Time Series Project

Metin Mirza VATANSEVER

**Time Series Analysis of Microsoft Stock Data (2015–2021)**

This project involves a comprehensive time series analysis of Microsoft stock data using R. The dataset spans from April 2015 to April 2021 and includes daily stock prices such as Open, Close, High, Low, and Volume.

**Tools & Libraries**

- `R`, `fpp2`, `forecast`, `TTR`, `xts`, `zoo`, `ggplot2`, `imputeTS`, `lubridate`, `tseries`

**Objective**

- To apply time series decomposition, forecasting, and indexing techniques on real-world financial data.

- To compare and evaluate different modeling and forecasting methods (Mean, Naïve, Seasonal Naïve, SARIMA).

- To examine seasonality, stationarity, autocorrelation, and structural breaks within the time series.

**Dataset**

For this assignment, I used a dataset from Kaggle that includes Microsoft's stock values between 04.2015-04.2021

Data Address: <https://www.kaggle.com/datasets/vijayvvenkitesh/microsoft-stock-time-series-analysis>

**Definition of yield variables:**

**Date**: Date

**Open**: Opening price of the stock on that date

**High**: Highest value reached by the stock during the day

**Low**: Lowest value reached by the stock during the day

**Close**: Closing price of the stock on that date

**Volume**: Number of stocks traded on that date

A screenshot of a computer

Description automatically generated

• I ran some simple codes to get information about the data and check if the data is ready to work on it.**A screenshot of a computer screen

Description automatically generated**

**Our Purpose and Codes**

**1. Downloading real national or international data (may be related to prices, production) and calculating fixed and compound based indexes related to this data.**

**2. Downloading international monthly or quarterly data from data sources and revealing, purifying, and modeling the time series components related to the data**

**3. Implementing the topics covered since the beginning of the term on the relevant data with explanations and comments**

**Part-1**

**A computer screen with text

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**Part-2**

**A computer screen with text and images

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A black background with white and orange text

AI-generated content may be incorrect.**

• I ran the summary code to evaluate my models and looked at the total error values ​​I got. Root mean square error, mean absolute error etc.

• The residual autocorrelation of Model 1 (0.0003) is lower than Model 2 (-0.1291), indicating that Model 1 models the dependency in the data better.

The residual variance of Model 1 is 5.14, while that of Model 2 is 34.04. A lower variance indicates that the estimates are less scattered and the model fits better.

Conclusion:

Model 1 has lower values ​​in all error measures and its residual variance is also smaller. This indicates that Model 1 is more successful and fits the data better. Model 2 may be useful in certain cases because it includes a seasonal structure (12 periods), but the current results show that Model 1 performs better overall.**A graph showing the growth of a stock market

Description automatically generatedA graph of growth in a line

Description automatically generated with medium confidence**

**Part-3**

**• When adding Part-3 to the report, since it would take a long time to add each code and explain it one by one, I will paste it into the report as pure code and share my general experiences. I am adding the outputs I received outside of the code below.**

**Comment**

**This project allowed me to do a comprehensive study on time series analysis. First, although I had difficulty with the data processing steps, I prepared the data as I wanted. I converted the raw data to suitable formats, checked for missing values ​​and made it suitable for time series analysis.**

**I used various methods to examine the trend, seasonality and structural changes of the time series. I separated the trend and seasonal components of the data with the decomposition method. Apart from this, I performed a more flexible seasonality analysis by performing STL decomposition. During this process, I noticed structural breaks such as the COVID-19 pandemic and tried to purify the data from such anomalies.**

**I used the Dickey-Fuller test for the stationarity analysis in the data and determined that the series was not stationary by interpreting the p-values. I tried to make the series stationary by taking differences and used ACF and PACF graphs to analyze autocorrelation in this process. No matter how hard I tried, I could not completely turn the data into a white noise series. **

**I tried different methods to eliminate seasonality and autocorrelation. I tried to smooth the series with techniques such as seasonal difference and moving average. In addition, I applied data transformation methods such as logarithmic transformation and scaling. But despite trying so many methods, I could not completely remove autocorrelation from my autocorrelated series.**

