```
##xgboost model###############
# Load the desired library
library(xgboost)
library(lubridate)
library(xlsx)
library(dplyr)
set.seed(123)

#    reading data
data <- read.csv("flu1.csv", header = TRUE)
# Conversion date format
data$Date <- as.Date(data$Date)
data$cases <- as.numeric(data$cases)
# Sort the data by date
data <- data[order(data$Date), ]
# Set the time range for the training and testing datasets
train_start <- as.Date("2013-01-01")
train_end <- as.Date("2021-12-01")
test_start <- as.Date("2022-01-01")
test_end <- as.Date("2022-12-01")
# Create the training and testing datasets
train_data <- subset(data, Date >= train_start & Date <= train_end)
test_data <- subset(data, Date >= test_start & Date <= test_end)

# Create sample data
train_df <- train_data %>%
   mutate(Date = as.character(Date)) %>%
   mutate(as.numeric(train_data$cases)) %>%
   select(-Date)    # Delete date column

# Training of the xgboost model
xgb_model <- xgboost(data = as.matrix(train_df[, -1]),
                              label = train_df[, 1],
                              nrounds =300,    #      iterations
                              objective = "reg:squarederror",    # Return to the task
                              max_depth = 7,    # The depth of the tree
                              eta = 0.07,    #    learning rate
                              subsample = 0.7,     # The portion of samples used for each round
iteration
                              MinChild=2,
                              colsample_bytree = 0.4) # The portion of features used for each tree
```

```
# Prediction was performed on the training set and the fitting effect was calculated
train_pred <- predict(xgb_model, as.matrix(train_df[, -1]))
write.xlsx(train_pred,file = "C:/Users/User/Desktop/ARIMA--X/2013-2021/xgboost/The training set was fitted to the value.xls")
train_mse <- mean((train_pred - train_df[, 1])^2)
train_mae <- mean(abs(train_pred - train_df[, 1]))
train_rmse <- sqrt(train_mse)
print(paste("MSE:", train_mse))
print(paste("MAE:", train_mae))
print(paste("RMSE:", train_rmse))

data <- data.frame(observed = train_pred, predicted = train_df[, 1])
# By specifying type = "all", we can calculate the confidence intervals for the MSE, MAE, and RMSE simultaneously
boot_MSE <- boot(data, calc_MSE, R = 1000)
boot_MAE <- boot(data, calc_MAE, R = 1000)
boot_RMSE <- boot(data, calc_RMSE, R = 1000)

# Output results
boot_MSE_ci <- boot.ci(boot_MSE, type = "all", conf = 0.95)
boot_MAE_ci <- boot.ci(boot_MAE, type = "all", conf = 0.95)
boot_RMSE_ci <- boot.ci(boot_RMSE, type = "all", conf = 0.95)

print("95% Confidence Interval for MSE:")
print(boot_MSE_ci)
print("95% Confidence Interval for MAE:")
print(boot_MAE_ci)
print("95% Confidence Interval for RMSE:")
print(boot_RMSE_ci)

# Prediction was performed on the test set and the fitting effect was calculated
test_pred <- predict(xgb_model,as.matrix(test_data$cases))
write.xlsx(test_pred,file = "C:/Users/User/Desktop/ARIMA--X/2013-2021/xgboost/The test set fits the value.xls")
test_mse <- mean((test_pred - as.matrix(test_data$cases))^2)
test_mae <- mean(abs(test_pred - as.matrix(test_data$cases)))
test_rmse <- sqrt(test_mse)
print(paste("MSE:", test_mse))
print(paste("MAE:", test_mae))
print(paste("RMSE:", test_rmse))

data <- data.frame(observed = test_pred, predicted = as.matrix(test_data$cases))
# By specifying type = "all", we can calculate the confidence intervals for the MSE, MAE, and RMSE simultaneously
```

```
boot_MSE <- boot(data, calc_MSE, R = 1000)
boot_MAE <- boot(data, calc_MAE, R = 1000)
boot_RMSE <- boot(data, calc_RMSE, R = 1000)

# Output results
boot_MSE_ci <- boot.ci(boot_MSE, type = "all", conf = 0.95)
boot_MAE_ci <- boot.ci(boot_MAE, type = "all", conf = 0.95)
boot_RMSE_ci <- boot.ci(boot_RMSE, type = "all", conf = 0.95)

print("95% Confidence Interval for MSE:")
print(boot_MSE_ci)
print("95% Confidence Interval for MAE:")
print(boot_MAE_ci)
print("95% Confidence Interval for RMSE:")
print(boot_RMSE_ci)

train_pred<- ts(train_pred, start = c(2013, 1), frequency = 12)
test_pred<- ts(test_pred, start = c(2022, 1), frequency = 12)
# Draw the training set, prediction set and prediction results
plot(data$Date, data$cases, xlim = c(2013, 2023), ylim = c(0, 2500), main = "Forecasts from XGBoost ",
     ylab="Influenza cases",xlab="")
axis(1, at = seq(2013, 2022, 1), labels = seq(2013, 2022, 1))
lines(flu_ts,pch=1, lty=1,col = "grey", lwd = 2)    #    training set
lines(train_pred, lty=1,col = "red", lwd = 2)    # The training set was fitted to the value
lines(test_pred, lty=6,col = "red", lwd = 2)     # Forecast resultslegend("topleft", legend = c("Actual", "XGBoost fitted", "XGBoost forecast"), col = c("grey", "red", "red"),lty=c(1, 1, 6), lwd = 3)
```