Table des matières :

Table des figures :

Introduction

Le cancer du sein est la maladie la plus redoutée par les femmes, par sa fréquence mais surtout son extrême gravité, qui mène inexorablement à la mort quand le diagnostic est établi tardivement.

Un cancer signifie la présence de cellules anormales qui se multiplient de façon incontrôlée. Dans le cas du cancer du sein, les cellules peuvent rester dans le sein ou se répondre dans le corps par les vaisseaux sanguins ou lymphatiques. La plupart du temps, la progression d'un cancer du sein prend plusieurs mois et même quelques années. Plusieurs recherches ont été menées ces dernières années afin de développer des outils d'aide au diagnostic de cette maladie.

Le but de notre travail est de concevoir un système d'aide aux médecins radiologues pour diagnostiquer cette maladie. Ce projet a pour axe d'étude la détection des tumeurs cancéreuses à partir d'image numérisée d'une aspiration d'aiguille fine (FNA) d'une masse mammaire. Les techniques de Machine Learning, plus précisément les méthodes de classification seront utilisées pour résoudre cette problématique.

Nous commencerons donc par la description de notre Data Set, Puis, nous analyserons les données de notre Data Set, ensuite, nous travaillerons sur la modélisation de ses données, en fin, nous évaluerons nos résultats pour améliorer les performances.

Ce rapport présente l'ensemble des étapes suivies pour développer notre solution. Il contient sept chapitres organisés comme suite :

Le rapport est constitué de sept chapitres, le premier chapitre est consacrée à la présentation des motivations et du contexte du projet, le deuxième chapitre concerne les connaissances liées à la problématique, le troisième chapitre liste les algorithmes du machine learning qu'on a utilisé, le quatrième chapitre décrit la base de données qu'on a employé, pendant que les chapitres 5,6 et traitent la consécutivement l'analyse de données, la modélisation du problème et l'évaluation des modèles.

Chapitre 1 : Motivation et contexte du projet

1.1.Motivation

Le cancer du sein est actuellement le plus fréquent des cancers féminins. Dans le monde, chaque année, l'on compte plus de 1 050 000 de nouveaux cas diagnostiqués et plus de 400000 décès causés par le cancer du sein.

La gestion du cancer du sein peut se résumer en trois problèmes principaux: diagnostic, pronostic et prédiction de bénéfice thérapeutique. Bien que le diagnostic du cancer du sein puisse être entièrement assuré par des outils d'imagerie médicale, chaque tumeur est différente et nécessite une expertise que même les meilleurs oncologues du monde ont du mal à maintenir à jour avec leur cerveau biologique.

Dans ce travail, nous proposons une approche pour surmonter de manière appropriée un tel challenge. Nous essayerons d'utiliser les outils de Machine Learning pour mieux comprendre la diversité de ces tumeurs, les dépister plus vite et pouvoir ainsi les traiter plus efficacement.

1.2.Contexte

1.2.1. Cancer du sein

Un cancer signifie la présence de cellules anormales qui se multiplient de façon incontrôlée. Dans le cas du cancer du sein, les cellules peuvent rester dans le sein ou se répandre dans le corps par les vaisseaux sanguins ou lymphatiques. La plupart du temps, la progression d'un cancer du sein prend plusieurs mois et même quelques années.

Le cancer du sein est le cancer le plus diagnostiqué chez les femmes à travers le monde, autant avant qu'après la ménopause1. Une femme sur 9 sera atteinte d'un cancer du sein au cours de sa vie et 1 femme sur 27 en mourra.

Le plus souvent, le cancer du sein survient après 50 ans. Le taux de survie 5 ans après le diagnostic varie de 80 % à 90 %, selon l'âge et le type de cancer.

Le nombre de personnes atteintes a progressé légèrement mais régulièrement, au cours des 3 dernières décennies. Par contre, le taux de mortalité a continuellement diminué au cours de la même période, grâce aux progrès réalisés en matière de dépistage, de diagnostic et de traitement.

1.2.2. Type de tumeur

Une tumeur est une masse qui se développe aux dépens d'un organe et à partir des cellules qui le constituent.

Il existe deux types de tumeurs : les tumeurs bénignes et les tumeurs malignes.

- **Les tumeurs bénignes** ne sont pas cancéreuses, c'est à dire qu'elles n'envahissent pas les organes voisins et ne font que les repousser ; elles ont un développement généralement limité ; elles n'essaiment pas leurs cellules ailleurs, ce qui signifie

qu'elles ne font pas de métastases . Les tumeurs bénignes peuvent malgré tout poser des problèmes selon l'endroit où elles se situent (le tympan par exemple qu'elles peuvent détruire, ou les intestins qu'elles peuvent boucher).

- **Les tumeurs malignes** font exactement le contraire : elles envahissent toute la région, infiltrent les organes avoisinants et surtout elles envoient des métastases dans d'autres endroits du corps. Elles peuvent devenir énormes et récidivent souvent une fois qu'on les a retirées. Toutefois, ces tumeurs cancéreuses ne sont pas toutes mortelles, tout dépend de leur degré d'extension, de la précocité du traitement et du type de cellules qui les constituent.

1.3.Objectifs du projet

Le cancer du sein est la maladie la plus redoutée par les femmes, par sa fréquence mais surtout son extrême gravité, qui mène inexorablement à la mort quand le diagnostic est établi tardivement.

Un cancer signifie la présence de cellules anormales qui se multiplient de façon incontrôlée. Dans le cas du cancer du sein, les cellules peuvent rester dans le sein ou se répondre dans le corps par les vaisseaux sanguins ou lymphatiques. La plupart du temps, la progression d'un cancer du sein prend plusieurs mois et même quelques années. Plusieurs recherches ont été menées ces dernières années afin de développer des outils d'aide au diagnostic de cette maladie.

Le but de notre travail est de concevoir un système d'aide aux médecins radiologues. Pour ce faire, nous devrons détecter des tumeurs cancéreuses à partir d'image numérisée d'une aspiration d'aiguille fine (FNA) d'une masse mammaire.

1.4.Méthodologie

Cette problématique est bien connue dans le domaine de Machine Learning, c'est un problème de classification supervisé. Donc, Nous avons adoptée comme méthodologie, des algorithmes de classification pour le diagnostic de la tumeur du sein.

Les tumeurs sont de types malignes ou bégnines, notre objectif sera d'entrainer des algorithmes de classification afin de nous donner comme résultat une prévision sur le type de tumeurs.

Plusieurs algorithmes seront utilisés pour notre modélisation, en fin, une comparaison entre ces algorithmes sera essentiel pour évaluer et améliorer les performances.

Nous commencerons donc par la description de notre Data Set, Puis, nous analyserons les données de notre Data Set, ensuite, nous travaillerons sur la modélisation de ses données, en fin, nous évaluerons nos résultats pour améliorer les performances.

La figure suivante résume le processus data science qu'on a suivit

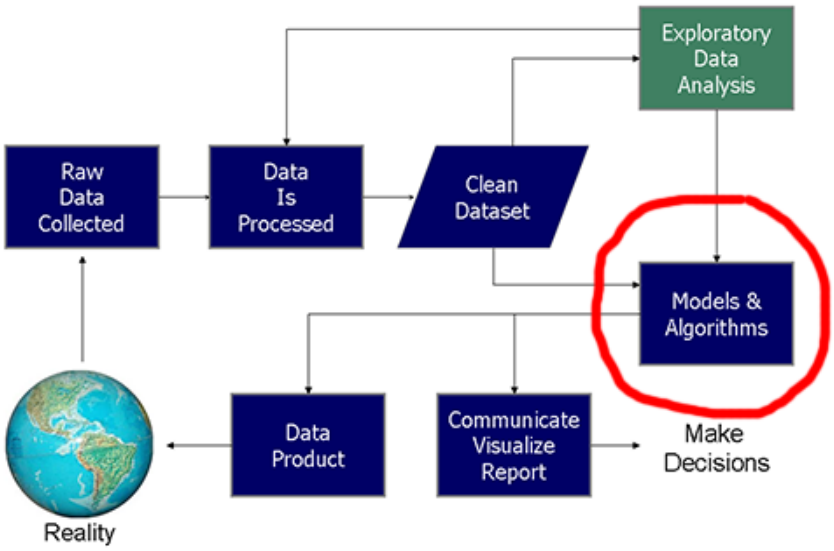


Figure 1 - Le processus du Data Science

Chapitre 2: Connaissances préalables du projet

2.1.Big data et Machine Learning

Les origines du Big Data remontent à 1941, date à laquelle les premières références ont été faites à la notion d'« explosion de l'information » dans l'Oxford Dictionary of English. James Maar a mis en exergue dès 1996 dans un rapport de la National Academy of Sciences la notion de « massive data set » (jeux de données massives). Mais c'est seulement en 1997 que le terme précis de Big Data a fait son apparition dans un article de la bibliothèque numérique de l'Association for Computing Machinery, faisant référence au challenge technique que représente l'analyse de grands ensembles de données. Le terme Big Data a récemment été introduit dans les dictionnaires français avec son équivalent officiel « mégadonnées » proposé par la Commission générale de terminologie et de néologie. Il est depuis utilisé pour désigner « des données structurées ou non, dont le très grand volume requiert des outils d'analyse adaptés ». Les géants du Web (Google, Amazon, Facebook, Apple, Twitter) ont développé depuis dix ans de tels outils, permettant ainsi d'assurer un coût marginal constant d'exploitation des données, indépendamment de leur volume.

