 2 = été, 3 = automne, 4 = hiver)
- **Holiday** : valeur binaire indiquant si, oui ou non, la journée pendant laquelle la donnée a été recueillie était une journée fériée.
- **WorkingDay** : valeur binaire indiquant si, oui ou non, la journée pendant laquelle la donnée a été recueillie était une journée de travail, c'est-à-dire ni un férié, ni une journée de fin de semaine.
- **Weather** : la météo au moment où la donnée a été recueillie (1 = pas de pluie, pas ou faiblement nuageux, 2 = faible pluie avec présence de quelques nuages, 3 = pluie/neige modérée avec plusieurs nuages, 4 = forte pluie/neige ou tempête, brouillard, orages et ciel complètement couvert par les nuages).
- **Temp** : la température, en degrés Celsius, à ce moment.
- **Atemp** : la température ressentie à ce moment, en degrés Celsius.
- **Humidity** : l'humidité relative, en pourcentage, au moment où la donnée a été recueillie.
- **Windspeed** : la vitesse du vent, en km/h.
- **Casual (pour l'ensemble d'entraînement uniquement)** : le nombre de locations effectuées par des utilisateurs qui ne sont pas enregistrés dans le système de locations.
- **Registered (pour l'ensemble d'entraînement uniquement)** : le nombre de locations effectuées par des utilisateurs enregistrés dans le système.
- **Count (pour l'ensemble d'entraînement uniquement)** : le nombre total de locations effectuées à ce moment ; cela représente la somme de Casual et Registered.

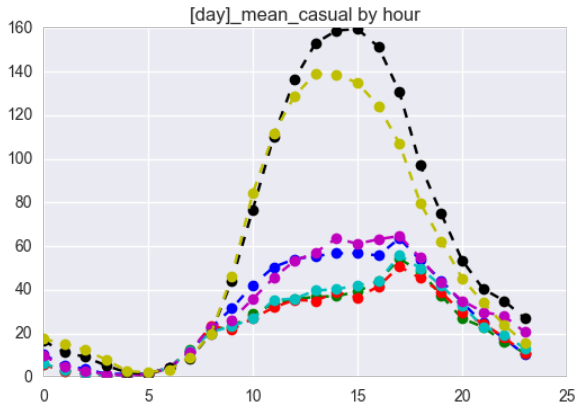
I-2-b) Représentation des données et pré-traitement :

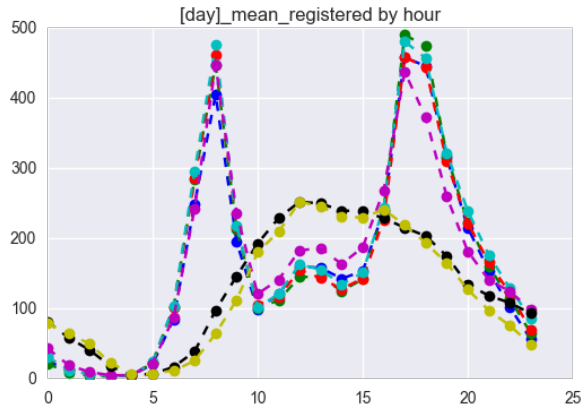
Suite à la prise de connaissance des données fournies afin de résoudre le problème, nous avons fait une analyse préliminaire de ces dernières afin de déterminer s'il était possible, par exemple, de réduire le bruit, de trouver des données manquantes ou encore d'appliquer un pré-traitement permettant de générer un modèle plus précis.

Nous avons ainsi commencé par traduire Datetime (sous forme de timestamp) en différents attributs tels que l'année, le mois, le jour dans le mois et le jour dans la semaine afin de pouvoir exploiter l'information, inaccessible auparavant.

Nous avons ensuite généré divers graphiques (heatmaps, diagrammes à moustache, graphes de valeurs moyennes/médianes selon plusieurs paramètres) et procédé à une analyse de corrélations afin de déterminer les paramètres que nous estimions dispensables à notre modèle.







