Market Volatility Forecasting using GARCH models - TGARCH

Tommaso Grandi

```r
library(readr)
library(xts)
```

Load and manipulate data exported from Python notebook

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
returns <- read_csv("returns.csv", show_col_types = FALSE)
returns$Date <- as.Date(returns$Date, format = "%d/%m/%Y")
colnames(returns) = c('Date', 'Returns')
data = data.frame(returns)

ret = xts(data$Returns, order.by = data$Date)
head(ret)
```

```
##                  [,1]
## 2004-01-05  0.5317418
## 2004-01-06  0.4421739
## 2004-01-07 -1.6959140
## 2004-01-08  1.1556172
## 2004-01-09 -0.2256844
## 2004-01-12  0.1092657
```

```r
## LOAD REALIZED VOL

realized <- read_csv("realized.csv", show_col_types = FALSE)
realized$Date <- as.Date(realized$Date, format = "%d/%m/%Y")
colnames(realized) = c('Date', 'Volatility')
data_vol = data.frame(realized)

vol = xts(data_vol$Volatility, order.by = data_vol$Date)
head(vol)
```

```
##                   [,1]
## 2004-01-05 0.6785438
## 2004-01-06 0.6698834
## 2004-01-07 0.6738177
## 2004-01-08 0.5615371
## 2004-01-09 0.5580599
## 2004-01-12 0.5543670
```

```r
tail(ret)
```

```
##                     [,1]
## 2023-12-06  0.80452022
## 2023-12-07 -0.67496286
## 2023-12-08  0.93183769
## 2023-12-11  0.07561934
## 2023-12-12 -0.27974808
## 2023-12-13 -0.15172007
```

```r
tail(vol)
```

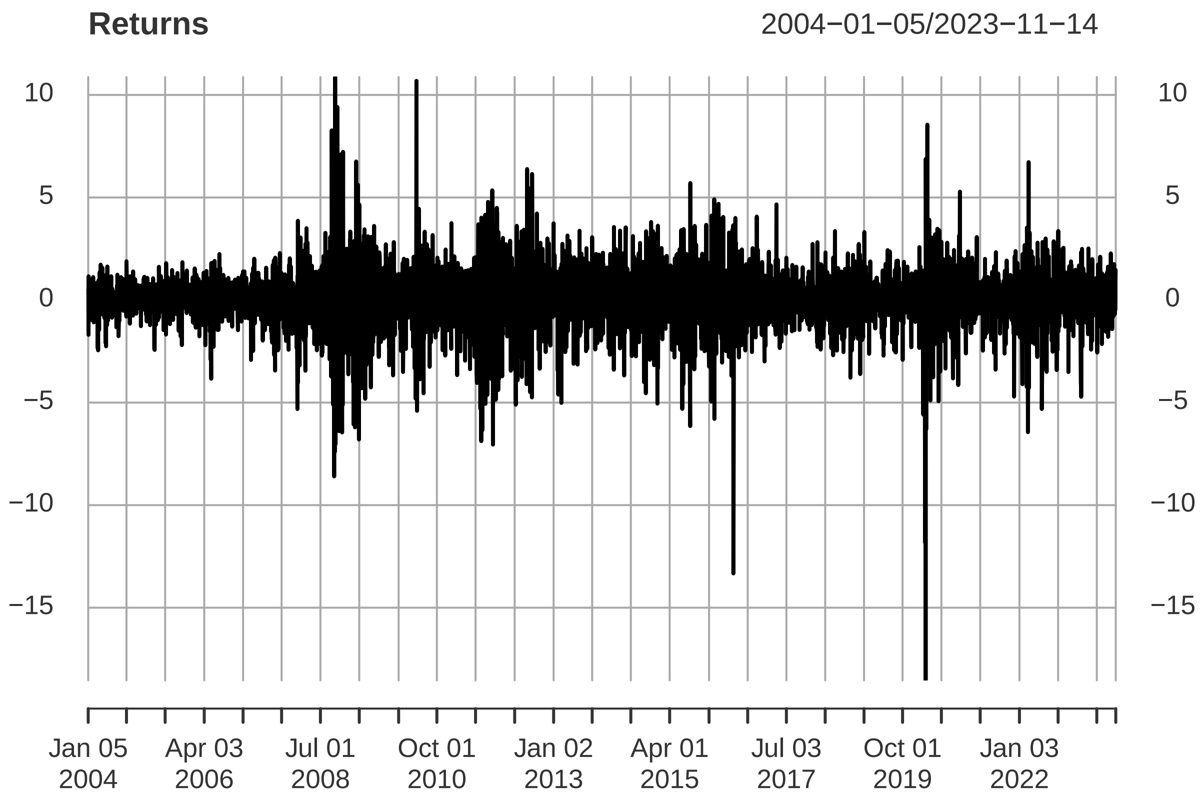
```
##                   [,1]
## 2023-11-07 0.6968787
## 2023-11-08 0.6958120
## 2023-11-09 0.7091035
## 2023-11-10 0.7038279
## 2023-11-13 0.6943186
## 2023-11-14 0.6472314
```

The two series ends at different dates, let's cut the return series.

