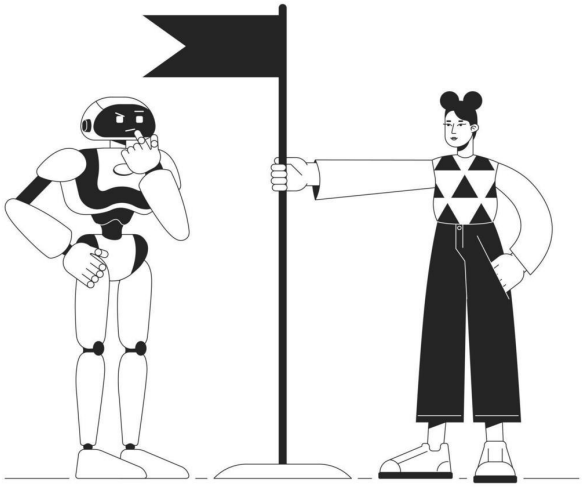
IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

EXECUTIVE SUMMARY

Challenge 2 : Apprentissage supervisé

Prévision de la qualité d'une solution d'un problème d'optimisation de transport



Realisé par

NADIA SYLLA
UYEN NGUYEN
RUXIN ZHENG
JIHANE LAGNAOUI
HICHAM CHEKIRI

Table des matieres

Contexte & Introduction

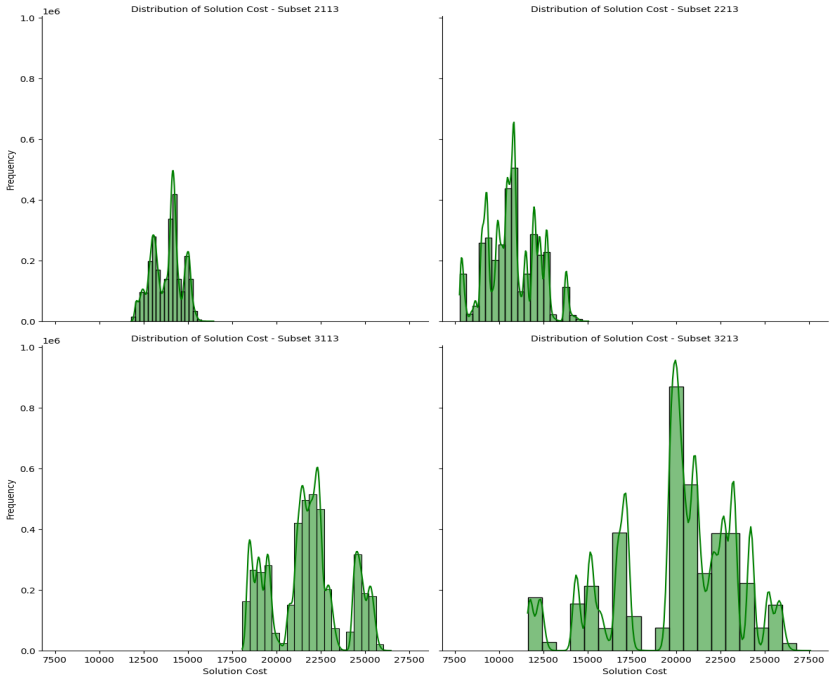
Ce projet s'inscrit dans le cadre d'un défi d'apprentissage supervisé ayant pour objectif de comprendre et prédire la qualité des solutions générées par un algorithme génétique d'optimisation pour le problème de tournées de véhicules (CVRP).
Le CVRP consiste à déterminer quel véhicule doit livrer quel client et dans quel ordre pour minimiser la distance totale parcourue.
Il ne s'agit pas de générer des solutions au problème lui-même, mais plutôt d'utiliser les méthodes d'apprentissage supervisé pour prédire la qualité des solutions sans connaître les détails ou les coûts des solutions.

Traitement Préalable de données

Comme première étape, nous avons regroupé l'ensemble des instances d'une configuration du problème en une seule base de données. Ainsi, pour chaque configuration **2113, 2213, 3113 et 3213**, nous avons établi une base de données, regroupant les 27 instances respectives.
Pour chacune des 4 bases de données créées, nous avons vérifié la présence ou non de données manquantes (Null), vérifié le type et nombre de features (colonnes). Cette analyse descriptive a permis d'observer que la variable **S7** contenait que des valeurs nulle "0", due à un bug lors de l'extraction des données (en amont). Nous avons donc fait le choix de l'écarter de l'analyse et de la supprimer des 4 bases de données.

Distribution de la variable Cost

Les histogrammes suivants montrent la distribution des coûts des solutions pour les scénarios **2113, 2213, 3113 et 3213**. Globalement, les coûts des solutions au problème avec **'dépôt' centré** (2113 et 2213) ont un coût moindre que les deux autres configurations (3113, 3213).