Aujourd'hui, le Big Data se caractérise par 5 « V » : volume, vitesse, variété, véracité et valeur des données exploitées. La chute des prix de stockage et l'augmentation des capacités de calcul sont à l'origine des gros Volumes et de la grande Vitesse de traitement des données. La Variété des données (images, textes, bases de données, objets connectés, etc.) est principalement due à la digitalisation croissante des supports d'information. Enfin, la Véracité des données, dont découle la valeur des travaux, constitue un enjeu central pour tout projet d'analyse automatisée des données. En effet, un algorithme est d'autant plus performant que les données sont nombreuses, exactes, et bien adaptées à la question à résoudre. Multiplier les sources et les croisements sans se soucier de la qualité des données ne peut que mener à des résultats erronés. Le développement du Big Data s'est accompagné de l'apparition des « Open Data » qui correspondent à des données générées et conservées par différents organismes et mises à la disposition des citoyens et des entreprises.

Les 5 « V » sont cependant insuffisants pour caractériser l'essence de l'innovation apportée par le Big Data. En effet, celle-ci provient avant tout de la combinaison des outils permettant de gérer ces 5 « V » avec un sous-domaine de l'intelligence artificielle dénommé « machine learning » (apprentissage automatique). Ce dernier permet de construire des algorithmes capables d'accumuler de la connaissance et de l'intelligence à partir d'expériences, sans être humainement guidés au cours de leur apprentissage, ni explicitement programmés pour gérer telle ou telle tâche particulière, d'où leur rôle central dans la chaîne de valeur de la donnée. La maîtrise de ces algorithmes est au cœur du métier de data scientist .

2.2.Approche Machine Learning

Le cycle de travail du data scientist comprend :

- la récupération des données utiles à l'étude
- le nettoyage des données pour les rendre exploitables
- une longue phase d'exploration des données afin de comprendre en profondeur l'articulation des données
- **la modélisation des données**

- l'évaluation et interprétation des résultats
- la conclusion de l'étude : prise de décision ou déploiement en production du modèle

Au sein de ce cycle, le **machine learning** désigne l'ensemble des méthodes de **modélisation statistique à partir des données**, et se situe bien au coeur du travail de data scientist.

Zoom sur la partie Machine Learning :

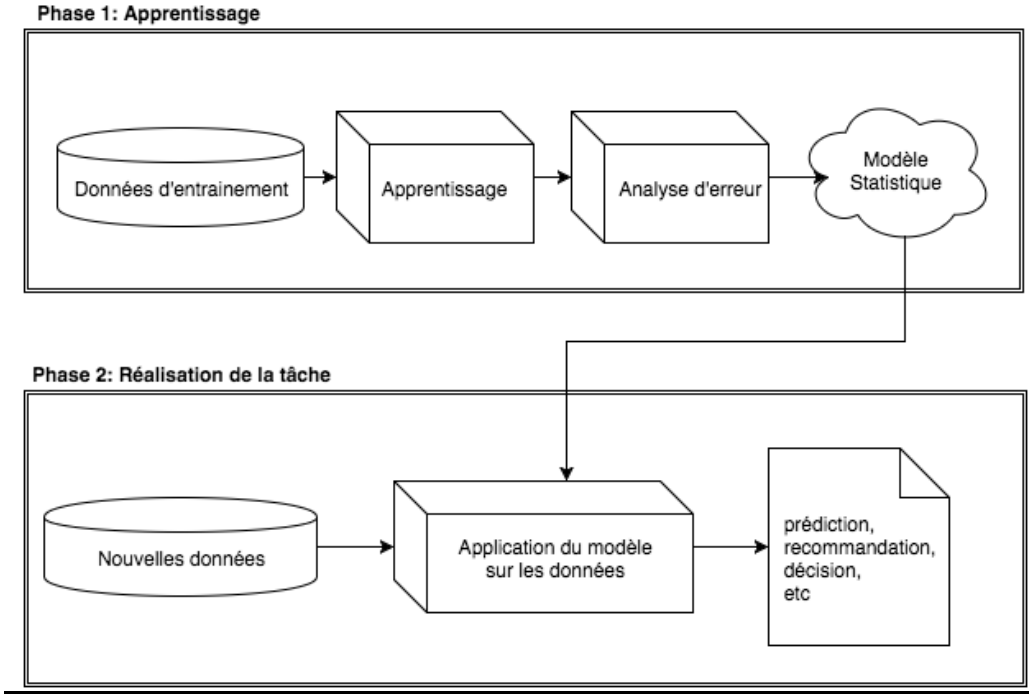


Figure 2 - processus machine learning

2.3.L'application du Big data au domaine de la santé

Les outils Big Data ont permis le lancement de nombreux projets médicaux fondés sur l'exploitation de données massives, à l'image de l'algorithme de « Support Vector Machine » permettant, à partir de l'analyse de 368 gènes, de discriminer les tumeurs mammaires basales de pronostic péjoratif de celles dont le pronostic est plus favorable. Les computer-aided diagnosis (CAD), qui peuvent aider les radiologues pour l'interprétation des mammographies, en sont un autre exemple. Plus récemment, le projet Senometry, porté conjointement par des médecins et des data scientists , vise à analyser pour 10 000 patientes atteintes d'un cancer du sein, et suivies pendant plusieurs décennies, des données non structurées provenant de leur histoire personnelle, de l'imagerie (scanner, IRM, mammographies, échographie, scintigraphie, imagerie par émission de positrons, etc.), de la biologie, de l'analyse anatomo-pathologique (caractéristiques tumorales, facteurs prédictifs et pronostiques), des thérapeutiques et de leur évolution. L'analyse croisée de ces multiples données devrait apporter un certain nombre de réponses à des questions non résolues en sénologie dans ses différents domaines (épidémiologie descriptive, analyse des facteurs de risque et protecteurs, évaluation de nouvelles pratiques, détermination plus précise du pronostic, etc.).

À la lumière des possibilités offertes par ce type de projets Big Data, le temps médical prend une autre dimension. À titre d'exemple, il aura fallu presque 30 années de suivi de cohortes pour prouver que le travail de nuit constitue un facteur de risque de cancer du sein et quantifier ce risque. Les technologies Big Data devraient permettre de répondre à ce type de questions en très peu de temps en analysant les données existantes, avec un impact économique important (réduction du coût des études) et une applicabilité immédiate en santé publique. Ces nouvelles technologies vont probablement accélérer les résultats de la recherche médicale et réduire l'écart qui existe aujourd'hui entre la temporalité du malade, qui a besoin de réponses immédiates et la temporalité de la recherche « classique ».

De manière plus globale, l'accès rapide à des jeux massifs de données pourrait changer nos paradigmes médicaux. Alors que le raisonnement médical traditionnel consistait à émettre une hypothèse puis à la vérifier sur des séries de patientes, l'arrivée du Big Data permet dans certains cas une démarche originale où la découverte de corrélations inattendues est postérieure à la récolte des données (c'est le concept de sérendipité).

Le Big Data aura probablement à l'avenir un impact fort sur la compréhension et le traitement du cancer du sein, à condition que la collecte des données et leur finalité soient dès aujourd'hui guidées par des médecins et des data scientists. À ce titre, le Big Data ne se conçoit pas sans interdisciplinarité, avec apprentissage d'une sémantique commune entre ces deux métiers. Les unités de sénologie, par leur organisation historiquement transverse avec mise en place des réunions pluridisciplinaires, constituent un cadre adapté pour de telles collaborations.

2.4.Les données radiologiques

L'imagerie médicale regroupe de nombreuses techniques d'exploration, appelées modalités, faisant appel à l'utilisation de rayons X, d'ultrasons, de lumière visible ou encore de résonnance électromagnétique. Nous utilisons ici le terme d'imagerie radiologique par opposition au terme « imagerie médicale » très générique et souvent utilisé dans le traitement d'images cellulaires obtenues à partir d'un microscope via une caméra numérique.

On peut classer l'utilisation médicale de l'imagerie radiologique selon quatre objectifs :

- la réalisation d'un diagnostic, pour identifier une pathologie,
- la mesure de l'efficacité d'un traitement par la mesure de l'évolution de lésions
- la décision et la planification d'une intervention, dans laquelle les images servent de support au choix des paramètres de l'intervention.
- l'utilisation à but thérapeutique, permettant d'aider à la réalisation d'un geste chirurgical, par exemple le suivi de la trajectoire d'un outil. On parle dans ce cas d'imagerie interventionnelle.

2.5.Analyse automatique d'image médicale

Les images médicales fournissent des informations sur la forme et le fonctionnement des organes du corps humain. Malheureusement, ces informations sont extrêmement difficiles à exploiter de manière quantitative et objective. En effet, bien que les images 3-D soient originellement numériques, leur examen est typiquement réalisé en observant sur un support analogique (un film), une succession de coupes bidimensionnelles (2-D). Le résultat est généralement purement qualitatif et subjectif.

La création de logiciels dédiés à l'analyse d'images médicales doit permettre d'optimiser leur exploitation, pour le plus grand bénéfice du patient et du médecin.

