# Projet Apprentissage Statistiques M1MINT

ROHIMUN SHAKIL

2024-04-23

## Introduction

L'objectif de ce projet est de prédire les mouvements du marché boursier, notamment de déterminer si le prix d'une action sera plus élevé ou plus bas le lendemain par rapport à aujourd'hui. Pour ce faire, nous utiliserons l'algorithme de machine learning appelé "Random Forest Classifier", en tenant compte de la problématique des classes déséquilibrées. Dans ce cadre, nous attribuerons la classe :

- +1 lorsque le prix de l'action augmente de manière significative entre les dates n et n+1
- -1 lorsque le prix diminue de manière significative
- 0 si le prix reste proche du prix précédent.

L'objectif final à travers ce système est de créer un bot qui achète l'action lorsque la prédiction est dans la classe +1, vend à la baisse si elle est dans la classe -1, et reste inactif si la prédiction est dans la classe 0.

## Obtention des données du marché

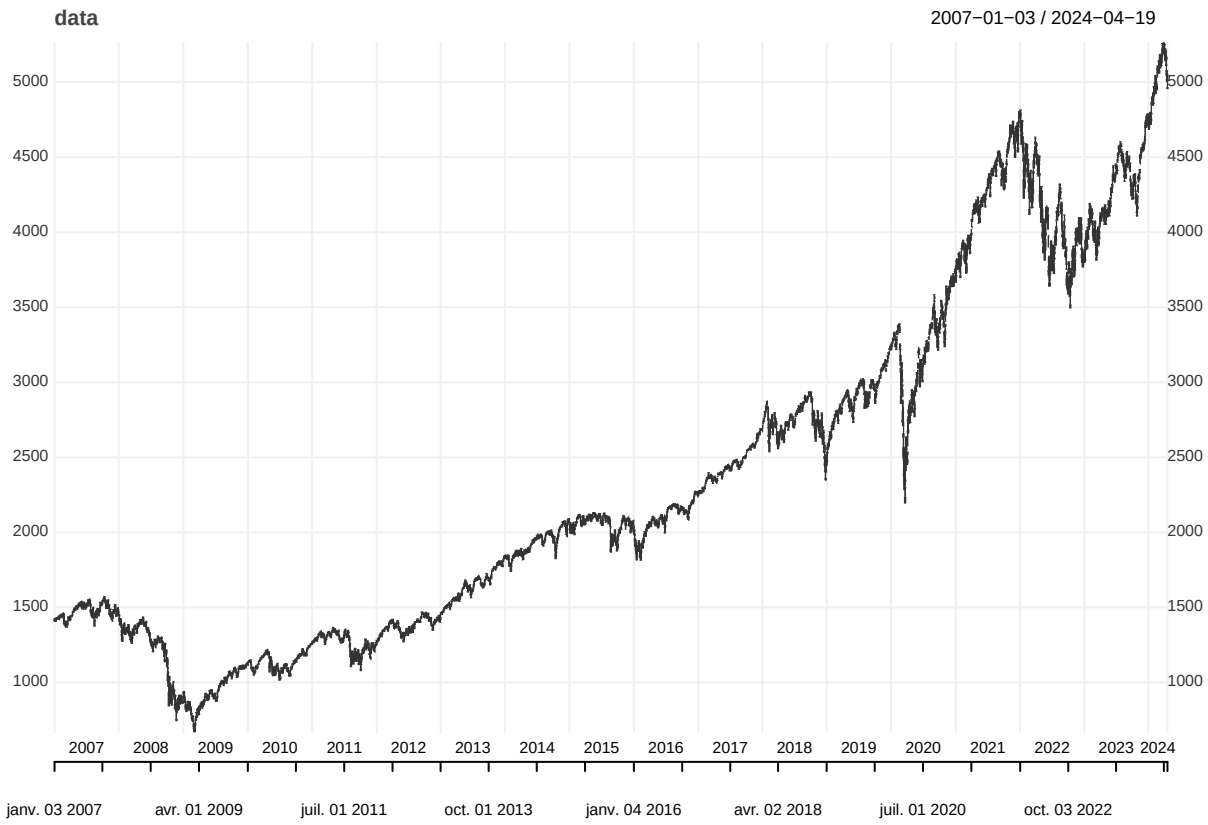
Pour acquérir les données, nous utilisons le package quantmod, qui récupère les données quotidiennes du marché financier. Voici quelques exemples de données boursières couramment utilisées.

```
getSymbols("AAPL", warnings = FALSE, auto.assign = TRUE) # Apple
getSymbols("MSFT", warnings = FALSE, auto.assign = TRUE) # Microsoft
getSymbols("AMZN", warnings = FALSE, auto.assign = TRUE) # Amazone
getSymbols("GOOG", warnings = FALSE, auto.assign = TRUE) # Google

getSymbols("^GSPC", warnings = FALSE, auto.assign = TRUE) # SP500
getSymbols("^FCHI", warnings = FALSE, auto.assign = TRUE) # CAC40
```

