Web page Phishing Detection

Projet Machine Learning - Master D3S

Thomas Fernandes, Vanessa Kenniche, Yassine Ouerghi, Mario Miron Ramos

2023-11-09

Le phishing est une technique d'escroquerie en ligne où des individus malveillants se font passer pour des entités de confiance afin de tromper les utilisateurs et de leur soutirer des informations personnelles sensibles.

La plupart du temps, ce phénomène se concrétise par le biais de faux sites web imitant des sites légitimes, mais ces factices présentent néanmoins des caractéristiques spécifiques pouvant être détectées.

L'étude vise à prédire la légitimité des sites web en utilisant diverses techniques de machine learning. La variable que nous cherchons à prédire est "status", qui indique si un site web est légitime ou potentiellement frauduleux (phishing).

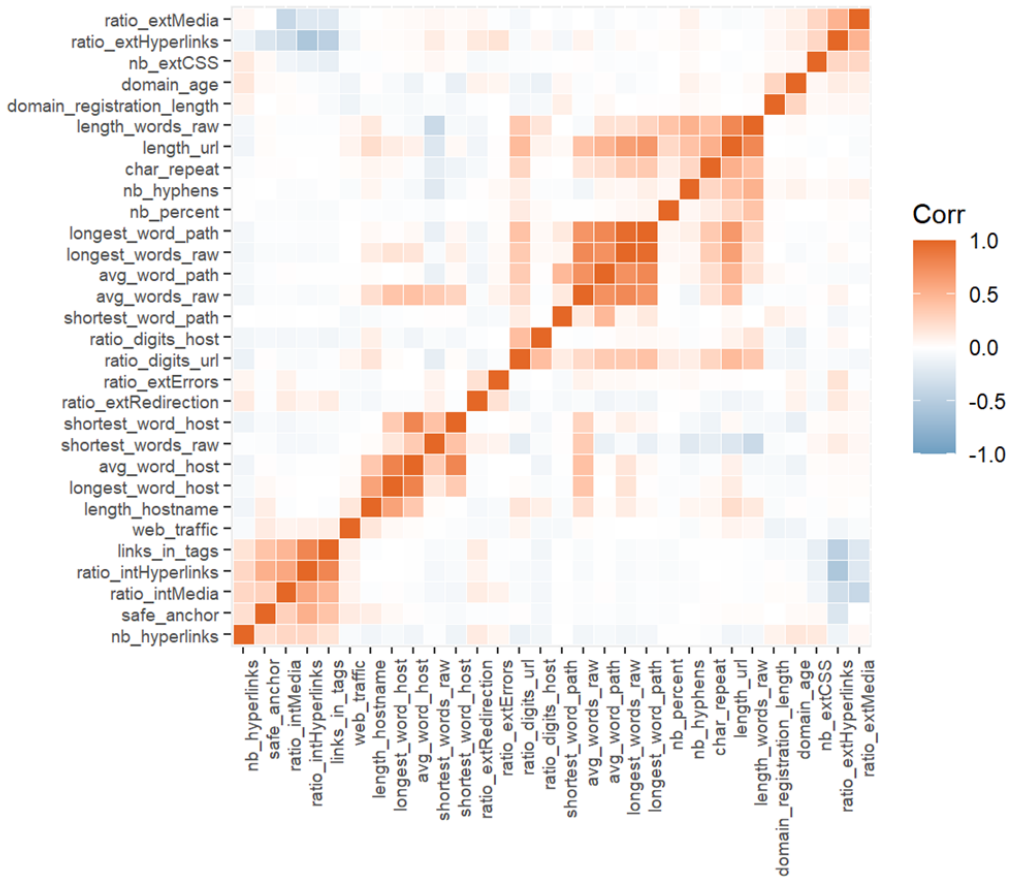
Pour ce faire, nous disposons d'un ensemble de données équilibré de 87 variables explicatives différentes, chacune fournissant des informations sur divers aspects de 11430 sites web différents. Ces données incluent 56 variables basées sur la structure, 24 extraites du contenu des pages web correspondantes, 7 obtenues par des requêtes auprès de services externes.

1. Présentation des données

1.1. Corrélation entre les variables quantitatives

Avant de commencer les différentes modélisations, regardons comment se structurent nos données.

Premièrement, analysons les corrélations entre les variables quantitatives à l'aide d'une carte de chaleur.

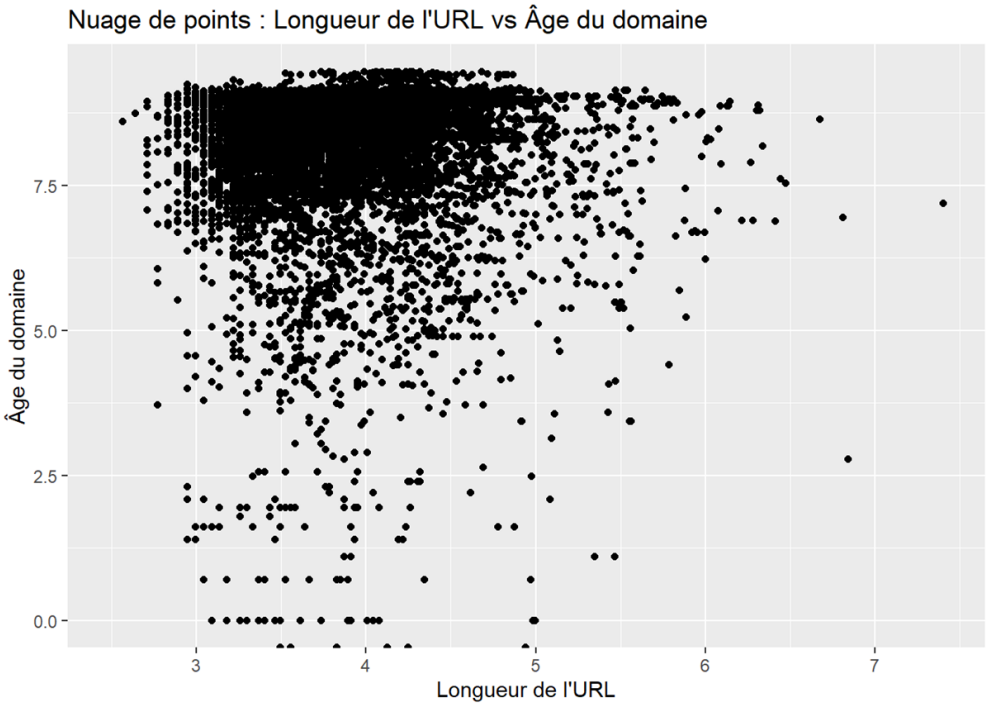


Comme anticipé, plusieurs variables présentent des corrélations entre elles. La carte de chaleur se présente comme suit : les zones rouges indiquent une corrélation positive et les zones bleues indiquent une corrélation négative.

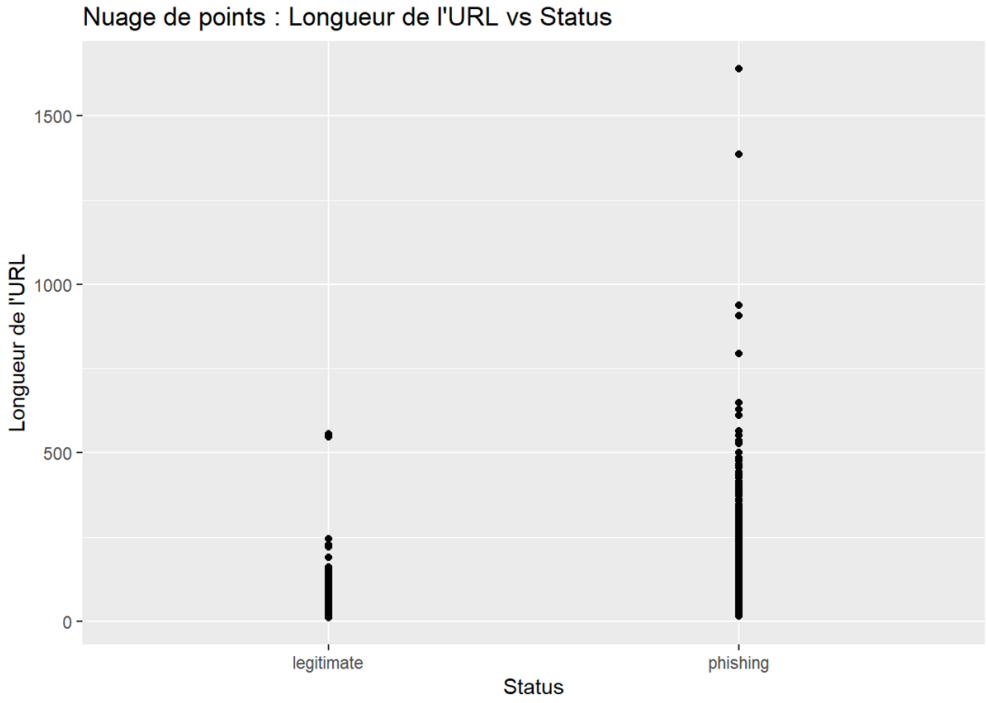
Ainsi, les variables qui montrent des cases rouges foncées entre elles, telles que nb_underscore et nb_hyphens, ont une forte corrélation positive, suggérant dans cet exemple que les URL de phishing ont tendance à utiliser à la fois des tirets bas et des tirets dans leur structure pour imiter les URL légitimes.