L'analyse automatique des images médicales peut offrir un ensemble de nouveaux outils d'aide au diagnostic. Parmi ceux-ci, on peut citer

- l'extraction de paramètres quantitatifs objectifs sur les formes et leur texture. Ceci doit pouvoir être appliqué à n'importe quelle structure anatomique ou pathologique en trois dimensions.
- La détection de changements entre deux images. On doit offrir au médecin une détection automatique et une mesure quantitative de tous les changements apparus entre deux images acquises avec la même modalité sur le même patient à deux instants différents. Ceci peut servir à établir un diagnostic plus précoce, mais aussi à évaluer l'efficacité d'un traitement thérapeutique.
- La fusion d'informations provenant de plusieurs modalités. On doit pouvoir combiner les informations complémentaires sur un même patient provenant de modalités d'imagerie différentes, en les superposant dans un référentiel commun.
- La comparaison des images de deux patients différents. Il faut concevoir des outils permettant de confronter les images provenant d'une même modalité, mais correspondant à des patients différents. Ces outils doivent permettre de comparer la nature et la gravité de pathologies similaires, ou bien d'extraire des images présentant certaines similitudes dans une base de données d'images.

Chapitre 3: Les Algorithmes du Machine Learning

On a essayé plusieurs algorithmes de classifications, mais on s'est focalisé sur trois algorithmes principaux :

- **Régression logistique**

La **régression logistique** est l'un des modèles d'analyse multivariée les plus couramment utilisés dans les problèmes de classification. Elle permet de mesurer l'association entre la survenue d'un évènement (variable expliquée qualitative) et les facteurs susceptibles de l'influencer (variables explicatives). **Elle** modélise la probabilité qu'une observation appartienne à la classe positive comme une transformation logistique d'une combinaison linéaire des variables.

- **Random Forest Classifier** (Méthode ensembliste)

Les forêts aléatoires sont un ensemble d'arbres de décisions entraînés individuellement, légèrement différents les uns des autres. Pour prédire une nouvelle valeur, on effectue la classification pour chaque arbre de cette forêt. La forêt choisit la valeur ayant le plus de votes parmi tous ses arbres. C'une autre manière pour réduire l'overfitting (d'utiliser une méthode ensembliste.

Le schéma suivant résume cet algorithme :

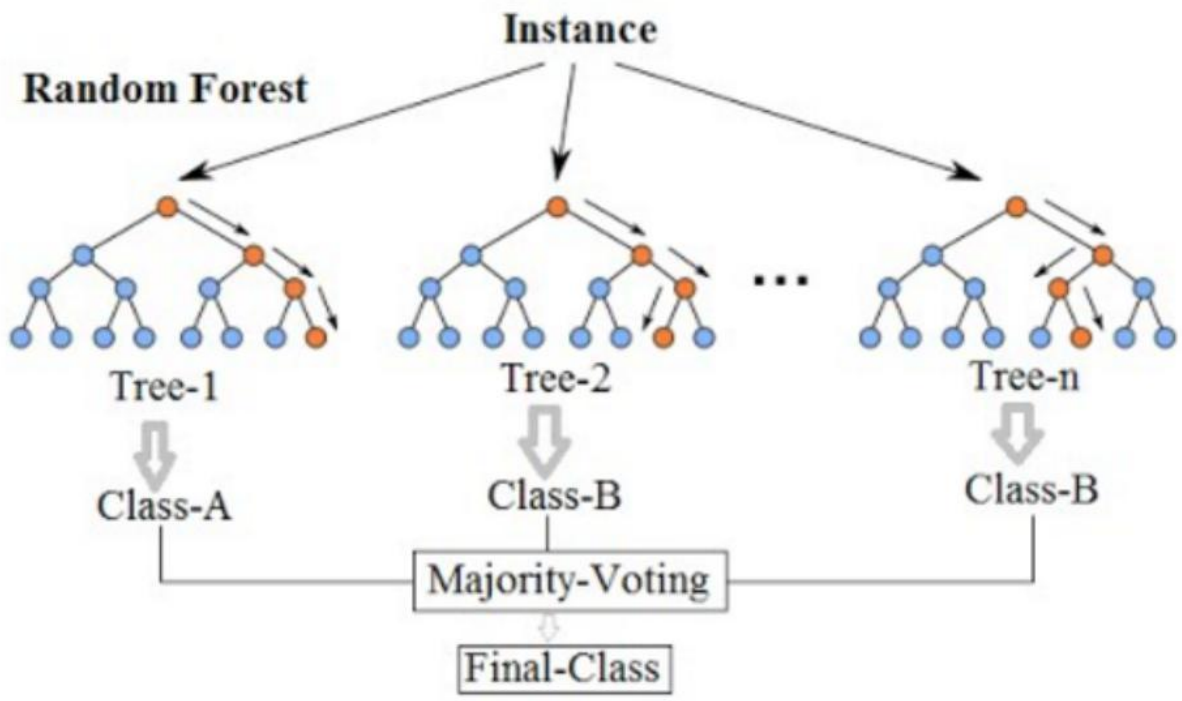


Figure 3 - Principe du Random Forest

- **Support vector machines**

Les séparateurs à vastes marges sont des classificateurs qui reposent sur deux idées clés, la première idée clé est la notion de *marge maximale*. La marge est la distance entre la frontière de séparation et les échantillons les plus proches. Ces derniers sont appelés *vecteurs supports*. Dans les SVM, la frontière de séparation est choisie comme celle qui maximise la marge. Le problème est de trouver cette frontière séparatrice optimale, à partir d'un ensemble d'apprentissage

Afin de pouvoir traiter des cas où les données ne sont pas linéairement séparables, la deuxième idée clé des SVM est de transformer l'espace de représentation des données d'entrées en un espace de plus grande dimension (possiblement de dimension infinie), dans lequel il est probable qu'il existe une séparation linéaire. Ceci est réalisé grâce à une fonction noyau.

Le schéma suivant résume cet algorithme dans le cas linéaire :

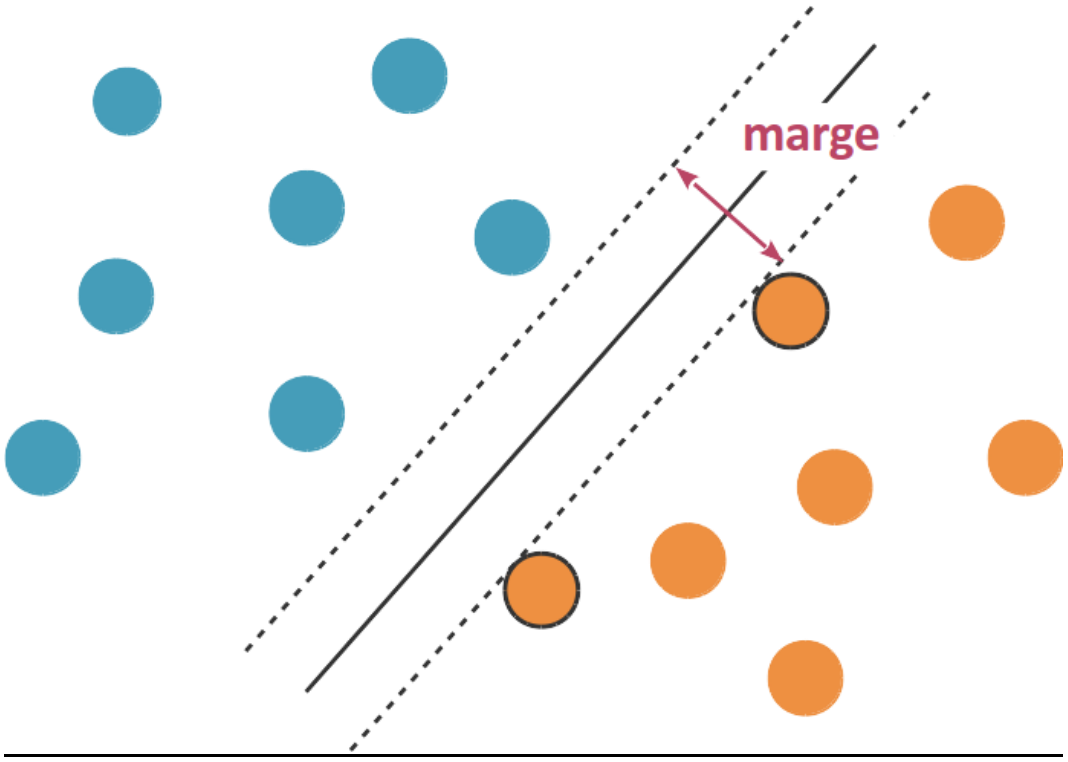


Figure 4 - Pricipe des SVM

Chapitre 4: Data set

La base de données qu'on choisit de l'étudier contient des informations sur le cancer du sein, cette base de données est offerte par UCI Machine Learning, dans le cadre d'étude prédictive sur les tumeurs malignes et bénignes.

Les caractéristiques sont calculées à partir d'une image numérisée d'une aspiration d'aiguille fine (FNA) d'une masse mammaire. Elles décrivent les caractéristiques des noyaux cellulaires présents dans l'image.