Dans notre étude, nous utiliserons uniquement les données du S&P 500, qui représentent la capitalisation boursière pondérée des 500 plus grandes entreprises des États-Unis. Il s'agit de l'un des indices les plus largement utilisés, souvent considéré comme un baromètre de la santé globale du marché boursier américain.

```
data <- GSPC
data <- data[, 1:5]
colnames(data) <- c("Open","High","Low","Close","Volume")
head(data)
```

Ci-dessous, nous observons l'évolution du prix du S&P 500 au fil du temps.




# Transformation des données

```
data <- as.data.frame(data)
epsilon_percent <- 0.01
# Ajout Epsilon
data <- data %>%
  mutate( Epsilon = data$Close * epsilon_percent)
# Ajout Tomorrow
data <- data %>%
  mutate(Tomorrow = lead(Close))
# Ajout Target
data <- data %>%
  mutate(Target = case_when(
    Tomorrow > Close + Epsilon ~ 1,
    Tomorrow < Close - Epsilon ~ -1,
    TRUE ~ 0
  ))
data$Target <- factor(data$Target)
tail(data)
```

```
##                Open    High     Low   Close     Volume Epsilon Tomorrow Target
## 2024-04-12 5171.51 5175.03 5107.94 5123.41 3963220000 51.2341  5061.82     -1
## 2024-04-15 5149.67 5168.43 5052.47 5061.82 3950210000 50.6182  5051.41      0
```

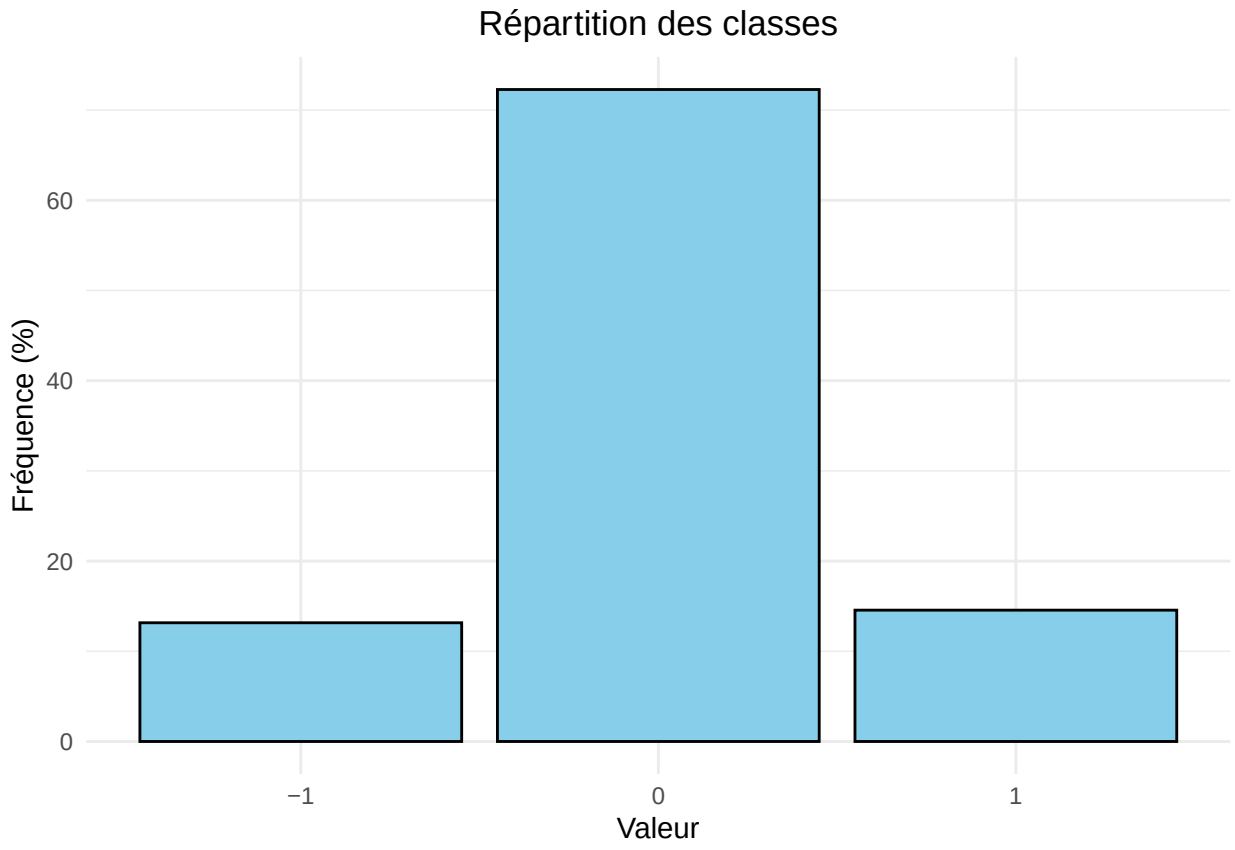
```
## 2024-04-16 5064.59 5079.84 5039.83 5051.41 4006200000 50.5141  5022.21      0
## 2024-04-17 5068.97 5077.96 5007.25 5022.21 3596130000 50.2221  5011.12      0
## 2024-04-18 5031.52 5056.66 5001.89 5011.12 3619760000 50.1112  4967.23      0
## 2024-04-19 5005.44 5019.02 4953.56 4967.23 3878750000 49.6723       NA      0
```

