Rapport du challenge

**Bearingpoint - Challenge**

**Groupe**:

Raphael Attali

Renata Baptista

Hugo Hubert

Saousan Kaddami

Table des matières

[**I - Introduction**](#_gmrneu1c8otq) **3**

[**II - Preprocessing**](#_lyu007v0q98o) **4**

[**III - Features**](#_379fdf0jbdm) **5**

[1 - Features existentes](#_ki129e95kftz) 5

[2 - Features crées](#_f05ur11efwkx) 6

[3 - Features sélectionnés](#_f3hvcztlb5mx) 7

[**III - Machine learning**](#_61dubaoqfegx) **8**

[1- Logistic regression :](#_q5b3cqc073an) 8

[Choix :](#_try7nwrr2ndd) 8

[Résultats :](#_jorbjc5qygih) 8

[2- Random forest :](#_uqvgyqjhkql0) 9

[Choix :](#_7f48mcm424mg) 9

[Résultats :](#_4ib6d8w0yo2e) 9

[3- Adaboost :](#_p55royk4zrsy) 10

[Choix :](#_ekci7n5xuu6x) 10

[Résultats :](#_7eg7arqfcwhv) 10

[4- Gradient Boosting classifier:](#_hyll173ym1k) 11

[Choix:](#_q90ptmx257iv) 11

[Résultats :](#_42vucvyfe8xk) 11

[**IV - Conclusion**](#_tt0h0ddr9ols) **13**

# 

# I - Introduction

Le but de ce rapport est d’expliciter les étapes de la solution du challenge Bearing Point. Dans ce challenge, nous essayons de mieux comprendre le comportement des acheteurs de chaussures pour répondre à la question: Qui retourne les achats et pour quelles raisons?

Il est possible de diviser ce travail en quatre étapes: comprendre le base de données, extraire de features d’importance pour ce problème, modéliser et choisir le meilleur modèle parmi ceux étudiés. On finira notre rapport par une conclusion générale.

La base de données contient 1067290 entrées avec la réponse correcte. En contraste, 800468 sans réponse. Pour éviter le *overfitting*, le premier conjoint set a été divisé en training et test set avec la proportion 70/30. Ce projet a été codé en Python 3.

# II - Preprocessing

1- Elimination de NaNs (cleaning):

2- Conversion en types appropriés :

Les détails de la conversion sont purement informatiques et nous n’avons pas trouvé pertinent de les faire figurer dans le rapport. Nous avons notamment effectué les conversions suivantes :

* Tous les données de date utilisés ont été convertis en *Data Time Series*. En effet, ce type permet de faciliter l’étude de ces dates car de nombreuses fonctions permettent de tirer pleinement parti de ces données.
* Le feature prix était du type string et nous l’avons transformée en *float64,* donc en donnée numérique exploitable.

3- Etude de corrélation:

La corrélation donne une indication sur les relations de variations au sein de deux variables ou features. Si deux variables varient dans la même direction, elle sont positivement corrélées. En revanche, elles le sont négativement si jamais elles varient dans des directions opposées.

Nous pouvons calculer la corrélation entre deux attributs et construire ainsi la matrice de corrélation. On va alors afficher et étudier cette matrice pour avoir une idée sur le degré de corrélation entre toutes les *features,* deux à deux.

Étudier la corrélation est très utile car dans certains problèmes de machine learning, comme les algorithmes linéaires ou la régression logistique, puisqu’on peut avoir une mauvaise performance si on a des variances très corrélées en entrée. Ainsi, nous avons jugé utile de d’abord étudier la corrélation avant de procéder à l’extraction de features et à la classification. Cette première étape nous a permis de mieux recentrer nos choix de features.

# 

# 

# 

# 

# 

# III - Features

## 1 - Features existantes

Le premier pas pour résoudre le problème est évaluer les données disponibles. Il y a 4 type de tables différents: products, customers, order, return. Ces tables ont des différents features avec lesquels est possible de jouer.

La premiere, il y a 25 features: *VariantId, ProductColorId, ProductId, GenderLabel, MarketTargetLabel, SeasonLabel, SeasonalityLabel, BrandId, UniverseLabel, TypeBrand, ProductType, SupplierColor, MinSize, MaxSize, CalfTurn, UpperHeight, HeelHeight, PurchasePriceHT, IsNewCollection, SubtypeLabel, UpperMaterialLabel, LiningMaterialLabel, OutSoleMaterialLabel, RemovableSole, SizeAdviceDescription*.