Cette base de données est également disponible via le serveur ftp UW **CS: ftp ftp.cs.wisc.edu cd math-prog / cpo-dataset / machine-learn / WDBC /**

Vous pouvez également la trouver sur le référentiel UCI Machine Learning: **https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29**

Dimensions et variables :

La base de données contient 540 observations

Les dimensions du dataset se résument dans le tableau suivant :

Variables	Description
id	Id Number
diagnosis	The diagnosis of breast
radius_mean	mean of distances from center to points on the perimeter
texture_mean	standard deviation of gray-scale values
perimeter_mean	
area_mean	
smoothness_mean	mean of local variation in radius lengths
compactness_mean	mean of perimeter^2 / area - 1.0
concavity_mean	mean of severity of concave portions of the contour
concave points_mean	mean for number of concave portions of the contour
symmetry_mean	
fractal_dimension_mean	mean for "coastline approximation" - 1
radius_se	standard error for the mean of distances from center to points on the perimeter
texture_se	standard error for standard deviation of gray-scale values
perimeter_se	
area_se	
smoothness_se	standard error for local variation in radius lengths
compactness_se	standard error for perimeter^2 / area - 1.0
concavity_se	standard error for severity of concave portions of the contour
concave points_se	standard error for number of concave portions of the contour
symmetry_se	
fractal_dimension_se	standard error for "coastline approximation" - 1
radius_worst	"worst" or largest mean value for mean of distances from center to points on the perimeter

Figure 5 - Les variables du dataset

Remarque :

La signification de chaque variable ne sera pas prise en compte. Elles vont être traitées numériquement de la même façon.

Chapitre 5 : Analyse de données

Le but de cette partie est de rassembler le maximum d'informations possibles sur chaque variable afin de détecter des remarques ou des patterns exploitables dans la sélection des variables ou même dans la modélisation.

5.1.Analyse basique de données

5.1.1. Description des données

Voyons concrètement la composition de nos données à l'aide de quelque graphe et des descripteurs statistiques.

L'entête :

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980

Figure 6 - L'entête du dataset

Résumé statistiques :

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200

8 rows × 32 columns

Figure 7 - description statistique des données

Les colonnes qui ont des valeurs NaN :

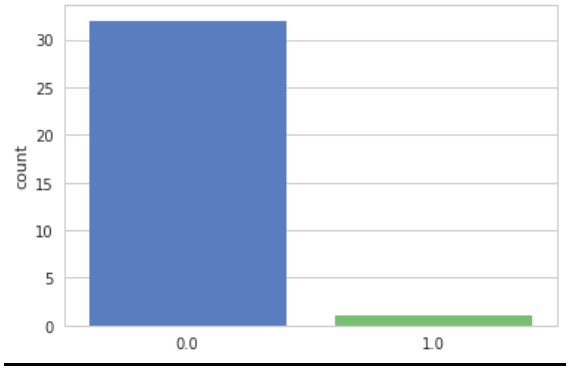


Figure 8 - Les colonnes avec des informations null

5.1.2. Remarque

D'après cette petite enquête, ces observations ont surgi :

- Les données ne sont pas normalisées
- La colonne **id32** ne porte aucune information supplémentaire
- Les colonnes **id** et **diagnosis** doivent être retirées avant l'entrainement du modèle

5.2.Visualisation

La visualisation de données permet de synthétiser les informations que contiennent ces données pour **mettre en évidence les informations** clés qu'ils renferment et **identifier la bonne information le plus rapidement possible**.

5.2.1. Violin Plot

Le but de ce graphe est de montrer la distribution d'une manière élégante et on séparant les catégories de chaque variable.