Après avoir préalablement nettoyé mes données, ajouté de nouveaux indicateurs et finalement transformé le tout en data frame, voici nos données :

- Index : La date
- Open : Le prix d'ouverture du jour
- High : Le prix le plus élevé enregistré au cours de la journée
- Low : Le prix le plus bas enregistré au cours de la journée
- Close : Le prix de clôture du jour
- Volume : La quantité d'actions échangées au cours de la journée
- Epsilon : Le prix de epsilon pour cent du close
- Tomorrow : Le prix de clôture du lendemain
- Target : Renvoie +1 si le prix de clôture du lendemain est supérieur au prix de clôture du jour plus epsilon, -1 si le prix de clôture du lendemain est inférieur au prix de clôture du jour moins epsilon, et 0 sinon

Notre objectif est de prédire la valeur de Target pour le lendemain.

## Visualisation du problème de classes



Ci-dessus, il est clairement observé que la classe 0 est sur-représentée par rapport aux deux autres classes lorsque l'on choisit un epsilon assez grand.

### Pourquoi et comment choisir ce epsilon

En réalité, lorsqu'on investit sur les marchés financiers, il est nécessaire de passer par un intermédiaire, généralement une société de courtage en trading. À chaque passage d'ordre d'achat ou de vente, ces sociétés prélèvent un pourcentage appelé spread. Le choix de l'epsilon est crucial pour éviter de subir des pertes même si la prédiction était correcte. De plus, opter pour un epsilon élevé assure une plus grande sécurité, car lorsque epsilon tend vers l'infini, le bot n'ouvre aucune position, évitant ainsi toute perte d'argent, mais également toute possibilité de gain. Enfin, sur des indices boursiers tels que le S&P500, le CAC40 ou encore Apple, nous observons des marchés relativement stables, où les prix évoluent peu sur de courtes périodes de temps. Cette stabilité réduit l'aléa et nous permet d'obtenir de meilleures prédictions potentielles grâce à notre algorithme.

## Problème théorique du Random Forest Classifier : Classification and Regression Trees (CART)

Soit $p$ variables explicatives $X^1, ..., X^p$ (quantitatives ou qualitatives), une variable qualitative Y a expliquer et le $N$-echantillon :

$(x_1, y_1), ..., (x_N, y_N)$, avec $x_i = (x_i^1, ..., x_1^p)$ pour tout $i \in [1, N]$.

CART construit l'arbre récursivement de la façon suivante : à chaque étape, on choisit de diviser un nœud, et pour cela on détermine une valeur de j et une coupure optimale du domaine de $X^j$. Supposons qu'une coupure génère la partition $R_1, .., R_M$ et notons $c_m$ l'ajustement optimal (constant) de $Y$ dans chaque région $R_m$. Le vecteur $y = (y_1, ..., y_N)$ est alors ajuste par :

$f(x) = \sum_{m=1}^{M} c_m \mathbb{1}_{R_m}(x)$, où la valeur optimale de $c_m$ est celle qui minimise la somme $S_M = \sum_i (y_i - f(x_i))^2$.

Pour tout $m = 1, 2, ..., M$, la meilleure estimation de $c_m$ est :

$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$, avec $N_m = card(i | x_i \in R_m)$.

