Time Series Modeling and Forecasting of Netflix Stock Prices and Hotel Bookings

Aekank Patel

MA 641: Time Series Analysis
Fall 2025

1 Motivation and Introduction of the Project

Time series analysis is useful in a lot of real world problems such as finance, tourism, and business planning. Many datasets are collected over time and past behavior helps to understand future pattern. Because of this, time series models are important in decision making.

In this project, two different time series datasets are analyzed. One dataset is non-seasonal and the other dataset is seasonal. This is done to understand how modeling changes when seasonality is present or not present in data.

The first dataset is Netflix adjusted closing stock price. The stock prices usually change over time with the trend and random movement, and also they are affected by a lot of external factors. Although precise values cannot be predicted, stock prices forecasting is useful for understanding a general future direction.

The second dataset is hotel bookings data which depends on months and seasons. Hotel demand usually increases during holidays and travel seasons, so it is important to model seasonal behavior correctly.

The objective of this project is to apply Box–Jenkins approach, select suitable ARIMA or SARIMA models and forecast future values for both datasets. The results help in comparing non-seasonal and seasonal time series modeling.

2 Data

All analysis in this project is performed using the R programming language. R is used for data cleaning, visualization, statistical testing, model fitting, and forecasting.

Two datasets are used in this project. One dataset is non-seasonal financial data and the other dataset is seasonal business data.

2.1 Netflix Stock Price Data

The Netflix dataset contains daily adjusted close prices from 2002 onwards [1]. The data is obtained from Kaggle and originally sourced from Yahoo Finance. The adjusted close price is used as it gives more accurate information by accounting for stock splits and dividends.

This dataset represents the historical stock price behavior of Netflix over time.

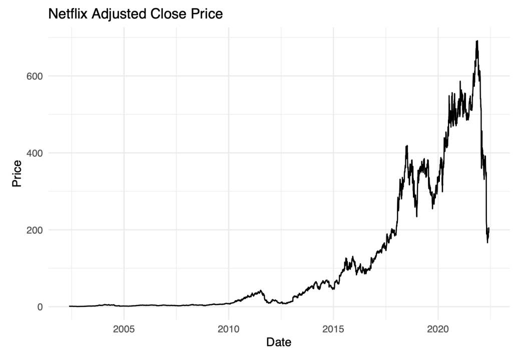


Figure 1: Netflix Adjusted Close Price

The plot in the Figure 1 shows a clear upward trend and increasing variance. There are no missing values in the dataset. Since variance increases with time, logarithmic transformation is applied to stabilize the variance.

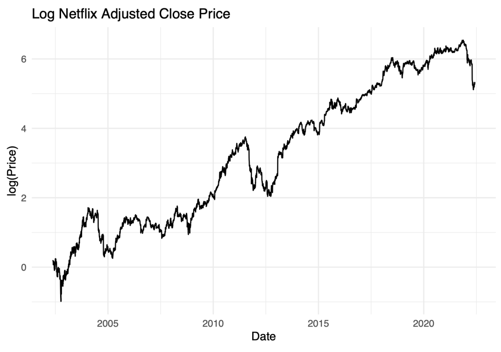


Figure 2: Log Netflix Adjusted Close Price

After log transform in Figure 2, variance looks more stable compared to original series, but upward trend is still present.

No outliers are removed since large movements are expected in stock price data. Differencing is applied later to remove trend and achieve stationarity.

2.2 Hotel Bookings Data

The hotel bookings dataset is obtained from Kaggle [2] and it contains information related to hotel arrival records. The data is aggregated into monthly total bookings for seasonal analysis.

This dataset represents the number of hotel bookings per month across multiple years.

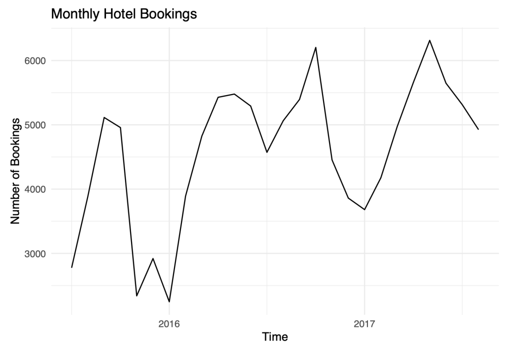


Figure 3: Monthly Hotel Bookings

The series in Figure 3 shows repeating yearly pattern which indicates seasonality.