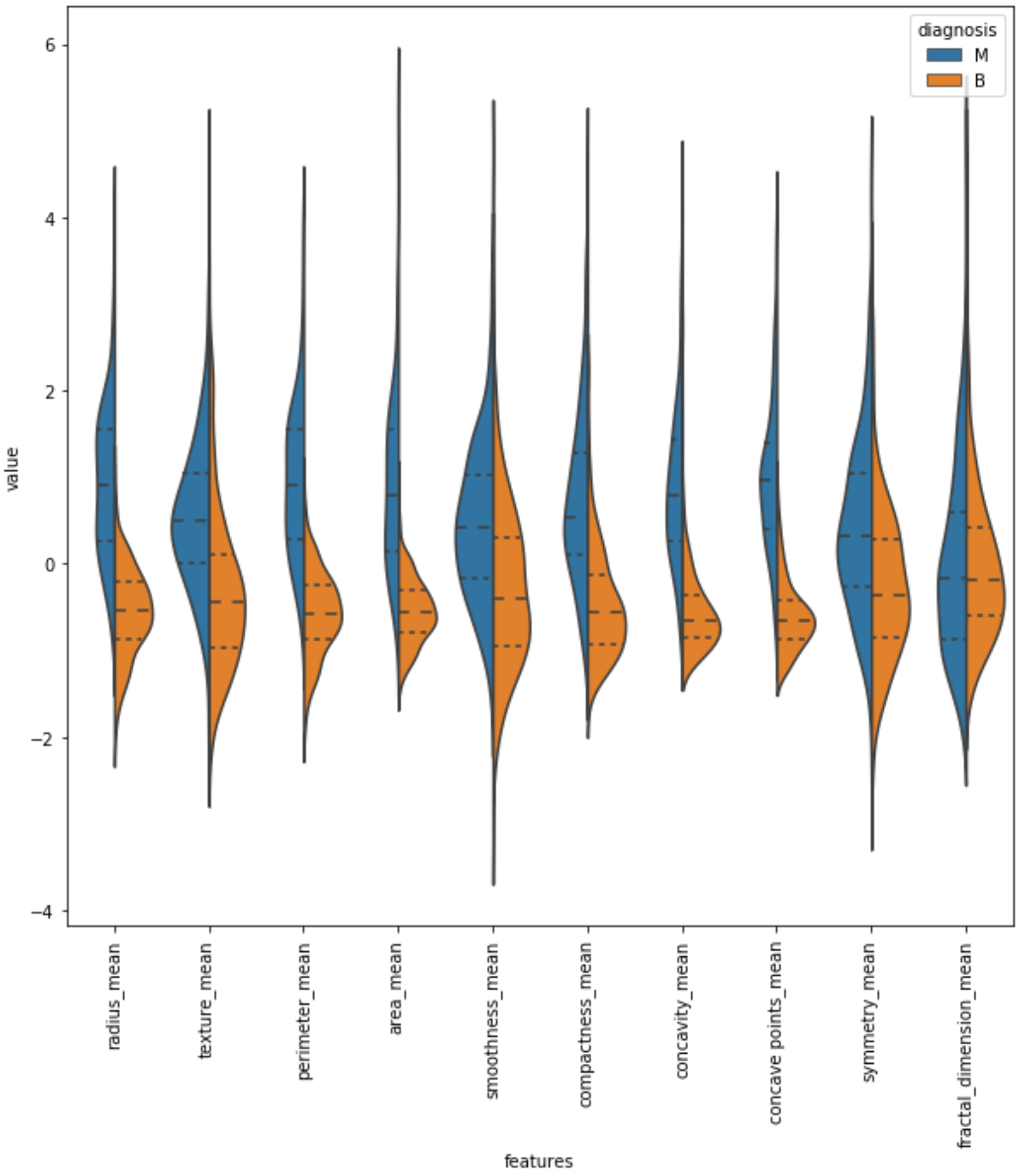


Figure 9 - Violin Plot des 10 premieres variables

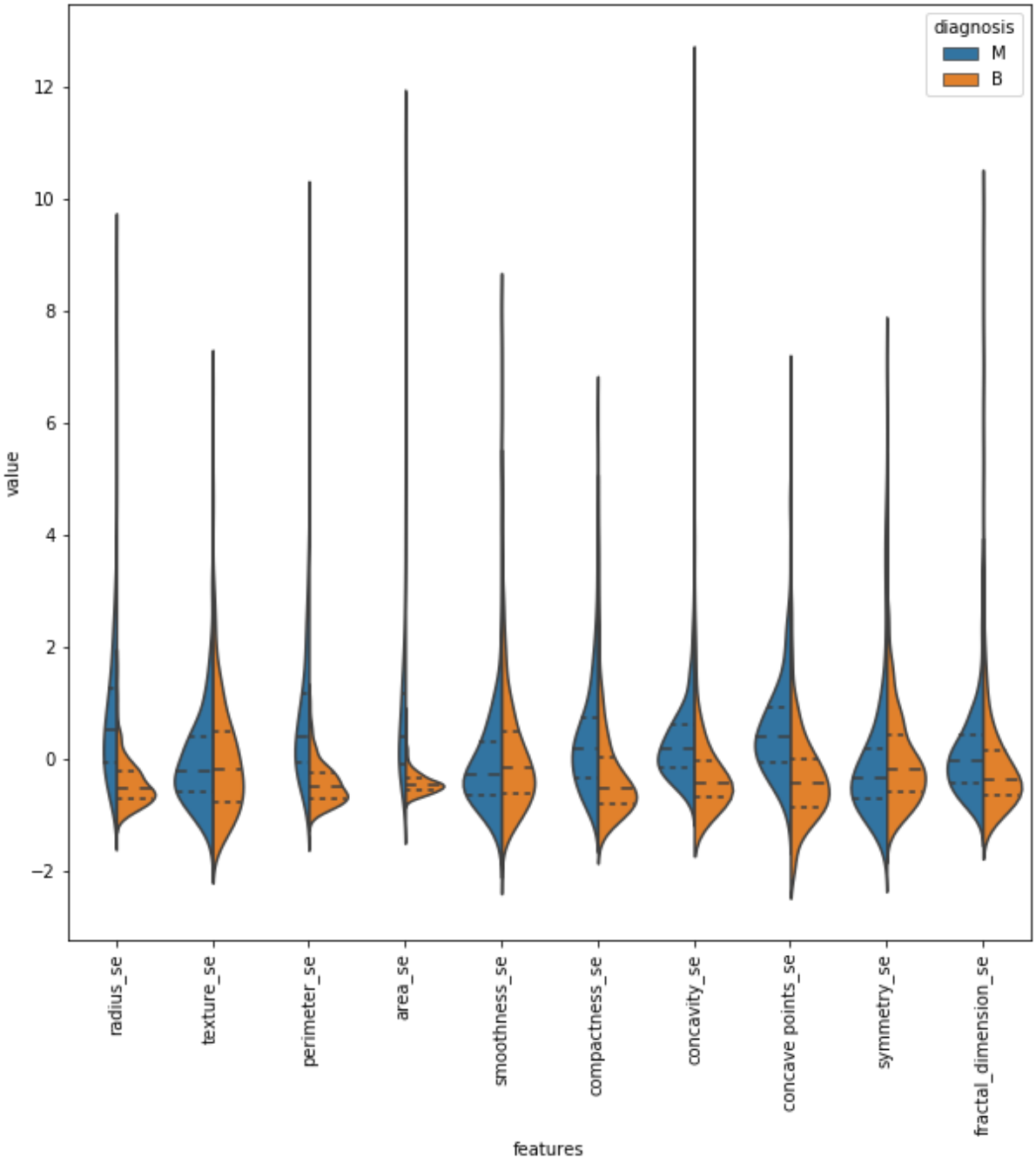


Figure 10 - Violin Plot des 10 variables intermediaires

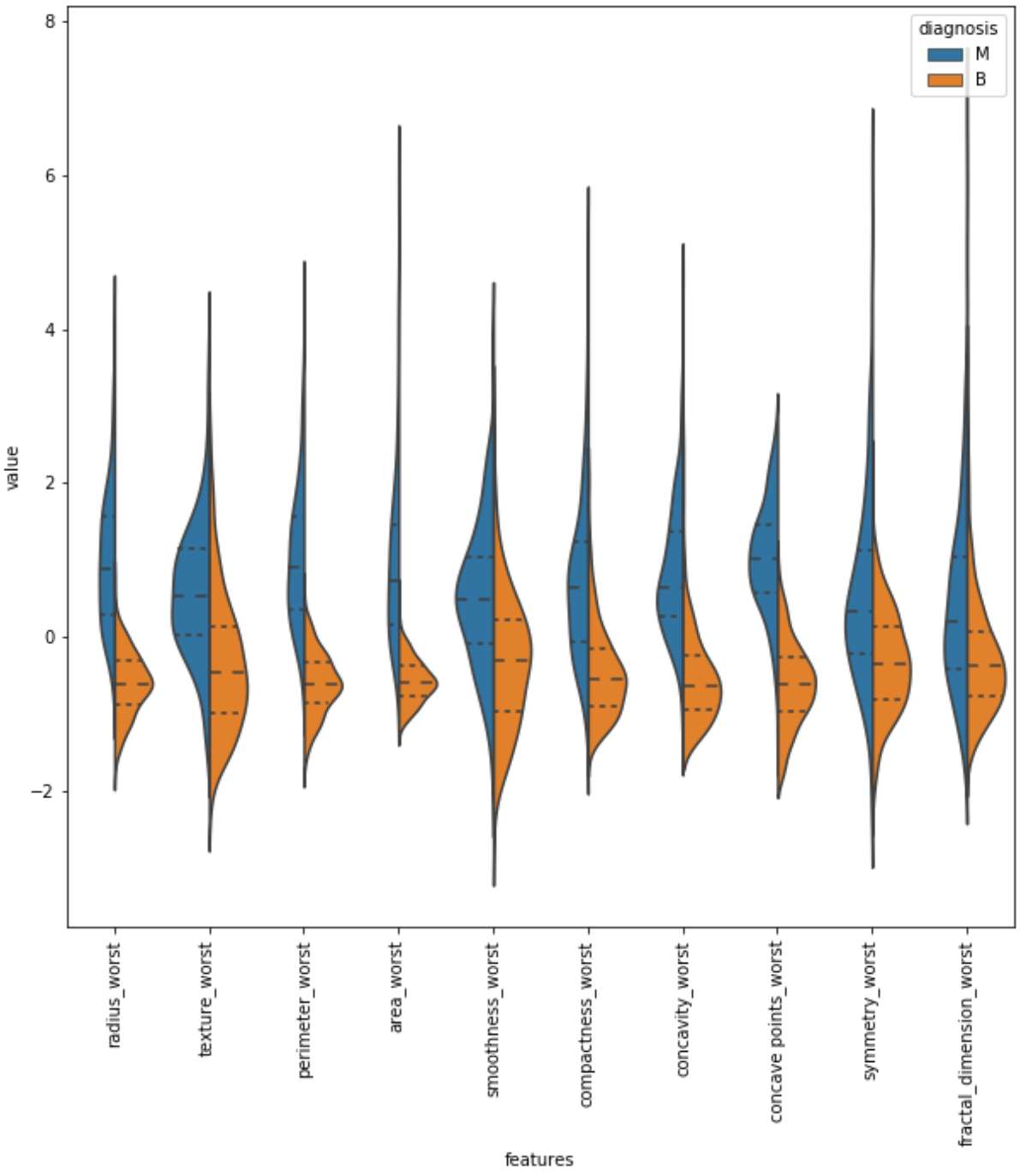


Figure 11 - Violin Plot des 10 dernieres variables

Remarque :

- ✓ les deux dernières variables ont à peu près la même distribution, elles peuvent être corrélées
- ✓ les médianes des catégories de la variable **radius_mean** sont écartées, cette variable peut être bonne pour la classification

Le but n'est pas de sélectionner des variables ou de les éliminer mais juste d'avoir une idée claire sur chaque variable et d'avoir des hypothèses qu'on doit après les tester et vérifier.

5.2.2. Joint Plot

Le dernier type de graphe ne nous permet pas de comparer deux variables entre eux, l'avantage de JOINTPLOT c'est qu'il nous permet de voir le type d'interaction entre deux variables avec une mesure de corrélation de plus.

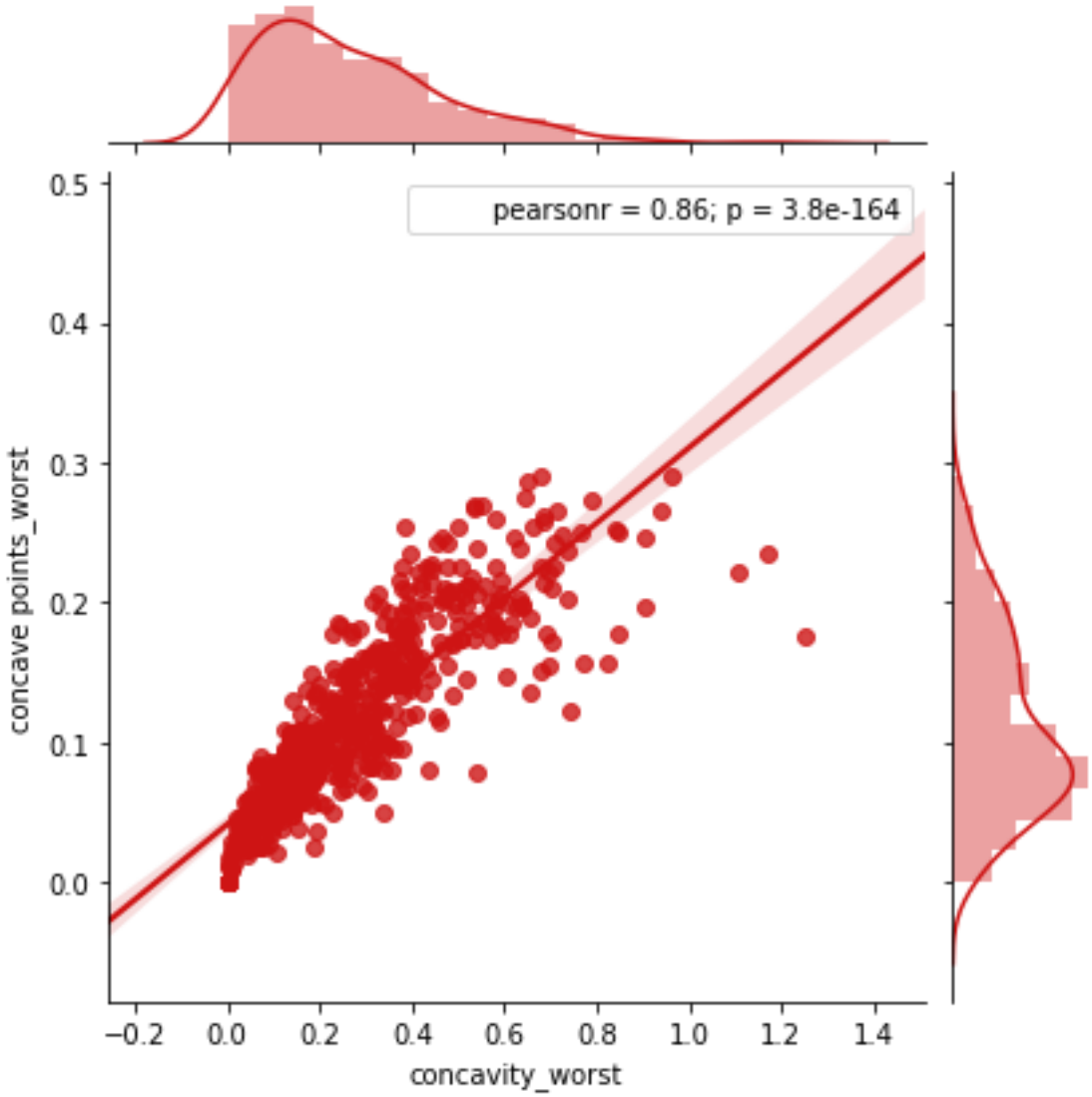


Figure 12 - Joint plot de deux variables

Remarque :

- ✓ Les deux variables semblent corrélées.