OLS Regression Results

Dep. Variable:	atemp	R-squared:	0.972
Model:	OLS	Adj. R-squared:	0.972
Method:	Least Squares	F-statistic:	1.249e+05
Date:	Mon, 13 Feb 2017	Prob (F-statistic):	0.00
Time:	21:32:50	Log-Likelihood:	-19291.
No. Observations:	10886	AIC:	3.859e+04
Df Residuals:	10882	BIC:	3.862e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf.	Int.]
const	2.2438	0.071	31.424	0.000	2.104	2.384
temp	1.0712	0.002	609.772	0.000	1.068	1.075
humidity	0.0038	0.001	5.043	0.000	0.002	0.005
windspeed	-0.0386	0.002	-21.852	0.000	-0.042	-0.035

Omnibus:	15324.444	Durbin-Watson:	0.273
Prob(Omnibus):	0.000	Jarque-Bera (JB):	7722991.616
Skew:	-8.095	Prob(JB):	0.00
Kurtosis:	132.478	Cond. No.	359.

C’est l’analyse de tels graphiques qui nous a permis d’émettre nos hypothèses. Tout d’abord, concernant le fait d’établir deux modèles différents : l’un pour les casuals, l’autre pour les registered. Ensuite, concernant la pertinence de certaines variables telles que Season, WorkingDay, WindSpeed et Atemp ; les variables jugées redondantes ont été écartées pour notre modélisation.

I-3) Mise en place d’une démarche pour établir une solution :

I-3-a) Démarche de test suivie pour notre solution :

Le nombre de soumissions sur Kaggle étant limité et chaque entrée possédant un score indépendant, il nous a été nécessaire développer un processus de sélection du modèle – « Framework » – nous permettant de comparer le résultat produit par chacune des versions de l’algorithme. Ainsi, nous pourrions de déterminer si un changement apporté à celui-ci possède un impact positif ou négatif sur le résultat final, et ce avant de soumettre les réponses produites à Kaggle.

Les données d’entraînement ne comptant que 11 000 entrées, nous avons décidé de charger directement en mémoire les données à l’aide de la librairie Pandas, puis d’appliquer la k-cross validation sur 20% des données afin d’isoler un ensemble de test pour sélectionner l’algorithme générant les meilleurs résultats. Pour cela, nous avons choisi de mesurer l’erreur pour chaque modèle via l’erreur quadratique moyenne résultante de 200 itérations.

I-3-b) Choix des algorithmes pour chacun des types d’utilisateurs :

Pour les deux types d’utilisateurs, la méthode finale retenue découle de plusieurs itérations essai/mesure de l’erreur tel que défini dans la partie ***I-3-a) Démarche de test suivie pour notre solution ;*** *nous ne décrivons toutefois que la dernière solution choisie pour chaque type d’utilisateur (la justification de l’évolution du modèle ayant déjà été décrite en détails dans le rapport précédent)*

I-3-b-i) Utilisateurs Casual :

Pour les utilisateurs casual, nous avons choisi d’implémenter une régression linéaire suivant une loi de répartition de Poisson ; c’est un modèle relativement simple, peu coûteux en temps CPU et assez efficace pour ce type d’utilisateur puisque cela représentait le plus faible proportion de l’erreur totale de notre prédiction (soit environ 24 % de l’erreur quadratique totale).

I-3-b-i) Utilisateurs Registered:

Pour les utilisateurs registered, nous avons choisi d’implémenter une solution légèrement plus complexe et coûteuse en temps CPU, mais nécessaire puisque l’erreur totale était engendrée en majorité par ces utilisateurs ; nous nous devions donc d’être plus précis. C’est pourquoi nous avons choisi d’implémenter une régresseur utilisant l’algorithme Random Forest.

I-4) Implémentation, résultats :

Afin d'implémenter ces solutions, puisque nous développions en Python, nous avons choisi d'utiliser les outils du package scikit-learn – c'est un puissant outil pour le data mining/machine learning reconnu par la communauté des utilisateurs de Kaggle.