The monthly plot in Figure 4 shows strong seasonal behavior. No missing values are observed after aggregation. Since seasonality is present, the seasonal differencing is required during modeling.

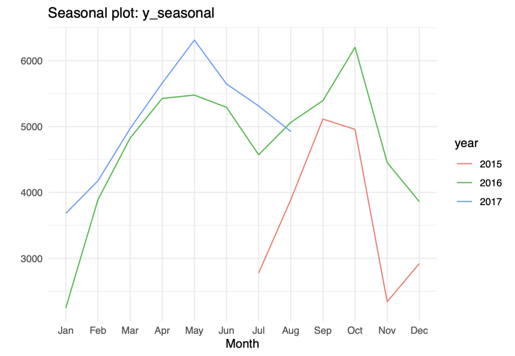


Figure 4: Seasonal Plot of Hotel Bookings

In the above plot, same months show similar booking behavior across different years.

No major outliers are removed because booking counts naturally change due to travel seasons and holidays.

3 Stationarity Analysis (Box–Jenkins Step 1)

For Netflix data, ADF test on log series shows non-stationarity. First differencing is applied to remove trend.

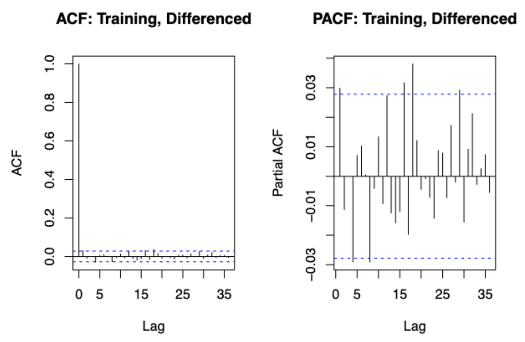


Figure 5: First Differenced Log Netflix Series

The differenced series in Figure 5 fluctuates around zero indicating stationarity.

For hotel bookings data, the series shows strong seasonal pattern. ADF test indicates stationarity after accounting for seasonality. Since the data has yearly seasonality, seasonal differencing is required before fitting SARIMA model.

4 Box–Jenkins Model Building

Box–Jenkins methodology is used for both datasets. First, stationarity is checked by using plots and ADF test. Based on stationarity results, the differencing is applied to remove trend and seasonality.

After differencing, the ACF and PACF plots are examined to identify possible AR and MA terms. For Netflix data, multiple ARIMA models such as ARIMA(0,1,1), ARIMA(1,1,1), and ARIMA(1,1,2) were considered. For hotel bookings data, seasonal ARIMA models are considered due to yearly seasonality.

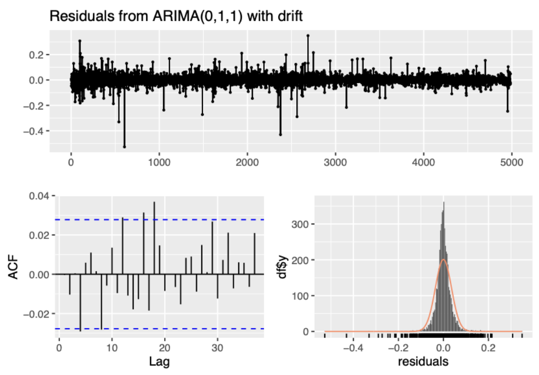
All candidate models are fitted and compared using information criterias such as AIC and BIC. Residual diagnostics such as residual plots and Ljung–Box test are used to check model assumptions. The final model is selected based on lowest AIC value and satisfactory residual behavior.

The selected ARIMA and SARIMA models are then used for forecasting future values.

5 Model Identification and Statistical Conclusions

5.1 Netflix ARIMA Model

Multiple ARIMA models are tested based on Box–Jenkins approach. Based on the AIC values and residual diagnostics, ARIMA(0,1,1) with drift is selected as the final model.