5.2.3. Pair grid

Comme Joint plot ne nous permet de visualiser que deux variables, PAIRGRID vient pour généraliser cette idée. Il nous permet donc de visualiser les différentes interactions entre n'importe quel nombre de variables.

Voyons le Pair Grid de trois variables: **radius_worst**, **perimeter_worst**, **area_worst**

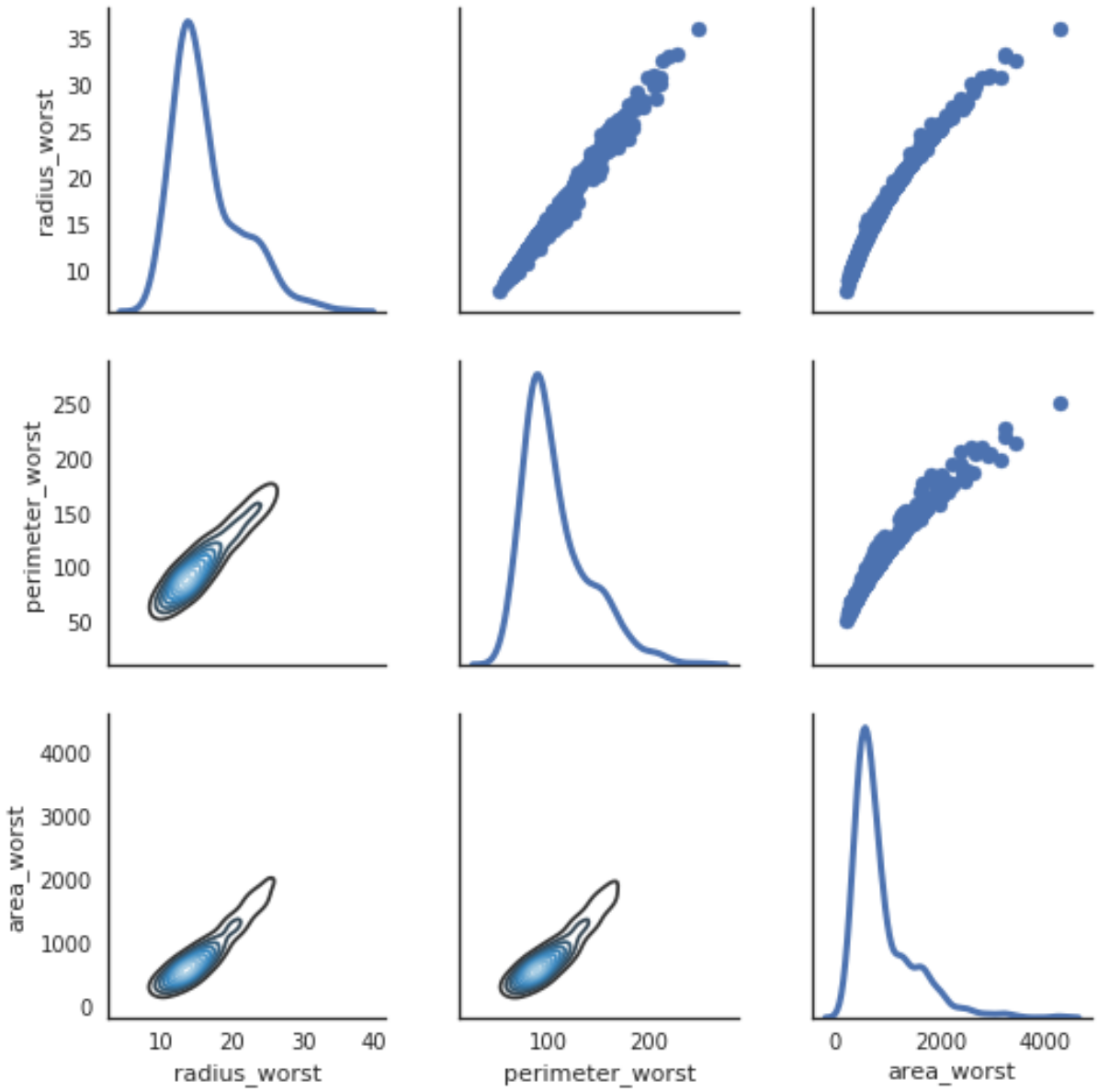


Figure 13 – Pair Grid de trois variables

Remarque :

- ✓ Les trois variables semblent corrélées entre eux.

5.2.4. Heatmap

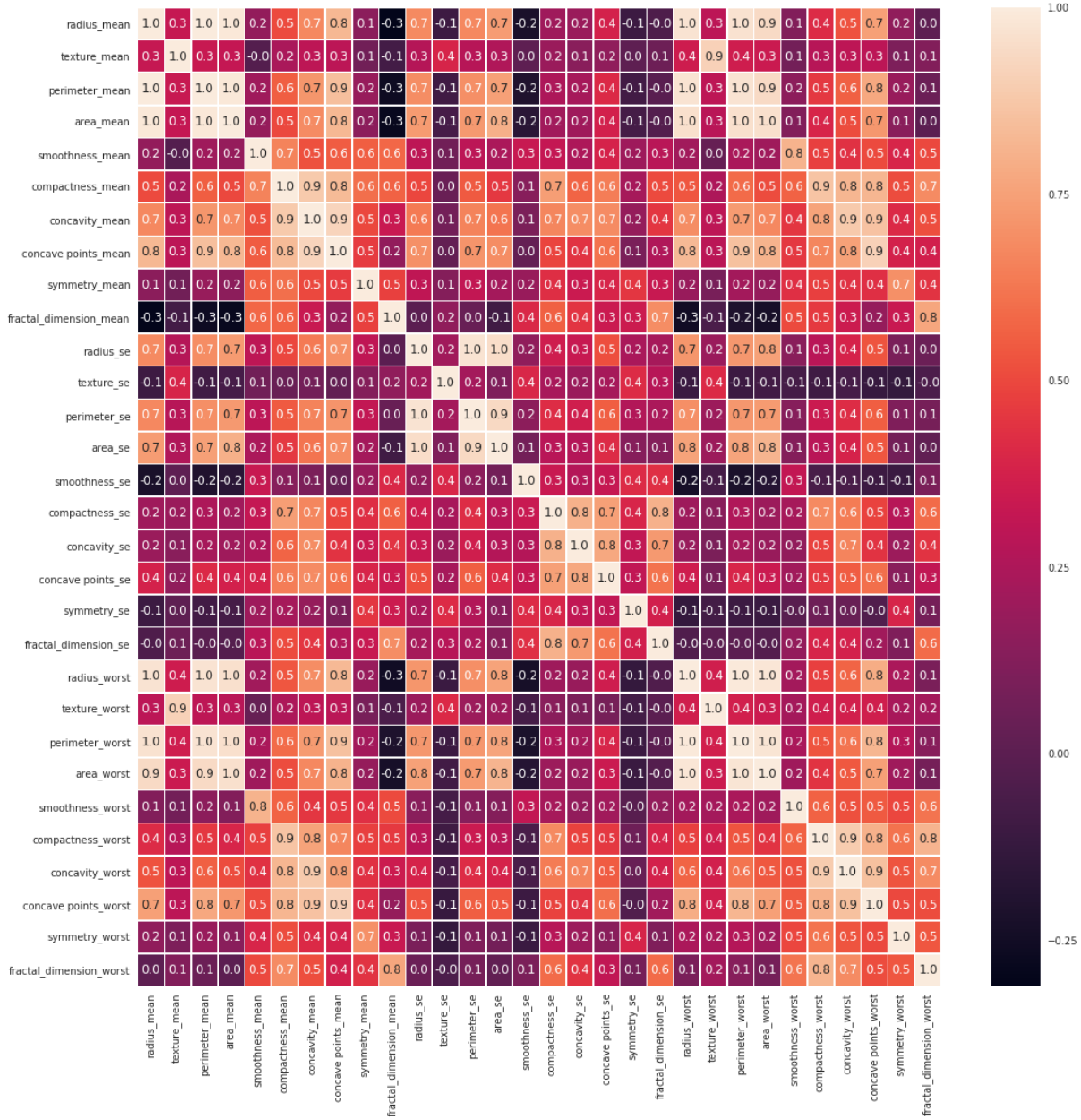


Figure 14 - heatmap des variables - correlation

Remarque :

Ce graphe nous permet de visualiser les corrélations entre les variables et de valider ou de rejeter les hypothèses émises par les différentes techniques de visualisation concernant les groupes qui semblaient être corrélés.

5.3.Sélection des variables

La sélection de caractéristiques est une technique permettant de choisir les caractéristiques les plus pertinentes, celles adaptées à la résolution d'un problème particulier, pour minimiser la complexité de calcule et éviter la redondance des données.