Sur la deuxième, il y a 5 features: *CustomerId, CountryISOCode, BirthDate, Gender, FirstOrderDate*.

Sur la troisième, il y a 20 features: *OrderNumber, VariantId, CustomerId, LineItem, OrderStatuslabel, OrderTypelabel, SeasonLabel, PayementModeLabel, CustomerTypeLabel, IsoCode, DeviceTypeLabel, PricingTypeLabel, TotalLineItems, Quantity, UnitPMPEUR, OrderCreationDate, OrderShipDate, OrderNumCustomer, IsOnSale, BillingPostalCode.*

Enfin La quatrième contient 4 features: *OrderNumber, LineItem, ReturnQuantityBin ReturnQuantity.*

Une tentative naturelle serait d’essayer tous les features disponibles et après faire un selection. Cependant, en raison de la taille de la base de données et du nombre important de features, cette approche n’est pas viable à cause de limitations de mémoires.

En analysant ces données, nous avons essayé de sélectionner celles qui décrivent le produit, le type d’achat et le client. Finalement, nous avons choisi les features ci-dessous, et nous avons pris le soin d’expliquer pourquoi nous avons pensé qu’elles étaient importantes pour chacune d’entre elles.

* LineItem: les premières produits dans la commande sont la raison pour laquelle le consommateur a accédé au site.
* OrderTypelabel: si le produit est un premier achat ou avait déjà été retourné.
* SeasonLabel\_x: différents types de produit peuvent être plus ou moins facile de choisir en ligne.
* PayementModeLabel: est-ce qu’il est facile d’être remboursé vis à vis du moyen de paiement utilisé ? Si c’est facile, on s’attend à plus de retours.
* CustomerTypeLabel: Si le client achète pour la première fois dans ce magasin, il est possible qu’il n’adhère pas au service
* IsoCode: différents pays peuvent avoir des habitudes différentes, et donc on peut imaginer qu’un pays aura un taux de retours plus important selon la culture.
* DeviceTypeLabel: acheter via smartphone peut rendre difficile la visualisation de certaines informations par exemple.
* TotalLineItems: un client qui achète plusieurs articles peut vouloir juste les essayer, et donc on s’attend au retour de certains d’entre eux. Mais il peut aussi être un client habitué au magasin, aux tailles et aux produits en général, et donc ne rien renvoyer finalement.
* Quantity: Même constat que précédemment, on ne sait pas si le client achète beaucoup pour essayer.
* UnitPMPEUR: le valeur, si un produit est peu cher, on aura plus de réticences à vouloir le changer, car la démarche ne vaut pas le coup pour se faire rembourser aussi peu d’argent.
* OrderCreationDate: voir la prochaine section.
* BrandId: une marque peut avoir plus de retours compte tenu de la qualité, des modèles et des tailles
* ProductType: différents types de produit peuvent être plus ou moins simples à commander en ligne.
* IsOnSale: quelqu’un peut acheter un produit en raison d’une promotion et du prix attractif, et finalement ne pas l’apprécier.
* CountryISOCode: différents pays peuvent avoir des habitudes différentes, et donc on peut imaginer qu’un pays aura un taux de retours plus important selon la culture.
* BirthDate: on s’attend à ce que la différences de générations influe sur les retours, des personnes plus jeunes seraient par exemple plus habituées à renvoyer les chaussures
* Gender: les hommes et les femmes n’ont généralement pas les mêmes habitudes d’achat et de commande en termes de chaussures.
* GenderLabel: même constat que précédemment.
* SeasonLabel\_y: différents types de produit peuvent être plus ou moins facile à choisir en ligne.
* TypeBrand: produits plus chers, de meilleure qualité ? Est-ce que les clients sont fidélisés ?
* HeelHeight: plus difficile de se sentir à l’aise dans des chaussures avec des talons haut, il faut absolument essayer les chaussures à la réception et on s’attend forcément à des retours plus élevés si les talons sont hauts.
* IsNewCollection: personne connaît comment est le produit et il n’a pas encore reçu d’avis.
* SizeAdviceDescription: la description de taille peut ne pas être correct et entraîner des retours.