I-4-a) Premier résultat – implémentation naïve :

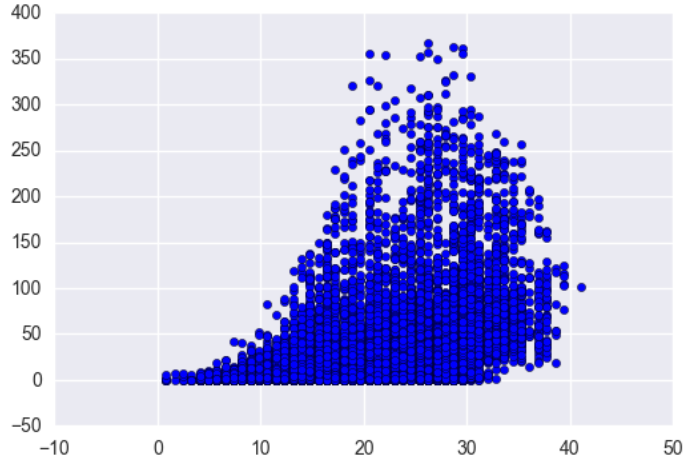
Pour chacun des modèles, après avoir développé une solution correspondant à notre choix d'algorithme, nous avons rendu notre résultat à Kaggle et obtenu le score de **0,561, soit une 2085ème place**. Sachant que nous visions à obtenir un score en dessous de 0.5, nous avons revu nos algorithmes plus en détail afin d'être plus performants.

I-4-b) Résultat final – (ré)-insertion de variables :

Afin d'améliorer notre score et donc notre classement, nous avons repris notre démarche à zéro en revenant sur nos hypothèses concernant les variables mises à l'écart lors de l'analyse des données. Nous avons procédé à une analyse de l'erreur quadratique moyenne en fonction des variables utilisées dans nos deux algorithmes. Les résultats nous ont confortés dans le fait de revoir les variables utilisées pour chaque type d'utilisateur – *nous avons ainsi écarté Weekday et Season pour les Casuals, Windspeed et Season pour les Registered.*

Valeurs manquantes !	CASUAL (0,40) – 50 ite			MOYENNE	REGISTERED (0,20) – 50 ite			MOYENNE	SOMME
¤	370.8360407	386.5085065	371.6833509	376.342633	981.3954916	1001.734361	1074.820498	1019.316784	1395.659416
year	449.0542807	433.0680799	432.660891	438.261084	3244.271015	3124.883475	2859.856596	3076.337029	3514.598113
weekday	352.3445037	372.4507165	377.0162294	367.2704832	1147.101218	1230.91925	1223.743855	1200.588108	1567.858591
season	371.4740149	363.9339642	378.248373	371.218784	1032.535949	950.6176865	973.1791995	985.4442784	1356.663062
humidity	401.4568869	401.3404948	392.5589207	398.452101	1143.730382	1148.992306	1130.229661	1140.984116	1539.436217
workingday	543.0819085	529.6514098	535.0488421	535.927387	1041.535079	1103.395138	1019.329033	1054.753083	1590.68047
windspeed	385.5799108	381.9055452	383.5335647	383.673007	1002.563783	1048.204974	961.7860485	1004.184935	1387.857942
month	383.4923727	377.64326	402.2339171	387.78985	1113.918473	1133.899134	1177.341824	1141.71981	1529.50966
weekday, season	361.0315198	368.8867546	347.1275893	359.0152879	1166.310003	1147.72708	1194.531819	1169.522967	1528.538255
windspeed, season	341.5324115	368.1295236	372.2025592	360.621498	954.2346478	929.7762788	972.9531061	952.3213442	1312.942842
weekday, windspeed, season	376.0021099	374.2318772	367.6684662	372.634151	1242.563616	1127.115398	1270.150373	1213.276462	1585.910613
weather	385.8810701	390.4264482	366.5683261	380.958615	1134.015395	1170.081097	1158.797248	1154.297913	1535.256528

Nous avons également amélioré l'efficacité de notre algorithme en créant une variable catégorielle pour la température indiquant s'il fait froid (< 13 °C), confortable (14 à 27 °C), chaud (28 à 36 °C) ou très chaud (> 37 °C). En effet, nous nous sommes rendu compte de l'existence de paliers de température lorsque nous avons tracé un scatter-plot donnant le nombre de locations en fonction de la température.



Ex : répartition des locations Casuals en fonction de la température

Enfin, nous avons passé du temps à déterminer au mieux nos paramètres pour le Random Forest en mesurant l'impact, par exemple, de la profondeur maximale définie sur l'erreur obtenue sur l'ensemble retenu par la K cross-validation.

