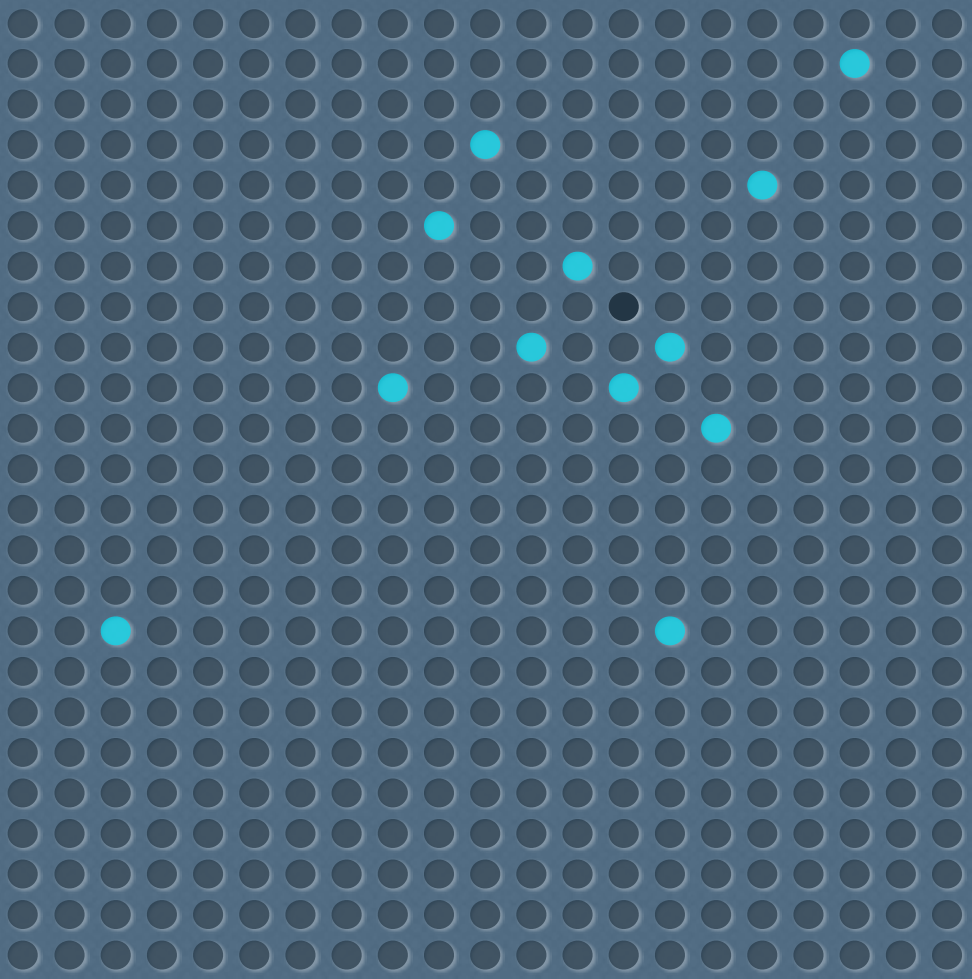


Machine Learning

Boufala Yacine / Fournier Jean-Charles / Idoumou Zein / Khalfoun Rayan

Sous la direction de Jean-Christophe Janodet et Hanczar Blaise

31 Mars 2022

Table des matières

INTRODUCTION

Ce projet consiste à traiter les données que nous avons à disposition et créer un modèle de prédictions pour faire les prédictions les plus précises possible. Les données concernent des établissements Rossmann, une chaîne de droguerie. Ils mettent à notre disposition diverses informations sur ses différents établissements comme, les jours d'ouverture, le nombre de clients ou encore la distance par rapport au concurrent direct, et c'est à nous, grâce à ses informations, de prédire le chiffre d'affaires potentiel d'un nouveau magasin avec le plus de fiabilité possible.

Nous nous servirons des bibliothèques Seaborn et Matplotlib pour l'analyse des données, Numpy et Pandas pour les manipuler et Scikit-learn pour les modèles de prédictions.