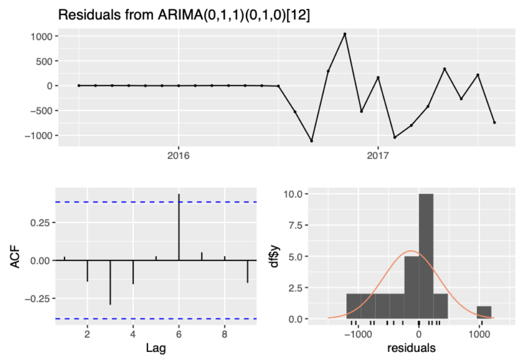
```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1) with drift
## Q* = 10.613, df = 9, p-value = 0.3032
##
## Model df: 1.   Total lags used: 10
##
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1) with drift
## Q* = 10.613, df = 9, p-value = 0.3032
```

Figure 6: Residual Diagnostics for ARIMA(0,1,1)

In the Figure 6 residuals move around zero and no clear pattern is visible. ACF plot does not show strong autocorrelation. Ljung–Box test from diagnostics suggests residuals are mostly uncorrelated, so model fit is reasonable.

5.2 Hotel Bookings SARIMA Model

For the seasonal data, SARIMA(0,1,1)(0,1,0)[12] model is selected since it captures both trend and yearly seasonality.



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)(0,1,0)[12]
## Q* = 4.1593, df = 4, p-value = 0.3849
##
## Model df: 1.   Total lags used: 5
```

Figure 7: Residual Diagnostics for SARIMA Model

In the Figure 7 residuals move around zero and no clear pattern is seen. ACF plot does not show strong autocorrelation and most bars are inside limits. Ljung–Box test suggests residuals are mostly uncorrelated and model fit is reasonable.

6 Forecasting

6.1 Netflix Forecast

Forecasting is performed to understand the future behavior of Netflix stock price based on past information. The ARIMA(0,1,1) model selected in the previous section is used for this purpose. The model is fitted on the full available log-transformed price series and then used to generate forecasts beyond the observed data.

In this analysis, forecasting is done in terms of steps ahead. One step represents one future observation forward in the series. Since the time series is modeled without explicit calendar dates, the x-axis of the forecast plot represents steps and not actual time such as days or months.

Figure 8 shows the future forecast of Netflix stock price. The forecast is presented on the original price scale for better interpretation. The plot displays only the future forecast values, making it easier to visualize the expected movement of the series. The forecast appears as a smooth increasing line, which is expected from an ARIMA model with drift.

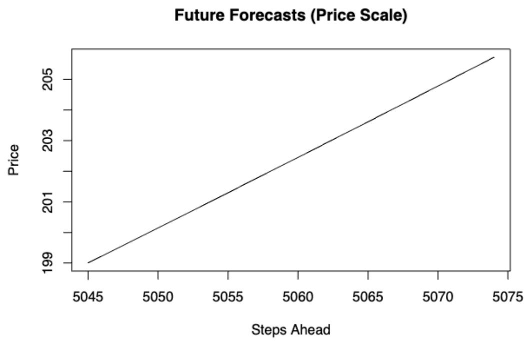


Figure 8: Future Forecast of Netflix Stock Price (Price Scale)

The forecast shows a steady upward movement in future steps. The model captures the overall trend but does not reflect short-term fluctuations, which is expected for ARIMA based forecasting. Forecast uncertainty increases as the steps move further ahead.

Overall, the future forecast provides an estimate of the expected direction of Netflix stock prices based on historical trend. Sudden market changes are not captured, but the model gives a reasonable long-term projection.

6.2 Hotel Bookings Forecast

A 12-month forecast is generated using the selected SARIMA model. The forecast is produced after fitting the model on the full seasonal time series.

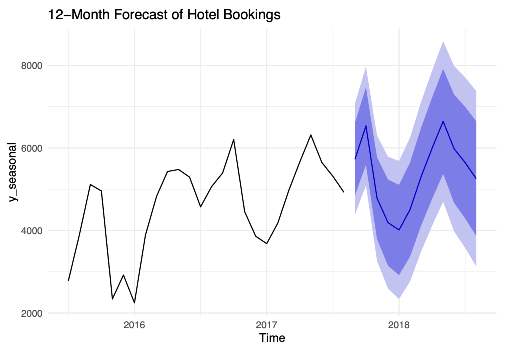


Figure 9: 12-Month Forecast of Hotel Bookings

In the Figure 9 forecast keeps the seasonal pattern seen in past data. High booking months and low booking months are repeated in the forecast. This shows the model captures seasonality in a reasonable way.

7 Conclusion

In this project, non-seasonal and seasonal time series data are analyzed using Box–Jenkins approach. The Netflix stock price data shows strong trend with random changes, while hotel bookings data clearly shows seasonality over time.

For Netflix data, ARIMA(0,1,1) with drift works in a reasonable way. The model follows the overall trend but it is not able to capture sudden large changes in stock price. For hotel bookings data, SARIMA(0,1,1)(0,1,0)[12] model is selected and it captures the yearly seasonal pattern in bookings.

The results show that it is important to first check trend and seasonality before choosing a model. Using correct model based on data behavior gives better forecasting results.

Overall, this project shows that Box–Jenkins method can be applied to real datasets and can be used to understand future movement of time series data, even though exact prediction is difficult.

References

1. Netflix Stock Price Dataset (2002–2022), Kaggle. `https://www.kaggle.com/datasets/meetnagadia/netflix-stock-price-data-set-20022022`

2. Hotel Booking Demand Dataset, Kaggle. `https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand`

A Appendix

The following R code is used for data preprocessing, time series modeling, diagnostics, and forecasting in this project.

A.1 R Code Used for Analysis of Netflix Stock Price