**I did a lot of data visualization in the project. While doing visualization, I used powerful visualization libraries such as ggplot2 and visualized time series trends, seasonality and lag diagrams at a level that I could understand. I can say that this visualization process played a big role in my understanding of the course.**

**Finally, I did estimation and forecast modeling. I evaluated different estimation approaches by comparing Mean, Naive and Seasonal Naive methods as in PDFs. I applied these models on both logarithmically transformed and raw data. In this way, I realized how enjoyable it is to work with manipulated data.**

**In general, this study was a study where I was able to put my theoretical knowledge on time series analysis into practice and systematically address the challenges I encountered. It allowed me to practice concepts such as seasonality, autocorrelation and stationarity, as well as data visualization and modeling techniques. It advanced my knowledge of time series quite a bit. My "I'll give up" note at the end of the project shows how complex time series analysis can be and how patience it requires. ☺**

**Plots**

Decomposition Graphs

**A graph of different types of growth

Description automatically generated with medium confidence** A graph of a graph of time

Description automatically generated with medium confidence

A graph of different types of growth

Description automatically generated with medium confidence

A graph with red and blue lines

Description automatically generated

Seasonally Adjusted Plot

A graph with lines and numbers

Description automatically generated

3rd Degree Moving Avarage Plot

A graph with lines and numbers

Description automatically generated

10th Degree Moving Avarage Plot

A graph showing the time and the time

Description automatically generated with medium confidence

Seasonal Colored & Polar Plots

A graph of stock data

Description automatically generated A graph showing a diagram of a season

Description automatically generated with medium confidence

Autoplot of Close Variable in Yield

A graph showing the growth of a stock market

Description automatically generated

Seanonality Plots

A graph of different colored lines

Description automatically generatedA diagram of a graph

Description automatically generated with medium confidence

Quartile Plots and Autocorrelation Estimation

A graph showing the stock data

Description automatically generated

Lag-Plot (That I can’t get make sense of)

A graph of a number of different types of objects

AI-generated content may be incorrect.

Autocorrelation Plot for Close Column – Introduction-Development-Conclusion

A graph with lines and numbers

Description automatically generatedA graph with lines and numbers

Description automatically generated

Autocorrelation Plot for Close Column– Introduction-Development-Conclusion Part-2

A graph with lines and numbers

Description automatically generatedA graph with lines and numbers

Description automatically generated

Scaled ACF Plot, Dickey-Fuller and Box L-jung test

A graph with lines and numbers

Description automatically generated

**A black screen with white text

Description automatically generated**A black background with white text

Description automatically generated

I also tried the ACF chart by converting the data to weekly data.

A graph with lines and numbers

Description automatically generated

I Wanted To See The Same ACF Chart DailyA graph showing a sound wave

Description automatically generated

1. Forecast Chart and Model Success Metrics

A graph showing the growth of a stock market

Description automatically generatedA screenshot of a computer

Description automatically generated

2. Forecast Chart and Model Success Metrics

A graph of growth in a line

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated

PURE CODE

install.packages("ggplot2")

library('ggplot2')

install.packages("zoo")

library("zoo")

install.packages("TTR")

library("TTR")

install.packages("xts")

library(xts)

install.packages('fpp2', dependencies = TRUE)

library("fpp2")

library(xts)

library(tseries)

library(zoo)

library(stats)

install.packages("imputeTS")

library(imputeTS)

library(dplyr)

install.packages("lubridate")

library(lubridate)

install.packages("ggfortify")

install.packages("forecast")

library("ggfortify")

library(forecast)