Ici, il s'agira d'un modèle de régression puisque nous cherchons à prédire le chiffre d'affaires qui est donc une valeur numérique. Pour nous aider dans notre mission, Scikit-learn mets à notre disposition pléthore de modèles d'apprentissage dont nous détaillerons ceux utilisés.

Ce rapport s'articulera autour de deux axes qui suivent globalement le cours chronologique de notre projet, même si nous avons fait de nombreux aller-retour entre les deux afin d'y apporter des amélioration.

Dans un premier temps, nous verrons plus en détail l'analyse et le traitement des données et l'importance de cette tâche dans un exercice comme celui-ci. Puis nous parlerons dans un second temps des modèles testés et comment nous en sommes arrivés au modèle choisi, avant de conclure par un bilan de notre expérience et les difficultés que nous avons rencontrées.

PREPROCESSING

Le préprocessing, ou traitement des données, est la première étape dans un exercice de Machine Learning comme celui qui nous a été donné.
Cette étape est sûrement la plus importante et celle à laquelle il faut accorder le plus de soin et de rigueur. Elle consiste à analyser les données brutes que l'on a en notre possession et d'en dégager, à l'aide de différentes méthodes, le contenu le plus pertinent et utile pour résoudre notre problème. En effet le plus souvent les DataFrame fournis ne sont pas directement utilisables et nécessitent des modifications afin d'être entièrement exploitable et fournir un maximum d'informations.

Ces modifications peuvent prendre plusieurs formes comme par exemple la suppression de colonnes ou la normalisation des données. Certaines peuvent sembler superficielles ou inutiles dans un premier temps, mais ont en réalité une importance capitale sur la qualité de la prédiction.

Dans notre sujet, il nous a été fourni 4 DataFrame différents :

- train.csv - Contenant les données des magasins
- test.csv - Notre DataFrame de test
- sample_submission.csv – le format du fichier attendu par Kaggle
- store.csv - contenants des informations supplémentaires sur les magasins

Nous les avons dans un premier temps importé, avant de fusionner « train » et « store » afin d'obtenir un DataFrame contenant toutes nos données relatives aux magasins, il en va de même pour «test » et « store » pour les mêmes raisons. Comme nous pouvons le voir sur le notebook grâce à la commande ".info()" nos nouveaux train et test comportent respectivement 17 et 16 colonnes (les 17eme étant notre chiffre d'affaires à prédire)

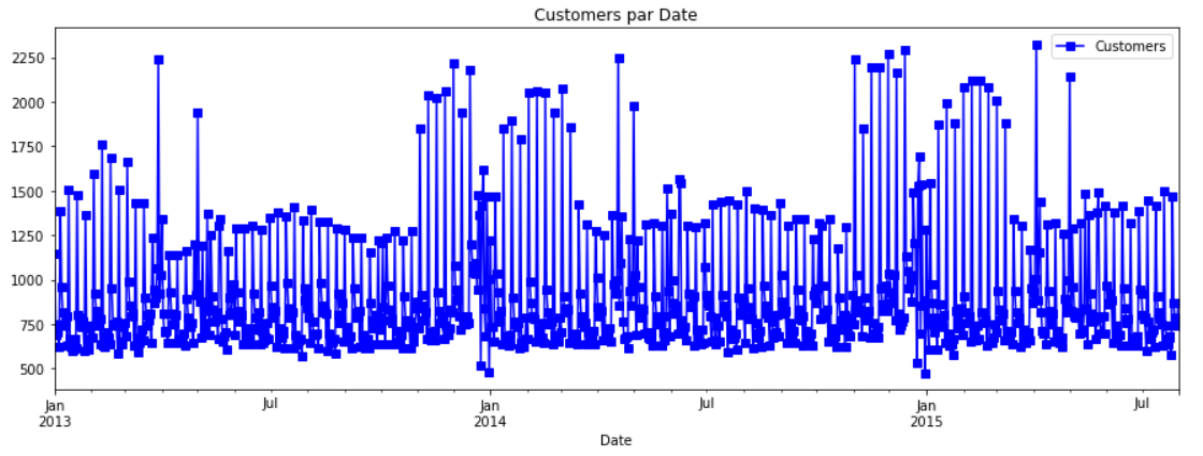
Nous avons ensuite pris la décision de retirer du jeu de données les magasins fermés (Colonne [open]) car un magasin fermé ne nous intéresse pas dans la prédiction d'un chiffre d'affaires.
Nous avons par la suite supprimé cette colonne devenue inutile car ne contenant que des 1.