```r
# NFLX

# Loading the libraries
library(tidyverse)
library(lubridate)
library(forecast)
library(tseries)
library(ggplot2)
library(ggfortify)
library(tseries)

# Step 1: Load and prepare data
# Read CSV
nflx_raw <- read.csv("NFLX.csv")

# Convert Date column to Date type
nflx_raw$Date <- as.Date(nflx_raw$Date)

# Sort by Date (just to be safe)
nflx_raw <- nflx_raw %>% arrange(Date)

# Choose the variable to model (Adj Close is best)
nflx <- nflx_raw %>%
  select(Date, Adj.Close = Adj.Close)

# Quick check
str(nflx)
head(nflx)
summary(nflx$Adj.Close)

# Step 2: Basic Time Series Plot (EDA)
# Line plot of Adjusted Close over time
ggplot(nflx, aes(x = Date, y = Adj.Close)) +
  geom_line() +
  labs(title = "Netflix Adjusted Close Price", x = "Date", y = "Price") +
  theme_minimal()

nflx <- nflx %>%
  mutate(LogAdj = log(Adj.Close))

ggplot(nflx, aes(x = Date, y = LogAdj)) +
  geom_line() +
  labs(title = "Log Netflix Adjusted Close Price", x = "Date", y = "log(Price)") +
  theme_minimal()

# Step 3: Create a Time-Series Object
# Extract the log series as a numeric vector
y_log <- nflx$LogAdj
```

```
# Create a ts object with frequency = 1
y_ts <- ts(y_log)   # frequency defaults to 1

# Step 4: Stationarity Check (Plots + ADF)
# i. Plot + ACF/PACF of original series
# Plot the series
autoplot(y_ts) +
  labs(title = "Log Netflix Price (y_ts)", x = "Time", y = "log(Price)") +
  theme_minimal()

# ACF and PACF
par(mfrow = c(1, 2))
acf(y_ts, main = "ACF of y_ts")
pacf(y_ts, main = "PACF of y_ts")
par(mfrow = c(1, 1))

# ii. Augmented Dickey{Fuller (ADF) Test
adf.test(y_ts)

# As p-value=0.5193>0.05 so we proceed with differencing

# Step 5: Differencing to Achieve Stationarity
# i. First Difference
y_diff1 <- diff(y_ts, differences = 1)

autoplot(y_diff1) +
  labs(title = "First Difference of y_ts", x = "Time", y = "Diff log(Price)") +
  theme_minimal()

par(mfrow = c(1, 2))
acf(y_diff1, main = "ACF of First Difference")
pacf(y_diff1, main = "PACF of First Difference")
par(mfrow = c(1, 1))

adf.test(y_diff1)

# As p-value=0.01<0.05 and as ACF/PACF looks stationary, so we use d=1. So, we can assume ARIMA(p,d=1,q)

# Step 6: Train{Test Split
n <- length(y_ts)
h <- 60  # forecast horizon for evaluation

y_train <- window(y_ts, end = n - h)
y_test  <- window(y_ts, start = n - h + 1)

length(y_train)
length(y_test)

# Step 7: Box{Jenkins: Model Identification & Estimation
# i. ACF/PACF on differenced training data
y_train_diff1 <- diff(y_train, differences = 1)

par(mfrow = c(1, 2))
acf(y_train_diff1, main = "ACF: Training, Differenced")
pacf(y_train_diff1, main = "PACF: Training, Differenced")
par(mfrow = c(1, 1))
```

