Projet : Cancer du Sein

Echandouri Loubna

30/03/2021

Objectif

Le cancer du sein est le cancer le plus mortel et le plus fréquent chez les femmes. En 2020, 2,261,419 cas de cancers du sein ont été comptés dans le monde entier, dont 684,996 décès. Malgré une baisse du taux d'incidence d'année en année, il semblerait que cette baisse soit irrégulière et peu rapide. La survie nette est de 97% à 1 an et de 88% à 5 ans, selon les derniers chiffres publiés en 2020 par Santé Publique France. Ce cancer du sein peut aussi toucher l'homme, mais dans une très faible proportion (moins de 1% de l'ensemble des cas). Il devient donc nécessaire de dépister tôt.

Ce projet a pour but de faire un travail de prédiction sur le caractère bénin ou malin d'une masse se trouvant dans le sein. Le jeu de données utilisé est le suivant : https://www.kaggle.com/uciml/breast-cancer-wisconsin-data. Il présente les caractéristiques de noyaux cellulaires, dont l'image en 3D a été récupérée après une biopsie par aspiration de la masse en question.

```
dataset <- read.csv('breast_cancer_wisconsin.csv', sep=',')
colnames(dataset)
```

```
##  [1] "id"                      "diagnosis"
##  [3] "radius_mean"             "texture_mean"
##  [5] "perimeter_mean"          "area_mean"
##  [7] "smoothness_mean"         "compactness_mean"
##  [9] "concavity_mean"          "concave.points_mean"
## [11] "symmetry_mean"           "fractal_dimension_mean"
## [13] "radius_se"               "texture_se"
## [15] "perimeter_se"            "area_se"
## [17] "smoothness_se"           "compactness_se"
## [19] "concavity_se"            "concave.points_se"
## [21] "symmetry_se"             "fractal_dimension_se"
## [23] "radius_worst"            "texture_worst"
## [25] "perimeter_worst"         "area_worst"
## [27] "smoothness_worst"        "compactness_worst"
## [29] "concavity_worst"         "concave.points_worst"
## [31] "symmetry_worst"          "fractal_dimension_worst"
## [33] "X"
```

Voilà les variables dans leur globalité qu'on peut trouver :

- id
- diagnosis (M = malignant, B = benign) : la variable cible

et il y a 10 types de variables explicatives continues, dont les valeurs sont données pour chaque noyau cellulaire:

- radius (moyenne des distances du centre aux points sur le périmètre)
- texture (écart type des valeurs des niveaux de gris)
- perimeter (périmètre)
- area (aire)
- smoothness (variation locale des rayons)
- compactness (perimeter^2 / area - 1.0)
- concavity (sévérité des parties concaves du contour)
- concave points (nombre de parties concaves du contour)
- symmetry (symétrie)
- fractal dimension ("approximation littorale" - 1)

Pour chacun de ces types de variables, on trouve différentes mesures. Par exemple, pour le type `radius`, on trouve `radius_worst`, `radius_se` et `radius_means`.

```
## [1] 569  33
```

Nous commençons donc avec un jeu de données de 569 observations, une variable cible (`diagnosis`), une variable d'identification (`id`), et 31 variables explicatives (10 types différents). On peut les observer ci-après :

```
##                          1          2          3          4
## id                       "  842302" "  842517" "84300903" "84348301"
## diagnosis                "TRUE"     "TRUE"     "TRUE"     "TRUE"
## radius_mean              "17.99"    "20.57"    "19.69"    "11.42"
## texture_mean             "10.38"    "17.77"    "21.25"    "20.38"
## perimeter_mean           "122.80"   "132.90"   "130.00"   " 77.58"
## area_mean                "1001.0"   "1326.0"   "1203.0"   " 386.1"
## smoothness_mean          "0.11840"  "0.08474"  "0.10960"  "0.14250"
## compactness_mean         "0.27760"  "0.07864"  "0.15990"  "0.28390"
## concavity_mean           "0.3001"   "0.0869"   "0.1974"   "0.2414"
## concave.points_mean      "0.14710"  "0.07017"  "0.12790"  "0.10520"
## symmetry_mean            "0.2419"   "0.1812"   "0.2069"   "0.2597"
## fractal_dimension_mean   "0.07871"  "0.05667"  "0.05999"  "0.09744"
## radius_se                "1.0950"   "0.5435"   "0.7456"   "0.4956"
## texture_se               "0.9053"   "0.7339"   "0.7869"   "1.1560"
## perimeter_se             "8.589"    "3.398"    "4.585"    "3.445"
## area_se                  "153.40"   " 74.08"   " 94.03"   " 27.23"
## smoothness_se            "0.006399" "0.005225" "0.006150" "0.009110"
## compactness_se           "0.04904"  "0.01308"  "0.04006"  "0.07458"
## concavity_se             "0.05373"  "0.01860"  "0.03832"  "0.05661"
## concave.points_se        "0.01587"  "0.01340"  "0.02058"  "0.01867"
## symmetry_se              "0.03003"  "0.01389"  "0.02250"  "0.05963"
## fractal_dimension_se     "0.006193" "0.003532" "0.004571" "0.009208"
## radius_worst             "25.38"    "24.99"    "23.57"    "14.91"
## texture_worst            "17.33"    "23.41"    "25.53"    "26.50"
## perimeter_worst          "184.60"   "158.80"   "152.50"   " 98.87"
## area_worst               "2019.0"   "1956.0"   "1709.0"   " 567.7"
## smoothness_worst         "0.1622"   "0.1238"   "0.1444"   "0.2098"
## compactness_worst        "0.6656"   "0.1866"   "0.4245"   "0.8663"
## concavity_worst          "0.7119"   "0.2416"   "0.4504"   "0.6869"
## concave.points_worst     "0.2654"   "0.1860"   "0.2430"   "0.2575"
## symmetry_worst           "0.4601"   "0.2750"   "0.3613"   "0.6638"
## fractal_dimension_worst  "0.11890"  "0.08902"  "0.08758"  "0.17300"
## X                        NA         NA         NA         NA
```

Mise en forme de la donnée

Avant de commencer l'analyse des données, on commence par examiner les valeurs nulles s'il y en a. On remarque qu'il existe une colonne `X` contenant des `NaN` exclusivement. Le reste des variables ne contient pas de `NaN`.

```
dataset <- subset(dataset, select=-c(X, id)) #deletion of X column full of NaN
# and id because of high cardinality (not useful for now)
skim(dataset)
```

Table 1: Data summary

Name	dataset
Number of rows	569
Number of columns	31
Column type frequency:	
factor	1
numeric	30
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
diagnosis	0	1	FALSE	2	FAL: 357, TRU: 212