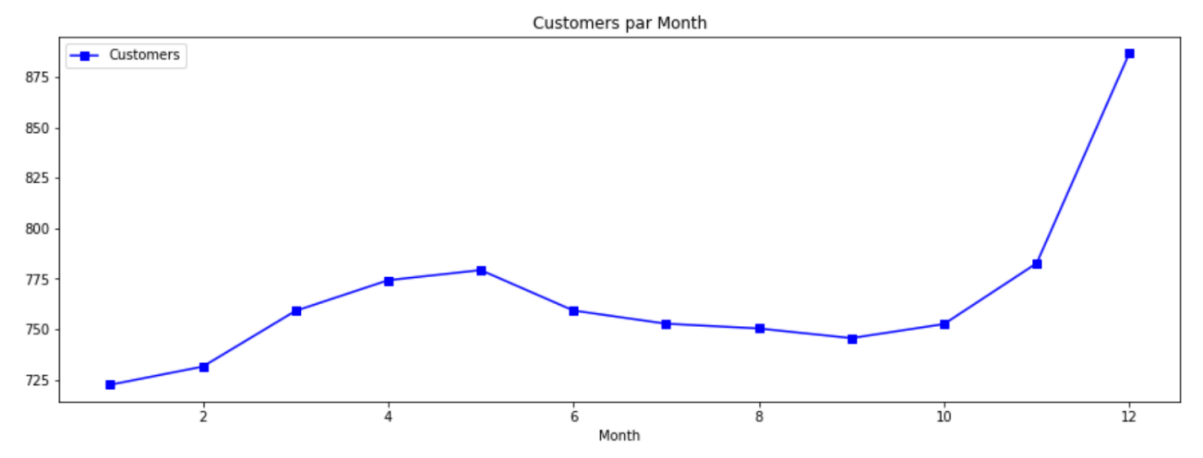
L'étape suivante fut le remplissage des valeurs vides dans « train » et «test ». Effectivement, il arrive très fréquemment que les données comportent des valeurs manquantes qu'il faut combler pour lancer l'apprentissage.

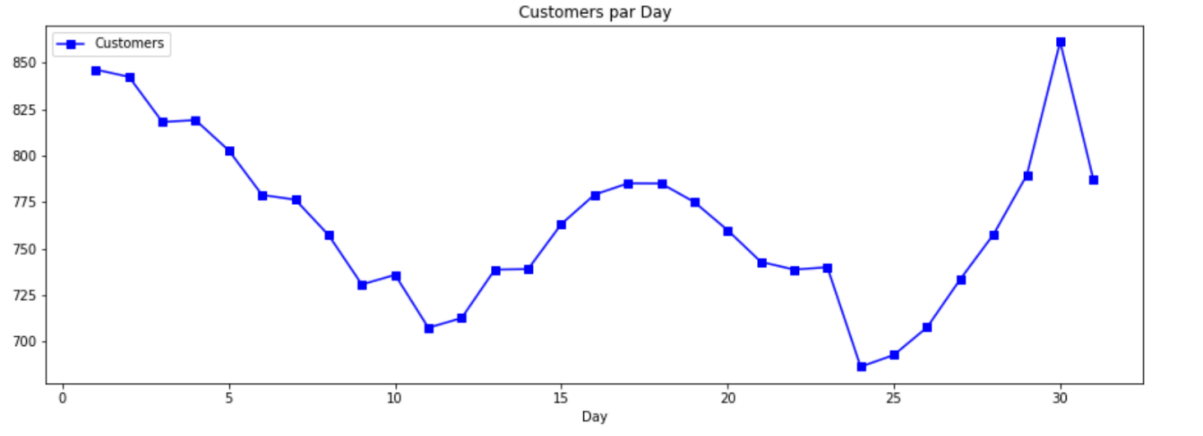
Dans les colonnes avec des valeurs manquantes, nous remplaçons toutes les valeurs manquantes par la moyenne des valeurs de la colonne pour essayer d'éviter au maximum un biais.