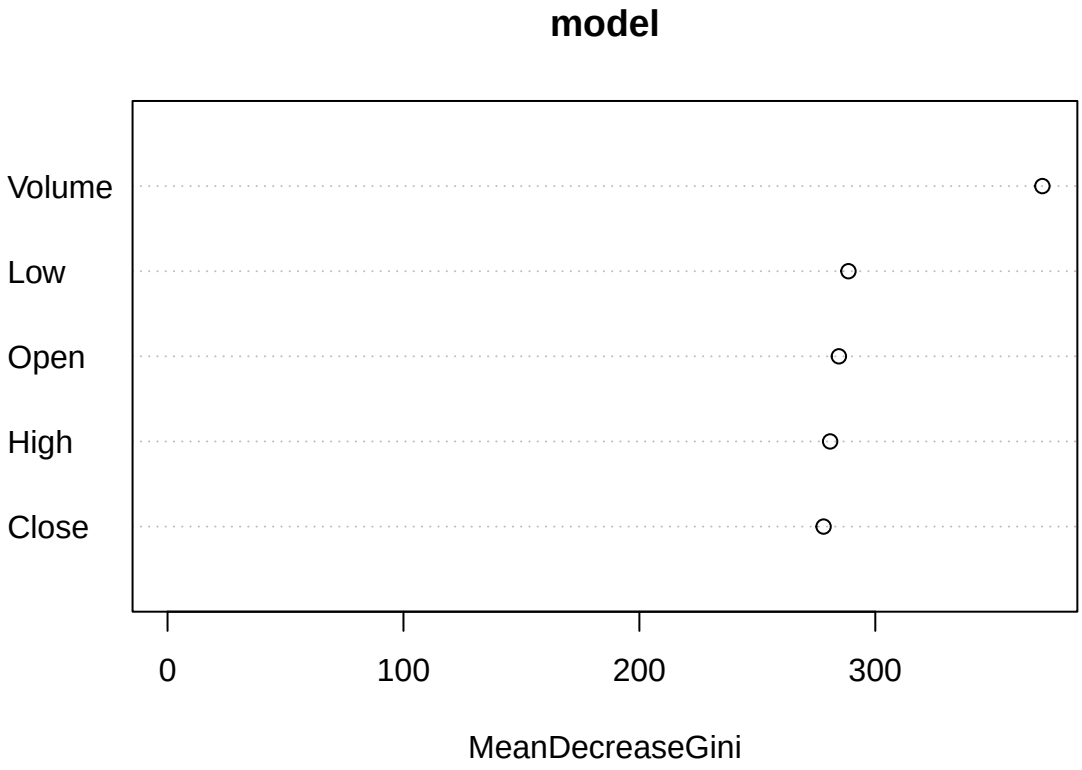
Dans notre cas, $p = 6$ et $N = 4354$ pour le S&P500.

## Application du Random Forest Classifier

```
set.seed(123)
train_size <- 0.8
train_index <- head(1:nrow(data), round(train_size * nrow(data)))
train_data <- data[train_index, ]
test_data <- data[-train_index, ]

model <- randomForest(Target ~ .,
                      data = train_data[, c("Open", "High", "Low", "Close", "Volume", "Target")])

print(model)
```

```
##
## Call:
```

```
##  randomForest(formula = Target ~ ., data = train_data[, c("Open",      "High", "Low", "Close", "Volu
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 30.2%
## Confusion matrix:
##     -1    0   1 class.error
## -1  37  367  50  0.91850220
## 0  102 2339 100  0.07949626
## 1   57  376  55  0.88729508
```




On peut faire un premier graphique qui illustre l'importance des variables explicatives dans la distinction des classes prédites. On constate que la seule variable explicative indépendante du temps est celle qui revêt le plus d'importance lors de la prise de décision de l'algorithme. En effet, le volume passé n'influe pas sur le volume futur. Ce dernier dépend principalement d'événements économiques ou sociologiques.