# ts() func 2020-present random veri

sales <- c(0.982716, 1.697239, 0.323146, 1.606725, 0.758019, 1.225429, 1.705847,

0.679039, 0.849232, 1.149105, 0.926340, 1.392482, 1.416663, 1.684588,

1.091457, 0.895126, 1.385147, 1.216367, 1.197553, 0.601648, 0.846092,

1.493991, 0.884929, 1.357743, 0.675743, 1.004754, 1.115058, 1.396485,

0.704323, 1.447149, 1.703308, 1.536081, 0.944853, 1.198320, 1.053133,

1.542146, 1.095249, 0.520796, 1.370186, 0.504466, 1.144440, 1.270736,

1.384385, 0.670116, 1.157045, 1.314007, 0.777445, 0.493667, 1.523826,

0.492337, 1.394343, 1.362239, 1.357225, 0.348570, 0.441050, 1.219929,

0.798493,1.294245,0.926323)

sales\_ts <- ts(sales, start = c(2020, 1), frequency = 12)

print(sales\_ts)

plot.ts(sales\_ts, main = "Monthly Time Serie", ylab = "Sales", xlab = "Year")

logsales\_ts <- log(sales\_ts)

plot.ts(logsales\_ts)

dytime <- ts(sales\_ts, frequency=4, start=c(2010,1))

dycomponents <- decompose(dytime)

print(dycomponents)

dycomponents$seasonal

dycomponents$trend

----------------------------------------------------------------------------------

#lecturedata pdf

births <- scan("http://robjhyndman.com/tsdldata/data/nybirths.dat")

birthstimeseries <- ts(births, frequency=12, start=c(1946,1))

birthstimeseries

birthstimeseriescomponents <- decompose(birthstimeseries)

birthstimeseriescomponents$seasonal

birthstimeseriescomponents <- decompose(birthstimeseries)

birthstimeseriesseasonallyadjusted <- birthstimeseries - birthstimeseriescomponents$seasonal

plot(birthstimeseriesseasonallyadjusted)

plot(birthstimeseriescomponents) #lecturedata

#--------------------------------------------------------------------------------

#My Data

file\_path <- "/Users/metinvs/Desktop/Akademi/Statistics/Present/Time Series/Microsoft\_Stock.csv"

dataset <- read.csv(file\_path)

colnames(dataset) <- c("Date", "Open", "High", "Low", "Close", "Volume")

dataset$Date <- as.Date(dataset$Date, format="%m/%d/%Y")

sum(is.na(dataset$Date))

mcstimeseries <- ts(dataset, frequency=260, start=c(2016))

head(dataset)

summary(dataset)

mcstimeseries2 <- ts(dataset$Close, frequency=260, start=c(2015, 1))

mcstimeseriescomponents <- decompose(mcstimeseries2)

mcstimeseriescomponents$seasonal

mcstimeseriescomponents <- decompose(mcstimeseries2)

mcstimeseriesseasonallyadjusted <- mcstimeseries2 - mcstimeseriescomponents$seasonal

plot(mcstimeseriescomponents)

seassontestdata <- ts(dataset$Close, start = c(2015, 1), frequency = 260)

seasonal\_decomposition <- stl(seassontestdata, s.window = "periodic")

plot(seasonal\_decomposition)

# As we see in the plots there was a seassonal effect in our data. Also we see increasing trend.

#And our data has structural break at 2020 global pandemic. after the pandemic our data looks more unstable

#So we delete after pandemic data

index(mcstimeseries) <- as.Date(index(mcstimeseries))

dataset\_filtered <- mcstimeseries["2015-01-01/2019-12-28"]

dataset\_filtered2 <- ts(dataset\_filtered$Close, frequency=260, start=c(2015, 1))

filtreddatacomponents <- decompose(dataset\_filtered2)

filtreddatacomponents$seasonal

filtreddatacomponents <- decompose(dataset\_filtered2)

dataset\_filteredseasonallyadjusted <- dataset\_filtered2 - filtreddatacomponents$seasonal

plot(filtreddatacomponents)

plot(dataset\_filtered)

plot(dataset)

plot(dataset$Volume) #mydata

plot(dataset$High) #mydata

plot(dataset$Low) #mydata

plot(dataset$Close) #mydata

plot(dataset$Open) #mydata

ggplot(data = dataset, aes(x = Date)) +

geom\_line(aes(y = Close, color = "Close"), size = 1) + # Close çizgisi

geom\_line(aes(y = Open, color = "Open"), size = 1) + # Open çizgisi

labs(title = "Microsoft Stock Prices (Open ve Close)",

x = "Date",

y = "Price",

color = "Price Type") +

scale\_color\_manual(values = c("Close" = "blue", "Open" = "red"))