Avec toutes ces attentions nous avons pu améliorer significativement notre score sur Kaggle, passant à un score de **0,47404, soit une 1161ème place**. De cette manière, nous avons donc rempli avec succès notre objectif de passer sous le score de 0,5

I-5) Remarques sur le XGBoost Algorithm :

Lors des deux dernières semaines de travail sur le projet, nous nous sommes intéressé à l'algorithme XGBoost pour prédire le nombre de locations pour les utilisateurs Registered. Il s'agit en effet d'un régresseur par boost de gradient qui a été utilisé dans de nombreuses solutions gagnantes de problèmes Kaggle.

Cependant, nous nous sommes rencontré à une difficulté majeure : le tuning des paramètres. En effet, sans avoir une bonne connaissance de l'outil nous avons rencontré de la difficulté à définir, à l'aveugle, les nombreux paramètres : eta, max_depth, gamma, etc. Nous avons donc fait de notre mieux en utilisant des valeurs usuelles pour XGBoost. Nous avons alors soumis notre résultat à Kaggle mais malheureusement nous n'avons pas réussi à avoir un score en dessous de 0,54 – soit, malgré tout, légèrement mieux que le Random Forest avant la ré-insertion de variables.

II – Rétrospective sur les outils, la démarche et les choix réalisés:

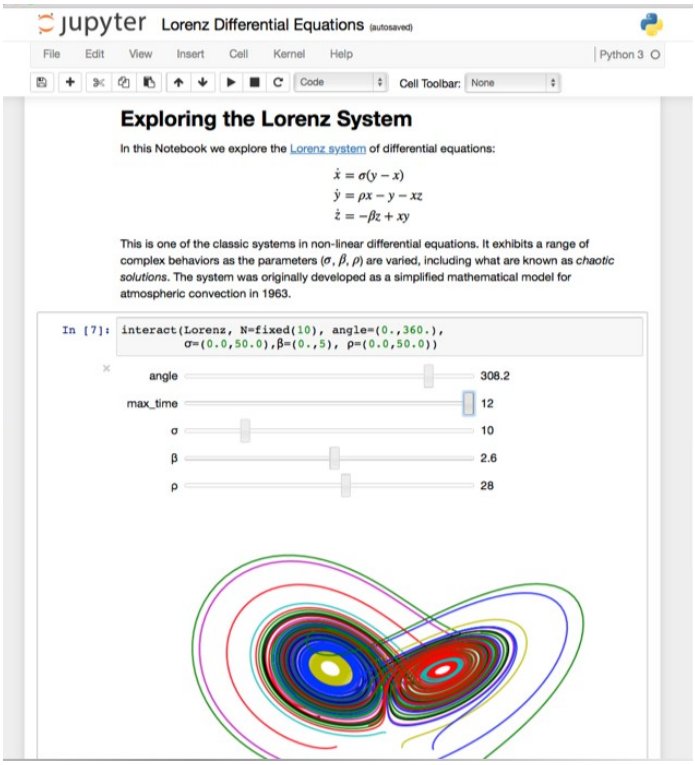
II-1) Retour sur les outils choisis :

Dans le cadre de notre projet, nous avons eu à utiliser de nombreux outils souvent utilisés dans le monde du data mining/machine learning. Le but de cette section est de les mettre en avant et de voir ce que chacun d'entre eux a pu nous apporter.

II-1-a) Langage et librairies utilisées :

Comme précisé plus haut, notre projet a été totalement développé en **Python** ; nous avons donc eu de nombreuses librairies à notre disposition pour nous aider, autant dans la lecture de données que dans l'analyse et l'implémentation de nos algorithmes. Ainsi, nous avons pu utiliser la librairie **Pandas** permettant de manipuler aisément les fichiers csv fournis pour la compétition. La librairie **mathlib** nous a quant à elle permis de générer facilement l'ensemble des graphiques que nous avons utilisés dans notre analyse de données. Enfin, les fonctions de **scikit-learn** nous ont fait gagner beaucoup de temps puisque de nombreux algorithmes d'apprentissage y étaient déjà implémentés, notamment Random Forest et la régression linéaire.

II-1-b) Jupyter Notebook – carnet de nos démarches :



Un autre outil a également eu un apport majeur dans la bonne avancée de notre projet : **Jupyter Notebook**. En effet, cette application web nous a permis de garder une trace de l'ensemble de notre raisonnement sous forme d'un « carnet » ainsi que de mettre à jour dynamiquement les différents paramètres de nos algorithmes. De cette manière, nous avons pu garder une trace de tout ce qui a été fait et avons pu nous en servir comme une force afin de converger vers une solution optimale rapidement.