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
radius_mean	0	1	14.13	3.52	6.98	11.70	13.37	15.78	28.11	
texture_mean	0	1	19.29	4.30	9.71	16.17	18.84	21.80	39.28	
perimeter_mean	0	1	91.97	24.30	43.79	75.17	86.24	104.10	188.50	
area_mean	0	1	654.89	351.91	143.50	420.30	551.10	782.70	2501.00	
smoothness_mean	0	1	0.10	0.01	0.05	0.09	0.10	0.11	0.16	
compactness_mean	0	1	0.10	0.05	0.02	0.06	0.09	0.13	0.35	
concavity_mean	0	1	0.09	0.08	0.00	0.03	0.06	0.13	0.43	
concave.points_mean	0	1	0.05	0.04	0.00	0.02	0.03	0.07	0.20	
symmetry_mean	0	1	0.18	0.03	0.11	0.16	0.18	0.20	0.30	
fractal_dimension_mean	0	1	0.06	0.01	0.05	0.06	0.06	0.07	0.10	
radius_se	0	1	0.41	0.28	0.11	0.23	0.32	0.48	2.87	
texture_se	0	1	1.22	0.55	0.36	0.83	1.11	1.47	4.88	
perimeter_se	0	1	2.87	2.02	0.76	1.61	2.29	3.36	21.98	
area_se	0	1	40.34	45.49	6.80	17.85	24.53	45.19	542.20	
smoothness_se	0	1	0.01	0.00	0.00	0.01	0.01	0.01	0.03	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
compactness_se	0	1	0.03	0.02	0.00	0.01	0.02	0.03	0.14	
concavity_se	0	1	0.03	0.03	0.00	0.02	0.03	0.04	0.40	
concave.points_se	0	1	0.01	0.01	0.00	0.01	0.01	0.01	0.05	
symmetry_se	0	1	0.02	0.01	0.01	0.02	0.02	0.02	0.08	
fractal_dimension_se	0	1	0.00	0.00	0.00	0.00	0.00	0.00	0.03	
radius_worst	0	1	16.27	4.83	7.93	13.01	14.97	18.79	36.04	
texture_worst	0	1	25.68	6.15	12.02	21.08	25.41	29.72	49.54	
perimeter_worst	0	1	107.26	33.60	50.41	84.11	97.66	125.40	251.20	
area_worst	0	1	880.58	569.36	185.20	515.30	686.50	1084.00	4254.00	
smoothness_worst	0	1	0.13	0.02	0.07	0.12	0.13	0.15	0.22	
compactness_worst	0	1	0.25	0.16	0.03	0.15	0.21	0.34	1.06	
concavity_worst	0	1	0.27	0.21	0.00	0.11	0.23	0.38	1.25	
concave.points_worst	0	1	0.11	0.07	0.00	0.06	0.10	0.16	0.29	
symmetry_worst	0	1	0.29	0.06	0.16	0.25	0.28	0.32	0.66	
fractal_dimension_worst	0	1	0.08	0.02	0.06	0.07	0.08	0.09	0.21	

Il n'existe plus aucune autre donnée nulle (`NaN`) dans le jeu de données. Le reste des variables explicatives sont bien continues comme le montre le data summary ci-dessus. Toutefois, elles sont d'ordre différent. Ainsi, nous allons standardiser les données afin d'éviter des impacts négatifs sur la modélisation plus tard.

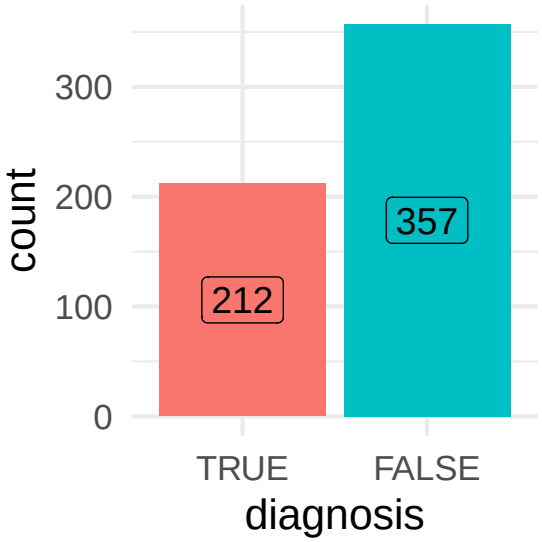
```
##  TRUE FALSE
##   212   357
```

Pour ce qui est de la variable cible `diagnosis`, on peut voir qu'elle n'est pas n'est pas excessivement déséquilibrée: sur 569 observations, 212 ont le label 'positif' (`Malignant`, qu'on a redéfini en `TRUE`) et 357 ont le label négatif (`Benign`, qu'on a redéfini en `FALSE`).

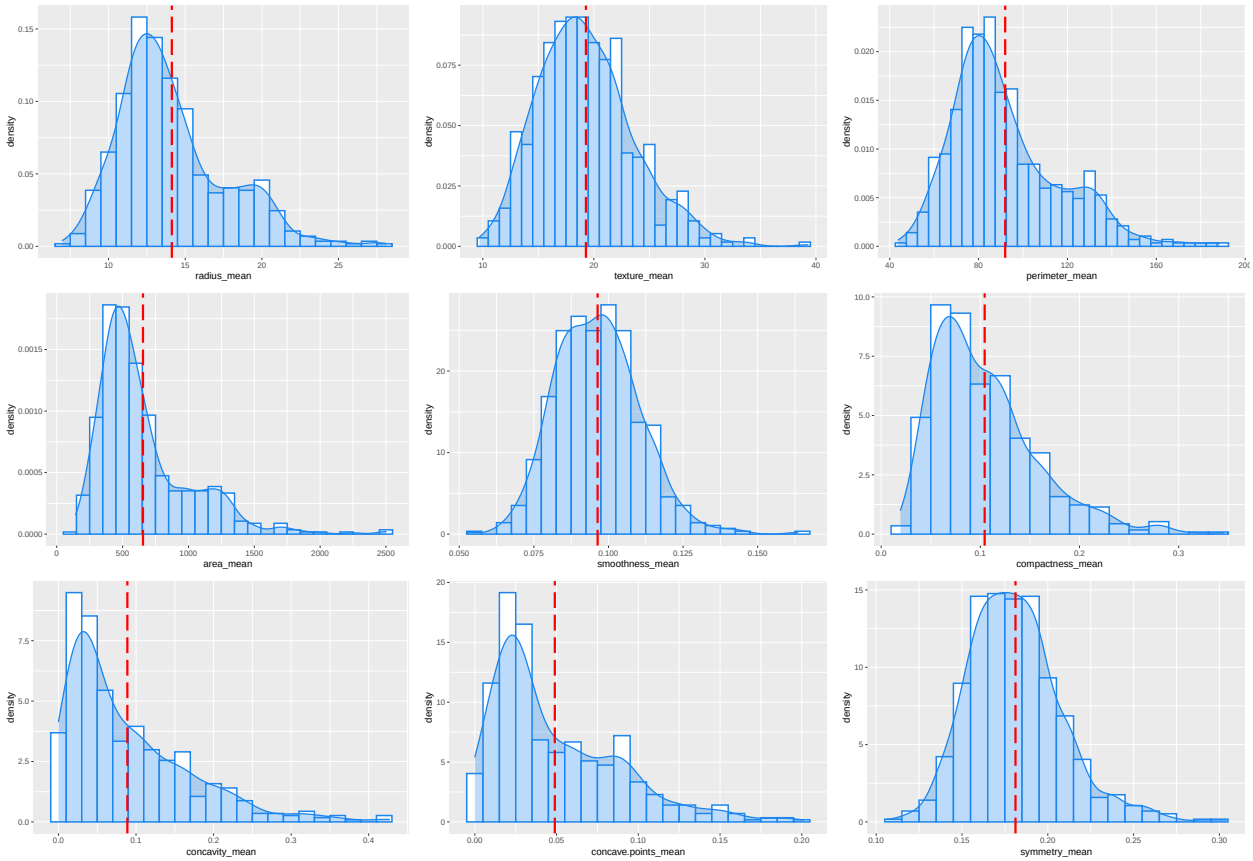
Analyse univariée des variables explicatives et de la cible