```r
# extract last date of vol as char
last = as.character(index(vol)[length(vol)])
last = gsub('-', '', last)

# extract first date as char
begin = as.character(index(vol)[1])
begin = gsub('-', '', begin)

# paste the two chars
cut = paste(begin, last, sep = '/')

# Cut ret so that they end in the same day of vol
ret <- ret[cut]
tail(ret)
```

```
##                    [,1]
## 2023-11-07 -0.6913642
## 2023-11-08  0.1302152
## 2023-11-09  0.7393555
## 2023-11-10 -0.4899569
## 2023-11-13  1.4661845
## 2023-11-14  1.4381823
```

```r
plot(ret, main = "Returns", type = "l")
```



```r
library('PerformanceAnalytics')
```

Load the necessary libraries

```
##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend
```

```r
library('rugarch')
```

```
## Loading required package: parallel

##
## Attaching package: 'rugarch'
## The following object is masked from 'package:stats':
##
##     sigma
```

```
library('urca')
```

TGARCH

The Threshold GARCH was developed by Zakoian in 1994 and is similar to GJR GARCH. The main difference is that specification is on conditional standard deviation instead on the conditional variance.

For a TGARCH(1,1), the variance process become:

$$\sigma_t = \omega + \alpha_1^+ \epsilon_{t-1}^+ - \alpha_1^- \epsilon_{t-1}^- + \beta_1^+ \sigma_{t-1}^+ - \beta_1^- \sigma_{t-1}^-$$

where $\epsilon_{t-1}^+ = \epsilon_{t-1}$ if $\epsilon_{t-1} > 0$, and $\epsilon_{t-1}^+ = 0$ if $\epsilon_{t-1} \leq 0$. Likewise $\epsilon_{t-1}^- = \epsilon_{t-1}$ if $\epsilon_{t-1} \leq 0$, and $\epsilon_{t-1}^- = 0$ if $\epsilon_{t-1} > 0$. This allows to have different estimated parameters for positive (α_1^+, β_1^+) and negative (α_1^-, β_1^-) news shocks and past conditional volatilities that evidently will have a diversified effect on the actual conditional volatility, helping to capture the asymmetry in the market.

Let's specify a TGARCH(1,1) model with a skewed t-distribution for the residuals.

```
### TGARCH Model Specification
# As before, model specification is:
# - AR(1) for the mean process
# - (1,1) for the variance process (in this case a TGARCH)
# - skewed t distribution for the residuals (shape + skew parameters)

spec = ugarchspec(
  variance.model = list(
    model = "fGARCH", garchOrder = c(1, 1), #fGARCH=familyGARCH
                        submodel = "TGARCH",
                        variance.targeting = FALSE),
    mean.model = list(armaOrder = c(1, 0), include.mean = TRUE),
    distribution.model = "sstd" #skewed t dist
  )