```
# Step 8: Automatic Model Suggestion (auto.arima)
fit_auto <- auto.arima(
  y_train,
  d = 1,
  seasonal = FALSE,
  stepwise = FALSE,
  approximation = FALSE
)

fit_auto


# We get ARIMA (0,1,1) with drift and AIC=-19026.71

# Step 9: Fit a Few Candidate ARIMA Models
# Auto-selected model
fit1 <- fit_auto

# Manual models examples
fit2 <- Arima(y_train, order = c(1, 1, 1))
fit3 <- Arima(y_train, order = c(2, 1, 1))
fit4 <- Arima(y_train, order = c(1, 1, 2))

# Compare AIC/BIC
AIC(fit1, fit2, fit3, fit4)
BIC(fit1, fit2, fit3, fit4)

# Step 10: Residual Diagnostics
# Diagnostics for all models: fit1, fit2, fit3, fit4
models <- list(
  fit1 = fit1,
  fit2 = fit2,
  fit3 = fit3,
  fit4 = fit4
)

for (name in names(models)) {
  cat("Diagnostics for", name, "\n")

  m <- models[[name]]

  # Residual ACF
  acf(residuals(m), main = paste("ACF -", name))

  # Ljung-Box test
  print(Box.test(residuals(m), lag = 20, type = "Ljung-Box"))

  # Combined diagnostic plots + test
  print(checkresiduals(m))
}

# Step 11: Forecasting on Test Set (Evaluation)
fit_best <- fit1

# Forecast h steps ahead (same length as y_test)
```

```
fc <- forecast(fit_best, h = h)

# Plot forecast vs actual (log scale)
autoplot(fc) +
  autolayer(y_test, series = "Actual") +
  labs(title = "Forecast vs Actual (Log Price)", x = "Steps") +
  theme_minimal()

# Accuracy measures on test set
accuracy(fc, y_test)

# Step 12: Forecasting the Future (Full Data)
# Fitting the best model on full series and forecast into the future (next 30 days/steps).
fit_final <- Arima(y_ts, model = fit_best)

fc_future <- forecast(fit_final, h = 30)

autoplot(fc_future) +
  labs(title = "Future Forecasts (Log Price)", x = "Steps", y = "log(Price)") +
  theme_minimal()
# Fitting forecasts back on original price scale:
# approximate back-transform (assuming log-normal)
future_vals <- exp(fc_future$mean)

plot(future_vals, type = "l",
     main = "Future Forecasts (Price Scale)",
     xlab = "Steps Ahead", ylab = "Price")
```

A.2 R Code Used for Analysis of Hotel Booking Price

```
library(dplyr)
library(lubridate)
library(forecast)
library(ggplot2)
library(tseries)

# Step 1: Load dataset
data <- read.csv("hotel_bookings.csv", stringsAsFactors = FALSE)

# Step 2: Create proper Date column
data$arrival_date <- as.Date(
  paste(data$arrival_date_year,
        data$arrival_date_month,
        data$arrival_date_day_of_month),
  format = "%Y %B %d"
)

# Step 3: Aggregate to MONTHLY data (Seasonal series)

monthly_bookings <- data %>%
  mutate(month = floor_date(arrival_date, "month")) %>%
  group_by(month) %>%
  summarise(bookings = n()) %>%
  arrange(month)
```

```
# Step 4: Convert to time series
start_year  <- year(min(monthly_bookings$month))
start_month <- month(min(monthly_bookings$month))

y_seasonal <- ts(monthly_bookings$bookings,
                 start = c(start_year, start_month),
                 frequency = 12)

# Step 5: Descriptive analysis
autoplot(y_seasonal) +
  labs(title = "Monthly Hotel Bookings",
       x = "Time",
       y = "Number of Bookings") +
  theme_minimal()

summary(y_seasonal)

ggseasonplot(y_seasonal) + theme_minimal()
ggsubseriesplot(y_seasonal) + theme_minimal()

# Step 6: Stationarity check
adf.test(y_seasonal)

# Step 7: Fit SARIMA model
fit_sarima <- auto.arima(y_seasonal, seasonal = TRUE)
summary(fit_sarima)

# Step 8: Residual diagnostics
checkresiduals(fit_sarima)

# Step 9: Forecasting
fc <- forecast(fit_sarima, h = 12)

autoplot(fc) +
  labs(title = "12-Month Forecast of Hotel Bookings") +
  theme_minimal()
```