Par contraste, une corrélation négative est observable entre des variables comme domain_age et nb_extHyperlinks, ce qui pourrait indiquer que les sites de phishing plus récents ont tendance à intégrer davantage de liens externes dans le but de se présenter comme plus crédibles.

Regardons de plus près des variables arbitraires afin d'analyser leur lien dans le détail. Prenons "domain_age" et "length_url".

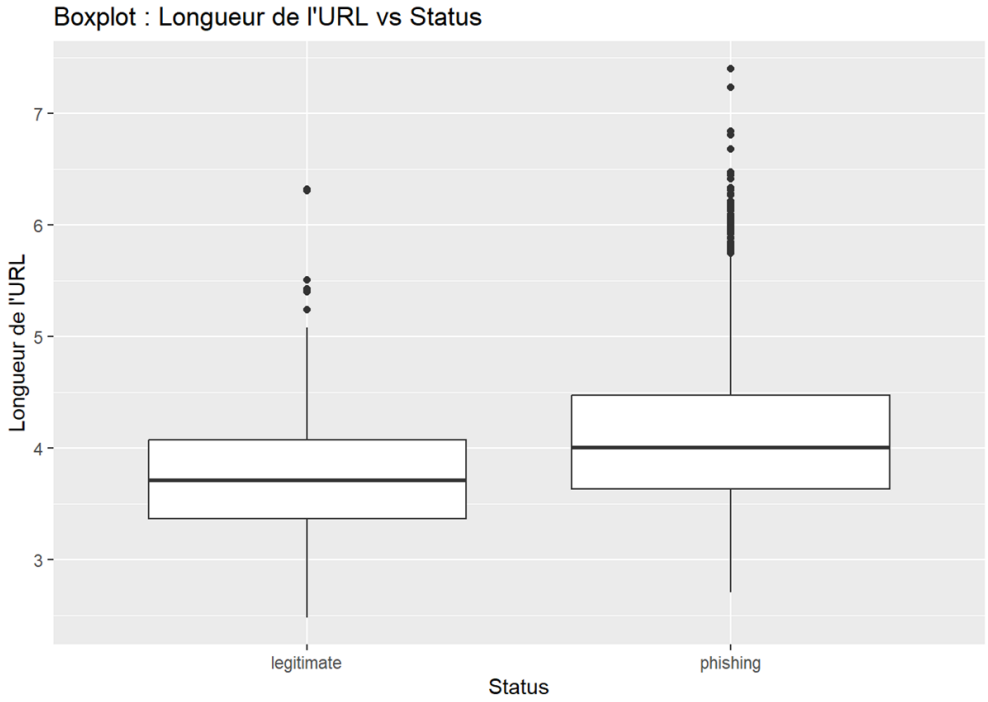


Nous avons réalisé un nuage de point de l'âge du domaine en fonction de la longueur de l'URL. Il y a une concentration dense de points autour d'une certaine valeur sur l'axe des x, ce qui indique que beaucoup de domaines ont une longueur d'URL similaire. Restons sur la longueur de l'URL, et regardons son influence sur la légitimité d'un site. Pour cela, nous avons réalisé un nuage de point dans un premier temps.



De manière générale, le plot nous indique que plus un URL est long, plus un site peut être frauduleux. En particulier lorsque la longueur de l'URL semble dépasser la valeur des 250 caractères.

Dans un second temps, nous avons généré un boxplot :



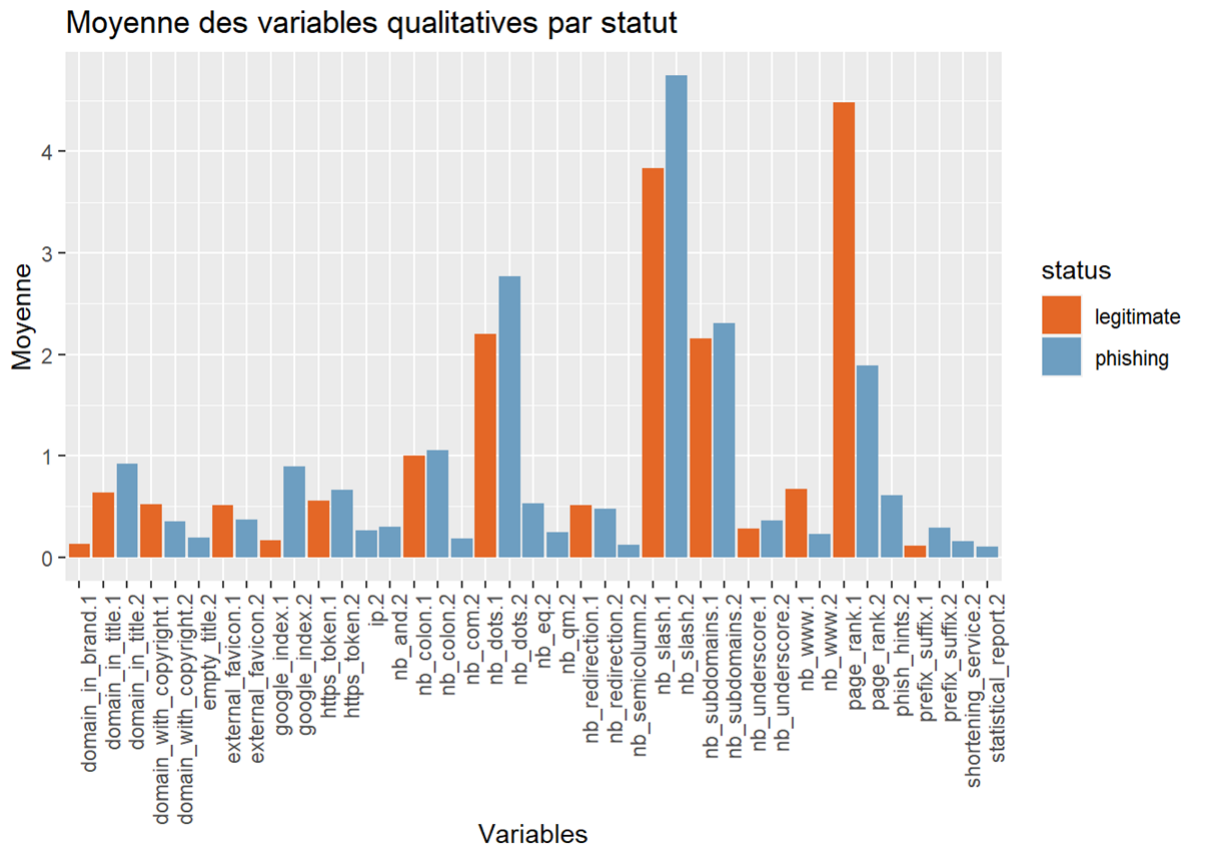
Comme sur le nuage de points, les informations sont cohérentes : la médiane de la valeur de la longueur de l'URL est supérieure pour les sites de phishing que pour les sites légitimes. Par ailleurs, pour les URL légitimes, la boîte est relativement petite, ce qui suggère que la majorité des URL légitimes ont une longueur assez uniforme et moins de valeurs extrêmes.

Les URL de phishing, en revanche, ont une boîte plus grande et un plus grand nombre de valeurs extrêmes (indiquées par les points au-dessus de la "moustache" supérieure), ce qui indique une variabilité plus importante dans la longueur des URL de phishing.

1.2. Moyenne par statut

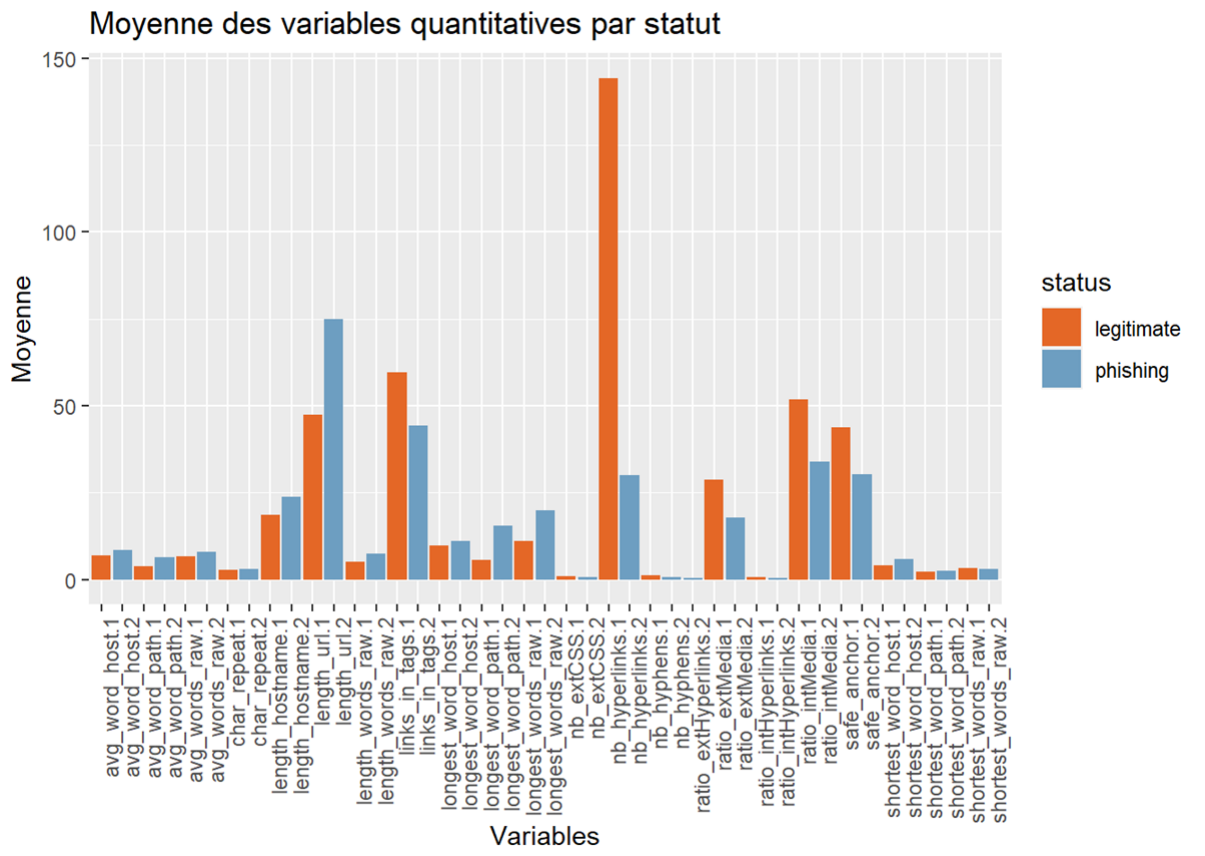
La moyenne par statut des variables permet d'opposer la moyenne d'apparition des variables selon si le statut du site est légitime ou frauduleux.