#Mevsimsellik var dersek seriyi arındırma

# Veriyi yalnızca 'Close' sütunu ile ts formatına dönüştürme

ts\_data <- ts(coredata(mcstimeseries[,"Close"]), start = c(2015, 1), frequency = 260)

seasonal\_decomposition <- stl(ts\_data, s.window = "periodic")

detrended\_data <- seasadj(seasonal\_decomposition)

sales\_tsseasonaladjusted <- dytime - dycomponents$seasonal

plot(sales\_tsseasonaladjusted)

# Mevsimsellik yok dersek seriyi düzleştirme.. Bu HO'da QTR'ler üstündeyse çok mantıklı değil gibi

sales\_tsseasonaladjustedSMA3 <- SMA(sales\_tsseasonaladjusted,n=3)

plot.ts(sales\_tsseasonaladjustedSMA3)

sales\_tsseasonaladjustedSMA10 <- SMA(sales\_tsseasonaladjusted,n=10)

plot.ts(sales\_tsseasonaladjustedSMA10)

#lecturedata

#---------------------------------------------------------------------------------------------------------

install.packages('fpp2', dependencies = TRUE)

library("fpp2")

data(a10)

autoplot(a10)

library(forecast)

ggseasonplot(ts\_data, year.labels = TRUE, year.labels.left = TRUE) + ggtitle("Seasonal Plot: Microsoft Stock Data")

autoplot(mcstimeseries[,"Close"])

#ggseasonplot(ts\_data, polar = TRUE) +

# ylab("$") +

#ggtitle("Polar Seasonal Plot: Microsoft Stock Price Data") +

#scale\_x\_continuous(breaks = seq(1, length(ts\_data), by = 65),

# labels = c("Q1", "Q2", "Q3", "Q4"))

#veri 1511 elemanlı asal sayı olduğu için düzgün biçimde çeyreklere bölemedik. o yüzden quarter grafik yapamadım.

# Zaman serisini kontrol etme

head(ts\_data)

# ggseasonplot ile mevsimsel grafik oluşturma

ggseasonplot(ts\_data, polar = TRUE) +

ylab("$") +

ggtitle("Polar Seasonal Plot: Microsoft Stock Data") +

scale\_x\_continuous(

breaks = seq(1, length(ts\_data), by = 65), # 65 günlük aralıklarla

)

#Similar Data (auscafe)

autoplot(ts\_data)

ggseasonplot(sales\_tsseasonaladjusted, year.labels=TRUE, year.labels.left=TRUE) + ylab("$") + ggtitle("Seasonal plot: Microsoft Stock Data")

ggseasonplot(sales\_tsseasonaladjusted, polar=TRUE) + ylab("$") + ggtitle("Polar seasonal plot: Microsoft Stock Data")

ggsubseriesplot(sales\_tsseasonaladjusted) + ylab("$") + ggtitle("Seasonal subseries plot: Microsoft Stock Data")

#lecturedata

data(elecdemand) > autoplot(elecdemand[,c("Demand","Temperature")], facets=TRUE) + xlab("Year: 2014") +

ylab("") + ggtitle("Half-hourly electricity demand: Victoria, Australia")

#lecturedata (visnights)

autoplot(visnights[,1:5], facets=TRUE) + ylab("Number of visitor nights each quarter (millions)")

autoplot(uschange[,1:4], facets=TRUE) + ylab("US Personal Consumption and Income Growth Rates")

#Another comparison

install.packages('GGally', dependencies = TRUE)

GGally::ggpairs(as.data.frame(ts\_data))

GGally::ggpairs(as.data.frame(visnights[,1:5]))

#delay diagrams comparisons

data(ausbeer)

beer2 <- window(ausbeer, start=1992)

gglagplot(beer2)

data(qcement)

qcement2 <- window(qcement, start=1992)

gglagplot(qcement2)

data(qauselec)

qauselec2 <- window(qauselec, start=1992)

gglagplot(qauselec2)

data(euretail)

euretail2 <- window(euretail, start=1992)

gglagplot(euretail2)

