L2 MI – Mini Projet

Challenge **"Solve Xporters traffic volume problem"** – Groupe **AUTOMOBILE**
Équipe composée de 5 membres :

- Martin BERTHIER <martin.berthier@u-psud.fr>
- Quentin BERTRAND <quentin.bertrand@u-psud.fr>
- Yohann BLACKBURN <yohann.blackburn@u-psud.fr>
- Toufic TALHA <toufic.talha@u-psud.fr>
- Victor LEMAIRE <victor.lemaire@u-psud.fr>

URL du challenge : https://codalab.lri.fr/competitions/652
Dossier GitHub du projet : https://github.com/Feynallein/automobile
URL de la vidéo : https://youtu.be/kttRHi0NZHY

Contexte et description du projet :

Nous sommes un groupe de 5 personnes et nous avons décidé de travailler sur le projet Xporters. L'objectif du projet en tant que responsable d'un stand de limonade situé à côté d'une autoroute, est de prédire le nombre de voitures qui passeront à une date, une heure et des conditions météorologique données. L'ensemble de données contient 58 caractéristiques appelées *features* dont la solution est le nombre de voitures (de 0 à 7280) en une heure.

Ce projet est un problème de régression. Pour évaluer nos données nous utilisons le métrique R^2 **[1]**. Cette dernière mesure la qualité de la prédiction d'une régression linéaire, autrement dit, nous cherchons le un résultat le plus proche de 1 possible. En effet, plus le résultat se rapproche de 1, plus le résultat se rapproche de la réalité et inversement, plus le résultat est loin de 1, plus le résultat s'éloigne de la réalité.

Certains d'entre nous avaient déjà touché à des problèmes de régression le semestre dernier en ayant pris l'option *Introduction à l'apprentissage automatique*, et c'est pour cela qu'ils ont choisi de travailler sur ce projet en particulier, pour d'autres, ce n'était que par curiosité.

Pour résoudre notre problème, nous avons divisé le projet en 3 parties :

- **Pré-processing :** qui sera charger de retravailler les données brutes reçues dans les différents fichiers pour que le modèle puisse les utilisés.
- **Modélisation :** qui sera charger de choisir le meilleur modèle de régression pour notre problème.
- **Visualisation :** qui, à partir des résultats obtenus des deux parties précédentes, sera charger de visualiser nos données à l'aide de tableaux et graphes pour une meilleure compréhension des résultats.

Description du travail de chaque partie :

Partie 1 : Pré-processing (Toufic et Victor) :

En préprocessing notre but est de commencer d'ores et déjà l'entraînement de nos données en vue du résultat final voulu. C'est dans cette partie du machine learning que l'on va préparer nos données d'entraînement grâce à un ensemble de méthodes en vue de les insérer dans notre model pour un résultat optimal. Pour pouvoir faire ça de façon encadrée et guidée, il nous a été demandé de faire certaines actions à travers des questions posées. Ici nous décrirons comment nous les avons résolues.

Tout d'abord la première partie de préparation des données a été de rechercher et soit de « réparé » ou de supprimer des points de données appelés « outlier ». Ces points de données sont dans la réalité soit des erreurs dans les prises de mesures, soit de la nouveauté (dans le monde de la physique par exemple cela serait un nouveau phénomène physique non découvert), pour notre modèle en revanche cela se matérialiserait en tant que points divergeant grandement du ou des clusters principaux. Ainsi ici nous avons utilisé un estimateur qui va donner un « Local Outlier Factor», qui est en réalité le score d'anomalie de chaque donnée. Ce « Local Outlier Factor» mesure la déviation de la donnée par rapport à ces voisins. Enfin les données marquées en tant qu'outlier auront un label -1, il suffit ensuite de les enlever de notre base de données.

Ensuite la deuxième partie du préprocessing a été de réduire la dimension de nos points de données, ici le but est regardé nos features et d'essayer de voir si certaines dépendent d'autres, et ainsi de les fusionner. Afin de faire cela nous avons pris l'estimateur PCA ou « Principal Component Analysis », qui est une réduction linéaire de l'espace dimensionnel en utilisant le « Singular Value Decomposition » ou SVD qui correspond à une opération en algèbre linéaire sur des matrices permettant ainsi de les factoriser, tout en centrant nos données. Cela permet de réduire les dimensions de notre base de données.

Enfin la dernière étape a été de sélectionner les features les plus importantes, ceux qui auront le plus d'impact sur la prédiction finale et ainsi d'éliminer ceux qui ne seront pas essentiels à notre model. Pour faire cela nous avions plusieurs choix mais il se trouve que le plus simple à implémenter a été « ExtraTreesClassifier » couplé à « SelectFromModel ». Le premier permet d'avoir un score basé sur l'importance de chaque features ainsi le deuxième permet de ne prendre que ceux ayant le score le plus élevé.

Lien vers le code : preprocessing.ipynb

Partie 2 : Modélisation (Martin et Quentin) :

Dans cette partie, nous sommes allés sur le site scikit learn**[2]** afin de récupérer plusieurs modèles, nous avons regardé leurs algorithmes et sélectionné ceux qui nous donnaient les meilleurs scores avec nos jeux de données. Le « Linear_model » de Ridge**[3]** est le model que nous avons choisit de garder car il s'agit de celui avec lequel nous obtenions de plus grands scores.