Dans cette partie, nous allons sélectionner des caractéristiques avec différentes méthodes qui sont la sélection de caractéristique avec corrélation, la sélection de caractéristiques univariées, l'élimination récursive des caractéristiques (RFE), et l'élimination récursive des caractéristiques avec validation croisée (RFECV).

5.3.1. Corrélation

La méthode de sélection des caractéristiques en utilisant la corrélation permet de diviser nos caractéristiques en sous-groupes tels que chaque sous-groupe contient des variables corrélées entre eux, ainsi on pourra choisir de chaque sous-groupe une variable significative.

Comme on peut le voir dans le diagramme map heat radius_mean, perimeter_mean et area_mean sont corrélés les uns avec les autres, donc nous utiliserons uniquement area_mean. Compactness_mean, concavity_mean et concaves points_mean sont corrélés entre eux. Par conséquent, nous choisissons concavity_mean. En dehors de ceux-ci, radius_se, perimeter_se et area_se sont corrélés donc nous n'utiliserons que area_se. radius_worst, perimeter_worst et area_worst sont corrélés donc nous utiliserons area_worst. Compactness_worst, concavity_worst et concavePoints_worst donc nous utilisons concavity_worst. Compactness_se, concavity_se et concavePoints_se donc nous choisirons concavity_se. texture_mean et texture_worst sont corrélés alors nous utiliserons texture_mean. area_worst et area_mean sont corrélés donc nous choisirons area_mean.

5.3.2. Univariate feature selection

La sélection de caracteristique univariée fonctionne en sélectionnant les meilleures caractéristiques basées sur des tests statistiques univariés. Il peut être vu comme une étape de prétraitement d'un estimateur.

Les tests statistiques univariés consistent à étudier les variables séparément via des techniques descriptives ou probabilistes. Les objectifs de ces études sont :

- Décrire et résumer chaque variable
- Généraliser les informations à la population entière
- Tester des hypothèses faites à priori

- Comparer 2 variables ou 1 variable sur plusieurs groupes

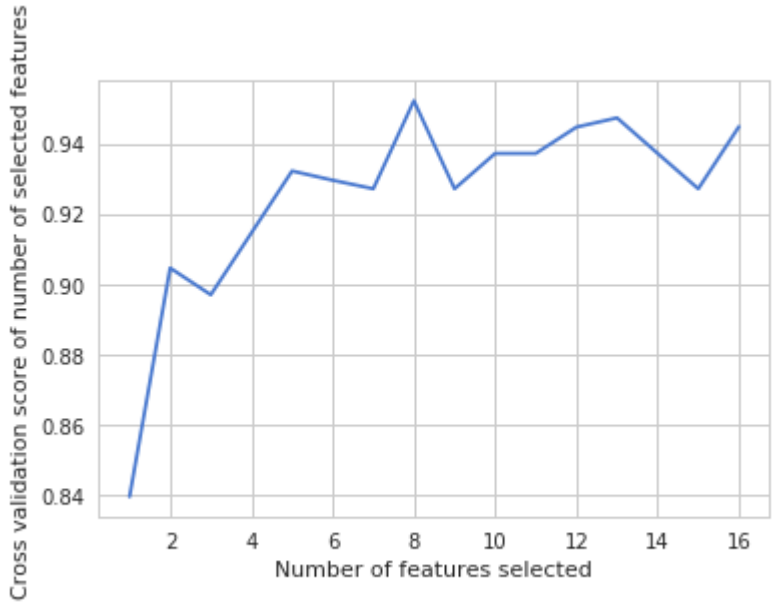
Dans la sélection de caractéristiques univariées, nous utiliserons SelectKBest qui supprime toutes les caractéristiques, à l'exception des k de score le plus élevée.

5.3.3. Recursive feature selection

Étant donné un estimateur externe qui attribue des pondérations aux entités (par exemple, les coefficients d'un modèle linéaire), le but de l'élimination récursive des fonctions (RFE) consiste à sélectionner des entités de manière récursive, des ensembles d'entités de plus en plus petits sont choisies récursivement jusqu'à obtenir le nombre de caractéristique désirée. Premièrement, l'estimateur est entrainé sur l'ensemble initial de caractéristiques, les caractéristiques sont ensuite classées par ordre d'importance. En fin, les caractéristiques les moins importantes sont supprimées de l'ensemble actuel. Cette procédure est répétée de manière récursive sur l'ensemble élagué jusqu'à ce que le nombre désiré de caractéristiques à sélectionner soit finalement atteint.

5.3.4. Tree based feature selection

Recursive features elimination with cross-validation RFECV exécute Recursive features elimination (RFE) dans une boucle de cross-validation pour trouver le nombre optimal de caractéristiques pour notre entrainement.



Chapitre 6 : Modélisation

L'étape de modélisation et de l'évaluation des modelés sont naturellement confondues et traitées en parallèle, surtout en ce qui concerne la division des données et les choix des paramètres de l'algorithme final.

6.1.Approche de division des données

Pourquoi diviser nos données ?

Le jeu de données dont nous disposons constitue une ressource précieuse, il faut pouvoir l'utiliser à bon escient afin de pouvoir à la fois **choisir** un modèle et mais aussi de pouvoir **tester** la qualité de ce modèle.

L'entraînement d'un modèle revient à mesurer l'erreur de la sortie de l'algorithme avec les données d'exemples, et chercher à la minimiser.

Un premier piège à éviter est donc d'évaluer la qualité de votre modèle final à l'aide des mêmes données qui ont servies pour l'entraînement. En effet, le modèle est complètement optimisé pour les données à l'aide desquelles il a été créé. L'erreur sera précisément minimum sur ces données. Alors que l'erreur sera toujours plus élevée sur des données que le modèle n'aura jamais vues.

Pour minimiser ce problème, la meilleure approche est de séparer **dès le départ** notre jeu de données en deux parties distinctes : *training set* et *testing set*

Techniques de divisions de données :

Il existe beaucoup de techniques de division de données, on cite :

- Train-test split :

  Cette technique consiste à **couper notre jeu de données en deux parties** : un jeu d'entraînement, et un jeu de test. On n'utilise ensuite pas du tout le jeu de test quand on choisit et qu'on entraîne notre modèle. Comme ça, on peut calculer la performance sur le jeu de test, et le résultat est une bonne approximation de la performance sur des données inconnues.

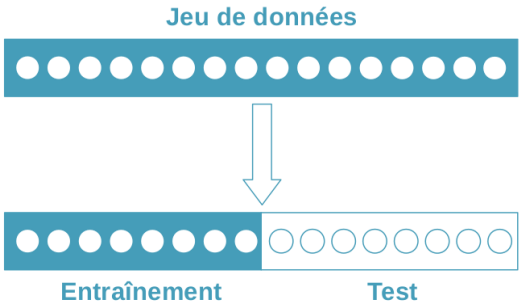


Figure 15 - principe du train/test spliting

- Cross validation split : cette technique consiste à faire plusieurs train/test split diffèrent, elle sera traitée en détails dans la partie de l'évaluation des modèles.

En ce qui concerne la partie de modélisation et pour évaluer nos algorithmes, on va adopter la premier technique, tous les scores seront calculé à partir des données de test.

6.2.Modélisation

On a utilisé la version par défaut de chaque algorithme sans personnalisé les hyperparamètres, le choix des hyperparamètres sera traité dans le chapitre suivant.

6.2.1. Implémentation

Pour la modélisation, on a utilisé principalement cette fonction qui regroupe les étapes suivantes : standardisation, division de données, modélisation et scoring :

```
def classification_accuracy(x,y,modele,predicteurs):
    # x : nos caracteristiques
    # y : ce qu'on cherche a predire
    # model : modele utilise
    # predicteurs : les colonnes a choisir pour l'entrainement >> va nous permettre une grande flexibilite

    #division de donnees en deux parties : entrainemet / test

    xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.4)

    #Standardisation des donnees :

    scaler = StandardScaler().fit(xtrain)
    xtrain=scaler.transform(xtrain)
    xtest = scaler.transform(xtest)


    #L'entrainement et test
    model = modele
    model.fit(xtrain,ytrain)
    prediction=model.predict(xtest)
    return metrics.accuracy_score(prediction,ytest)
```

Figure 16 - implémentation de classification_accuracy

6.2.2. Résultats

La figure suivante résume les résultats obtenus :

Toutes les VARIABLES

```
print("Score de SVM : ",classification_accuracy(x,y,svm.SVC(),x.columns))
Score de SVM :  0.9736842105263158

print("Score de RandomForrest",classification_accuracy(x,y,RandomForestClassifier(n_estimators=100),x.columns))
Score de RandomForrest 0.9780701754385965

print("Score de la Regression Logistique :",classification_accuracy(x,y,LogisticRegression(),x.columns))
Score de la Regression Logistique : 0.9736842105263158
```

TOP 6 des VARIABLES

```
print("Score de SVM : ",classification_accuracy(x,y,svm.SVC(),carac_plus_signifiantes))
Score de SVM :  0.9605263157894737

print("Score de RandomForrest",classification_accuracy(x,y,RandomForestClassifier(n_estimators=100),carac_plus_signifiante
Score de RandomForrest 0.9692982456140351

print("Score de la Regression Logistique :",classification_accuracy(x,y,LogisticRegression(),carac_plus_signifiantes))
Score de la Regression Logistique : 0.9824561403508771
```

Figure 17 - Résultats du scoring des algorithmes

6.2.3. Remarques

On remarque que les trois algorithmes sont performants

Les performances sont élevées (plus que 93%) même si on a passée de 33 variables a 6 variables.