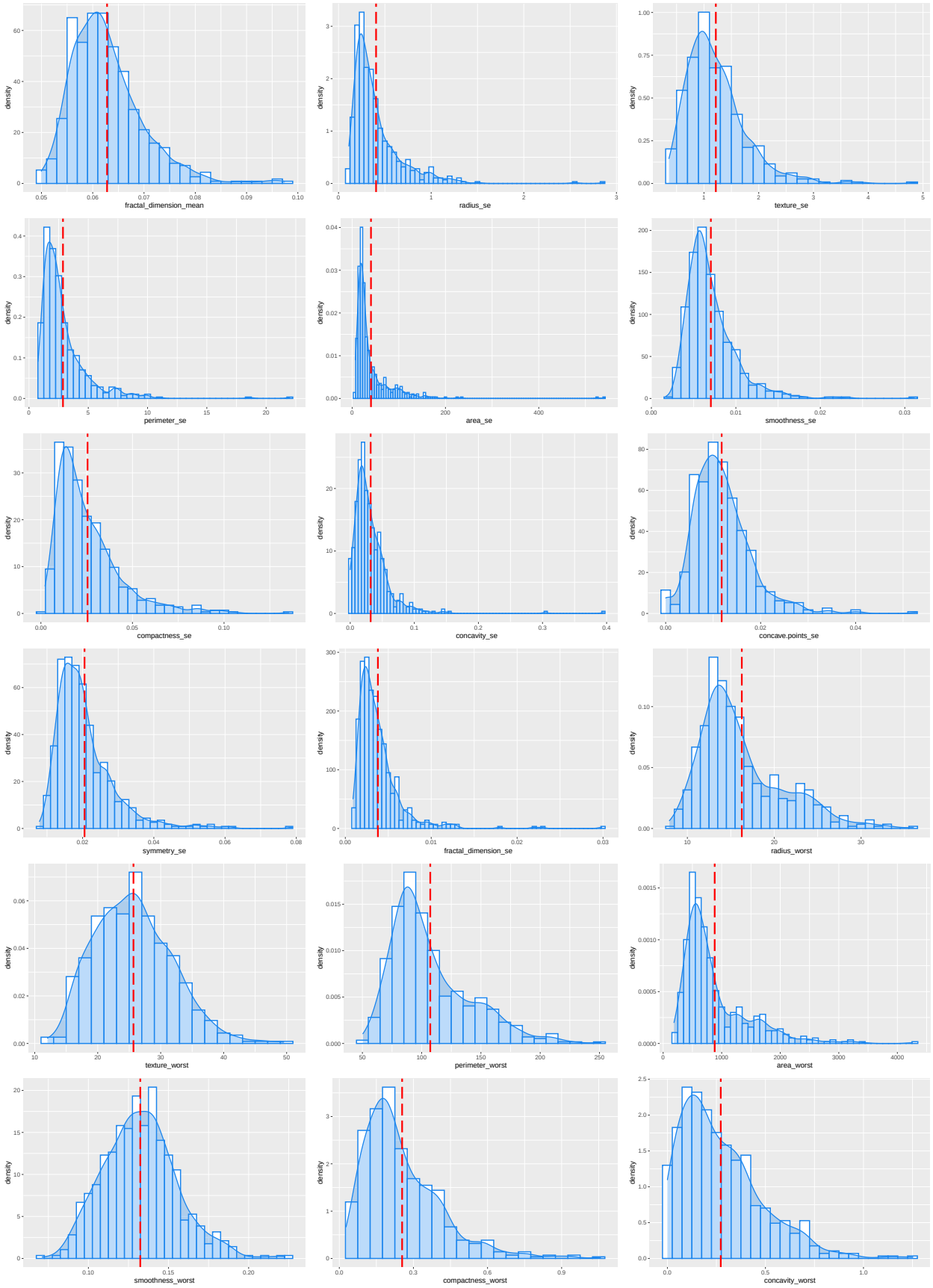
Dans cette partie, nous allons essayer de regarder les distributions générales des différentes variables explicatives, dans le but de voir comment évoluent les variables et la distribution de la variable cible.

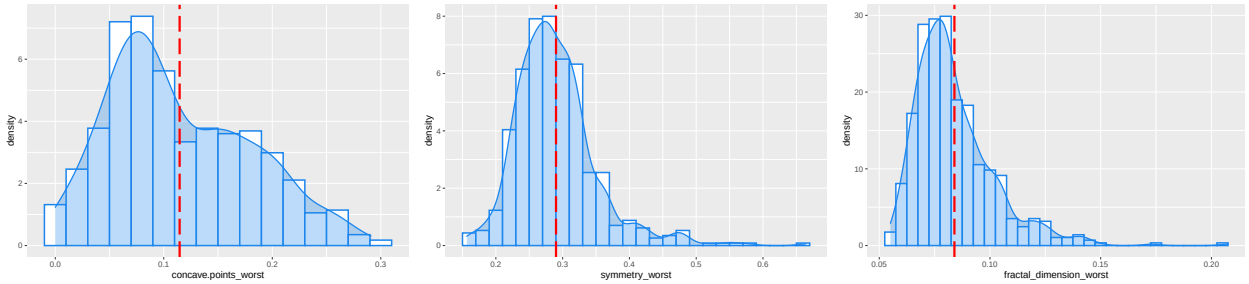
Variable cible : `diagnosis`



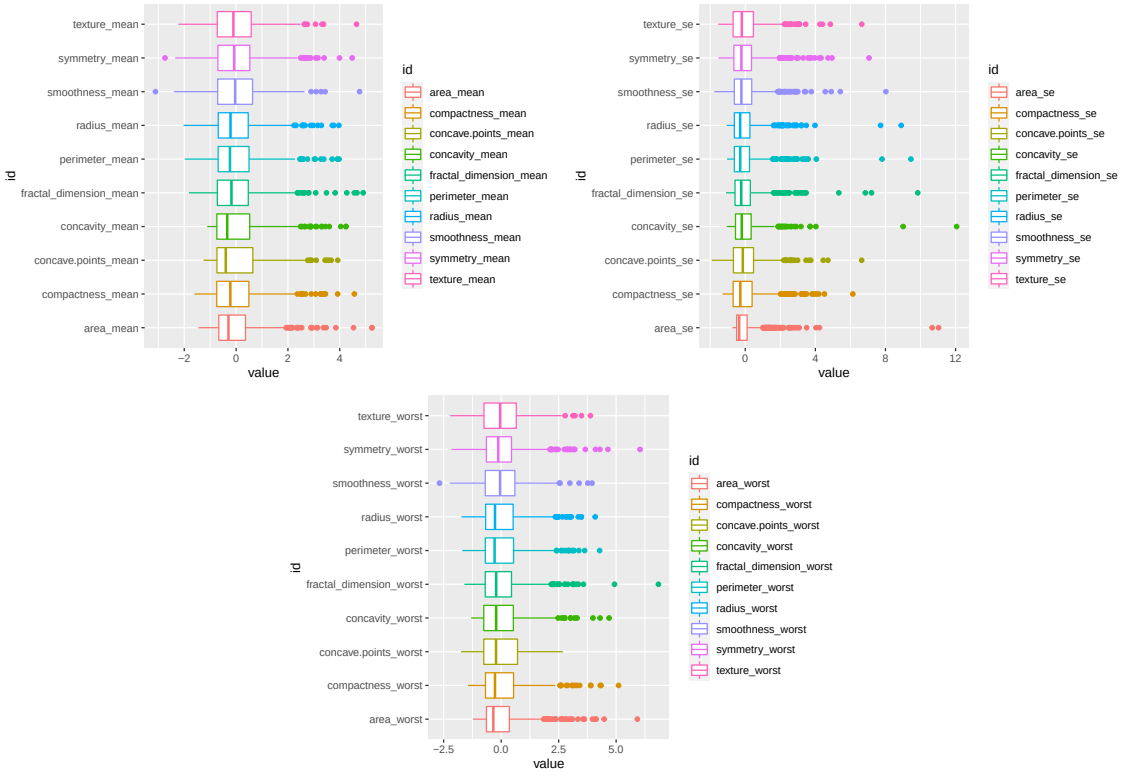
Variables explicatives




density
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave.points_se
symmetry_se
fractal_dimension_se
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst



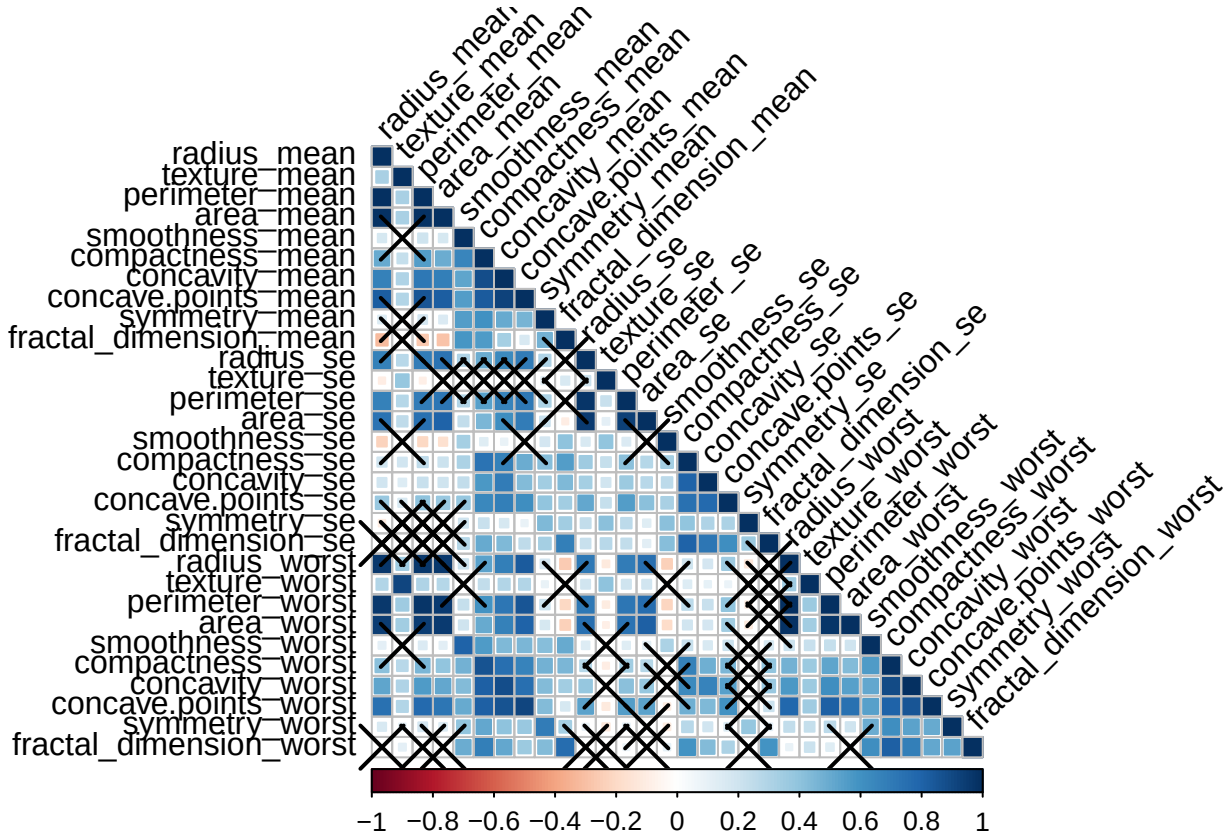
Détection d'outliers



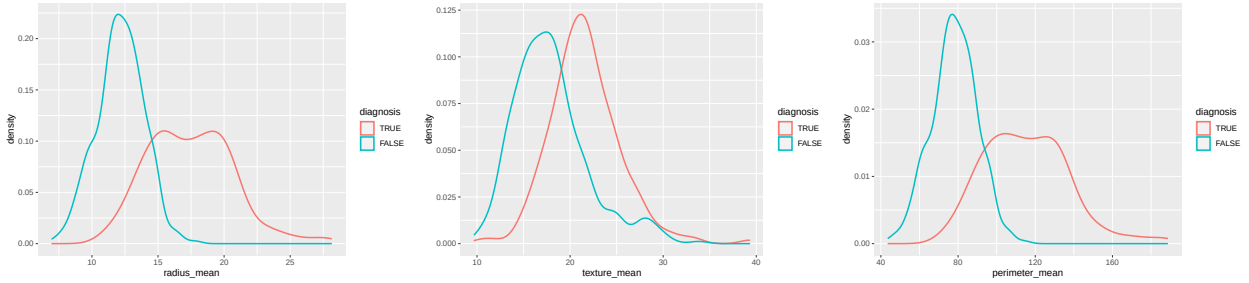
Les box-plots, ci-dessous, mettent en évidence les différents outliers pour chacune des variables explicatives, toutefois certains des ces points sont à préserver. En effet, certains points 'outliers' sont considérés comme 'leverage points', et contiennent de l'information importante malgré leur valeur distincte. Nous verrons plus tard comment séparer les vrais 'outliers' des 'leverage points'.

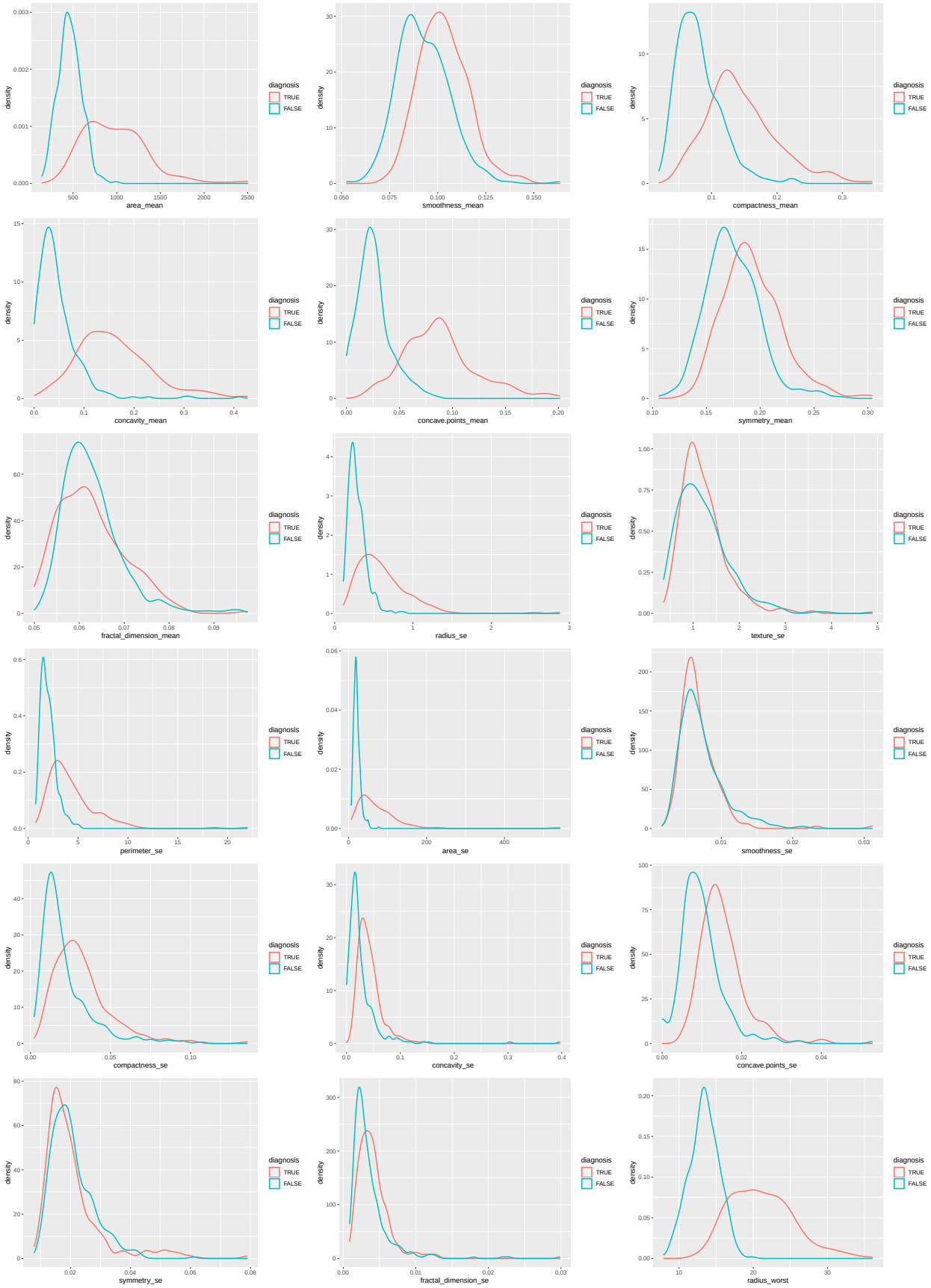
Analyse multivariée des variables explicatives et de la cible