Lien vers le code : model.py

Partie 3 : Visualisation (Yohann) :

Dans cette partie, j'ai créé une première fonction *def set_plot_config()* qui me permettra de donner les mêmes propriétés à toute nos figures qu'on obtiendra. À l'aide de la librairie **seaborn[2]**, j'ai créé deux autres fonctions *(def plot_test_distrib(y_proba, y_test, save_path, title) ET def plot_scores(scores, scores_std, save_path, title))*, qui elles, seront chargées de créer nos graphes pour une meilleure compréhension de nos résultats. En effet, je vais à partir des données récoltées dans les parties précédente, tracer un graphe qui nous permettra de comprendre mieux nos résultats. Il m'a, cependant, été impossible de tester mes fonctions, en effet, je n'ai pu installer la librairie scikit learn**[2]** sur mon ordinateur.
Lien vers le code : visualization.py

Résultats et figure :

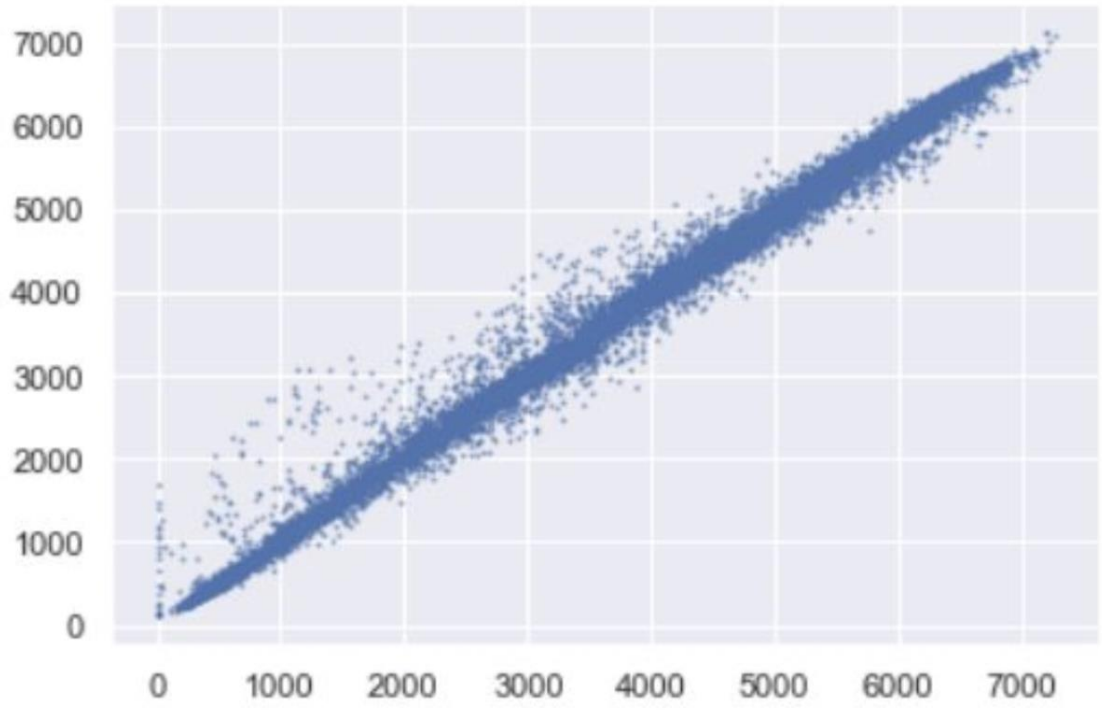

Figure 1 : Graphique entre les résultats réels et ceux prédits

Sur ce graphique, les points bleus représentent les résultats de notre modèle. Les coordonnées de nos points bleus en *(x, y)* sont respectivement, le résultat réel et le résultat prédit. Plus ces derniers forment une diagonale, plus notre modèle est juste, le score se rapprochera donc de 1.

Discussion et Conclusion :

Pour conclure, la leçon tirée de ce projet est que chaque partie est interdépendante, au tout début nous pensions comme la majorité des personnes selon nous, que la partie model est la partie la plus importante du machine learning or cela s'est avéré faux. En effet, la partie préprocessing sert à optimiser le résultat obtenu à partir du model, sans elle le model travaillerait sur un jeu de données erroné ainsi ses résultats ne seraient pas aussi précis qu'avec un travail d'élimination et d'épuration de données.

La partie model qui s'occupe du choix de l'algorithme utilisé est bien évidemment tout aussi importante, nous avons appris qu'ils existent une multitude d'algorithmes différents et que chacun a des caractéristiques qui lui sont propres. Ainsi, le choix d'un l'algorithme erroné peut s'avérer fatal, ce choix peut être basé sur des critères de performances ou de ressources dont on dispose (puissance de calcul) ou d'optimisation alliant rapidité avec efficacité.

Enfin la partie visualisation est probablement celle la plus sous-estimée, notre cours du premier semestre sur l'apprentissage automatique aurait probablement dû s'y attarder un peu. En effet, on s'est rendu compte que, pour nous qui avons le nez plongé dedans durant 6 mois, obtenir des résultats à travers de tableaux et nombres ne dérangeait pas or à la toute fin, des personnes extérieures au groupe ou même le grand public (dans un cas réel) doivent pouvoir faire tourner le programme et obtenir des résultats lisibles et claires qu'ils pourront comprendre facilement. Ainsi cette partie, souvent sous-estimée, fait partie intégrante du machine learning.

Nos conseils pour les étudiants des prochaines années seraient de prendre comme dit plus haut chaque partie au sérieux ainsi que d'avoir un bon groupe où chacun est intégré et travail. Et enfin de ne pas prendre peur au tout début car la bibliothèque sklearn peut être un peu dure à prendre en main, elle utilise des fois des principes mathématique inconnus, mais avec de la détermination et du temps on peut tout faire.

Références :

[1] Coefficient de détermination
[2] Scikit Learn
[3] Model linear Ridge
[4] Seaborn