Tous les features choisis, sauf BrandId, ont eu les résultats attendus.

## 2 - Features crées

Après cette première sélection, nous nous sommes penchés sur les raisons pour lesquels quelqu’un pourrait retourner des chaussures/produits.

Ainsi, nous avons décidé de regarder le moment auquel les clients ont acheté les produits, car peut être que ce facteur était très en rapport avec le fait de retourner les produits. En effet, les consommateurs pourraient être plus impulsifs dans leurs achats tel ou tel jour de la semaine, par exemple si un client achète dans la semaine, exténué par son travail, peut être qu’il aura plus de chances de renvoyer le produit que s’il avait commandé le dimanche calmement. Nous avons donc, dans cette démarche, d’étudier quatre nouvelles features, à savoir le jour de la semaine, la semaine, le mois et l’année d’achat, obtenus à partir de la date de création de la commande.

Une autre idée serait de vérifier si des chaussures avec des descriptions de taille différents de l’usuel sont plus retournés. Bien que le site indique si la chaussure taille grand ou petit le cas échéant, on peut imaginer que cette information n’ait pas été prise en compte par le client, voire trop prise en compte. Pour étudier cette hypothèses, nous avons regroupé tous les indicateurs différents de “pointure normale”, et nous avons comparé les résultats.

Ensuite, nous avons utilisé l’année de naissance pour calculer l’âge du consommateur et déterminer s’il jouait un rôle important ou non.

Enfin, une dernière idée que nous avons eue : nous nous sommes demandé si des femmes achetaient des produits pour homme et inversement. En effet, dans ce cas, on peut intuiter que cette personne n’a pas acheté le produit pour elle-même mais pour une autre. Le risque de retour est alors bien plus grand, car l’utilisateur n’a pas choisi lui-même sa paire de chaussures ! Cependant, nous ne sommes pas sûr à 100% que ce dernier n’a pas choisi lui-même sa paire : en effet, peut-être qu’un homme pourrait utiliser le compte de sa femme pour commander par exemple, voire qu’il était avec elle au moment de la commande.

En résumé, nous avons créé ces features:

1. X.Hour
2. X.DayOfMonth
3. X.DayOfWeek
4. X.Month
5. X.YearBirth
6. X.DifferentSize
7. X.DifferentGenre

Finalement, quelles ont été les features qui ont été vraiment liées aux retours de chaussures ? Nous avons alors fait les constats suivants : le moment de la journée et l’année ont beaucoup influé sur le taux de retour. Contre toute attente, l’âge n’a pas joué un rôle très important. Le facteur “pointure différente” est également non négligeable, car ce facteur figure parmi les 20 premières features après régression logistique. Enfin, le facteur DifferentGenre a également été pertinent.

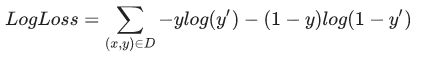
# III - Machine learning

## 1- Logistic regression :

#### ***Choix :***

Pour une classification binaire des problèmes, la régression logistique était notre première méthode d’approche à ce problème d’identification de retour de chaussures.

La fonction de perte pour la régression logistique est la **perte logistique**, définie de la façon suivante :



où *D* est l'ensemble de données , *y* est l'étiquette dans un exemple étiqueté, et y’ est la valeur prédite. L'équation pour la perte logistique est étroitement liée à [la mesure d'entropie de Shannon de la théorie de l'information](https://wikipedia.org/wiki/Entropy_(information_theory)).

La régression logistique est une méthode linéaire, toutefois, les prédictions sont transformées par le biais d’une fonction de logistique (par exemple sigmoïde). On s’attendait donc à obtenir un résultat qui n’était pas une combinaisons linéaires des entrées, comme dans le cas d’une régression linéaire.