Une fois, cela fait, il a fallu gérer le format de "date". Pour se faire, elle a été découpée et 5 colonnes : « year », « date », « month », « day » et « weekOfYear », de type int, pour permettre leur future manipulation.

On peut à présent analyser nos données de manières plus fines. Ci-dessous le chiffres d'affaires en fonction de la date, puis des mois et enfin des jours. On peut observer que le chiffre d'affaires est bien plus important en décembre, et au début et à la fin du mois.







Pour réduire le nombre de colonne dans un souci d'optimisation, nous avons mis en place la méthode scale_df qui a pour rôle de réunir « CompetitionOpenSinceYear » et « CompetitionOpenSinceMonth » en « CompetitionOpen ». Il en va de même pour « Promo2SinceYear » « Promo2SinceWeek » qui deviennent « Promo2Open ».

Cette manipulation permet de réduire 4 colonnes en 2 sans perdre en qualité d'information.

Nous décidons ensuite de séparer nos données en deux parties : les données numériques et les données textuelles. Sur la première catégorie, on applique un MinMaxScaler qui permet une mise à l'échelle de ces valeurs.
Sur les données textuelles, on applique la méthode du OneHotEncoder qui permet de créer une colonne composée de valeur binaire pour chaque possibilité (StateHoliday_0, StateHoliday_a, StateHoliday_b, StateHoliday_c par exemple) avant de supprimer les colonnes de base et ainsi pouvoir utiliser les data dans notre prédiction (uniquement numérique).

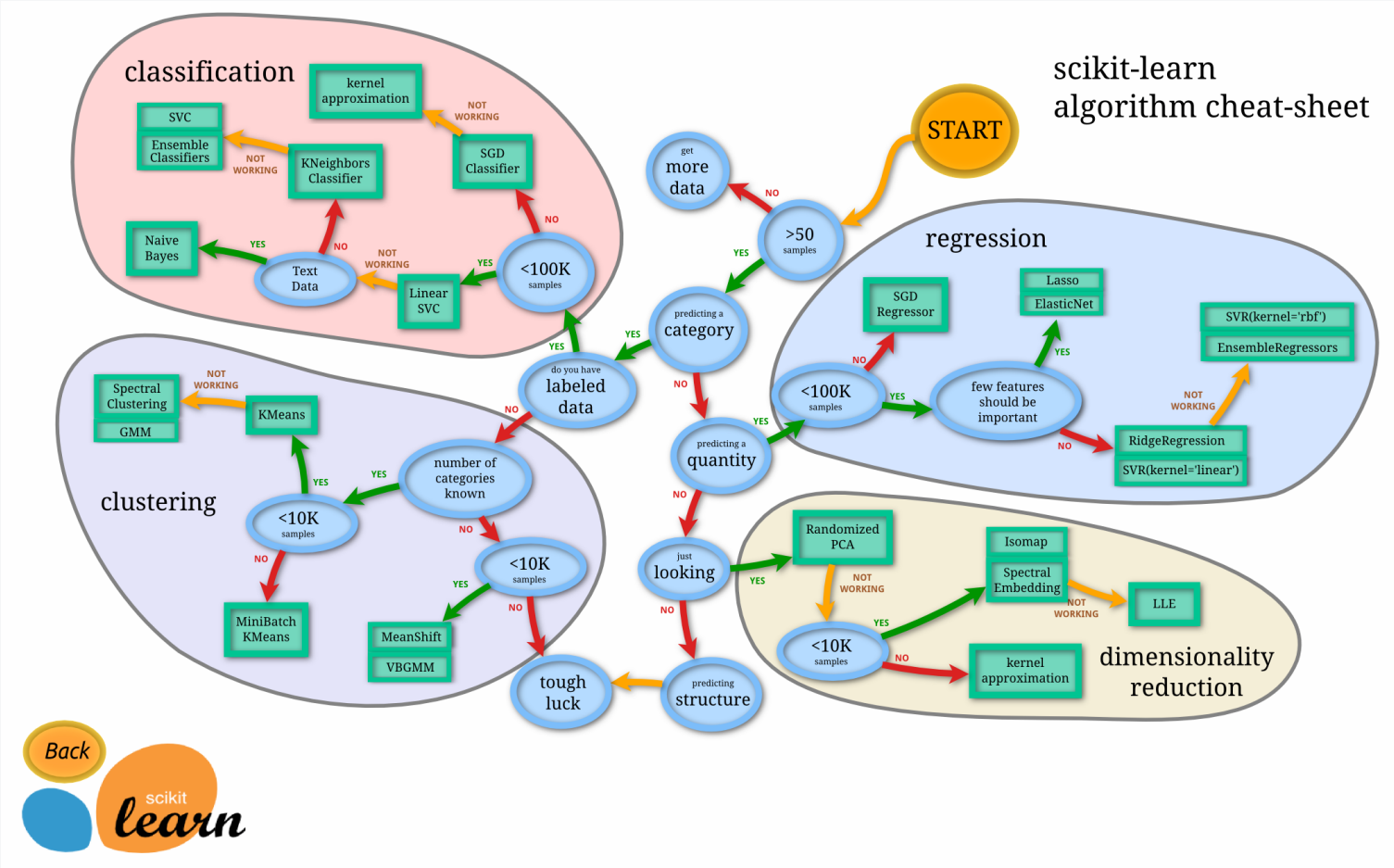
Enfin, on définit notre Y comportant la colonne des ventes et X qui se compose de toutes les autres colonnes. Puis l'on applique la méthode Train_test_split qui va se charger de diviser en 2 parties notre X et notre Y. Une partie qui servira à l'apprentissage du modèle, l'autre partie servira à le tester. Ici et après plusieurs essais, nous avons décidé de réduire la proportion de données dédiée au test à 0,001 (0,1%) des données complètes.