## Caret : une méthode plus optimisée

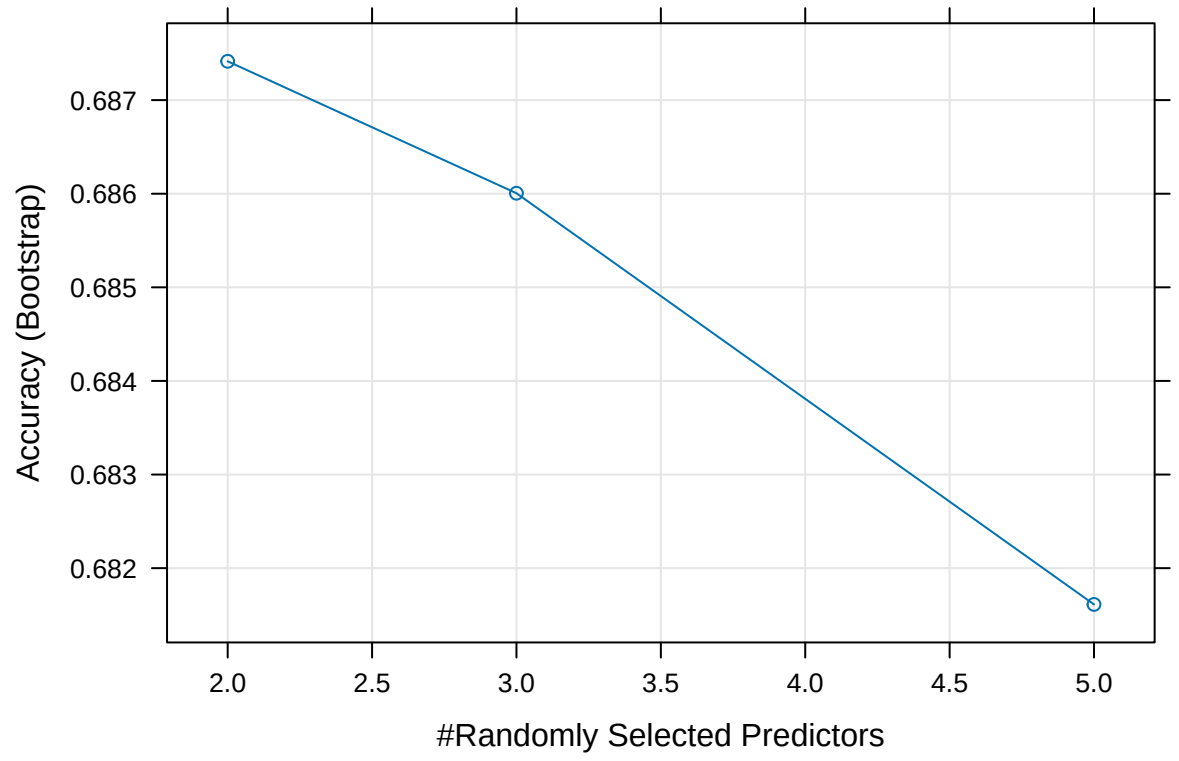
Une méthode plus efficace pour utiliser le modèle Random Forest consiste à utiliser le package Caret, car il propose des valeurs optimales pour les paramètres "ntree" et "mtry".

```
set.seed(123)
model_caret <- train(Target ~ .,
```

```
                    data = train_data[, c("Open", "High", "Low", "Close", "Volume", "Target")],
                    method = "rf")
print(model_caret)
```

```
## Random Forest
##
## 3483 samples
##    5 predictor
##    3 classes: '-1', '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3483, 3483, 3483, 3483, 3483, 3483, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.6874147  0.1067497
##   3     0.6860052  0.1101117
##   5     0.6816127  0.1123845
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Il est notable que le