print(spec)
```

```
##
## *---------------------------------*
## *       GARCH Model Spec          *
## *---------------------------------*
##
## Conditional Variance Dynamics
## ------------------------------------
## GARCH Model      : fGARCH(1,1)
## fGARCH Sub-Model : TGARCH
## Variance Targeting   : FALSE
##
## Conditional Mean Dynamics
## ------------------------------------
## Mean Model       : ARFIMA(1,0,0)
## Include Mean     : TRUE
## GARCH-in-Mean        : FALSE
##
## Conditional Distribution
## ------------------------------------
## Distribution :  sstd
## Includes Skew    :  TRUE
## Includes Shape   :  TRUE
## Includes Lambda  :  FALSE
```

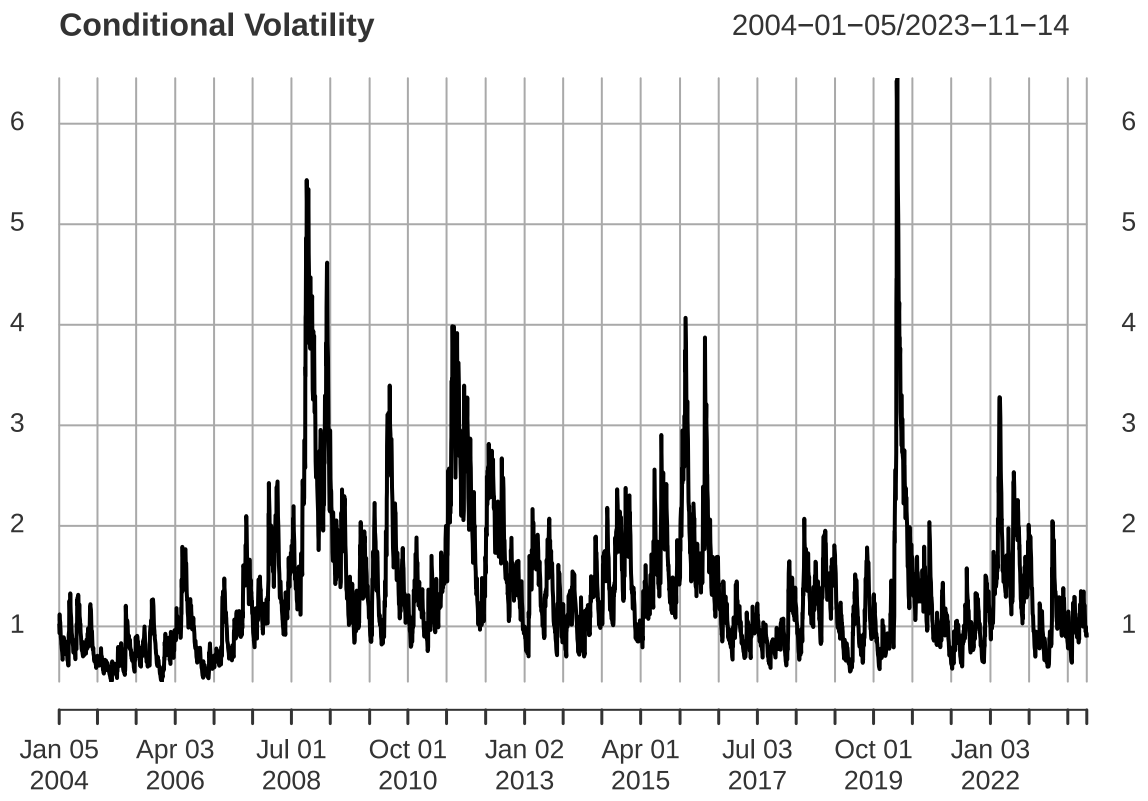
```
### TGARCH Model
tgarch.fit = ugarchfit(spec, ret)
```

```
# Examine the coefficients
tgarch.fit@fit$matcoef
```

```
##            Estimate  Std. Error    t value     Pr(>|t|)
## mu      0.007768533 0.012965615  0.5991643 5.490633e-01
## ar1    -0.042627898 0.012387056 -3.4413260 5.788707e-04
## omega   0.020883539 0.004167342  5.0112370 5.408126e-07
## alpha1  0.069771847 0.010908507  6.3960953 1.594005e-10
## beta1   0.931195356 0.010622066 87.6661203 0.000000e+00
## eta11   0.953785635 0.111747230  8.5352061 0.000000e+00
## skew    0.835324245 0.016950161 49.2811983 0.000000e+00
## shape   7.614289925 0.756058740 10.0710296 0.000000e+00
```

Comments: all coefficients are significant (expect for mu).