Premièrement les moyennes des variables qualitatives :



Certaines variables, telles que "nb_dots", "nb_slash", "nb_subdomain" ou encore "page_rank" ont des moyennes d'apparition significativement supérieures, que ce soit dans le cadre légitime ou du phishing. Cela peut être un indicateur de variables le plus souvent utilisées sur les sites web en général. Cependant, certaines variables ont une moyenne d'apparition tombant sous le phishing supérieure à la moyenne des sites légitimes. C'est le cas de "nb_slash". Cela signifierait que plus un site web possède de slashs, plus il est probable que c'est un site frauduleux. À l'inverse, pour la variable "page_rank", c'est la moyenne d'apparition dans les sites légitimes qui est significativement supérieure. Ce résultat semble cohérent étant donné que le rang d'une page est lié à son apparition dans les systèmes de recherches tels que Google.

Deuxièmement les moyennes des variables quantitatives :



En ce qui concerne les moyennes des variables quantitatives, celles qui ressortent le plus sont “length_url”, dont on observe une moyenne supérieure lorsque c’est du phishing, à l’inverse de “links_in_tags”, “ratio_in_media” et “safe_anchor”, et en particulier avec une écrasante significativité “nb_hyperlinks”, dont la moyenne de valeurs supérieure de ces variables semble indiquer la légitimité des site web.

2. Implémentation des méthodes de machine Learning

Nous avons divisé notre jeu de données en un échantillon d’entraînement et un échantillon de test. Nous utiliserons ces deux partitions pour tous nos modèles, de sorte à ce qu’on puisse les comparer dans les meilleures conditions. L’échantillon d’entraînement comporte 75% des données et l’échantillon d’évaluation 25%. Ce choix de faire 75-25 repose sur le fait qu’avec un échantillon d’entraînement suffisamment large, nos modèles auront de meilleures chances de se généraliser tout en laissant un échantillon de test suffisamment large pour éviter le surapprentissage et bien vérifier que la validation de notre modèle sur cet échantillon est robuste.

Concernant les paramètres de l’étude, nous avons décidé d’uniformiser la variable de contrôle pour tous nos modèles, optant pour une méthode de validation-croisée avec un nombre de divisions égal à 5. Le choix du nombre de divisions repose principalement sur des soucis de performance matériels. Un nombre de divisions plus élevé auraient allongé excessivement le temps d’entraînement pour certains modèles. 5 nous semble être l’équilibre idéal entre efficacités et contraintes matérielles.

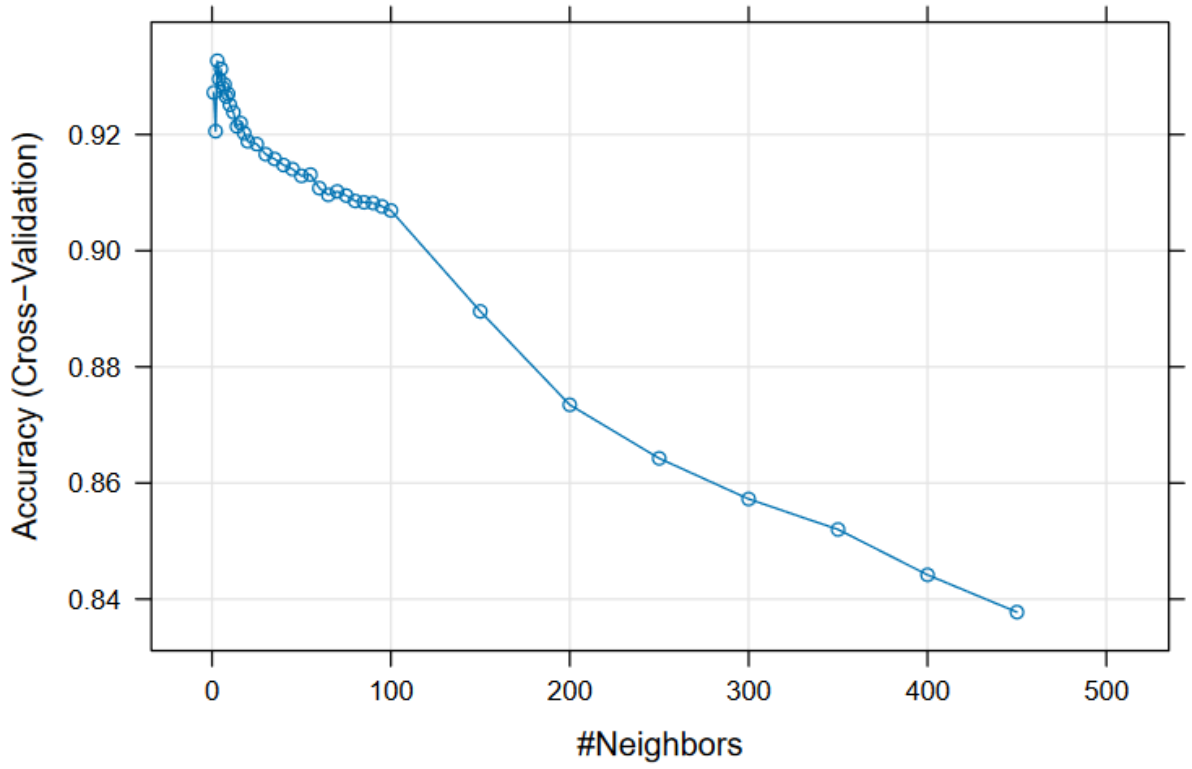
Nous avons utilisé plusieurs modèles de machine learning pour prédire la légitimité des sites web. Ces modèles incluent KNN (K-Nearest Neighbors), Régression Logistique, Naive Bayes (NB), Analyse Discriminante Linéaire (LDA), Analyse Discriminante Quadratique (QDA), ainsi que des Support Vecteurs Machines (SVM) avec séparateurs linéaire et quadratique.

2.1. KNN

Pour les KNN, nous avons commencé par créer une séquence de k allant de 1 à 500 avec un pas qui diminue au fur et à mesure que les k augmentent (Pas de 1 jusqu'à 10, pas de 2 jusqu'à 20, pas de 5 jusqu'à 100 puis pas de 50 jusqu'à 500). L'objectif de cette séquence est de tester un maximum de K pour un maximum d'ordre de grandeur.

Le modèle KNN a démontré une précision d'environ 93.27% sur les données d'entrainement, indiquant qu'il a correctement classé 93.27% des URLs en tant que légitimes ou de phishing. La matrice de confusion révèle que le modèle a identifié correctement 1369 URLs légitimes (vrais positifs) et 1341 URLs de phishing (vrais négatifs). Cependant, il a mal classé 59 URLs légitimes comme phishing (faux négatifs) et 87 URLs de phishing comme légitimes (faux positifs). Le taux d'erreur, représentant la proportion des prédictions incorrectes, est donc de 6.73%.

	Legitimate	Phishing
Legitimate	1369	87
Phishing	59	1341



Le graphique présenté illustre la relation entre la précision de la validation croisée et le nombre de voisins K. La première observation visible est que la précision semble assez stable pour des petits k, mais chute ensuite au fur et à mesure que les k augmentent (Toute mesure gardée puisque la chute n'est que d'une dizaine de pourcent). Cela suggère qu'un nombre plus réduit de voisins aide le modèle à mieux capturer les nuances des données sans tomber dans le surajustement, où le modèle est trop spécifique aux données d'entraînement et ne généralise pas bien aux nouvelles données.

La précision la plus élevée est atteinte pour k = 3, suggérant que le modèle classifie les données avec le plus de précision lorsqu'il considère les 3 plus proches voisins.