Pour le moment le modèle le plus performant est **RandomForestClassifier**

Voyons si les performances vont s'améliorer après la sélection des hyperparamètres.

Chapitre 7: Evaluation et amélioration des performances

Un **bon modèle de machine learning**, c'est un modèle qui généralise, c'est-à-dire, qui la capacité de faire des prédictions non seulement sur les données que vous avez utilisées pour le construire, mais surtout sur **de nouvelles données** : c'est bien pour ça que l'on parle **d'apprentissage**

Évaluer un modèle sur le jeu de données sur lequel on l'a construit ne nous permet donc pas du tout de savoir comment il se comportera sur de nouvelles données, celles sur lesquelles il est vraiment intéressant de faire de la prédiction.

7.1.Cross Validation

La validation croisée nous permet d'utiliser l'intégralité de notre jeu de données pour l'entraînement et pour la validation.

On découpe le jeu de données en k parties égales. Tour à tour, chacune des k parties est utilisée comme jeu de test. Le reste est utilisé pour l'entraînement.

Le schéma suivant présente une illustration d'une validation croisée de 5 folds :

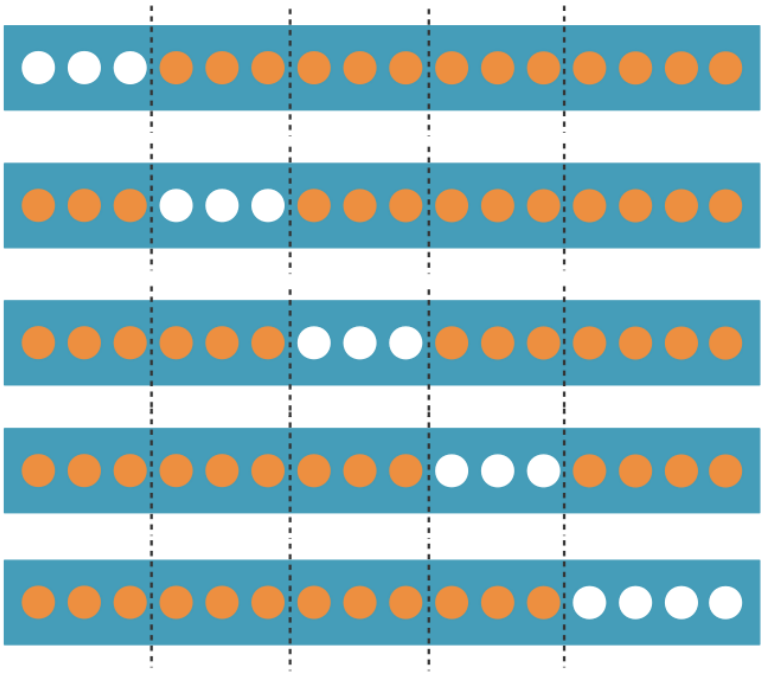


Figure 18 - principe de la validation croisée

Voici l'algorithme général d'une validation croisée :

```
Entrée : données X (dimension nxp), étiquettes y (dimension n), nombre de folds k

Couper [0, 1, ..., n-1] en k parties de taille (n/k). (La dernière partie sera un peu plus petite si n n'est pas un multiple de k)

for i=0 to (k-1):
    Former le jeu de test (X_test, y_test) en restreignant X et y aux indices contenus dans la i-ième partie.
    Former le jeu d'entraînement (X_train, y_train) en restreignant X et y aux autres indices.
    Entraîner l'algorithme sur le jeu d'entraînement
    Utiliser le modèle ainsi obtenu pour prédire sur le jeu de test
        Calculer l'erreur du modèle en comparant les étiquettes prédites aux vraies étiquettes contenues dans y_test

Sortie : la valeur moyenne des erreurs calculées sur les k folds.
```

Figure 19 - l'algorithme de la validation croisée

La figure suivante représente la fonction qu'on a implémentée de la validation croisée :

```python
def classification_accuracy_CV(x,y,modele,predicteurs):

    # x : nos caracteristiques
    # y : ce qu'on cherche a predire
    # model : modele utilise
    # predicteurs : les colonnes a choisir pour l'entrainement >> va nous permettre une grande flexibilite

    #index des splits

    kf = KFold(x.shape[0], n_folds=5)

    error = []

    # boucle sur chaque split
    for train, test in kf:

        xtrain,xtest,ytrain,ytest = x[predicteurs].iloc[train,:],x[predicteurs].iloc[test,:],y[train],y[test]

        scaler = StandardScaler().fit(xtrain)
        xtrain=scaler.transform(xtrain)
        xtest = scaler.transform(xtest)

        model = modele

        model.fit(xtrain,ytrain)
        prediction=model.predict(xtest)

        error += [ metrics.accuracy_score(prediction,ytest)]


    return sum(error)/len(error)
```

Figure 20 - Implémentation de la fonction classification_accuracy_CV

La figure suivante résume les résultats des scores après une validation croisée de 5 folds:

TOUTES les VARIABLES

```
print("Score de validation croisee du SVM : ",classification_accuracy_CV(x,y,svm.SVC(),x.columns))
```

```
Score de validation croisee du SVM :  0.9718832479428661
```

```
print("Score de RandomForrest",classification_accuracy_CV(x,y,RandomForestClassifier(n_estimators=100),x.columns))
```

```
Score de RandomForrest 0.9526005278683435
```

```
print("Score de la Regression Logistique :",classification_accuracy_CV(x,y,LogisticRegression(),x.columns))
```

```
Score de la Regression Logistique : 0.9771774569166279
```

TOP 6 des VARIABLES

```
print("Score de validation croisee du SVM : ",classification_accuracy_CV(x,y,svm.SVC(),carac_plus_signifiantes))
```

```
Score de validation croisee du SVM :  0.9419965843813072
```

```
print("Score de RandomForrest",classification_accuracy_CV(x,y,RandomForestClassifier(n_estimators=100),carac_plus_signifia
```

```
Score de RandomForrest 0.9525539512498058
```

```
print("Score de la Regression Logistique :",classification_accuracy_CV(x,y,LogisticRegression(),carac_plus_signifiantes))
```

```
Score de la Regression Logistique : 0.9473063188945815
```

Figure 21 - Résultats du scoring des algorithmes avec CV

La validation croisée va nous aider énormément dans le choix des paramètres des modèles.

7.2.Sélection des hyper paramètres

Cette partie est très importante, il va nous permettre de choisir les paramètres les plus adaptés pour chaque modèle, c'est une phase primordiale dans l'amélioration des performances d'un modèle.

La figure suivante décrit fonction qui gère le paramétrage des modèles :

Choix des hyperparametres :

```
def Classification_model_gridsearchCV(model,param_grid,predicteurs):

    clf = GridSearchCV(model,param_grid,cv=10,scoring="accuracy")

    xx = StandardScaler().fit_transform(x[predicteurs])

    clf.fit(xx,y)

    print("The best parameter found on development set is :",clf.best_params_)

    print("the bset estimator is ",clf.best_estimator_)

    print("The best score is ",clf.best_score_)
```

Figure 22 - implémentation de la fonction classification_accuracy_gridsearchcv

- **Paramétrage et résultats de SVM**

SVM

```
model=svm.SVC()
param_grid = [

              {'C': [1, 10, 100, 1000],
               'gamma': [0.001, 0.0001],
               'kernel': ['rbf','linear']
              }
 ]
Classification_model_gridsearchCV(model,param_grid,carac_plus_signifiantes)
```

```
The best parameter found on development set is : {'C': 100, 'gamma': 0.001, 'kernel': 'linear'}
the bset estimator is  SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
The best score is  0.9560632688927944
```

Figure 23 - Résultat du paramétrage des SVM

- **Paramétrage et résultats de Random Forest**

RandomForest

```
model = RandomForestClassifier()
param_grid = {
 'n_estimators': [200, 400, 600, 800, 1000]}
Classification_model_gridsearchCV(model,param_grid,carac_plus_signifiantes)
```

```
The best parameter found on development set is : {'n_estimators': 400}
the bset estimator is  RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=400, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
The best score is  0.9543057996485061
```

Figure 24 - Résultat du paramétrage du RandomForest

- **Paramétrage et résultats de la régression logistique**

Regression Logistique

```
model = LogisticRegression()

penalty = ['l1', 'l2']


C = np.logspace(0, 4, 10)


param_grid = dict(C=C, penalty=penalty)

Classification_model_gridsearchCV(model,param_grid,carac_plus_signifiantes)
```

```
The best parameter found on development set is : {'C': 2.7825594022071245, 'penalty': 'l1'}
the bset estimator is  LogisticRegression(C=2.7825594022071245, class_weight=None, dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='ovr', n_jobs=1, penalty='l1', random_state=None,
          solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
The best score is  0.9595782073813708
```

Figure 25 - Résultat du paramétrage de la régression logistique

7.3.Modèle finale

Apres la sélection des variables et des hyperparamètres, l'algorithme qui semble le plus capable à généraliser le problème et d'offrir la plus haute performance est : **LA REGRESSION LOGISTIQUE**

Avec le paramétrage suivant : **'C' = 2.78 ; 'penalty' = 'l1'**

Conclusion