- 2113 - Dépôt Centré, Clients Aléatoires

La distribution des coûts a plusieurs pics distincts. Ceci pourrait indiquer que certaines distances étant plus optimales que d'autres. La présence de multiples modes pourrait entraîner une variation de coûts des solutions en fonction de la position des clients.

- 2213 - Dépôt Centré, Clients en Clusters :

Cette distribution est similaire au premier scénario. Les clusters de clients pourraient permettre un circuit plus

efficace, les clusters peuvent conduire à des solutions qui exploitent la proximité des clients, résultant en des coûts potentiellement plus faibles pour certaines solutions.

- 3113 - Dépôt en Coin, Clients Aléatoires :

Cette distribution est largement dispersée, le coût général semble être plus élevé, ce qui peut être dû à l'augmentation des distances moyennes parcourues, puisque le dépôt est moins central.

- 3213 - Dépôt en Coin, Clients en Clusters :

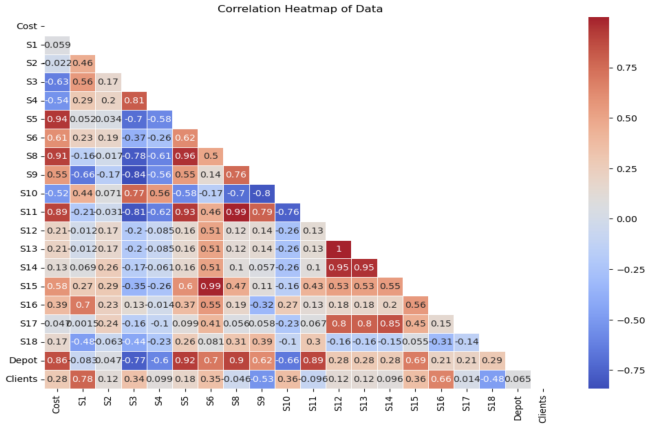
La distribution des coûts est très remarquable, ce qui suggère l'existence de certaines solutions qui sont largement optimales Cela pourrait être le résultat des clients au sein de clusters, même si le dépôt est situé dans un coin. Les coûts semblent concentrés autour de valeurs définies, ce qui permet de dire que les solutions les plus privilégiées sont celles qui émergent lorsque les clients sont regroupés en clusters malgré la position du dépôt.

Fusion des données en une seule base de données

Étant conscient qu'il faudra par la suite pouvoir prédire **un seul modèle de prédiction du coût**, capable de prédire, indépendamment des configurations, nous avons fait le choix de concaténer les 4 bases de données précédentes en une seule base de données. Afin de ne pas perdre l'information de la configuration, qui pourrait justifier certaines distributions des coûts et ou la qualité de la prédiction, nous avons décidé de définir 2 nouvelles features **Dépôt** et **Clients** par one hot encoding. Le tableau suivant regroupe les informations prises par ces 2 variables en fonction de la configuration.

Scenario	DEPOT	CLIENTS
depot centered and clients in random positions	0	1
depot centered and clients in clusters	0	0
depot in a corner and clients in random positions	1	1
depot in a corner and clients in clusters	1	0

Matrice de corrélation et analyse statistique des features



La matrice de corrélation ci -dessus montre les relations linéaires (Pearson) entre les différentes variables de cet ensemble de données. Ci-après les points clés que l'on peut en déduire:

- Corrélation avec la Variable Target(Cost) :

Plusieurs variables montrent une forte corrélation avec la variable Cost. Celle-ci est fortement corrélée positivement avec plusieurs variables explicatives à savoir S5,

S8, S11, **Dépôt**, et varient dans le même sens, on pourrait dire que ces variables peuvent être des **features déterminantes** pour déterminer la variable ‘Costs’ .

- Faible Corrélation ou Indépendance avec cost:

Certaines variables comme **S13 et S14** montrent une très faible corrélation, cela peut signifier qu'elles ne sont pas pertinentes pour prédire le coût. De plus, elles affichent une redondance (corrélation) avec la variable S12.

★ **Variable 'Depot'**

La variable 'Dépôt' a des corrélations notables avec plusieurs caractéristiques, ce qui suggère que la position du dépôt (centré ou non) pourrait affecter d'autres aspects des solutions.

Pour examiner de plus prêt les corrélations des features S5, S8, S11 et Dépôt avec la variable ‘Cost’, nous avons calculé la p-value afin de confirmer ou non nos observations. Ces résultats montrent que toutes les variables sont statistiquement significatives ce qui permet de dire qu' une relation linéaire avec le coût existe.