Il est également important de noter les écarts-types des précisions et des scores Kappa sont assez petits et stable, laissant supposer que le modèle est fiable.

k	Accuracy	Kappa	AccuracySD	KappaSD
1	0.9272210	0.8544423	0.007037996	0.014075344
2	0.9205740	0.8411486	0.008217546	0.016433645
3	0.9327036	0.8654076	0.004973536	0.009946071
4	0.9295545	0.8591097	0.005333699	0.010665375
5	0.9313040	0.8626088	0.004796734	0.009591570
6	0.9280380	0.8560767	0.006182806	0.012363033

Sur les données de test, on obtiens une précision 94.89%, un peu supérieur à celle sur les données d'entraînement.La sensibilité du modèle passe à 95.87% et la spécificité à 93.91%. Avec en plus un taux d'erreur global très faible, notre modèle avec k = 3 semble être bien spécifié.

2.2. Régression logistique

Pour la régression logistique, nous avons décidé de tester deux cas : avec et sans sélection de variables selon le critère de l'AIC. Pour les deux, nous obtenons un score similaire. Nous n'avons donc pas gardé le modèle avec AIC.

Les résultats obtenus sur les données d'entraînement sont bons, avec une précision de 94.62%. Les écart-types montrent également que le modèle est stable et fiable.

Accuracy	Kappa	AccuracySD	KappaSD
0.9462332	0.8924669	0.003402219	0.006804705

Parmi les 88 variables analysées, quatre se distinguent nettement : `google_index`, `page_rank`, `nb_www`, et `phish_hints`. Ces variables semblent avoir une influence plus marquée dans la détermination de la légitimité des sites web. Ce tableau nous sera utile par la suite pour déterminer les variables à utiliser dans d'autres modèles.

Variable	Importance
google_index	100.00
page_rank	81.32
nb_www	56.29
phish_hints	52.56
domain_age	34.17
nb_hyperlinks	32.68
shortening_service	32.05
longest_words_raw	29.62
ratio_digits_host	26.20
length_hostname	25.31
nb_hyphens	24.42
domain_in_title	23.75
nb_underscore	23.70
https_token	22.65
domain_registration_length	22.34

Sur les données d'entraînement, la régression logistique obtient un score de 94.64%. La matrice de confusion montre que sur les données de test, il a correctement identifié 1351 URL légitimes (vrais positifs) et 1352

URL de phishing (vrais négatifs). De plus, les autres métriques sont également solides : le coefficient Kappa de 0.8929 indique un bon niveau de concordance entre les prédictions du modèle et les données réelles.

Prediction	Legitimate	Phishing
Legitimate	1351	76
Phishing	77	1352

Pour le classificateur Bayésien naïf, l'analyse discriminante linéaire et quadratique, ainsi que les machines à vecteurs de support (SVM), nos ressources matérielles étaient insuffisantes pour estimer nos modèles en utilisant l'ensemble des variables de notre jeu de données.Par conséquent, nous avons effectué une sélection de variables en nous basant sur les critères d'importance fournis par le modèle de régression logistique. Nous avons classé les variables par ordre d'importance, puis avons exécuté chaque modèle en utilisant les variables de 1 à i, où i représente la dernière variable pour laquelle la complexité n'a pas limité les performances de l'ordinateur. De cette manière, nous avons testé toutes les combinaisons de variables en augmentant d'une variable à chaque itération jusqu'à atteindre i. Enfin, nous avons retenu le modèle pour lequel la précision était la plus élevée.

2.3. Bayésien naïf

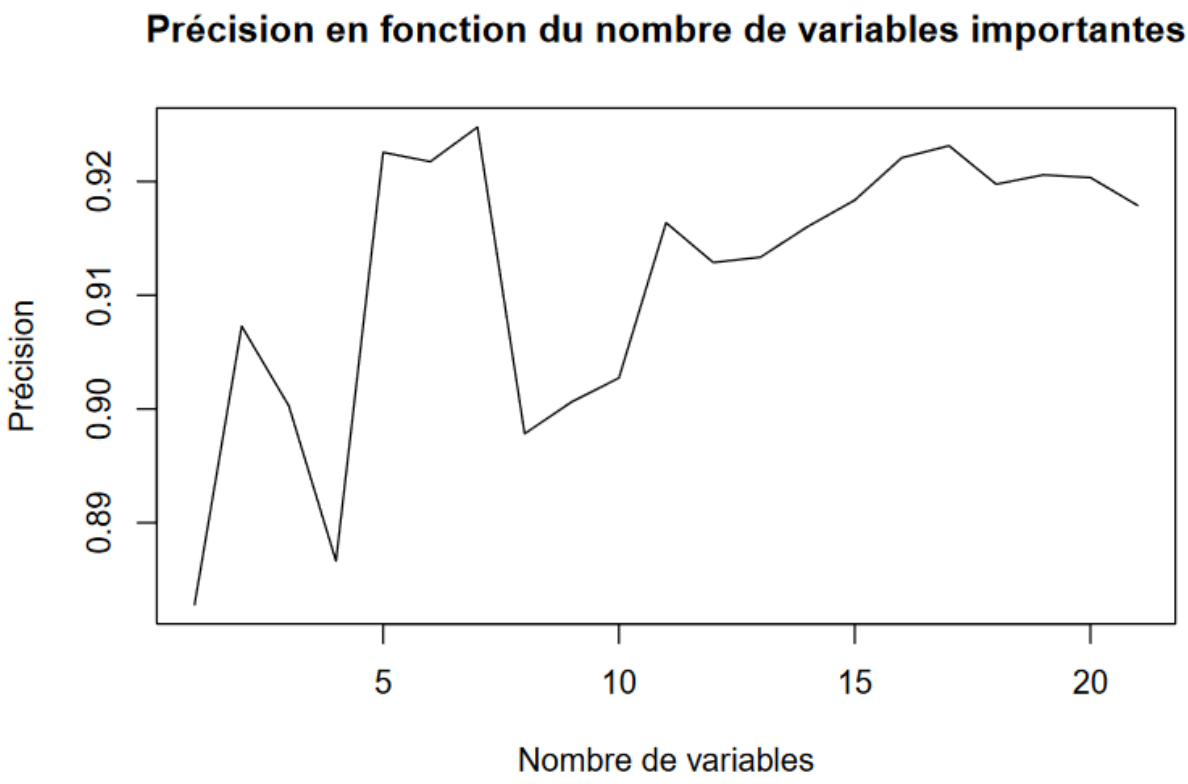
Les résultats de la classification Bayésien naïf sur les données d'entraînement montrent une précision de 92.34%, indiquant que le modèle Bayésien naïf a correctement classé 92.34% des URL en tant que légitimes ou de phishing dans l'ensemble d'entraînement. De plus, les écart-types des précisions et des scores Kappa sont assez petits, suggérant que le modèle est fiable et que ses performances ne varient pas de manière significative lorsqu'il est testé sur des sous-ensembles différents des données d'entraînement.

Accuracy	Kappa	AccuracySD	KappaSD
0.9233730	0.8467476	0.004991672	0.009980999

Sur les données de test, le modèle obtiens une précision de 93.3%. En termes de sensibilité, le modèle a obtenu un score de 95.80%, indiquant qu'il a réussi à détecter la grande majorité des URL légitimes présentes dans les données d'entraînement. La spécificité est de 90.27%, montrant que le modèle a également bien réussi à distinguer les URL de phishing des URL légitimes.

	Prédiction Legitimate	Prédiction Phishing
Legitimate	1368	139
Phishing	60	1289

On remarque que notre modèle a retenu les 7 variables les plus importantes. On ne distingue cependant pas de baisse flagrante de la précision au fur et à mesure que le nombre de variables ajoutées augmente. Ce qui peut laisser penser qu'on aurait potentiellement pu atteindre une meilleure précision avec un nombre de variables plus grand.



2.4. Analyse Discriminante Linéaire (LDA)

Pour la classification basée sur l'Analyse Discriminante Linéaire (LDA), les résultats sur les données d'entraînement montrent une précision de 92.72% et un coefficient Kappa de 0.8545. Cela signifie que le modèle LDA a correctement classé 92.72% des URL en tant que légitimes ou de phishing dans l'ensemble d'entraînement.

Accuracy	Kappa	AccuracySD	KappaSD
0.927239	0.8544778	0.01242653	0.02485308

Lorsque nous évaluons ce modèle sur les données de test, il maintient de bonnes performances avec une précision de 93.28%. La matrice de confusion indique qu'il a correctement identifié 1322 URL légitimes (vrais positifs) et 1342 URL de phishing (vrais négatifs).

	Légitime	Phishing
Légitime	1322	86
Phishing	106	1342

Pour la LDA, nous nous sommes limités à supprimer les variables qui avaient une importance de 0 ou presque.

2.5. Analyse Discriminante Quadratique (QDA)

Les résultats de la classification basée sur l'Analyse Discriminante Quadratique (QDA) sur les données d'entraînement montrent que le modèle QDA a une précision de 80.87% et un coefficient Kappa de 61.74%. Comparé aux autres modèles que vous avez évalués précédemment, ces performances sont bien inférieures.