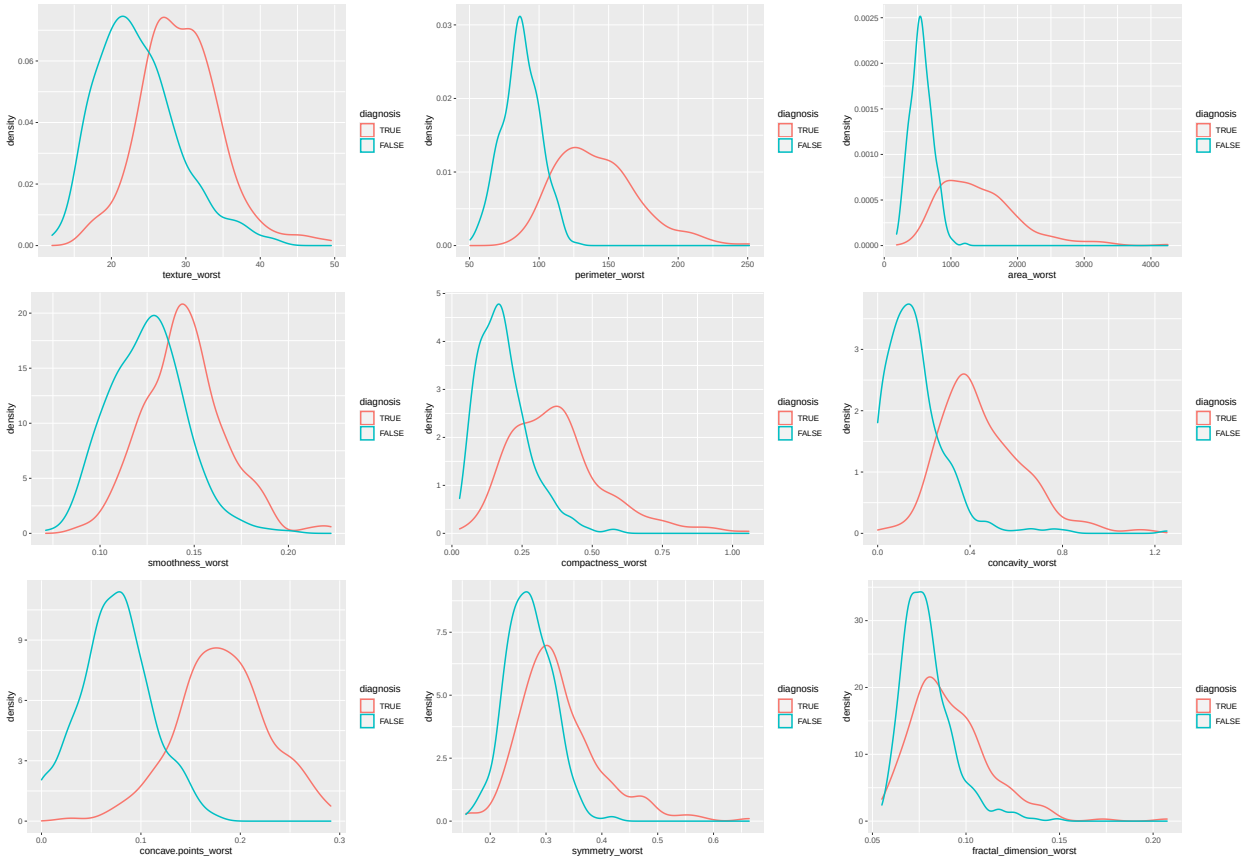
Matrix de correlation



On remarque que beaucoup de variables explicatives sont fortement corrélées, souvent positivement. Comme par exemple `radius_mean` et `perimeter_mean` ou encore `perimeter_worst`et `area_mean`. Cela nous sera utile dans la suite, et en vue de la préparation à la modélisation. En effet, sachant qu'on a un nombre assez faible d'observations (`569`), il est nécessaire de réduire le nombre de variables explicatives afin d'obtenir un modèle assez généralisable. Le fait que beaucoup de ces variables soient corrélées nous aide grandement, car cette réduction de dimension ne supprimera pas beaucoup d'information.




density
diagnosis
TRUE
FALSE
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave.points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave.points_se
symmetry_se
fractal_dimension_se
radius_worst



Grâce aux graphiques de distributions des valeurs croisées, on peut identifier à coup d'oeil les variables explicatives dont la distribution change en fonction de la cible. Ces variables ont donc un bon potentiel de prédiction. Par exemple, on remarque que la variable explicative `concave.points_worst` a une distribution gaussienne de moyenne `0.07` environ pour les diagnostics négatifs. Mais elle a une distribution gaussienne de moyenne `1.7` environ pour les diagnostics positifs. A l'inverse, le graphique de la variables explicative `symmetry_se` montre que la distribution change de manière infime entre un diagnostic positif et un diagnostic négatif.

L'analyse multivariée nous a permis de voir qu'une réduction de dimension est possible. D'un côté par le biais de fortes corrélations entre variables explicatives, mais également par le biais du potentiel de prédiction de ces dernières. L'importance de certaines variables plutôt que d'autres sera analysée de plus prêt dans la suite.

Préparation à la modélisation

Exclusion d'outliers

Afin d'exclure les outliers, nous allons utiliser la fonction `isolationForest` (package `solitude`). Cela va nous permettre de retirer du jeu de données uniquement les observations aux valeurs extrêmes ne contribuant pas à amener de l'information bénéfique au modèle.

```
iforest <- isolationForest$new()
iforest$fit(dataset_scaled)
dataset_scaled$pred <- iforest$predict(dataset_scaled)
dataset_scaled$outlier <- as.factor(ifelse(dataset_scaled$pred$anomaly_score >=0.65,
```

```
                                          'outlier', 'normal'))
summary(dataset_scaled$outlier)
```