II-1-c) Autres outils non utilisés :

Dès le début du projet, nous avions également identifié d'autres outils qui auraient pu nous servir mais que nous avons choisi d'écarter par peur de manque de temps pour maîtriser l'outil. Nous pensons notamment à la librairie **TensorFlow** qui permet l'analyse des données en utilisant l'approche par réseaux de neurones. En effet, même si cette approche est extrêmement puissante, elle est également complexe et il est fort probable que nous ne soyons pas parvenus à générer un résultat supérieur à celui obtenu avec les outils actuels par simple manque de maîtrise de l'outil.

II-2) Décisions pertinentes et maladresses :

II-2-a) Remise en cause du pré-traitement :

En prenant en considération à la fois les résultats obtenus et le déroulement du processus, nous croyons que si le projet s'est généralement bien déroulé, le choix ayant eu le plus d'impact est celui de remettre en question notre pré-traitement après avoir identifié le régresseur par Random Forest. Si le pré-traitement des données est une étape capitale au succès du développement de l'algorithme, nous jugeons que celui-ci devrait être ré-évalué au fur et à mesure que l'algorithme évolue afin de s'assurer que les données manipulées permettent un rendement optimal plutôt que d'exécuter cette étape une seule fois en début de projet et d'assumer que le résultat obtenu est valide dans toutes les circonstances. **Nous avions en effet assumé, à tort, que corrélation était synonyme de causalité, ce qui n'est pas le cas** ; la meilleur solution pour choisir nos variables a été de s'appuyer sur une analyse de la valeur de notre fonction de mesure de l'erreur.

II-2-b) Définir un modèle par type d'utilisateur :

Sans aucun doute, l'une des décisions les plus pertinentes que nous ayons pris a été de définir deux algorithmes de prédictions différents pour les utilisateurs casuals et registered ; **le temps passé à analyser la répartition du nombre de locations par type d'utilisateur nous a permis de créer un modèle pertinent**.

Nous avons eu à ce propos la curiosité d'explorer la démarche mise en place par le second de la compétition et il s'avère que, dans cette solution, les deux types d'utilisateurs avaient été séparés en deux modèles comme nous l'avons fait : https://github.com/logicalguess/kaggle-bike-sharing-demand.

II-3) Si on devait recommencer ?

Tout lecteur qui aurait accès à ce rapport prendrait probablement conscience du comportement à adopter vis-à-vis d'un problème Kaggle, riche de notre expérience : il ne faut pas se précipiter dans l'analyse des données et considérer le développement de solution comme un processus itératif prenant en compte la mesure de l'erreur comme marqueur de progrès.

Nous avons appris avec ce projet qu'il n'était pas aussi évident que cela de déterminer avec certitude les variables les plus pertinentes dans un modèle ; si nous devions recommencer le projet, nous développerions probablement un processus de test exhaustif des variables du problème et l'intégrerions à la démarche de test définie en **I-3-a) Démarche de test suivie pour notre solution**. De cette manière, nous pourrions nous assurer, peu importe le modèle choisi, que les variables choisies sont celles qui vont donner les meilleurs résultats pour notre problème.

III – Conclusion :

S’il est facile d’être critique vis-à-vis du déroulement de l’une ou l’autre des étapes du projet en ayant les résultats en main, il est important de se rappeler que **le processus de développement d’un algorithme d’apprentissage machine est un processus itératif** ; il ne peut généralement pas être complété sans commettre d’erreur. Même si nous aurions pu éviter de passer du temps sur l’optimisation de l’algorithme avec la ré-insertion de valeurs écartées hâtivement, ou encore considérer une répartition de Poisson pour la régression linéaire dès début de projet, nous jugeons que ces deux étapes ont tout de même permis de faire avancer le processus de manière globale, en nous offrant des nouvelles pistes de réflexion. Il faut garder à l’esprit que tout exercice d’analyse et de traitement de données massives reste avant tout un **sujet de recherche** et donc qu’**il faut accepter et comprendre l’erreur pour avancer et améliorer ses résultats**.