nombre optimal de variables à sélectionner aléatoirement à chaque division ("mtry") est de 2, comme indiqué dans le graphique suivant :



## Explication du calcul de la précision

Dans notre cas, nous nous concentrons particulièrement sur les faux positifs, qui se produisent lorsque l'algorithme prédit incorrectement une action d'achat ou de vente. Ainsi, pour évaluer la performance du modèle, nous utilisons l'indicateur FPR (False Positive Rate), calculé selon la formule suivante :

$FRP = \frac{FauxPositifs}{FauxPositifs+VraisNegatifs}$.

L'objectif est de minimiser les faux positifs, car ils représentent des prédictions incorrectes qui pourraient entraîner des pertes financières, on obtient donc la formule suivante :

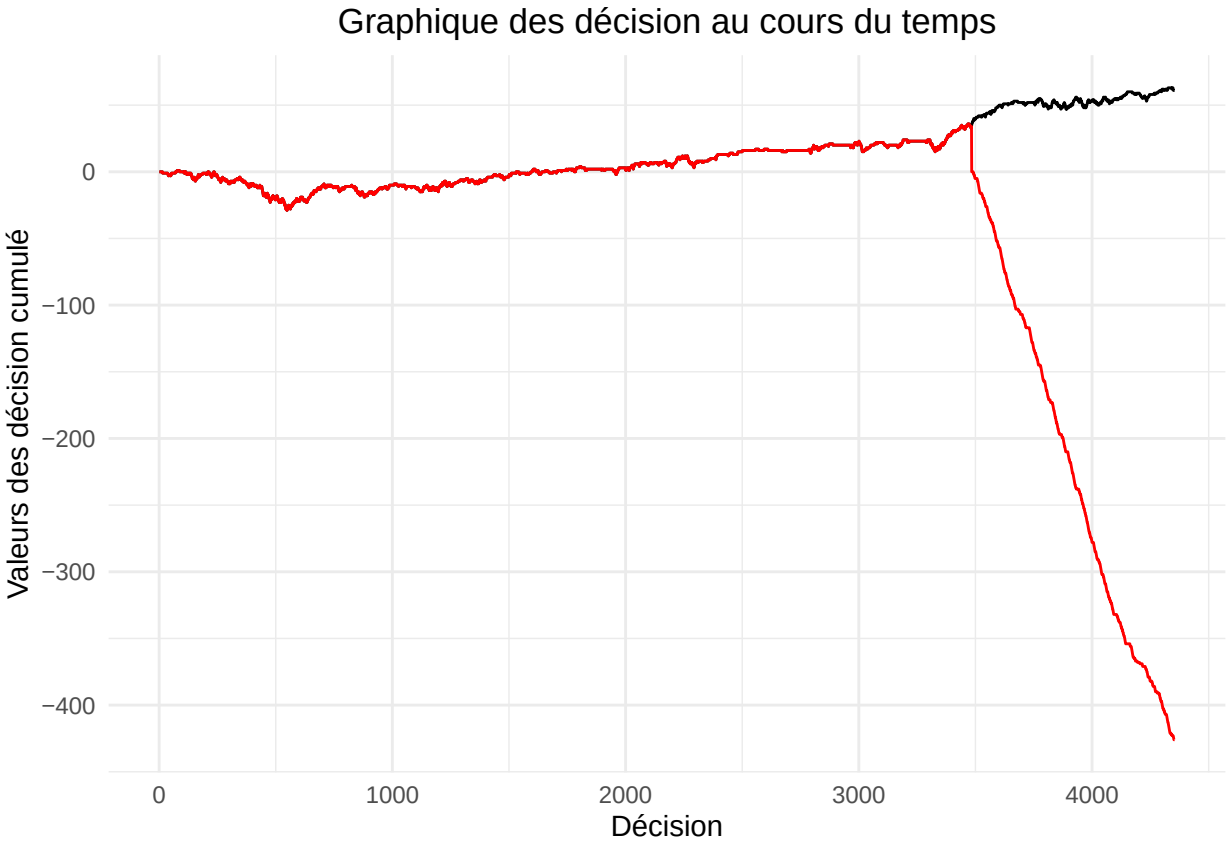
$FRP_{Totale} = \frac{FP_{classe=-1}+FP_{classe=0}+FP_{classe=1}}{Total_{observation}}$.