Variable	Pearson correlation	P value
S5	0.99	1,39 e-19
S8	0.97	2,5 e-13
S11	0.96	2,81 e-12
Depot	0.97	3,75 e-13

Prediction du cout

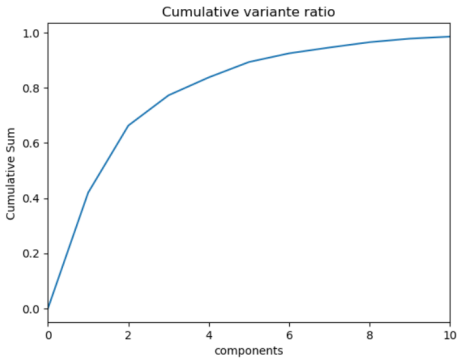
Afin de développer un algorithme capable de prédire la variable Cost en fonction des différentes features, nous avons eu recours a 3 modèles de régression : KNN, Decision tree régression, XGboost.

Traitement des données et création de données input et output

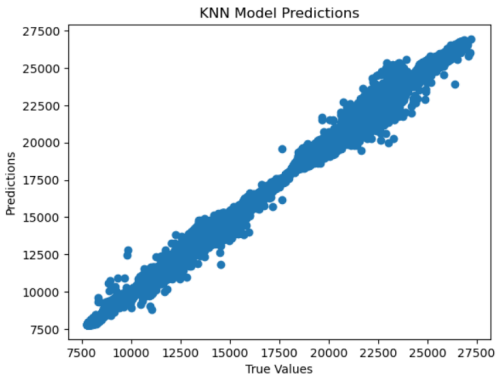
Compte tenu de la forte corrélation entre les features S12, S13 et S14, nous avons décidé d’en garder qu’une, S12, afin de réduire les dimensions pour l'implémentation des algorithmes d'éviter la redondance d’information dans nos données.
De ce fait, nous avons défini les inputs X= [S1, S2, …, S6, S8, …., S12,S15, …, S18, Depot, Client], et un output Y= [cost], qui est l’information étudiée.

Prediction avec KNN

Pour effectuer le KNN, nous avons effectué un PCA afin de réduire les dimensions des données d'entrées. Avec une dimension de 10, nous avons pu retenir plus de 98% des informations (cf courbe de ratio cumulé)



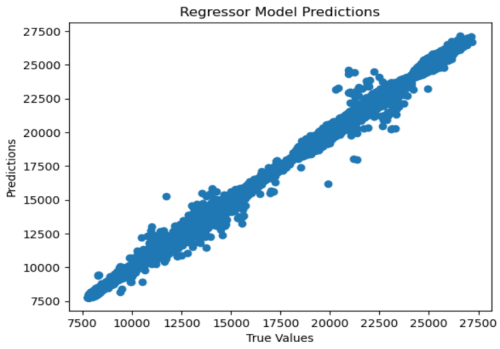
Après l'entraînement du modèle KNN et de son évaluation, nous avons pu obtenir la régression ci-après :
Le graphique montre que le modèle a bien pu définir une régression entre les données d'entrée et de sortie. Cette information a été confirmée par le calcul du MAE et du MSE.
On a obtenu un MAE de 57.69 , un MSE de 8415.85 et un RMSE: 91.74.



Decision Tree regressor

Pour ce modèle, nous avons repris directement les données d'entrée de sorties X et Y sans remettre à l'échelle ni réduire la donnée X. Nous avons toutefois utilisé la fonction train_test_split pour egalement creer deux données d'entraînement (X_train_2, y_train_2) et deux donnees test (X_test_2, y_test_2). Après avoir défini et entraîner le modèle, nous avons fait un test de prédiction du modèle decision tree regressor (model_DT) et nous avons obtenu la régression suivante:

En comparant cette courbe avec la régression KNN, on peut visuellement voir que la régression obtenue avec le DT est nettement plus fine (la droite créée par les nuages de points est plus fine et se rapproche plus d'une droite). Cette différence visuelle se confirme par la calcule du MAE et du MSE qui ont une valeur plus faible. En effet, pour ce deuxième modèle, on obtient un MAE de 46.69, un MSE de 5484.77 et un RMSE de 74.06.



XGBoost

Nous avons entraîné nos données par un modèle de régression basé sur la méthode XGBoost avec les données (X_train_2, y_train_2, X_test_2, y_test_2) définies précédemment. Avant toute interprétation de résultat, il faut noter qu'**une grande différence de temps** a été observée dans l'obtention de résultat.
Le temps de computation et de calcul a pris moins de temps que toutes les méthodes précédemment utilisées, avec le même jeu de données. Nous avons obtenu une valeur de MAE de 71.81, un MSE de 9971.53 et un RMSE de 99.86. Bien que le MSE soit plus élevé, cette méthode semble être moins complexe a implémenter et permet d'obtenir un modèle entraîné plus rapidement.

