R Notebook

Importation des librairies

```
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.
```

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

```
library(opera)
library(dplyr)
```

```
##
## Attachement du package : 'dplyr'

## L'objet suivant est masqué depuis 'package:randomForest':
##
##     combine

## Les objets suivants sont masqués depuis 'package:stats':
##
##     filter, lag

## Les objets suivants sont masqués depuis 'package:base':
##
##     intersect, setdiff, setequal, union
```

Lecture des données

```
df = read.csv("https://raw.githubusercontent.com/Tdjaaleb/MALIA/main/Time%20Series/Preprocessing/Data/cl
X_train = subset(na.omit(df[1:52561,]), select = -c(Conso, ConsoT.1, Date, Heure))
Y_train = df$Conso[337:52561]

X_test = subset(df[105216:121248,], select = -c(Conso, ConsoT.1, Date, Heure))
Y_test = df$Conso[105216:121248]
```

Construction des experts

Random forest

```
expert_rf <- randomForest(x=X_train, y=Y_train, ntree=100, maxnodes=5)
expert_rf_forecast <- predict(expert_rf, newdata=X_test)
```

TSLM

```
ts <- ts(Y_train, frequency = 48)
expert_tslm <- tslm(ts ~ tod+tow+Fourier+ConsoJ.1+ConsoJ.7+trend+season, data=X_train)
pred <- forecast(expert_tslm, newdata = X_test)
```

```
## Warning in predict.lm(predict_object, newdata = newdata, se.fit = TRUE, : les
## prédictions venant d'un modèle de rang faible peuvent être trompeuses

## Warning in predict.lm(predict_object, newdata = newdata, se.fit = TRUE, : les
## prédictions venant d'un modèle de rang faible peuvent être trompeuses
```

```
expert_tslm_forecast <- pred$mean
```

Gradient Boosting

```
expert_gbm <- gbm.fit(x=X_train, y=Y_train, n.trees=1000, distribution="gaussian", shrinkage=0.01)
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1 145074375.1147             nan     0.0100 1841809.5305
##      2 143258060.6429             nan     0.0100 1813164.6320
##      3 141468878.1376             nan     0.0100 1780220.9044
##      4 139727991.9448             nan     0.0100 1739019.4872
##      5 138015152.6243             nan     0.0100 1718477.8465
##      6 136340211.7465             nan     0.0100 1681554.6155
##      7 134706465.9586             nan     0.0100 1640946.6478
##      8 133089746.7433             nan     0.0100 1618670.0619
##      9 131514885.6803             nan     0.0100 1581268.4246
##     10 129960142.2863             nan     0.0100 1555199.2922
##     20 115918839.7275             nan     0.0100 1299849.1869
##     40 93936569.8425              nan     0.0100 938887.2948
```

```
##     60 77648428.3008             nan     0.0100 711715.2937
##     80 65227954.4071             nan     0.0100 549602.4852
##    100 55558142.8236             nan     0.0100 426764.8057
##    120 47937035.3766             nan     0.0100 339076.7486
##    140 41856027.3419             nan     0.0100 270273.2395
##    160 36971945.1121             nan     0.0100 218366.1544
##    180 33003847.6845             nan     0.0100 181321.2182
##    200 29756712.7958             nan     0.0100 146773.9092
##    220 27081888.2781             nan     0.0100 118325.0928
##    240 24869995.5041             nan     0.0100 99283.1364
##    260 23031933.3555             nan     0.0100 83370.4020
##    280 21481122.8591             nan     0.0100 71017.7950
##    300 20173663.6666             nan     0.0100 59506.9155
##    320 19072349.9659             nan     0.0100 51394.1688
##    340 18106097.3948             nan     0.0100 45237.4677
##    360 17248246.6561             nan     0.0100 40827.4792
##    380 16491016.0592             nan     0.0100 34132.5742
##    400 15820549.7390             nan     0.0100 31303.2528
##    420 15226204.7383             nan     0.0100 28166.7466
##    440 14697056.3428             nan     0.0100 24662.9900
##    460 14226419.2098             nan     0.0100 21804.1967
##    480 13806929.7979             nan     0.0100 19281.3266
##    500 13428028.2922             nan     0.0100 17980.0321
##    520 13078962.5100             nan     0.0100 16540.6075
##    540 12758242.6807             nan     0.0100 14505.1584
##    560 12462528.3145             nan     0.0100 12651.9954
##    580 12189129.7995             nan     0.0100 12603.3377
##    600 11935290.5349             nan     0.0100 11835.7315
##    620 11702115.8328             nan     0.0100 11059.0066
##    640 11485935.6781             nan     0.0100 10519.2992
##    660 11287270.2892             nan     0.0100 9465.6972
##    680 11103463.5577             nan     0.0100 7716.7010
##    700 10931872.0403             nan     0.0100 7907.9305
##    720 10774761.7197             nan     0.0100 7358.6084
##    740 10628825.0175             nan     0.0100 6532.5549
##    760 10493937.7411             nan     0.0100 6214.9347
##    780 10369602.1927             nan     0.0100 5300.1958
##    800 10253965.0779             nan     0.0100 4853.1054
##    820 10146212.3878             nan     0.0100 5087.5923
##    840 10045207.2144             nan     0.0100 4554.1943
##    860  9951954.1109             nan     0.0100 4422.8156
##    880  9865578.6190             nan     0.0100 3843.2414
##    900  9784856.9009             nan     0.0100 3753.7653
##    920  9710090.8762             nan     0.0100 3326.4180
##    940  9640657.6899             nan     0.0100 3300.7162
##    960  9575485.3636             nan     0.0100 3013.2123
##    980  9514810.3083             nan     0.0100 2797.0110
##   1000  9458392.0485             nan     0.0100 2531.2329
```

```
expert_gbm_forecast <- predict(expert_gbm, newdata = X_test)
```

```
## Using 1000 trees...
```

Aggregation

```
experts <- cbind(expert_rf_forecast, expert_tslm_forecast, expert_gbm_forecast)
colnames(experts) <- c("rf", "tslm", "gbm")
or <- oracle(Y_test, experts, model = "convex", loss.type = "square")
```

RMSE des experts seuls

```
rmse_exp <- apply(experts, 2, function(x){sqrt(mean((x - Y_test)^2))})
rmse_exp %>% round(, digits = 0) %>% sort
```

```
## tslm  gbm   rf
## 2721 2826 5527
```

Calcul de la valeur théorique du learning rate

```
M <- mean((Y_train - X_train$ConsoJ.7)^2, na.rm = T)
learning.rate <- (1/M) * sqrt(8*log(ncol(experts))) / length(Y_test)
```

Aggregation

```
agg.online_theoric<- mixture(Y = Y_test,
                             experts = experts,
                             model = 'EWA',
                             loss.type = "square",
                             loss.gradient = F,
                             parameter=list(eta=learning.rate))
```

Résultats

```
summary(agg.online_theoric)
```

```
## Aggregation rule: EWA
## Loss function:  squareloss
## Gradient trick:  FALSE
## Coefficients:
##     rf  tslm   gbm
##  0.044 0.493 0.463
##
## Number of experts:  3
## Number of observations:  16033
## Dimension of the data:  1
##
##         rmse   mape
## EWA     2830 0.0433
## Uniform 3200 0.0524
```

Visualisation

```
plot(agg.online_theoric)
```

```
## Warning in par(def.par, new = FALSE): argument 1 does not name a graphical
## parameter
```