```
conf_matrix <- model$confusion

FPm1 <- conf_matrix["-1", "0"] + conf_matrix["-1", "1"]
FP0 <- conf_matrix["0", "-1"] + conf_matrix["0", "1"]
FP1 <- conf_matrix["1", "-1"] + conf_matrix["1", "0"]
Total_obs <- conf_matrix["-1", "-1"] + conf_matrix["-1", "0"] + conf_matrix["-1", "1"] +
  conf_matrix["0", "-1"] + conf_matrix["0", "0"] + conf_matrix["0", "1"] +
  conf_matrix["1", "-1"] + conf_matrix["1", "0"] + conf_matrix["1", "1"]


FRP <- (FPm1 + FP0 + FP1)/Total_obs
FRP
```

```
## [1] 0.3020385
```

On retrouve le résultat de l'erreur OOB du Random Forest.

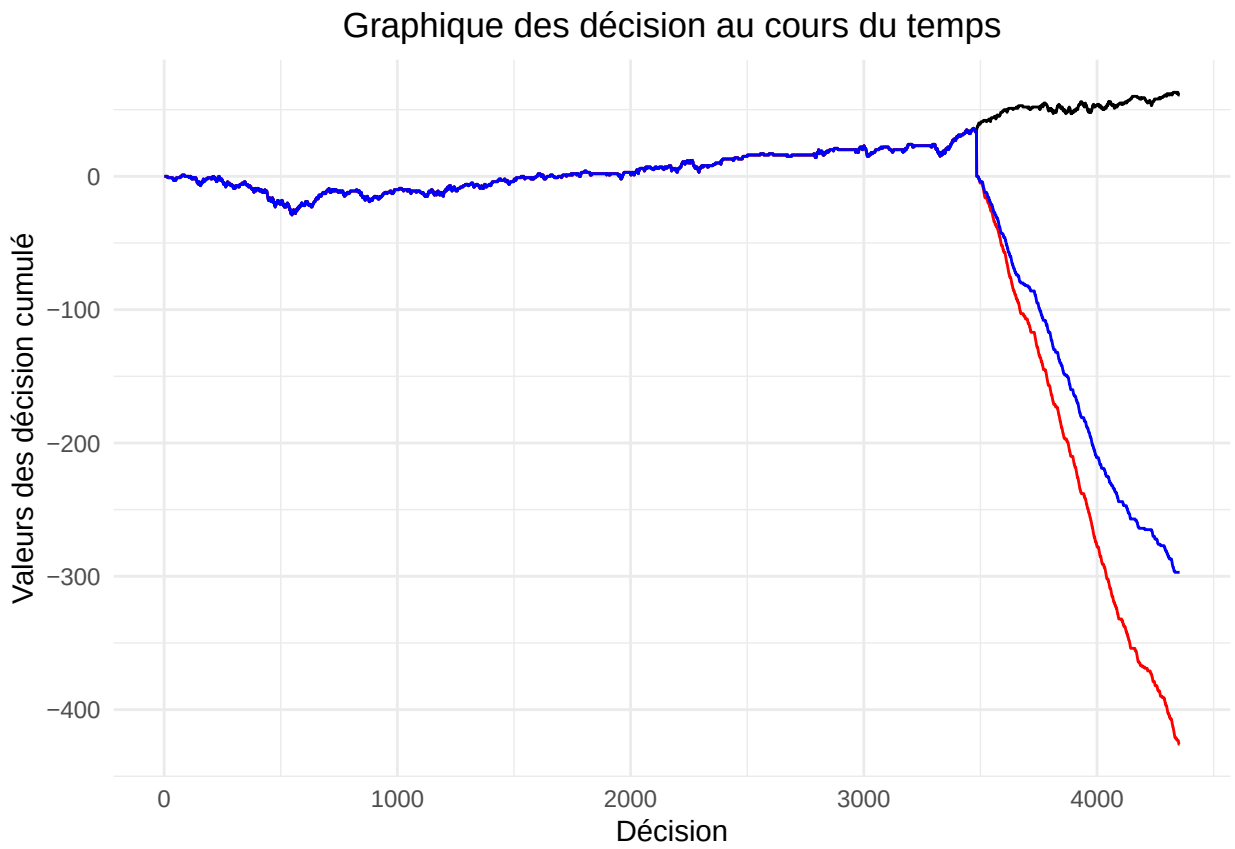
## Visualisation des résultats




On remarque que la prédiction renvoie très majoritairement la classe -1. Afin de résoudre ce problème, nous tentons de rééquilibrer les classes pour obtenir un résultat plus équilibré.

## Équilibrer les classes ?

```
##
## Call:
##  randomForest(formula = Target ~ ., data = train_data_w[, c("Open",      "High", "Low", "Close", "Vo
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 29.51%
## Confusion matrix:
##    -1    0  1 class.error
## -1 28  385 41  0.93832599
## 0  78 2385 78  0.06139315
## 1  46  400 42  0.91393443
```



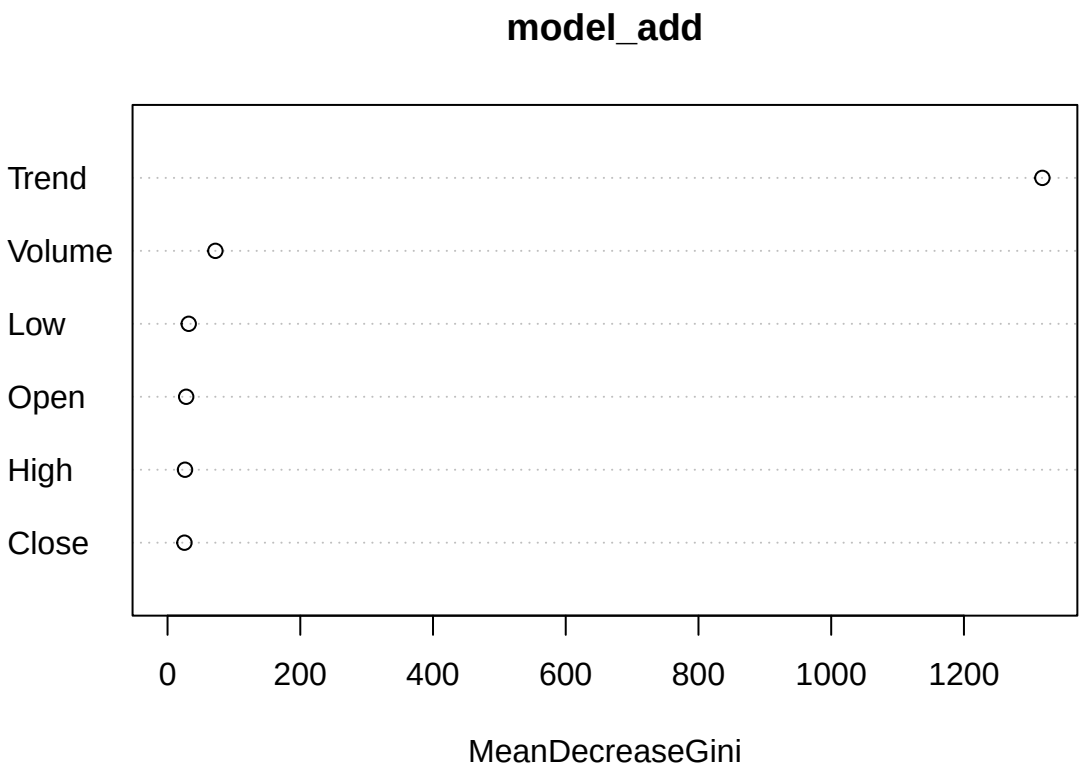

On remarque qu'il y a légèrement moins d'erreur mais ce n'est pas assez significatif pour être pris en compte.

## Les solutions envisageables

On pourrait corréler nos données avec le temps en créant une nouvelle colonne tendance comme ceci :