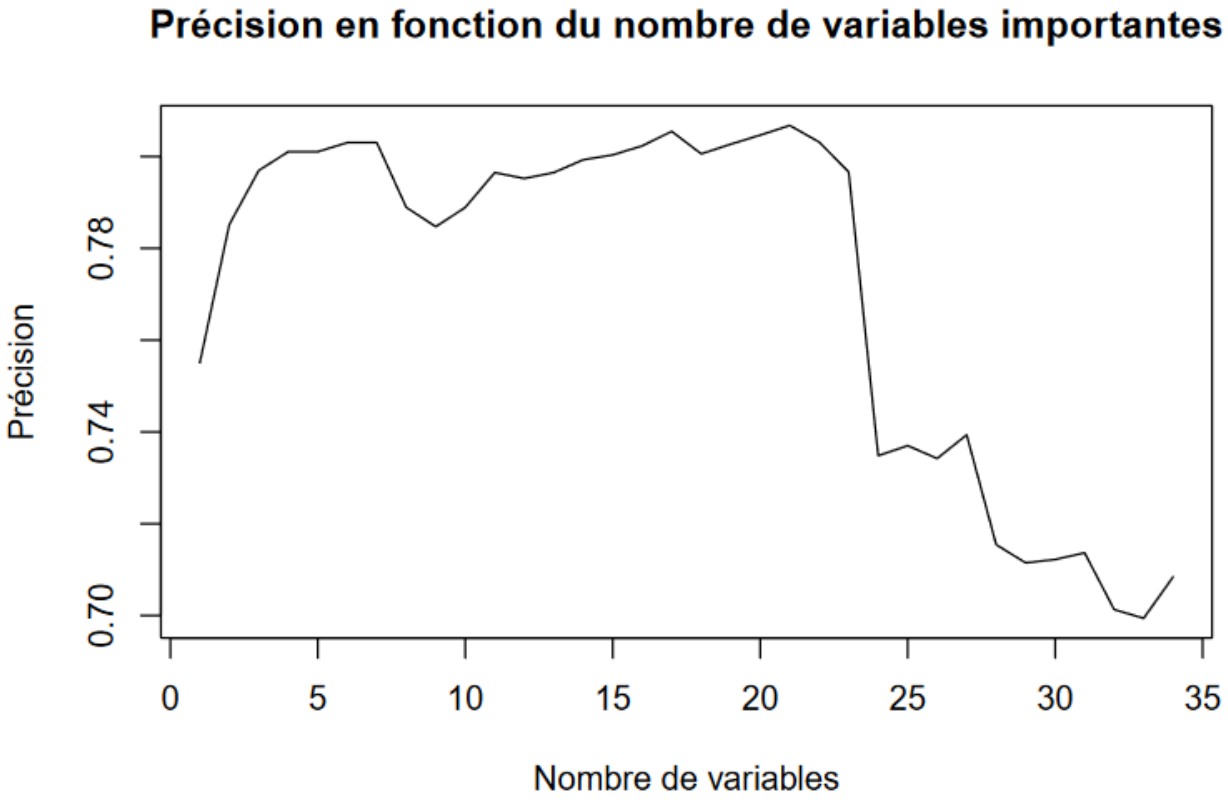
Accuracy	Kappa	AccuracySD	KappaSD
0.8087182	0.6174179	0.002107387	0.004196258

Lorsque nous évaluons le modèle QDA sur les données de test, il maintient un score de précision de 80.67%, ce qui est également inférieur aux autres modèles précédemment examinés. La sensibilité est de 96.92%, ce qui indique qu'il a réussi à détecter efficacement les URL légitimes. Cependant, la spécificité est de seulement 64.43%, montrant que le modèle a eu plus de difficulté à distinguer les URL de phishing des URL légitimes.

	Légitime	Phishing
Légitime	1384	508
Phishing	44	920

Il est à noter que le modèle pourrait être amélioré facilement en ajoutant des variables. Sur l'ensemble des variables de notre jeu de données et sans validation croisée, on obtient un score de 90.27%. Nous avons cependant décidé de garder la validation croisée pour des raisons d'uniformisations des paramètres de nos modèles. Pour les QDA, nous nous limitons ainsi au matériel.

Il est important de noter que notre modèle a retenu les 21 premières variables sur les 35 testées.



2.6. Support Vecteur Machine (SVM) - Linéaire

Pour le modèle de Support Vector Machine Linéaire (SVM linéaire), les résultats sur les données d'entraînement montrent que lorsque le meilleur paramètre C, fixé à 1e-04, le modèle atteint une précision de 91.19%.

C	Accuracy	Kappa	AccuracySD	KappaSD
1e-04	0.9119456	0.8238913	0.008085922	0.01617184

Lorsque nous évaluons le modèle sur les données de test, il maintient un score élevé de précision de 91.56%. La matrice de confusion indique qu'il a correctement identifié 1307 URL légitimes (vrais positifs) et 1308 URL de phishing (vrais négatifs). Le modèle s'est correctement généralisé.

	Legitimate	Phishing
Legitimate	1307	120
Phishing	121	1308

2.7. Support Vecteur Machine (SVM) - Quadratique avec noyau radial

Pour le modèle SVM Quadratique avec un noyau radial, les résultats montrent que lorsque les paramètres C = 0.0031 et Sigma = 0.03125 sont utilisés, le modèle atteint une précision de 91.68%.

C	Sigma	Accuracy	Kappa	AccuracySD	KappaSD
0.0031	0.03125	0.9168407	0.8336816	0.006671519	0.013343775

Lorsque nous évaluons le modèle avec ces paramètres sur les données de test, il maintient un score élevé de précision de 91.95%. La sensibilité (taux de vrais positifs) du modèle SVM Quadratique avec noyau radial sur les données de test est de 90.41%, ce qui signifie qu'il a réussi à détecter efficacement les URL légitimes.