```
plot(sigma(tgarch.fit), main='Conditional Volatility')
```



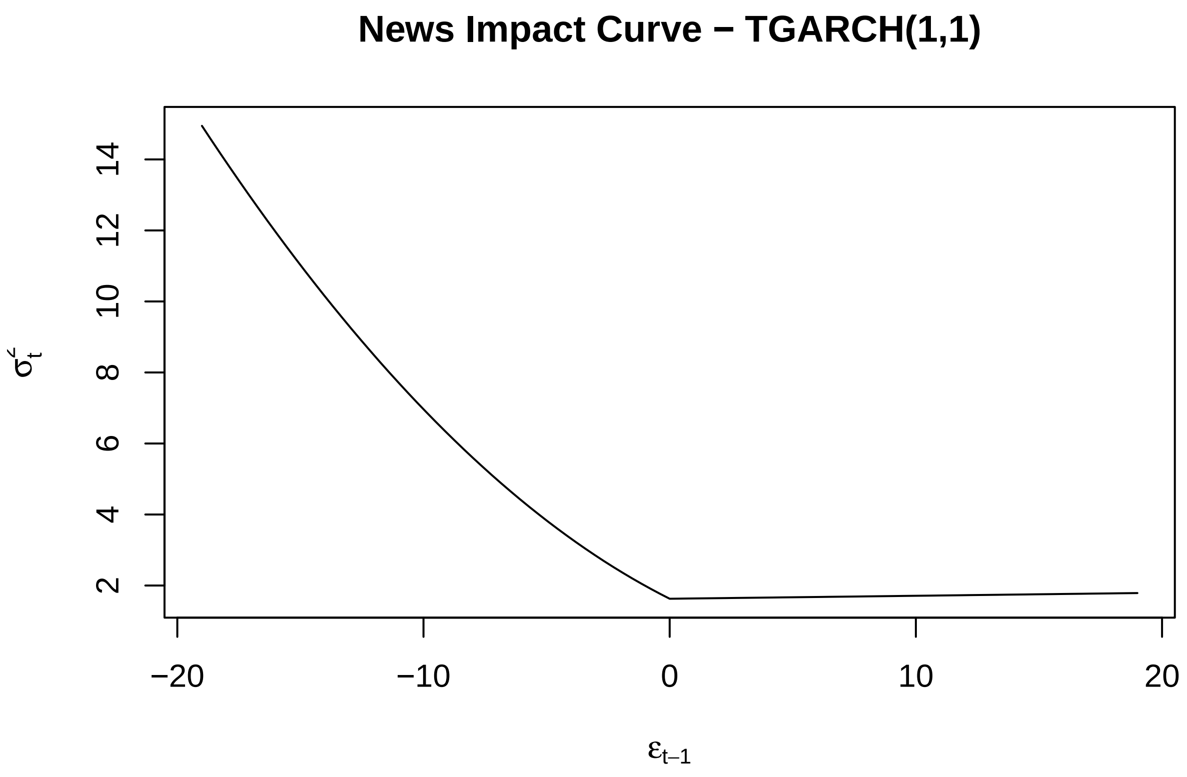
```
print(infocriteria(tgarch.fit))
```

```
##
## Akaike       3.185740
## Bayes        3.196064
## Shibata      3.185735
## Hannan-Quinn 3.189356
```

Comments : Values appeared to be on different scale than ICs calculated in Python.

Let’s look at the news impact curve:

```
NIC <- newsimpact(z= NULL, tgarch.fit)

plot(x = NIC$zx, y = NIC$zy, ylab = NIC$yexpr, xlab = NIC$xexpr, type = "l",
     main = "News Impact Curve - TGARCH(1,1)")
```



Comments: left part of the curve is less steep than GJR NIC and GARCH NIC.

Model Validation for the fitted data (In-Sample)

As before we want to validate the model error measures in the in sample prediction as well as the information criterias.