9 observations ont été retirées car ayant un `anomaly_score` supérieur à 65%, et plus le score est proche de 1 plus il est probable que l'observation en question est une anomalie. Ci-après, une partie des observations considérées comme anomalies :

```
##                          4             79            83            109
## diagnosis                "TRUE"        "TRUE"        "TRUE"        "TRUE"
## radius_mean              "-0.7682333"  " 1.7175438"  " 3.1477170"  " 2.3106116"
## texture_mean             "0.25350905"  "1.08819172"  "1.30674372"  "0.08843253"
## perimeter_mean           "-0.5921661"  " 2.1289356"  " 3.2730165"  " 2.5034369"
## area_mean                "-0.7637917"  " 1.6768605"  " 3.4755947"  " 2.4270435"
## smoothness_mean          "3.2806668"   "2.2923368"   "0.7067426"   "2.5767483"
## compactness_mean         "3.399917"    "4.564409"    "3.070452"    "3.265480"
## concavity_mean           "1.914213"    "3.595100"    "3.074527"    "4.234841"
## concave.points_mean      "1.450431"    "2.873007"    "3.494096"    "3.437399"
## symmetry_mean            "2.86486215"  "3.99201189"  "0.06340261"  "2.71530507"
## fractal_dimension_mean   "4.9066020"   "2.6375968"   "0.7113502"   "1.0753541"
## radius_se                "0.3260865"   "1.8986793"   "1.7746316"   "2.9202696"
## texture_se               "-0.1103120"  " 1.2111819"  " 0.4661422"  " 0.5948473"
## perimeter_se             "0.2863415"   "2.8602160"   "2.2335636"   "3.5531442"
## area_se                  "-0.2881246"  " 1.6720431"  " 1.7511796"  " 2.8502980"
## smoothness_se            " 0.6890953"  " 1.1120737"  " 0.3746925"  "-0.1751793"
## compactness_se           "2.741868"    "2.393982"    "1.756285"    "3.417537"
## concavity_se             "0.8187979"   "2.5576800"   "0.8416562"   "2.3887279"
## concave.points_se        "1.114027"    "2.290634"    "1.378196"    "2.107498"
## symmetry_se              " 4.728520"   " 7.065700"   "-1.196692"   " 1.279606"
## fractal_dimension_se     "2.0457109"   "0.8284344"   "0.7929100"   "0.4694115"
## radius_worst             "-0.281217"   " 1.469161"   " 2.840911"   " 2.509870"
## texture_worst            "0.1338663"   "0.9831636"   "1.2922948"   "0.3795443"
## perimeter_worst          "-0.2497196"  " 1.8760124"  " 3.1080621"  " 2.9622398"
## area_worst               "-0.5495377"  " 1.3039567"  " 2.9531856"  " 2.5983994"
## smoothness_worst         "3.391291"    "1.380992"    "1.091930"    "1.652536"
## compactness_worst        "3.889975"    "2.301659"    "2.245728"    "2.831098"
## concavity_worst          "1.987839"    "2.377056"    "1.799462"    "3.300726"
## concave.points_worst     "2.173873"    "2.071945"    "2.618099"    "2.683516"
## symmetry_worst           " 6.0407261"  " 4.1043288"  "-0.8821368"  " 1.8656724"
## fractal_dimension_worst  "4.9306719"   "0.8689414"   "1.1712458"   "0.7720490"
## pred.id                  "  4.0000000" " 79.0000000" " 83.0000000" "109.0000000"
## pred.average_depth       "  5.8200000" "  5.7300000" "  6.3400000" "  5.6900000"
## pred.anomaly_score       "  0.6743039" "  0.6784257" "  0.6509752" "  0.6802656"
## outlier                  "outlier"     "outlier"     "outlier"     "outlier"
```

Réduction de dimension

Comme nous l'avons observé précédemment, certaines variables explicatives sont fortement corrélées, et donc il y a redondance de l'information. Nous allons donc supprimer celles qui sont redondantes. Nous aurions pu également effectuer une ACP, mais par désir de préserver une intérprétabilité du modèle, on a préféré la méthode précédente.

```
full_formula <- as.formula(str_c('diagnosis ~ ', str_c(features, collapse = ' + ')))
redun <- redun(formula = full_formula, data = dataset_scaled, r2 = 0.9)
print(redun, long = FALSE)
redun_features <- redun$Out
print(redun_features)
```

Modélisation et prédiction

Nous pouvons désormais commencer la modélisation et pour ce problème de classification binaire, on va examiner les résultats d'une régression logistique et d'arbres aléatoires. Mais avant tout, nous allons diviser le jeu de données en deux ensembles : un ensemble d'apprentissage et un ensemble de test. L'ensemble d'apprentissage correspond à 75% du jeu de données initial.

```
reduced_features <- intersect(features, redun_features)
reduced_formula <- as.formula(str_c('diagnosis ~ ', str_c(reduced_features, collapse = ' + ')))
dataset_scaled_split <- initial_split(dataset_scaled, p = 0.7, strata = diagnosis)
train <- training(dataset_scaled_split)
train <- select(train , all_of(c(reduced_features,'diagnosis')))
test <- testing(dataset_scaled_split)
test <- select(test , all_of(c(reduced_features,'diagnosis')))
t(head(train))
```

```
##                         3             8             9
## radius_mean             " 1.5784992"  "-0.1184126"  "-0.3198854"
## perimeter_mean          " 1.56512598" "-0.07280278" "-0.18391855"
## area_mean               " 1.5575132"  "-0.2187724"  "-0.3838695"
## compactness_mean        " 1.0519999"  " 1.1391001"  " 1.6825294"
## concavity_mean          " 1.3622798"  " 0.0609721"  " 1.2180246"
## concave.points_mean     " 2.0354398"  " 0.2817024"  " 1.1496800"
## fractal_dimension_mean "-0.3976580"  " 1.6588935"  " 1.5710793"
## perimeter_se            " 0.8501802"  " 0.4896202"  "-0.2275432"
## area_se                 " 1.180297518" " 0.233516951" "-0.352093318"
## radius_worst            " 1.5105411"  " 0.1636190"  "-0.1612147"
## texture_worst           "-0.02395331" " 0.40069534" " 0.82208998"
## perimeter_worst         " 1.34629062" " 0.09936115" "-0.03158132"
## smoothness_worst        " 0.5269438"  " 1.4466882"  " 1.6612952"
## compactness_worst       " 1.0819801"  " 0.7241483"  " 1.8167112"
## concavity_worst         " 0.85422232" "-0.02103534" " 1.27890922"
## concave.points_worst    " 1.9532817"  " 0.6236470"  " 1.3903928"
## diagnosis               "TRUE"        "TRUE"        "TRUE"
##                         11            12            13
## radius_mean             " 0.5370834"  " 0.4689800"  " 1.4309417"
## perimeter_mean          " 0.44162208" " 0.47866067" " 1.66389556"
## area_mean               " 0.4060959"  " 0.3583570"  " 1.3301850"
## compactness_mean        "-0.7129146"  " 0.4707010"  " 2.6785008"
## concavity_mean          "-0.7000684"  " 0.1347304"  " 1.4764296"
## concave.points_mean     "-0.4043298"  " 0.4417422"  " 1.6205218"
## fractal_dimension_mean "-0.8253981"  "-0.2801003"  " 2.1532024"
## perimeter_se            "-0.1978675"  " 0.3451983"  " 4.0576315"
## area_se                 " 0.003801211" " 0.303860527" " 1.667646605"
## radius_worst            " 0.6043170"  " 0.8588046"  " 0.9705309"
## texture_worst           " 1.33459697" " 0.26077280" " 0.69355648"
## perimeter_worst         " 0.49218857" " 0.87013617" " 1.32248289"
## smoothness_worst        "-0.6249267"  " 0.3167164"  "-1.2556086"
## compactness_worst       "-0.6302737"  " 1.9489119"  " 0.8646116"
## concavity_worst         "-0.60533934" " 0.59586313" " 0.43960136"
## concave.points_worst    "-0.2260109"  " 1.0100626"  " 0.9446458"
## diagnosis               "TRUE"        "TRUE"        "TRUE"
```