Prediction	Legitimate	Phishing
Legitimate	1291	93
Phishing	137	1335

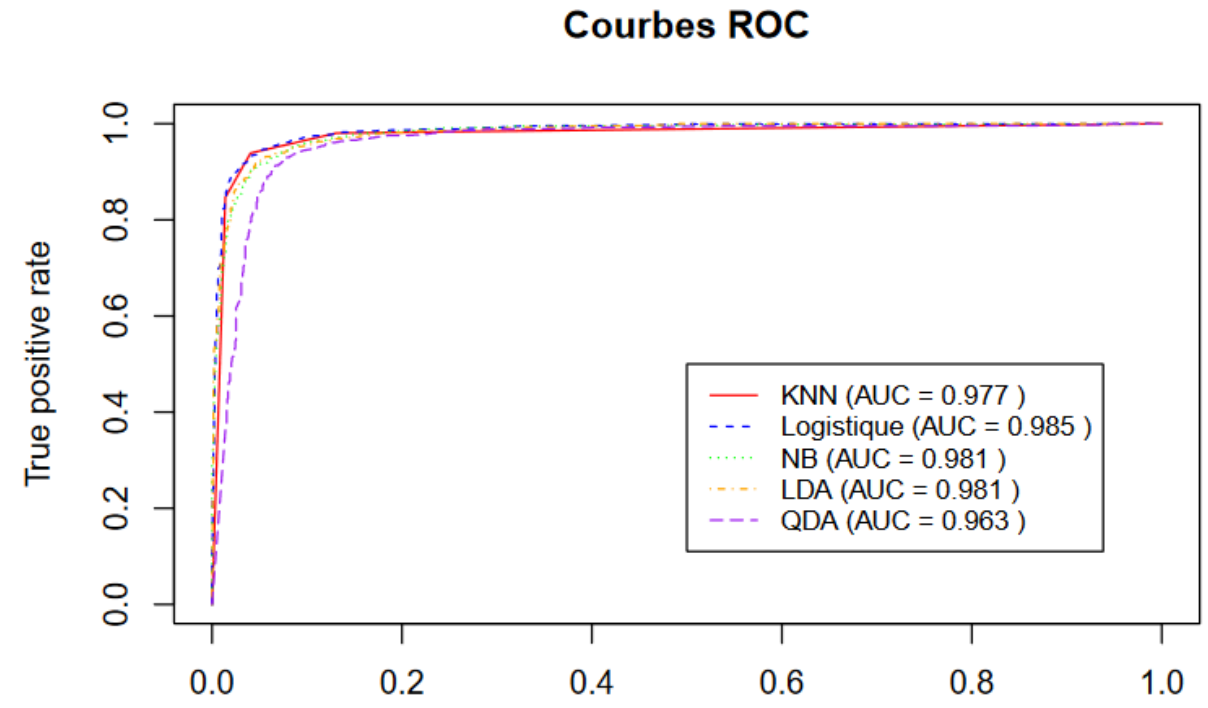
3. Comparaison des méthodes

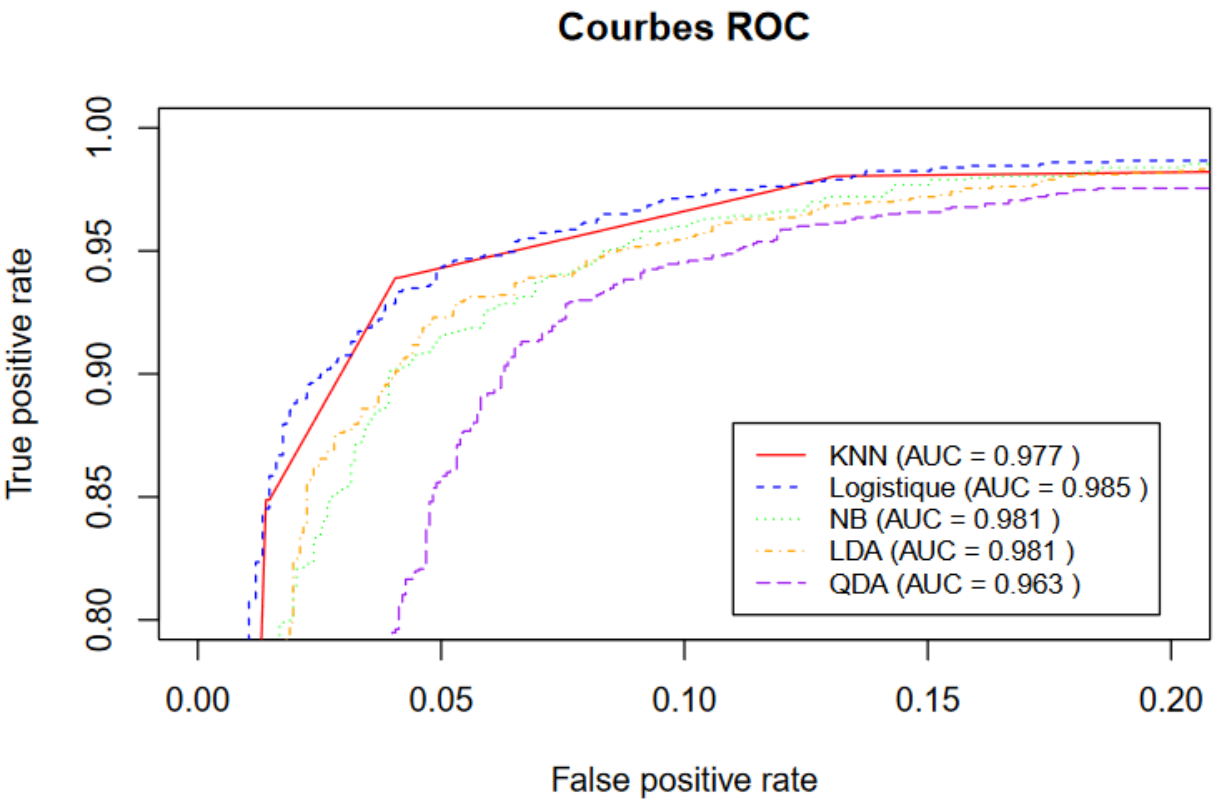
Modèle	Précision	Sensibilité	Spécificité	AUC
KNN	0.9489	0.9587	0.9391	0.9769
Logistique	0.9464	0.9461	0.9468	0.9853
NB	0.9303	0.9580	0.9027	0.9806
LDA	0.9328	0.9258	0.9398	0.9814
QDA	0.8067	0.9692	0.6443	0.9631
SVM Linéaire	0.9156	0.9153	0.9160	NA
SVM Quadratique	0.9195	0.9041	0.9349	NA

L'analyse des performances de différents modèles révèle des nuances significatives dans leurs capacités respectives. En particulier le modèle QDA, qui se distingue par des performances inférieures comparées aux autres modèles. Sa faible performance peut être principalement attribuée à sa tendance à classifier de manière incorrecte un nombre élevé de sites légitimes comme étant des sites d'hameçonnage, résultant ainsi en une spécificité réduite.

En ce qui concerne la précision, le modèle des KNN se positionne en tête, suivi de près par le modèle de régression logistique. À l'opposé, les modèles basés sur les SVM semblent rencontrer plus de difficultés.

Pour une comparaison plus robuste, nous pouvons tracer les courbes ROC et comparer les AUC.





La meilleure courbe ROC est celle qui est la plus proche du coin supérieur gauche du graphique, indiquant un taux élevé de vrais positifs tout en maintenant un faible taux de faux positifs. En d'autres termes, un modèle est considéré comme meilleur s'il maximise la sensibilité (taux de vrais positifs) tout en minimisant la probabilité de faux positifs.

Cette fois-ci, les modèles de régression logistique, bayésien naïf, et LDA affichent des performances comparables en termes de courbes ROC. Cependant, il est à noter que le modèle KNN se positionne légèrement en dessous du modèle de régression logistique, qui a la valeur d'AUC la plus élevée parmi nos modèles.

En conclusion, bien que le modèle KNN et le modèle de régression logistique soient presque équivalents en termes de performances globales, le facteur crucial dans la détection de phishing est d'éviter de classer les sites de phishing comme légitimes. Avec sa sensibilité plus élevée, le modèle KNN est légèrement meilleur pour détecter les sites de phishing. Par conséquent, en privilégiant la sensibilité comme critère principal, on peut considérer le modèle KNN comme le meilleur modèle.

Partie Code R

1. Présentation des données

1.1. Corrélation entre les variables quantitatives

```
df_present <- df

#Extraire les variables qualitatives
v_quali <- vector("logical", length = ncol(df_present) - 1)
for (i in 2:ncol(df_present)) {
  v_quali[[i]] <- (length(unique(df_present[[i]])) / sum(!is.na(df_present[[i]]))) < 0.002
}

num_cols <- character()
cat_cols <- character()

for (i in 1:length(v_quali)) {
  if (!v_quali[[i]]) {
    num_cols <- c(num_cols, names(df_present)[i])
  } else {
    cat_cols <- c(cat_cols, names(df_present)[i])
  }
}

corr <- cor(df_present[num_cols])

ggcorrplot(
  corr,
  hc.order = TRUE,
  type = "full",
  outline.color = "white",
  ggtheme = ggplot2::theme_gray,
  colors = c("#6D9EC1", "white", "#E46726"),
  show.diag = TRUE,
  tl.cex = 7,
  tl.srt = 90
)
```

```
ggplot(data = melt(df_present[, num_cols]), aes(x = variable, y = value)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

```
attach(df)

ggplot(df, aes(x = nb_underscore, y = nb_hyphens )) +
  geom_point() +
  labs(x = "Nombre tiret du bas", y = "Nombre de tirets") +
  ggtitle("Nuage de points : Nombre tiret du bas vs Nombre de tirets")
```

```
ggplot(df, aes(x = domain_age, y = ratio_extHyperlinks)) +
  geom_point() +
  labs(x = "Âge du domaine", y = "Nombre de liens externes") +
  ggtitle("Nuage de points : Âge du domaines vs Nombre de liens externes")
```

```r
max(domain_age)
```

```
## [1] 12874
```

```r
min(domain_age)
```

```
## [1] -12
```

```r
# Nuage de point y = longueur url, x = status
ggplot(df, aes(x = status, y = length_url)) +
  geom_point() +
  labs(x = "Status", y = "Longueur de l'URL") +
  ggtitle("Nuage de points : Longueur de l'URL vs Status")
```

```r
# Boxplot
ggplot(df, aes(x = status, y = log(length_url))) +
  geom_boxplot() +
  labs(x = "Status", y = "Longueur de l'URL") +
  ggtitle("Boxplot : Longueur de l'URL vs Status")
```

1.2. Moyenne par statut

```r
mean_by_status <- function(df, col_name) {
  df %>%
    group_by(status) %>%
    summarise(mean_value = mean(.data[[col_name]], na.rm = TRUE))
}
mean_values_list_cat <- list()

for (col in cat_cols) {
  mean_values_list_cat[[col]] <- mean_by_status(df_present, col)
}

mean_values_df_cat <- do.call(rbind, mean_values_list_cat)
mean_values_df_cat$col_names <- rownames(mean_values_df_cat)

mean_values_df_cat <- mean_values_df_cat[mean_values_df_cat$mean_value > 0.1 | mean_values_df_cat$mean_
mean_values_df_cat <- mean_values_df_cat[!is.na(mean_values_df_cat$mean_value), ]

ggplot(mean_values_df_cat, aes(x = col_names, y = mean_value, fill = status)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Variables", y = "Moyenne", title = "Moyenne des variables qualitatives par statut") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  scale_fill_manual(values = c("#E46726", "#6D9EC1"))
```

2. Modélisation

2.1. Préparation du jeu de données

```r
set.seed(123)
indxTrain <- createDataPartition(df$status, p = 0.75, list = FALSE)
DTrain <- df[indxTrain, ]
DTest <- df[-indxTrain, ]
```

2.2. Variable de controle

```
ctrl <- trainControl(method = "cv", number = 5)
```

2.3. Entrainement

KNN

```
set.seed(123)

k <- c(1:10, seq(12, 19, by = 2), seq(20, 99, by = 5), seq(100, 500, by = 50))

# fit.knn.cv <- train(status ~ .,
#                     data = DTrain,
#                     method = "knn",
#                     trControl = ctrl,
#                     tuneGrid = expand.grid(k = k),
#                     preProcess = c("center", "scale"),
#                     na.action = na.omit)
load("C:/Users/thoma/Desktop/Github/Web-page-Phishing-Detection/fit.knn.cv.RDATA")
plot(fit.knn.cv)
```

```
bestK <- fit.knn.cv$bestTune$k
#matrice de confusion
predictions <- predict(fit.knn.cv, newdata = DTest)
confusionMatrix(predictions, DTest$status)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   legitimate phishing
##   legitimate       1369       87
##   phishing           59     1341
##
##                Accuracy : 0.9489
##                  95% CI : (0.9402, 0.9567)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8978
##
##  Mcnemar's Test P-Value : 0.02545
##
##             Sensitivity : 0.9587
##             Specificity : 0.9391
##          Pos Pred Value : 0.9402
##          Neg Pred Value : 0.9579
##              Prevalence : 0.5000
##          Detection Rate : 0.4793
##    Detection Prevalence : 0.5098
##       Balanced Accuracy : 0.9489
##
##        'Positive' Class : legitimate
##
```

Régression logistique

```
#Sans AIC
fit.lr <- train(status ~ .,
                data = DTrain,
                method = "glm",
                trControl = ctrl,
                na.action = na.omit)
print(varImp(fit.lr))
```