Maintenant que les données ont été épurées et sont désormais utilisables par les modèles de prédiction, nous pouvons passer à la modélisation de notre solution.

MODÉLISATION

Une fois nos données traitées, normalisées et rendu utilisable, c'est le moment de les utiliser pour résoudre notre problème.Nous avons dans un premier temps cherché à utiliser les modèles vus en cours, à savoir le modèle SVM, qui dans notre cas est un SVR puisque l'on cherche une régression. Ce modèle était très long à entraîner ce qui nous a conduit a cherché dans la documentation de Scikit-Learn une solution à ce problème. Nous sommes alors tombés sur un guide qui oriente les utilisateurs sur le modèle qui convient le mieux à leur problème. Nous avons donc compris que le SVR était inadapté à la quantité de données en notre possession. Nous nous sommes alors dirigés vers le modèle « SGD Regressor » comme le conseillait la map.

Ce modèle a vite été abandonné puisque bien que beaucoup plus rapide, les résultats donnés était médiocre, malgré l'essai de différents paramètres (environ 0.40 de score public kaggle). Nous nous sommes donc penchés sur un autre type de modélisation, les ensemble learning.



L'ensemble Learning est une technique qui consiste à entraîner plusieurs modèles pour ensuite considérer l'ensemble de leurs prédictions.Les trois grandes méthodes à utiliser cette technique sont les suivantes :

Le Stacking : on entraîne différents modèles sur les données, puis on entraîne un modèle sur les prédictions des premiers.

Le Bagging : on entraîne plusieurs copies d'un même modèle avec une partie aléatoire du dataset avant de regrouper leurs prédictions pour n'en faire qu'une finale obtenue par la majorité des prédictions. Les parties du dataset sont produites par un algorithme d'échantillonnage appelé Bootstrapping qui permet aux modèles d'avoir certaines données en commun, ce qui est nécessaire pour obtenir une majorité parmi les prédictions. Ici, nous sommes dans un cas de régression alors il s'agira plutôt de la moyenne des prédictions.