Comparaison des modèles de prédictions du ‘cost’

Globalement, nous avons obtenu la meilleure prédiction du ‘cost’ par le modèle basé sur le decision tree regressor.

Model	KNN	Decision Tree	XGBoost
MAE	57.69	46.69	71.81
RMSE	8415.85	74.06	9971.53
MSE	91.74	5484.77	99.86

En se basant sur ces résultats, Decision Tree semble être le modèle le plus performant ,car il a les scores les plus bas en MAE et RMSE, indiquant une meilleure précision et une meilleure gestion des erreurs. XGBoost semble avoir la moins bonne performance parmi les trois ce qui peut paraître contre intuitif mais cela peut être expliqué par le choix des hyperparamètres de l’algorithme.

Transformation d'un problème de régression en classification pour l'optimisation de Coûts

Définition des Seuils pour les Catégories et création d’une feature

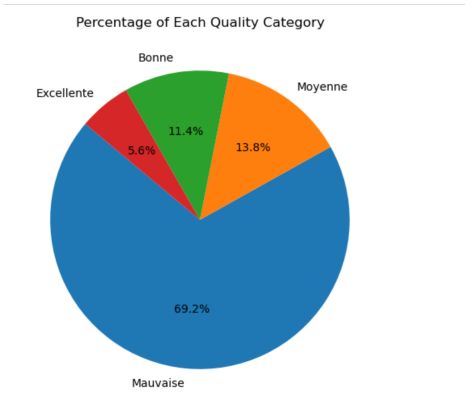
Pour chaque instance de notre ensemble de données, nous avons commencé par identifier le coût minimum, représentant la meilleure solution connue. Sur cette base, nous avons établi 4 catégories : excellente, bonne, moyenne et pauvre. Ces catégories ont été définies comme suit:

- Excellente : Coût ≤ Coût Minimum + 25%
- Bonne : Coût Minimum + 25% < Coût ≤ Coût Minimum + 50%
- Moyenne : Coût Minimum + 50% < Coût ≤ Coût Minimum + 75%
- Pauvre : Coût > Coût Minimum + 75%

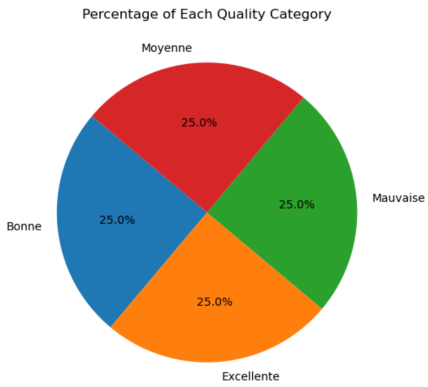
Ainsi, une nouvelle colonne ‘Quality’ a été ajoutée à notre jeu de données pour refléter cette classification du coût. Chaque instance a été assignée à l'une des quatre catégories en fonction de son coût par rapport au coût minimum.

Gestion du déséquilibre des classes

Nous avons rencontré le problème majeur de déséquilibre des classes dans notre ensemble de données qui contribuerait au bias de model. Pour y remédier, nous avons utilisé des techniques de rééchantillonnage, permettant d'équilibrer la distribution des classes et d'améliorer la fiabilité de notre modèle de classification. Donc on a mis en place Synthetic Data Generation SMOTE (Synthetic Minority Over-sampling Technique) pour le Suréchantillonnage et RandomUnderSampler pour le sous-échantillonnage. Les figures ci-dessous présentent les pourcentages des classes avant puis après le rééquilibrage par rééchantillonnage.



Dataset avant SMOTE



Dataset après SMOTE

Modélisation et évaluation

Pendant le processus d'entraînement, on a rencontré un problème de manque de stockage. Même si on a fait le PCA pour diminuer la dimension de dataset, on a décidé d'entraîner les modèles dans un moins grand dataset, qui contient 1% de la dataset original.
Nous avons essayé deux types de model: Random Forest Classifier et SVM, pour chaque modèle, on a trouvé des meilleurs paramètres grâce à la gride. Le modèle choisi pour cette classification était le Random Forest Classifier avec estimateur = 100 . Les performances ont été mesurées à l'aide de matrices de confusion et du score F1, offrant une vision complète de la précision et de la fiabilité de notre modèle.

	precision	recall	f1-score	support
Bonne	1.00	1.00	1.00	19633
Excellente	1.00	1.00	1.00	19712
Mauvaise	1.00	0.99	1.00	19471
Moyenne	0.99	1.00	0.99	19603
accuracy			1.00	78419
macro avg	1.00	1.00	1.00	78419
weighted avg	1.00	1.00	1.00	78419

Accuracy Optimized: 0.9968757571506905

Résultat de RF