```
##                Open    High    Low   Close     Volume Epsilon Tomorrow Target
## 2008-10-31  953.11  984.38 944.59  968.75 6394350000  9.6875   966.30      0
## 2008-11-03  968.67  975.57 958.82  966.30 4492280000  9.6630  1005.75      1
## 2008-11-04  971.31 1007.51 971.31 1005.75 5531290000 10.0575   952.77     -1
## 2008-11-05 1001.84 1001.84 949.86  952.77 5426640000  9.5277   904.88     -1
## 2008-11-06  952.40  952.40 899.73  904.88 6102230000  9.0488   930.99      1
## 2008-11-07  907.44  931.46 906.90  930.99 4931640000  9.3099   919.21     -1
## 2008-11-10  936.75  951.95 907.47  919.21 4572000000  9.1921   898.95     -1
## 2008-11-11  917.15  917.15 884.90  898.95 4998340000  8.9895   852.30     -1
## 2008-11-12  893.39  893.39 850.48  852.30 5764180000  8.5230   911.29      1
## 2008-11-13  853.13  913.01 818.69  911.29 7849120000  9.1129   873.29     -1
##             Trend
## 2008-10-31      0
## 2008-11-03      1
## 2008-11-04     -1
## 2008-11-05     -2
## 2008-11-06      1
## 2008-11-07     -1
## 2008-11-10     -2
```

```
## 2008-11-11    -3
## 2008-11-12     1
## 2008-11-13    -1
```




## Conclusion

La méthode de Random Forest, bien qu'amplement utilisée en ligne pour prédire les prix des actions, s'avère finalement peu adaptée à notre contexte. Une des raisons principales réside dans le fonctionnement du classifieur Random Forest, où l'algorithme mélange de manière aléatoire les données, altérant ainsi la temporalité inhérente à notre jeu de données. Dans le cas de données temporelles, ce mélange entraîne un biais dans les prédictions. De plus, l'algorithme a tendance à souffrir de surapprentissage, produisant théoriquement des prédictions presque parfaites, mais inefficaces dans la pratique.

Pour pallier ces problèmes, des solutions plus efficaces pour utiliser le Random Forest impliquent l'intégration de nouvelles données temporelles, telles que la variable "Trend", qui agit comme une moyenne mobile. Cette variable joue un rôle crucial dans les prédictions de l'algorithme, expliquant ainsi son importance croissante dans le processus. En intégrant ces aspects temporels de manière plus appropriée et en optimisant le paramètre epsilon, il est possible d'améliorer la performance prédictive et la pertinence des résultats obtenus.

Après des recherches approfondies en ligne, il apparaît que dans le monde professionnel, la prédiction des prix via l'apprentissage automatique repose sur l'utilisation d'un mélange d'algorithmes, notamment une combinaison bien connue : Random Forest et Gradient Boosting. De plus, la version de Random Forest utilisée est ajustée pour minimiser le mélange aléatoire pendant l'exécution de l'algorithme, ce qui conduit à des résultats améliorés sans risque de surapprentissage.

# Références


- https://cran.r-project.org/web/packages/quantmod/quantmod.pdf
- http://mehdikhaneboubi.free.fr/random_forest_r.html#creation-de-la-base-de-travail
- machine learning for algorithmic trading stefan jansen
- cours Université Paris-Dauphine : Arbre de décision, Patrice Bertrand