```
## glm variable importance
##
##   only 20 most important variables shown (out of 81)
##
##                              Overall
## google_index                  100.00
## page_rank                      81.32
## nb_www                         56.29
## phish_hints                    52.56
## domain_age                     34.17
## nb_hyperlinks                  32.68
## shortening_service             32.05
## longest_words_raw              29.62
## ratio_digits_host              26.20
## length_hostname                25.31
## nb_hyphens                     24.42
## domain_in_title                23.75
## nb_underscore                  23.70
## https_token                    22.65
## domain_registration_length     22.34
## ratio_extHyperlinks            21.37
## ratio_extRedirection           20.95
## ratio_extMedia                 19.75
## avg_words_raw                  19.56
## nb_space                       19.21
```

```
#Avec AIC
# fit.lr.aic <- train(status ~ .,
#                     data = DTrain,
#                     method = "glmStepAIC",
#                     trControl = ctrl,
#                     na.action = na.omit)
load("C:/Users/thoma/Desktop/Github/Web-page-Phishing-Detection/fit.lr.aic.RDATA")
```

```
importance <- varImp(fit.lr)$importance
variable_names <- rownames(importance)
importance_values <- importance$Overall
variable_indexes <- match(variable_names, names(DTrain))
important_vars_df <- data.frame(
  Variable = variable_names,
  Importance = importance_values,
  Index = variable_indexes
)
important_vars_df_sorted <- important_vars_df[order(important_vars_df$Importance, decreasing = TRUE), ]
```

NB

```
# accuracies_nb <- numeric()
# best_accuracy_nb <- 0
# best_combination_nb <- NULL
#
# for (i in 2:25) {
#   variable_names_nb <- names(DTrain)[important_vars_df_sorted$Index[1:i]]
#   fit.nb_temp <- train(status ~ .,
#                        data = DTrain[, c("status", variable_names_nb)],
#                        method = "nb",
#                        trControl = ctrl)
#   model_accuracy_nb <- max(fit.nb_temp$results$Accuracy)
#   accuracies_nb[i - 1] <- model_accuracy_nb
#
#   if (model_accuracy_nb > best_accuracy_nb) {
#     best_accuracy_nb <- model_accuracy_nb
#     best_combination_nb <- variable_names_nb
#   }
# }
#
# fit.nb <- train(status ~ .,
#                 data = DTrain[, c("status", best_combination_nb)],
#                 method = "nb",
#                 trControl = ctrl)
load("C:/Users/thoma/Desktop/Github/Web-page-Phishing-Detection/fit.nb.RDATA")
plot(accuracies_nb, type = "l",
     xlab = "Nombre de variables",
     ylab = "Précision",
     main = "Précision en fonction du nombre de variables importantes")
```

LDA

```
fit.lda = train(status ~ .,
                data = DTrain[, -c(9, 60, 62, 64, 69, 72)],
                method="lda",
                trControl=ctrl)
```

QDA

```
# accuracy_qda <- numeric()
# best_accuracy_qda <- 0
# best_combination_qda <- NULL
#
# for (i in 2:35) {
#   variable_names <- names(DTrain)[important_vars_df_sorted$Index[1:i]]
#
#   fit.qda <- train(status ~ .,
#                    data = DTrain[, c("status", variable_names)],
#                    method = "qda",
#                    trControl = ctrl)
#
#   model_accuracy <- max(fit.qda$results$Accuracy)
```

```
#
#   accuracy_qda[i - 1] <- model_accuracy
#
#   if (model_accuracy > best_accuracy_qda) {
#     best_accuracy_qda <- model_accuracy
#     best_combination_qda <- variable_names
#   }
# }
load("C:/Users/thoma/Desktop/Github/Web-page-Phishing-Detection/fit.qda.RDATA")
plot(accuracies_qda, type = "l",
     xlab = "Nombre de variables",
     ylab = "Précision",
     main = "Précision en fonction du nombre de variables importantes")
```

SVM - Séparateur Linéaire

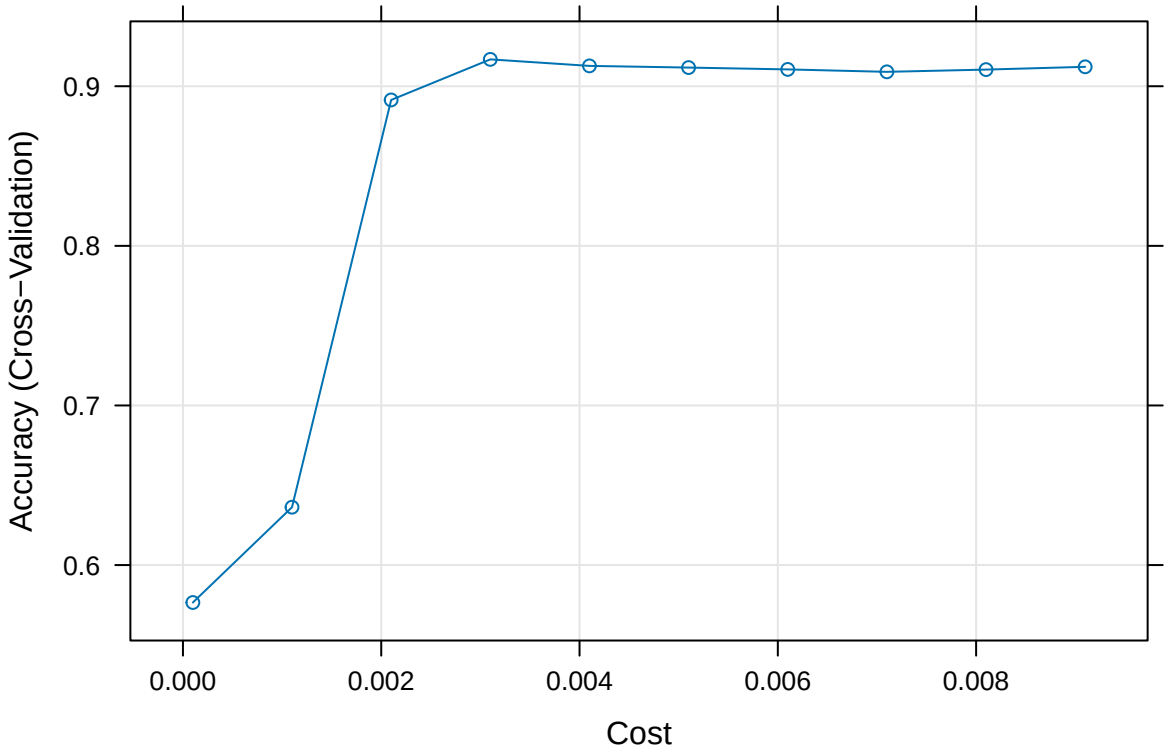
```
# svmGrid_lin <- seq(0.0001, 0.01 ,by = 0.001)
# best_accuracies_svm <- numeric()
# best_accuracy_svm <- 0
# best_combination_svm <- NULL
# best_C_svm <- NULL
#
# for (i in 2:20) {
#   variable_names_svm <- names(DTrain)[important_vars_df_sorted$Index[1:i]]
#   fit.Lin.svm <- train(status ~ .,
#                        data = DTrain[, c("status", variable_names_svm)],
#                        method = "svmLinear",
#                        type = "C-svc",
#                        trControl = ctrl,
#                        tuneGrid = data.frame(.C = svmGrid_lin))
#   max_accuracy <- max(fit.Lin.svm$results$Accuracy)
#   max_C <- fit.Lin.svm$bestTune$.C
#
#   best_accuracies_svm[i - 1] <- max_accuracy
#
#   if (max_accuracy > best_accuracy_svm) {
#     best_accuracy_svm <- max_accuracy
#     best_combination_svm <- variable_names_svm
#     best_C_svm <- max_C
#   }
# }

load("C:/Users/thoma/Desktop/Github/Web-page-Phishing-Detection/fit.Lin.svm.RDATA")
fit.Lin.svm$bestTune
```

```
##       C
## 1 1e-04
```

SVM - Séparateur Quadratique

```
#Noyau Radial
# svmGrid_quad <- expand.grid(.C = seq(0.0001, 0.01, by = 0.001),
#                             .sigma = seq(0.0001, 0.01, by = 0.001))
```

```
# best_accuracies_svm_radial <- numeric()
# best_accuracy_svm_radial <- 0
# best_combination_svm_radial <- NULL
# best_params_svm_radial <- NULL
#
# for (i in 2:20) {
#   variable_names_svm_radial <- names(DTrain)[important_vars_df_sorted$Index[1:i]]
#   fit.Quad.svm <- train(status ~ .,
#                         data = DTrain[, c("status", variable_names_svm_radial)],
#                         method = "svmRadial",
#                         type = "C-svc",
#                         trControl = ctrl_temp,
#                         tuneGrid = svmGrid_quad)
#
#   max_accuracy_radial <- max(fit.Quad.svm$results$Accuracy)
#   max_params_radial <- fit.Quad.svm$bestTune
#
#   best_accuracies_svm_radial[i - 1] <- max_accuracy_radial
#
#   if (max_accuracy_radial > best_accuracy_svm_radial) {
#     best_accuracy_svm_radial <- max_accuracy_radial
#     best_combination_svm_radial <- variable_names_svm_radial
#     best_params_svm_radial <- max_params_radial
#   }
# }
load("C:/Users/thoma/Desktop/Github/Web-page-Phishing-Detection/fit.Quad.svm.RDATA")
plot(fit.Quad.svm)
```