Avant de procéder à notre régression logistique, nous avons pris en compte plusieurs critères et procédures sur la data pour justifier la pertinence de ce classifieur, dont :

* Le fait que ce soit conçu pour un système binaire, constitué de deux classes (rendre ou garder les chaussures). La régression logistique est donc bien conçue pour un système binaire de classification. Elle a pour objectif de prédire la probabilité d’une instance d’appartenir à telle ou telle classe.
* Le nettoyage de la data des outliers et des éléments perturbateurs (tels que NaNs, ou chiffres inappropriés). Cela avait pour objectif d’enlever les instances potentiellements mal classées de notre training set.
* L’élimination des entrées corrélées : grâce à notre matrice de corrélation, nous avons pu identifier les variables corrélées, et par la suite les éliminer. L’objectif était d’éviter l’overfitting dans le cas où on a des features très corrélées en entrée. La matrice de corrélation permet donc de calculer les corrélations mutuelles au sein de toutes les pairs de features, et d’afficher une matrice avec des couleurs permettant de les discriminer.

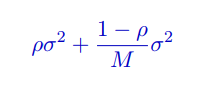
#### ***Résultats :***

Après avoir fait une sélection à la base des coefficients d’importance de features, nous obtenons un score de 0.658 à l’aide la la Logistic Regression.

## 2- Random forest :

#### ***Choix :***

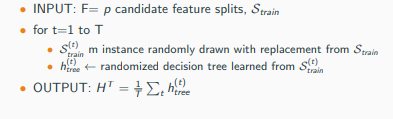
Une deuxième approche consistait à utiliser les arbres de décisions et le bagging. En revanche, nous avons convenu que tout arbre de décision généré à la base d’un bagging est identiquement distribué. Le biais est le même que la moyenne. On voudrait exploiter une méthode qui permette de réduire la variance. La variance de M identiquement distribuées variables de variance σ ² et de corrélation ρ est :



On voudrait donc utilisée des bagged trees qui ne soient pas corrélés pour réduire les deux termes de la variance. On utilise donc des Random Forests.

On a donc manipulé les hyperparamètres que sont le nombre de variables utilisées, le nombre d’observations dans chaque feuille et le nombre de variables aléatoires sélectionnées pour choisir des forêts aléatoires optimales par validation croisée.

On voulait donc avoir une approche aléatoire par le choix au hasard des classifieurs, suivant l’algorithme :



On s’attendait théoriquement aux résultats suivants :

* ***Avantages :*** 
  + Rapidité
  + Compatibilité avec le grand nombre de features
  + Facile à manipuler
  + Généralement performant dans les challenges.
* ***Inconvénients:***
  + Overfitting si la taille des arbres est très grande
  + Perte d’interprétabilité en dépit du fait qu’on puisse estimer l’importance d’une feature donnée.

#### ***Résultats :***

On obtient un score de 0.619, un peu inférieur au score obtenu pour la régression logistique. Le score change en revanche selon les hyperparamètres fixés à notre modèle.

## 3- Adaboost :

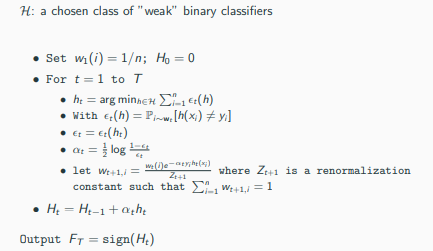
#### ***Choix :***

On se posait donc la question par rapport à la possibilité de booster un classifieur faible (erreur sur le training set est inférieur strictement à 0.5) pour en faire un de plus puissant. L’outil visualisé en SD 210 à ce sujet faisait référence à l’AdaBoost.

L’idée est d’entraîner une suite de classifieurs d’apprentissage sur une dataset pondérée avec des poids dépendants de la fonction de perte obtenue à ce moment là.

En d’autres termes, on n’aura pas besoin d’une architecture profonde pour faire la base de notre classifier. Un arbre court peut être utilisé pour fitter, un linear classifier peut aussi être utilisé à cette fin.

L’algorithme imaginé a la forme :



L’erreur sur le training set obtenue par l’AdaBoost à l’instant T est telle que :



On s’attendait par ailleurs à une possibilité d’overfitting pour AdaBoost, en dépit de l’apport qu’il promet à l’erreur de prédiction. L’idée de descente en coordonnées grâce à AdaBoost nous a ouvert sur d’autres alternatives, telles que le gradient Boosting, à titre d’exemples.