#My delayed data

install.packages("ggfortify")

install.packages("forecast")

library("ggfortify")

library(forecast)

ts\_datadelay <- window(ts\_data, start=2016)

gglagplot(ts\_datadelay)

ts\_datadelay <- window(ts\_data, start=2017)

gglagplot(ts\_datadelay)

ts\_datadelay <- window(ts\_data, start=2019)

gglagplot(ts\_datadelay)

ts\_datadelay <- window(ts\_data, start=2020)

gglagplot(ts\_datadelay)

#Autocorrelationtest

recent\_production <- acf(ts\_data, lag\_max = 9)

recent\_production <- acf(euretail2, lag\_max = 9)

recent\_production <- acf(qcement2, lag\_max = 9)

recent\_production <- acf(qgas2, lag\_max = 9)

recent\_production <- acf(beer2, lag\_max = 9)

recent\_production <- acf(qauselec2, lag\_max = 9)

#veri durağan değil

#Dickley-Fuller Testi

library(tseries)

adf.test(ts\_data)

#test sonucu p-value > 0.05 olduğu için serimiz durapan değildir. şimdi fark alma işlemi yapıcam

ts\_data\_diff <- diff(ts\_data)

acf(ts\_data\_diff, lag.max = 30) # İlk 30 günlük gecikmeleri analiz ettik ve güçlü bir otokorelasyon olduğunu gördük

ts\_data\_diff2 <- diff(ts\_data, differences = 2)

acf(ts\_data\_diff2, lag.max = 30)

library(forecast)

ts\_data\_stl <- stl(ts\_data, s.window = "periodic") # # STL ile ayrıştırma

seasonal\_component <- ts\_data\_stl$time.series[, "seasonal"]

ts\_data\_adjusted <- ts\_data - seasonal\_component

acf(ts\_data\_adjusted, lag.max = 30)

ts\_data\_log <- log(ts\_data)

ts\_data\_diff <- diff(ts\_data\_log)

acf(ts\_data\_diff, lag.max = 30)

ts\_data\_scaled <- scale(ts\_data)

ts\_data\_diff <- diff(ts\_data\_scaled)

acf(ts\_data\_diff, lag.max = 30)

# p-value = 0.01 çıktı.

recent\_production <- acf(ts\_data, lag\_max = 9)

recent\_production <- acf(msctimeseries, lag\_max = 9)

Box.test(ts\_data\_diff, lag = 30, type = "Ljung-Box")

#seride hala otokorelasyon var. pes edicem

ts\_data\_seasonal\_diff <- diff(ts\_data, lag = 7) # Haftalık mevsimsellik farklaması

acf(ts\_data\_seasonal\_diff, lag.max = 30)

##Sonuç olarak Mevsimsellik ve otokorelasyondan arındırmadım....

#same outout different code

ggAcf(euretail2)

ggAcf(qcement2)

ggAcf(qgas2)

ggAcf(beer2)

ggAcf(qauselec2)

ggAcf(ts\_data\_diff2)

#There was no seasonalling for our auscafe data because graph not like a tomb

auscafeds <- window(auscafe, start=1990)

autoplot(auscafeds) + xlab("Year") + ylab("SPP")

ggAcf(auscafeds, lag=48)

#Comparising lecture and random data// Both series has white noise because there was no outlier value

set.seed(30)

y <- ts(rnorm(50))

autoplot(y) + ggtitle("White noise")

set.seed(50)

x <- ts(rnorm(80))

autoplot(y) + ggtitle("White noise")

ggAcf(y)

ggAcf(x)

beer2 <- window(ausbeer,start=1992,end=c(2007,4))

autoplot(beer2) +

autolayer(meanf(beer2, h=11), series = "Mean", PI = FALSE) +

autolayer(naive(beer2, h=11), series = "Naïve", PI = FALSE) +

autolayer(snaive(beer2, h=11), series = "Seasonal naïve", PI = FALSE) +

xlab("Year") +

ylab("Megalitres") +

guides(colour = guide\_legend(title = "Forecast"))