```r
# Create the evaluate function

evaluate <- function(model, realized_vol) {

  mse <- mean((sigma(model) - realized_vol)^2)
  mae <- mean(abs(sigma(model) - realized_vol))

  IC <- infocriteria(model)[1:2, 1]


  return(c(mse, mae, IC))

}
```

```r
# test the function on the TGARCH
evaluate(tgarch.fit, vol)
```

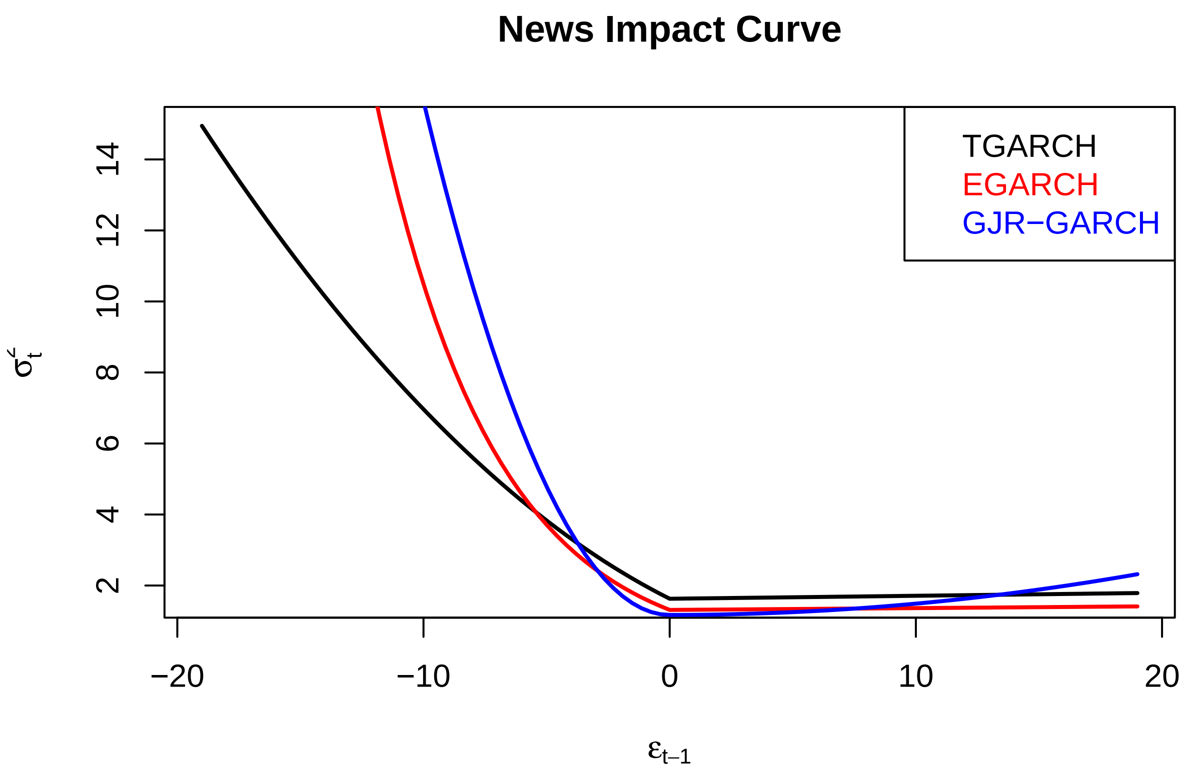
```
##                        Akaike     Bayes
## 0.3336846 0.3859930 3.1857396 3.1960640
```

Information Criterias appear to be on a different scale than those in Python, let's recreate all the models we had in Python in order to compare the TGARCH with the previous models.

```r
# BASIC GARCH
garch.fit <- ugarchfit(
  ugarchspec(
    variance.model = list(model = "sGARCH",
                          garchOrder = c(1, 1),
                          submodel = NULL,
                          variance.targeting = FALSE),
    mean.model = list(armaOrder = c(1, 0), include.mean = TRUE),
    distribution.model = "sstd" #skewed t dist
  ),
                       ret)

# EGARCH
egarch.fit <- ugarchfit(
  ugarchspec(
    variance.model = list(model = "eGARCH", garchOrder = c(1, 1),
                          submodel = NULL,
                          variance.targeting = FALSE),
    mean.model = list(armaOrder = c(1, 0), include.mean = TRUE
                     ),
    distribution.model = "sstd" #skewed t dist
  ),
                       ret)

# GJR GARCH
gjrgarch.fit <- ugarchfit(
  ugarchspec(
    variance.model = list(model = "gjrGARCH", garchOrder = c(1, 1),
                          submodel = NULL,
                          variance.targeting = FALSE),
    mean.model = list(armaOrder = c(1, 0), include.mean = TRUE),
    distribution.model = "sstd" #skewed t dist
  ),
                       ret)
```

Compare models News Impact Curves:

```r
nic_egarch = newsimpact(z= NULL, egarch.fit)
nic_gjr = newsimpact(z= NULL, gjrgarch.fit)

legend <- c("TGARCH","EGARCH","GJR-GARCH")
col  <- c("black", "red", "blue")

plot(x = NIC$zx, y = NIC$zy, ylab = NIC$yexpr, xlab = NIC$xexpr, type = "l", main = "News Impact Curve"

lines(x = nic_egarch$zx, y = nic_egarch$zy, lwd = 2, col = col[2])
lines(x = nic_gjr$zx, y = nic_gjr$zy, lwd = 2, col = col[3])

legend(x = "topright", y = NULL, legend = legend,
       text.col = col)
```



TGARCH left-side of the curve has the smaller steepness, while EGARCH is more steep and GJR has the biggest steepness. This means that GJR conditional variance reacts with more emphasis to negative variation in the past residuals (is the more sensitive model with respect to bad news) whereas TGARCH is the less sensitive model to bad news. Another observation is that the positive side of the GJR NIC curve has a more positive slope compared to the other two. This means that the conditional variance is more sensitive to positive variation in the past good news.

```
## CREATE DATAFRAME TO STORE THE STATISTICS

eval = data.frame()

eval = rbind(eval, evaluate(garch.fit, vol))
eval = rbind(eval, evaluate(egarch.fit, vol))
eval = rbind(eval, evaluate(gjrgarch.fit, vol))
eval = rbind(eval, evaluate(tgarch.fit, vol))


colnames(eval) = c('MSE', 'MAE', 'AIC', 'BIC')
row.names(eval) = c('GARCH', 'EGARCH', 'GJR-GARCH', 'TGARCH')

eval
```

```
##                 MSE       MAE      AIC      BIC
## GARCH     0.3561488 0.3897030 3.228030 3.237064
## EGARCH    0.3237467 0.3809633 3.187766 3.198090
## GJR-GARCH 0.3995823 0.4062296 3.200110 3.210434
## TGARCH    0.3336846 0.3859930 3.185740 3.196064
```

As before, EGARCH is overall the best model on the fitted data in terms of both Information Criterias and Error measures. TGARCH is the second best model and even surpassed EGARCH in the information criterias results.

These two models should be those to be used in the out of sample forecast of the volatility.

But what if we really want to choose a winner?

Diebold Mariano Test

We can use a very specific statistical test that measure the statistical difference between the prediction of two models.

This test is capable of comparing the predictive accuracy between two models and hence test if two models have a statistical difference in their prediction.

```
library(forecast)
```

GARCH vs TGARCH

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

```
# GARCH VS TGARCH
dm.test(e1 = sigma(garch.fit), e2 = sigma(tgarch.fit), h = 1, power = 1, alternative='greater')
```

```
## 
##  Diebold-Mariano Test
## 
## data:  sigma(garch.fit)sigma(tgarch.fit)
## DM = 5.5864, Forecast horizon = 1, Loss function power = 1, p-value =
## 1.22e-08
## alternative hypothesis: greater
```

We reject the null, hence the second method (TGARCH) is more accurate than method 1 (GARCH).

```
# EGARCH vs TGARCH
dm.test(e1 = sigma(egarch.fit), e2 = sigma(tgarch.fit), h = 1, power = 1, alternative='less')
```

EGARCH vs TGARCH

```
## 
##  Diebold-Mariano Test
## 
```

```
## data:  sigma(egarch.fit)sigma(tgarch.fit)
## DM = -4.2066, Forecast horizon = 1, Loss function power = 1, p-value =
## 1.318e-05
## alternative hypothesis: less
```

In this case we used as the alternative hypotheses 'less' so the alternative is that TGARCH is less accurate than EGARCH. We reject the null, hence the second method (TGARCH) has less accuracy than method 1 (EGARCH). This confirms our conclusion from the model evaluation statistics and hence crowns the EGARCH as the best suitable model for this data. Let's go back to the Python Notebook to test EGARCH in out of sample prediction.