#### ***Résultats :***

Nous avons obtenu, par conséquent, un score de 0.661 par le biais d’AdaBoost. Ce score est donc supérieur à celui obtenu par les deux précédents classifiers : Logistic Regression (0.658) ou encore random forests (0.619). Cela montre donc la puissance que peut avoir la fortifications de faibles classifieurs pour en avoir un de plus puissant.

## 4- Gradient Boosting classifier:

#### ***Choix:***

L’idée de Gradient Boosting vient donc du fait que AdaBoost utilisait une méthode de descente par coordonnées.

Dans chaque méthode de Boosting, nous avons à résoudre le problème d’optimisation :



On pourrait, donc théoriquement, faire une approximation de gradient, telle que :



Donc l’idée de Gradient Boosting consiste dans le fait de remplacer l’étape de minimisation par une descente de gradient :

* En choisissant la meilleure direction de descente 
* En choisissant un bon paramètre alpha qui minimise

On s’attend donc (compte tenu de calculs théoriques), à avoir une équivalence entre les deux méthodes de descente : AdaBoost et Gradient Boosting.

Il s’agit dans les deux cas d’une ‘greedy optimisation’ pour obtenir une combinaison linéaire de classificateurs faibles.

Là encore, on avait la possibilité de manipuler des hyper paramètres, tels que le nombre d’observations dans les feuilles de l’arbre, la profondeur maximale (max depth) et max\_features qui est le nombre de features à prendre en compte quand on procède à la classification dans l’arbre. Cela permettrait en l’occurrence, de jouer sur la complexité de l’arbre.

On savait par ailleurs que le gradient boosting était généralement utilisé pour des problèmes de Ranking (Yahoo, Yandex). En revanche, il n’y avait aucune raison pour ne pas pouvoir l’utiliser pour un problème de classification binaire.

#### ***Résultats :***

Nous obtenons donc pour le gradient boosting un taux de réussite à 0.667. Il dépasse ainsi les autres taux réalisés par les autres classificateurs. En revanche, il est à noter que ces comparaisons n’ont pas de sens absolu car on a dû travailler au fur et à mesure sur la sélection de features et donc on n’a pas forcément procédé par le même training set. Cette sélection reposait donc sur le calcul par validation croisée du score inhérent à chaque feature en particulier.

# IV - Conclusion

Notre démarche s’est résumée en trois grandes étapes. D’abord, un travail de recherche, le “preprocessing” : il a fallu étudier les liens entre les différentes données, notamment par l’étude de la corrélation, pour déterminer les données qui seront cruciales dans notre travail. Il a également fallu rendre la base de données exploitable, ce qui n’est pas évident lorsque les bases sont réelles. Ce challenge a donc particulièrement dénoté par rapport aux TPs, en étudiant une base de données avec des défauts, des erreurs et des entrées vides !

Mais ce n’est qu’après que nous avons réellement expérimenté le travail d’analyse du data scientist : la recherche de features. En effet, si l’étape de preprocessing nous a un peu aiguillés sur cette recherche, nous avons dû nous pencher sur le cas particulier de la vente de chaussures pour trouver les facteurs déterminants des retours. Par exemple : “Peut-être qu’en rentrant du travail, une personne sera plus impulsive qu’en achetant le dimanche, et sera plus amenée à renvoyer sa paire ?” : si cette question peut paraître anodine, c’est bien ce type de démarche que nous avons appliqué, et qui a sûrement fait la différence entre chaque groupe de travail. C’est précisément cette étape qui était la plus emblématique de ce data challenge : si avant, lors des TPs, nous ne faisions que classifier les données, désormais, nous devons faire ressortir des données annexes avant de faire l’apprentissage d’un modèle. C’est cette approche que nous appliquerons plus tard dans le monde professionnel.

Finalement, nous avons comparé méthodiquement plusieurs algorithmes pour réduire la variance des résultats et obtenir le meilleur score possible. Comme durant les TPs, nous avons essayé de trouver les meilleurs algorithmes avec les meilleurs paramètres pour atteindre notre but.

Ce data challenge nous a donc permis de mieux cibler le vrai travail et les vraies difficultés auxquels se confronte le data scientist dans le monde professionnel. Une base de données imparfaite, des paramètres qui ne peuvent être déterminés qu’en étudiant le domaine précis dans lequel on travaille … Ce data challenge était donc un véritable pas dans notre formation de data scientist.