Logistic Regression

La régression logistique est le premier choix de modèle, car il se prête très bien à ce genre de problème de classification binaire. De plus, il est relativement simple à utiliser et peu coûteux.

```r
set.seed(0)
glm_mod <- glm(reduced_formula, family=binomial(logit), data=train)
summary(glm_mod)
```

```
##
## Call:
## glm(formula = reduced_formula, family = binomial(logit), data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.55864   0.00000   0.00034   0.00976   1.80371
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -1.8318     1.7548  -1.044  0.29654
## radius_mean             128.0026    61.9300   2.067  0.03874 *
## perimeter_mean         -126.4985    60.8818  -2.078  0.03773 *
## area_mean                -1.6787    17.0413  -0.099  0.92153
## compactness_mean         15.4119     6.6365   2.322  0.02022 *
## concavity_mean            0.9958     4.2579   0.234  0.81509
## concave.points_mean      -5.7723     3.9239  -1.471  0.14128
## fractal_dimension_mean   -0.3950     1.6489  -0.240  0.81065
## perimeter_se             -5.8745     4.1390  -1.419  0.15581
## area_se                   0.2750     6.1811   0.044  0.96451
## radius_worst            -34.0235    15.3761  -2.213  0.02691 *
## texture_worst            -2.2476     0.7183  -3.129  0.00176 **
## perimeter_worst          19.8521    14.3747   1.381  0.16727
## smoothness_worst         -2.0796     0.9981  -2.084  0.03719 *
## compactness_worst        -4.4987     3.5984  -1.250  0.21123
## concavity_worst          -0.7747     2.9782  -0.260  0.79478
## concave.points_worst     -2.3106     2.3358  -0.989  0.32256
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 515.310  on 392  degrees of freedom
## Residual deviance:  38.145  on 376  degrees of freedom
## AIC: 72.145
##
## Number of Fisher Scoring iterations: 11
```

```r
odds.ratio(glm_mod)
```

```
## Waiting for profiling to be done...
```

```
##                                OR       2.5 %      97.5 %        p
## (Intercept)             1.6012e-01  2.7825e-03  4.1402e+00 0.296536
## radius_mean             3.8980e+55  3.1040e+09 3.8885e+120 0.038744 *
## perimeter_mean          1.1545e-55 8.5578e-119  0.0000e+00 0.037730 *
## area_mean               1.8663e-01  7.2438e-17  1.7469e+13 0.921531
## compactness_mean        4.9354e+06  1.3186e+02  2.0858e+14 0.020216 *
## concavity_mean          2.7068e+00  5.2846e-04  1.6682e+04 0.815091
## concave.points_mean     3.1125e-03  6.8502e-07  5.6209e+00 0.141277
## fractal_dimension_mean  6.7365e-01  2.3114e-02  2.0117e+01 0.810652
## perimeter_se            2.8103e-03  2.6034e-07  1.0337e+01 0.155813
## area_se                 1.3166e+00  1.6103e-06  3.1134e+05 0.964508
```

```
## radius_worst            1.6742e-15  3.5637e-32  8.0000e-04 0.026915 *
## texture_worst           1.0566e-01  1.5927e-02  3.3080e-01 0.001755 **
## perimeter_worst         4.1848e+08  5.3874e-04  9.2651e+22 0.167266
## smoothness_worst        1.2497e-01  1.2112e-02  7.0160e-01 0.037194 *
## compactness_worst       1.1124e-02  9.7604e-07  2.8131e+00 0.211234
## concavity_worst         4.6086e-01  8.6175e-04  1.7027e+02 0.794776
## concave.points_worst    9.9202e-02  5.9645e-04  6.9314e+00 0.322555
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On peut voir , d'après la valeur des `OR` (odd ratios), que les variables explicatives présentes dans le modèle ont toutes un effet sur la variable cible, car leur odd ratio est largement supérieur ou largement inférieur à 1.

```
pR2(glm_mod)
```

```
## fitting null model for pseudo-r2
```

```
##          llh      llhNull           G2     McFadden          r2ML          r2CU
##  -19.0725696 -257.6551659  477.1651925    0.9259764    0.7030409    0.9623980
```

Le pseudo-R^2 de MacFadden vaut 0.92 et indique donc que notre modèle est bon, mais nous allons confirmer ça avec l'ensemble de test.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction TRUE FALSE
##      TRUE    60     5
##      FALSE    1   101
##
##                Accuracy : 0.9641
##                  95% CI : (0.9234, 0.9867)
##     No Information Rate : 0.6347
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9236
##
##  Mcnemar's Test P-Value : 0.2207
##
##             Sensitivity : 0.9836
##             Specificity : 0.9528
##          Pos Pred Value : 0.9231
##          Neg Pred Value : 0.9902
##              Prevalence : 0.3653
##          Detection Rate : 0.3593
##    Detection Prevalence : 0.3892
##       Balanced Accuracy : 0.9682
##
##        'Positive' Class : TRUE
##
```

On observe une `accuracy` de 96% ce qui est un très bon résultat, mais il ne faut pas oublier le cœur du problème. Dans notre cas, la métrique `accuracy` est nécessaire mais pas suffisante. En effet, la métrique

`recall` ou `sensitivity` est la seconde métrique la plus importante. Celle-ci nous permet de nous assurer qu'on évite au maximum les faux négatifs, c'est à dire un diagnostic de cancer négatif alors qu'il est positif. L'effet d'une telle erreur est bien plus grave que celui d'un faux positif. Dans notre cas, la `sensitivity` vaut 98%, ce qui est très bon. Ce taux peut peut-être être amélioré, par exemple, en ajoutant des cas positifs dans le jeu de données de base (qui, on le rappelle, était un peu déséquilibré).

Random Forest

Nous allons voir maintenant le modèle des forêts aléatoires, et nous allons essayer de comparer les résultats trouvés précédemment avec le modèle de régression logistique.

```
set.seed(0)
rf_mod <- ranger(reduced_formula, data=train, mtry=4,
                 num.trees=200, write.forest=TRUE, importance='permutation')
rf_mod
```

```
## Ranger result
##
## Call:
##  ranger(reduced_formula, data = train, mtry = 4, num.trees = 200,      write.forest = TRUE, importan
##
## Type:                             Classification
## Number of trees:                  200
## Sample size:                      393
## Number of independent variables:  16
## Mtry:                             4
## Target node size:                 1
## Variable importance mode:         permutation
## Splitrule:                        gini
## OOB prediction error:             5.34 %
```

```
predtest <- predict(rf_mod, test)$prediction
confusionMatrix(predtest, test$diagnosis, positive='TRUE')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction TRUE FALSE
##      TRUE    57     2
##      FALSE    4   104
##
##                Accuracy : 0.9641
##                  95% CI : (0.9234, 0.9867)
##     No Information Rate : 0.6347
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.922
##
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.9344
##             Specificity : 0.9811
```

```
##          Pos Pred Value : 0.9661
##          Neg Pred Value : 0.9630
##              Prevalence : 0.3653
##          Detection Rate : 0.3413
##    Detection Prevalence : 0.3533
##       Balanced Accuracy : 0.9578
##
##        'Positive' Class : TRUE
##
```