Le Boosting : on entraîne l'un après l'autre des modèles qui tenteront de corriger les erreurs du modèle précédent. Les modèles sont complémentaires pour permettre de palier les faiblesses des autres.

Pour le bagging, les modèles sont surentraînés et la mise en parallèle permet de réduire la variance de la foule alors que dans le Boosting, les modèles sont sous-entrainés et la foule permet de réduire le biais.

Le Boosting étant interdit dans le cadre de ce projet, nous utiliserons RandomForrestRegressor qui est un modèle qui utilise du bagging, et StackingRegressor pour faire du Stacking.

Malgré de nombreux tests sur les différents modèles utilisés ainsi que leurs différents paramètres, le StackingRegressor n'a jamais donné de résultat particulièrement convaincant contrairement au RandomForrestRegressor qui a immédiatement donné des prédictions précises. Nous nous sommes donc concentrés sur son optimisation. Cette étape passe principalement par la modification des hyperparamètres et nous permet de nous approcher d'un score plus satisfaisant. (Environ 0.16 score public kaggle)

Pour trouver les paramètres les plus pertinents ainsi que leurs meilleurs combinaison, nous avons utilisés RandomSearchCv ainsi que GridSearchCv. Ils prennent chacun une liste des paramètres à tester, ainsi que les intervalles de valeur (ou les options possibles quand il ne s'agit pas de valeur) dans lesquels nous voulons chacun des paramètres.

Ils ne fonctionnent pas pareil, mais ils renvoient tous les deux une liste avec les combinaisons de paramètres testés sur le modèle, ainsi que le score qu'il a obtenu une fois entraîné. On peut accéder au meilleur modèle, aux meilleurs paramètres et à différentes données.

La différence entre les deux est que GridSearchCv teste chacune des combinaisons possibles alors que l'autre ne teste qu'un certain nombre donné. Avec RandomSearchCv, on peut balayer un grand nombre de combinaison et voir quels paramètres sont importants. Puis chercher précisément les meilleurs paramètres avec GridSearchCv.

Dans un souci d'optimisation maximale, nous avons décidé de reprendre la phase de préprocessing pour essayer d'améliorer encore la précision du modèle de prédiction. Nous avons modifié le remplacement des valeurs manquantes par des 0 plutôt que par la moyenne de la colonne.

À la suite de cette étape, le score public était d'environ 0.153. Nous avons alors essayé l'outil « SelectKBest » pour évaluer l'impact de nos différentes features pour mieux comprendre et cibler celles qu'on devrait supprimer. Nous utilisions la fonction de score par défaut, à savoir « f_regression » et obtenions des résultats incohérents. Cela nous a conduit à faire une sélection basée sur l'observation et l'analyse manuel des résultats et cela nous a permis de drastiquement réduire le score passant sous la barre de 0.15.

Le problème n'était donc pas l'outil mais la fonction utilisée avec, nous avons donc testé avec « mutual_info_regression ». Il s'agit d'une technique de sélection des features qui vise encore à réduire la taille de nos données en calculant le degré d'entropie par rapport à notre cible de chaque colonne du DataFrame. Cela nous permet d'éliminer les informations superflues et conduit à une amélioration de la précision du modèle.

Cette série d'amélioration nous a permis d'arriver à un score public de 0,136 et un score prive de 0,148

CONCLUSION

La réalisation de ce projet nous a permis d'en apprendre davantage sur l'intelligence artificielle et plus précisément sur le Machine Learning et le traitement de données. En effet, ces deux sujets, que nous avons l'occasion d'expérimenter dans un projet concret pour la première fois, nous ont permis de nous rendre compte à la fois de la multitude de domaines d'applications qu'offre ces techniques, mais aussi d'atténuer la vision très théorique et mathématique que nous pouvions nous en faire par l'intermédiaire des cours universitaires. C'est donc un exercice qui a été très formateur et nous servira dans la suite de notre cursus. De plus, chaque travail de groupe nous permet de nous familiariser davantage avec l'environnement dans lequel nous serons amenés à travailler en apprenant à nous organiser de plus en plus dans la gestion des tâches dans le cadre de gros projets.