#predict with log data

ts\_dataclose <- window(ts\_data\_log,start=2015,end=c(2020,4))

autoplot(ts\_dataclose) +autolayer(meanf(ts\_dataclose, h=11),series="Mean", PI=FALSE) +

autolayer(naive(ts\_dataclose, h=11),series="Naïve", PI=FALSE) +

autolayer(snaive(ts\_dataclose, h=11),series="Seasonal naïve", PI=FALSE) +

ggtitle("Forecasts for Stock Price") +xlab("Year") + ylab("Stock Price") +

guides(colour=guide\_legend(title="Forecast"))

#predict with normal data

ts\_dataclose <- window(ts\_data,start=2015,end=c(2019,4))

autoplot(ts\_dataclose) +

autolayer(meanf(ts\_dataclose, h=11), series = "Mean", PI = FALSE) +

autolayer(naive(ts\_dataclose, h=11), series = "Naïve", PI = FALSE) +

autolayer(snaive(ts\_dataclose, h=11), series = "Seasonal naïve", PI = FALSE) +

xlab("Year") +

ylab("Close Price") +

guides(colour = guide\_legend(title = "Forecast"))

###naive method more usefull than others

#------------------------------------------------------------

# Part 1-2

# Sabit Esaslı İndeks

base\_price <- dataset$Close[1] # İlk gözlem

dataset$Fixed\_Base\_Index <- (dataset$Close / base\_price) \* 100

# Bileşik Esaslı İndeks

dataset$Price\_Change\_Rate <- c(NA, diff(log(dataset$Close))) # Günlük değişim oranı

dataset$Chained\_Index <- cumprod(1 + ifelse(is.na(dataset$Price\_Change\_Rate), 0, dataset$Price\_Change\_Rate)) \* 100

fiyatlar <- ts\_data

miktarlar <- ts(runif(length(ts\_data), min = 50, max = 150), start = start(ts\_data), frequency = frequency(ts\_data))

baz\_yil <- start(ts\_data)[1]

baz\_fiyat <- window(close, start = baz\_yil, end = baz\_yil)

baz\_miktar <- window(miktarlar, start = baz\_yil, end = baz\_yil)

laspeyres <- sum(baz\_fiyat \* miktarlar) / sum(baz\_fiyat \* baz\_miktar)

paasche <- sum(fiyatlar \* miktarlar) / sum(baz\_fiyat \* miktarlar)

fischer <- sqrt(laspeyres \* paasche)

cat("Laspeyres Endeksi:", laspeyres, "\n")

cat("Paasche Endeksi:", paasche, "\n")

cat("Fischer Endeksi:", fischer, "\n")

# Date değişkenini zaman serisi indeksi olarak ayarla

dataset$Date <- as.Date(dataset$Date)

dataset\_xts <- xts(dataset$Close, order.by = dataset$Date)

# SARIMA

sarima\_model <- auto.arima(dataset\_xts, seasonal = TRUE, stepwise = TRUE, approximation = FALSE)

# Model summary

summary(sarima\_model)

# 30 day predict

forecast\_result <- forecast(sarima\_model, h = 30)

plot(forecast\_result, main = "SARIMA Modeli ile Günlük Tahmin")

lines(daily\_data, col = "blue", lwd = 2, type = "l")

##Seriyi aylık veriye ddönüştürüp çözdüğüm kısım aşağıda. Ben hem günlük hem aylık tahminleme denedim.

dataset\_xts <- xts(dataset$Close, order.by = dataset$Date)

monthly\_avg <- apply.monthly(dataset\_xts, function(x) mean(x, na.rm = TRUE))

# Aylık ortalama zaman serisini bir R ts objesine dönüştür

monthly\_ts <- ts(

as.numeric(monthly\_avg),

start = c(year(index(monthly\_avg)[1]), month(index(monthly\_avg)[1])),

frequency = 12

)

# Modelleme

sarima\_model2 <- auto.arima(monthly\_ts, seasonal = TRUE)

forecast\_result <- forecast(sarima\_model2, h = 12) # 12 aylık tahmin

plot(forecast\_result, main = "SARIMA Modeli ile Aylık Tahmin")

lines(monthly\_ts, col = "blue", lwd = 2, type = "l")

summary(sarima\_model2)