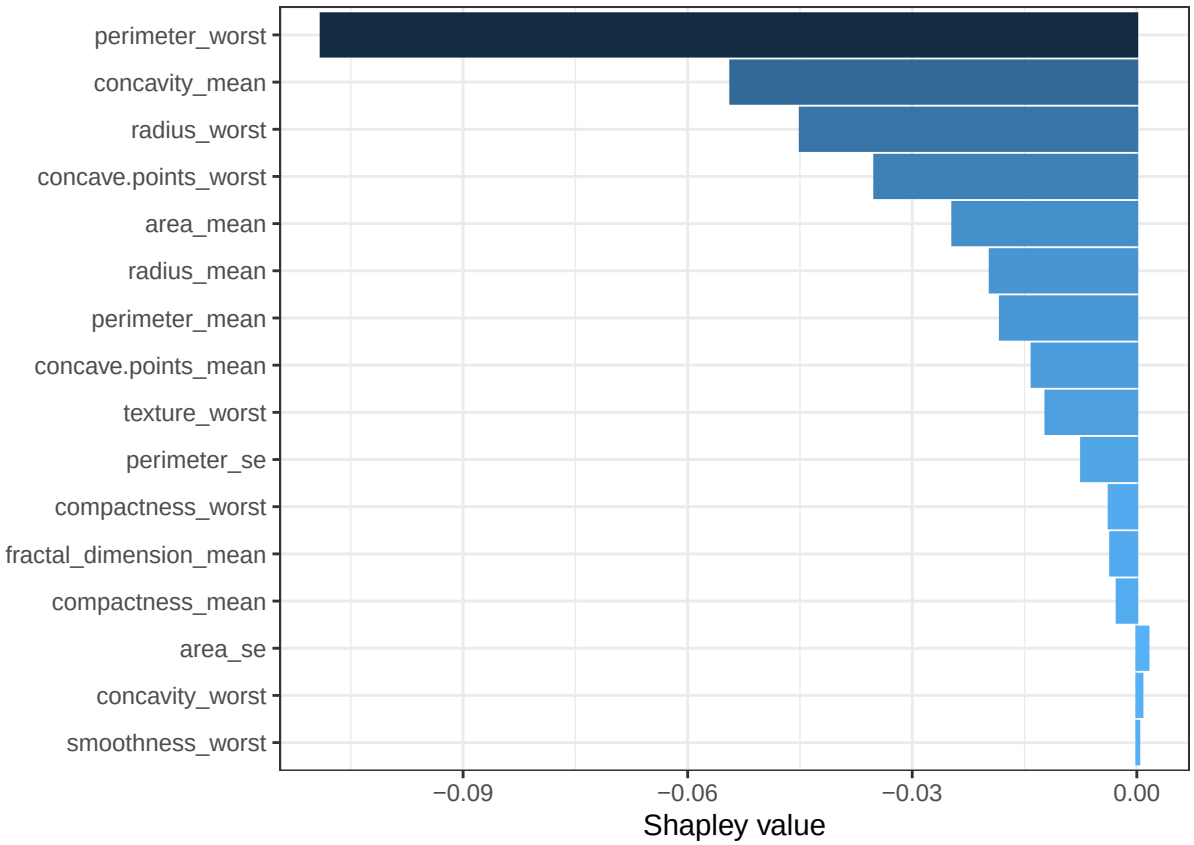
Le modèle `RandomForest` performe moins bien que le modèle `glm`. On voit que l'`accuracy` vaut 96%, un taux quasiment similaire au modèle de régression logistique. Mais en ce qui concerne la métrique `sensitivity`, elle vaut 93% pour `RandomForest`, ce qui est bien plus faible que pour le modèle de régression linéaire. De manière générale, on préférera le modèle de régression logistique, car il est plus simple et moins coûteux, et plus performant.

Interprétabilité

Dans cette dernière sous-partie, nous allons utiliser la librairie `fastshap`, et les valeurs Shap, afin de comprendre comment chaque variable explicative impacte la cible et à quel point. Des modèles complexes, comme `RandomForest`, nécessitent ce genre de méthode pour pouvoir faire une intérprétation claire.

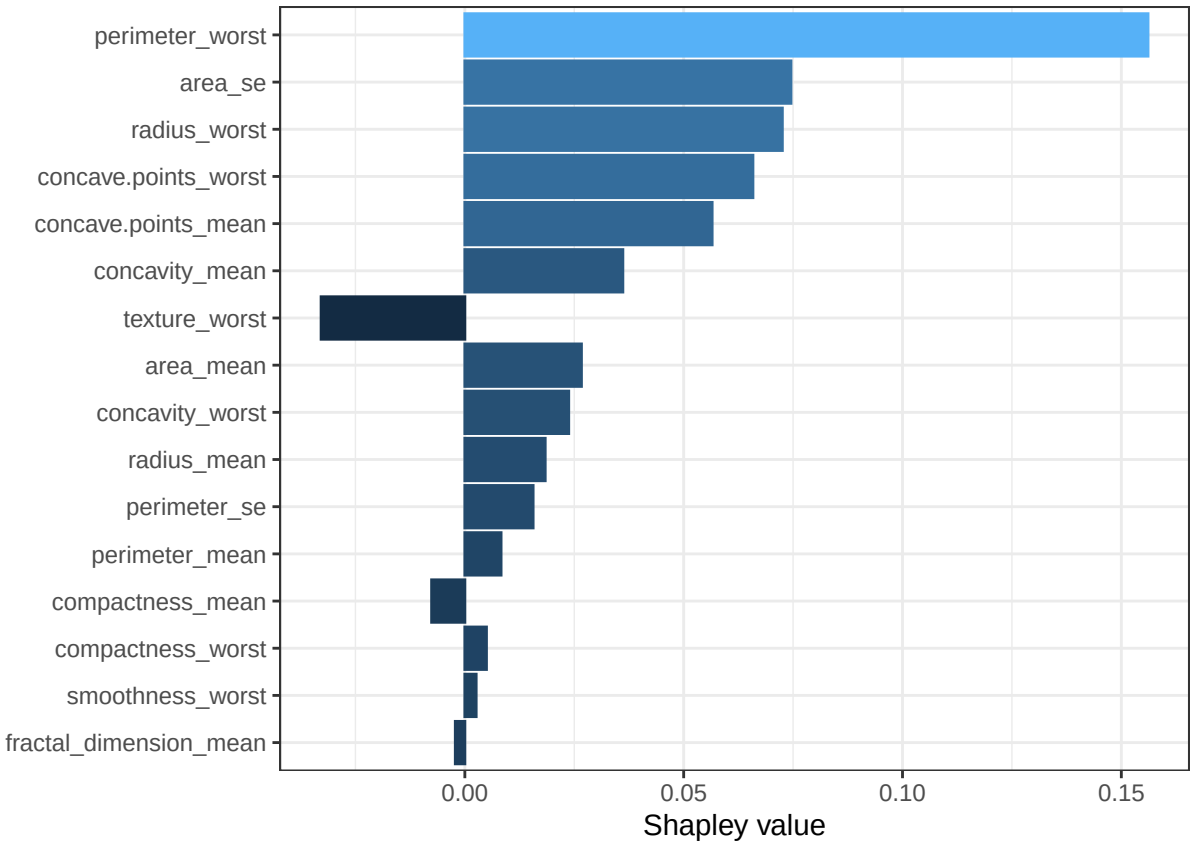
```
dataset_for_shap <- select(dataset_scaled, all_of(append(reduced_features,'diagnosis')))
dataset_for_shap$diagnosis <- if_else(dataset_for_shap$diagnosis == TRUE,1,0)
dataset_scaled_split <- initial_split(dataset_for_shap, p = 0.7, strata = diagnosis)
train <- training(dataset_scaled_split)
train <- select(train , all_of(c(reduced_features,'diagnosis')))
test <- testing(dataset_scaled_split)
test <- select(test , all_of(c(reduced_features,'diagnosis')))
rf_for_shap <- ranger(reduced_formula, data=train, mtry=4,
                      num.trees=200, importance='permutation')
```

```
##    user  system elapsed
##    6.82    0.80    3.34
```




```
## [1] "[test set, row 6] -> diagnosis : "
```

```
## [1] 0
```




```
## [1] "[test set, row 7] -> diagnosis : "
```

```
## [1] 1
```

Les variables explicatives sont classées dans l'odre décroissant d'importance. Dans l'exemple de la ligne 7 de l'ensemble de test ci-dessus, dont le diagnostic est positif, la variable `perimeter_worst` est la plus importante, et `fractal_dimension_mean` la moins importante. Le graphiqe met en lumière également la correlation positive ou négative avec la variable cible `diagnosis`. Par exemple, `concave.points_mean` et `area_se` sont fortement positivement corrélées à la variable `diagnosis`. Donc plus elles prennent de grandes valeurs, plus il est probable que le diagnostic soit positif.