```
fit.Quad.svm$bestTune
```

```
##   sigma      C
## 4   0.1 0.0031
```

Prédictions

KNN

```
set.seed(123)
predictionsBestK <- predict(fit.knn.cv, newdata = DTest)
confusionMatrixBestK <- confusionMatrix(predictionsBestK, DTest$status)
print(confusionMatrixBestK$overall['Accuracy'])
```

```
##  Accuracy
## 0.9488796
```

```
set.seed(123)
predictionsBestK <- predict(fit.knn.cv, newdata = DTest, type = "prob")

pred.knn <- prediction(predictionsBestK[,2], DTest$status)
perf.knn <- performance(pred.knn, "tpr", "fpr")
plot(perf.knn)
```

```r
auc.knn <- performance(pred.knn, "auc")@y.values[[1]]
```

Regression logistique

```r
class.lr <- predict(fit.lr, newdata = DTest)
print(varImp(fit.lr))
```

```
## glm variable importance
##
##   only 20 most important variables shown (out of 81)
##
##                             Overall
## google_index                 100.00
## page_rank                     81.32
## nb_www                        56.29
## phish_hints                   52.56
## domain_age                    34.17
## nb_hyperlinks                 32.68
## shortening_service            32.05
## longest_words_raw             29.62
## ratio_digits_host             26.20
## length_hostname               25.31
## nb_hyphens                    24.42
## domain_in_title               23.75
## nb_underscore                 23.70
## https_token                   22.65
## domain_registration_length    22.34
## ratio_extHyperlinks           21.37
## ratio_extRedirection          20.95
## ratio_extMedia                19.75
## avg_words_raw                 19.56
## nb_space                      19.21
```

```r
class.lr.aic <- predict(fit.lr.aic, newdata = DTest)

confusionMatrixLR <- confusionMatrix(class.lr, DTest$status)
confusionMatrixLRAIC <- confusionMatrix(class.lr.aic, DTest$status)
print(confusionMatrixLR$overall['Accuracy'])
```

```
##  Accuracy
## 0.9464286
```

```r
print(confusionMatrixLRAIC$overall['Accuracy'])
```

```
##  Accuracy
## 0.9457283
```

```r
class.lr <- predict(fit.lr, newdata = DTest, type = "prob")
class.lr.aic <- predict(fit.lr.aic, newdata = DTest, type = "prob")

pred.lr <- prediction(class.lr[,2], DTest$status)
perf.lr <- performance(pred.lr, "tpr", "fpr")
auc.lr <- performance(pred.lr, "auc")@y.values[[1]]
```

NB

```
#Accuracy
class.nb <- predict(fit.nb, newdata = DTest)
confusionMatrixNB <- confusionMatrix(class.nb, DTest$status)
print(confusionMatrixNB$overall['Accuracy'])
```

```
##  Accuracy
## 0.9303221
```

```
pred.nb = predict(fit.nb, newdata=DTest[,c(best_combination_nb)], type = "prob")

pred.nb <- prediction(pred.nb[,2], DTest$status)
perf.nb <- performance(pred.nb, "tpr", "fpr")
auc.nb <- performance(pred.nb, "auc")@y.values[[1]]
```

LDA

```
class.lda <- predict(fit.lda, newdata = DTest)
confusionMatrixLDA <- confusionMatrix(class.lda, DTest$status)
print(confusionMatrixLDA$overall['Accuracy'])
```

```
##  Accuracy
## 0.9327731
```

```
pred.lda = predict(fit.lda,
                   newdata=DTest[,-c(9, 60, 62, 64, 69, 72, 88)],
                   type = "prob")
pred.lda <- prediction(pred.lda[,2], DTest$status)
perf.lda <- performance(pred.lda, "tpr", "fpr")
plot(perf.lda)
```

```
auc.lda <- performance(pred.lda, "auc")@y.values[[1]]
```

QDA

```
class.qda <- predict(fit.qda, newdata = DTest)
confusionMatrixQDA <- confusionMatrix(class.qda, DTest$status)
print(confusionMatrixQDA$overall['Accuracy'])
```

```
##  Accuracy
## 0.8067227
```

```
pred.qda = predict(fit.qda, newdata=DTest[, best_combination_qda], type = "prob")
pred.qda <- prediction(pred.qda[,2], DTest$status)
perf.qda <- performance(pred.qda, "tpr", "fpr")
plot(perf.qda)
```

```
auc.qda <- performance(pred.qda, "auc")@y.values[[1]]
```

SVM - Séparateur Linéaire

```
class.Lin.svm <- predict(fit.Lin.svm, newdata = DTest)
confusionMatrixLinSVM <- confusionMatrix(class.Lin.svm, DTest$status)
print(confusionMatrixLinSVM$overall['Accuracy'])
```

```
##  Accuracy
## 0.9156162
```

SVM - Séparateur Quadratique

```
class.Quad.svm <- predict(fit.Quad.svm, newdata = DTest)
confusionMatrixQuadSVM <- confusionMatrix(class.Quad.svm, DTest$status)
print(confusionMatrixQuadSVM$overall['Accuracy'])
```

Noyau Radial

```
##  Accuracy
## 0.9194678
```

Comparaison des modèles

```
models <- c("KNN", "Logistique", "NB", "LDA", "QDA", "SVM Linéaire", "SVM Quadratique")
accuracy <- c(confusionMatrixBestK$overall['Accuracy'],
              confusionMatrixLR$overall['Accuracy'],
              confusionMatrixNB$overall['Accuracy'],
              confusionMatrixLDA$overall['Accuracy'],
              confusionMatrixQDA$overall['Accuracy'],
              confusionMatrixLinSVM$overall['Accuracy'],
              confusionMatrixQuadSVM$overall['Accuracy'])
sensibilite <- c(confusionMatrixBestK$byClass['Sensitivity'],
                 confusionMatrixLR$byClass['Sensitivity'],
                 confusionMatrixNB$byClass['Sensitivity'],
                 confusionMatrixLDA$byClass['Sensitivity'],
                 confusionMatrixQDA$byClass['Sensitivity'],
                 confusionMatrixLinSVM$byClass['Sensitivity'],
                 confusionMatrixQuadSVM$byClass['Sensitivity'])
specificite <- c(confusionMatrixBestK$byClass['Specificity'],
                 confusionMatrixLR$byClass['Specificity'],
                 confusionMatrixNB$byClass['Specificity'],
                 confusionMatrixLDA$byClass['Specificity'],
                 confusionMatrixQDA$byClass['Specificity'],
                 confusionMatrixLinSVM$byClass['Specificity'],
                 confusionMatrixQuadSVM$byClass['Specificity'])
auc <- c(auc.knn, auc.lr, auc.nb, auc.lda, auc.qda, NA, NA)

df_models <- data.frame(models, accuracy, sensibilite, specificite, auc)
df_models
```

```
##            models  accuracy sensibilite specificite       auc
## 1             KNN 0.9488796   0.9586835   0.9390756 0.9769111
## 2      Logistique 0.9464286   0.9460784   0.9467787 0.9852774
## 3              NB 0.9303221   0.9579832   0.9026611 0.9805765
## 4             LDA 0.9327731   0.9257703   0.9397759 0.9813622
## 5             QDA 0.8067227   0.9691877   0.6442577 0.9631335
## 6    SVM Linéaire 0.9156162   0.9152661   0.9159664        NA
## 7 SVM Quadratique 0.9194678   0.9040616   0.9348739        NA
```

```
legend_labels <- c(
    paste("KNN (AUC =", round(auc.knn, 3), ")"),
```

```
    paste("Logistique (AUC =", round(auc.lr, 3), ")"),
    paste("NB (AUC =", round(auc.nb, 3), ")"),
    paste("LDA (AUC =", round(auc.lda, 3), ")"),
    paste("QDA (AUC =", round(auc.qda, 3), ")")
)

plot(perf.knn, col = "red", main = "Courbes ROC", lty = 1)
plot(perf.lr, col = "blue", add = TRUE, lty = 2)
plot(perf.nb, col = "green", add = TRUE, lty = 3)
plot(perf.lda, col = "orange", add = TRUE, lty = 4)
plot(perf.qda, col = "purple", add = TRUE, lty = 5)

legend(0.5, 0.5,
       legend = legend_labels,
       col = c("red", "blue", "green", "orange", "purple"),
       lty = c(1, 2, 3, 4, 5), cex = 0.8)
```

On fait le même graphique mais en zoom sur la partie gauche.

```
legend_labels <- c(
    paste("KNN (AUC =", round(auc.knn, 3), ")"),
    paste("Logistique (AUC =", round(auc.lr, 3), ")"),
    paste("NB (AUC =", round(auc.nb, 3), ")"),
    paste("LDA (AUC =", round(auc.lda, 3), ")"),
    paste("QDA (AUC =", round(auc.qda, 3), ")")
)

plot(perf.knn, col = "red", main = "Courbes ROC", lty = 1, xlim = c(0, 0.2), ylim = c(0.8, 1))
plot(perf.lr, col = "blue", add = TRUE, lty = 2)
plot(perf.nb, col = "green", add = TRUE, lty = 3)
plot(perf.lda, col = "orange", add = TRUE, lty = 4)
plot(perf.qda, col = "purple", add = TRUE, lty = 5)

legend(0.11, 0.88,
       legend = legend_labels,
       col = c("red", "blue", "green", "orange", "purple"),
       lty = c(1, 2, 3, 4, 5), cex = 0.8)
```