Analysis and Forecasting of Skechers U.S.A., Inc. (SKX) Stock Prices

Univariate Time Series Analysis

- **Time Period Covered by the Data:** November 26, 2020 - February 9, 2024 - daily data
- **Date of Download:** February 15, 2024
- **Access Link:** https://finance.yahoo.com/quote/SKX/history?period1=1606348800&period2=1700524800&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true

Created by: Boglárka Póra

```
rm(list = ls())
graphics.off()
```

The first step is to install the necessary packages and then import these libraries so that they are available in the environment.

```
install.packages('strucchange')
install.packages('changepoint')
install.packages('fpp3')
install.packages('tidyverse')
install.packages('rsample')
install.packages('forecast')
install.packages('readr')
install.packages('ggplot2')
install.packages('plotly')
install.packages('stats')
install.packages('rsample')
install.packages('gridExtra')
install.packages('aTSA')
install.packages('urca')
install.packages('reshape')
install.packages('dplyr')
install.packages('mFilter')
install.packages('timetk')
install.packages('ggpubr')
install.packages('tidymodels')
install.packages('tidyquant')
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

also installing the dependencies 'zoo', 'sandwich'


Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

also installing the dependencies 'numDeriv', 'quadprog', 'warp', 'BH', 'distributional', 'progressr', 'ggdist', 'slider', 'anytime',


Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```

```
library(strucchange)
library(changepoint)
library(fpp3)
library(tidyverse)
library(rsample)
library(forecast)
library(readr)
library(ggplot2)
library(plotly)
library(stats)
library(rsample)
library(gridExtra)
library(aTSA)
library(urca)
library(reshape)
library(dplyr)
library(mFilter)
library(timetk)
library(ggpubr)
library(tidymodels)
library(tidyquant)
```

Loading, Transforming, and Plotting Time Series Data

```
skx <-  tq_get("SKX",
               from = "2020-11-26",
               to = "2024-02-10") %>%
  dplyr::select(date, adjusted)
head(skx)
```

```
Error in tq_get("SKX", from = "2020-11-26", to = "2024-02-10") %>% dplyr::select(date, : could not find function "%>%"
Traceback:
```

```
tail(skx)
```

```
Error in eval(expr, envir, enclos): object 'skx' not found
Traceback:

1. tail(skx)
```

Then, they need to be converted into a tsibble object: the [1D] indicates that the time series contains daily data.

```
skx <- skx %>%
  as_tsibble(index = date, key = adjusted)
head(skx)
```

The next step is to visualize the trends of the different variables, explore the data more deeply, and possibly clean the data.

Plotting the adjusted closing price trend:

```
ggplot(skx, aes(x = date)) +
  geom_line(aes(y = adjusted, color = "Adj Close"), linetype = "solid") +
  labs(title = "The trend of SKX stock's adjusted closing price between 2020 and 2024",
       x = "Timestamp",
       y = "Adjusted Closing Price") +
  scale_color_manual(values = c("Adj Close" = "#630742")) +
  theme_minimal()
```

```
skx <- skx %>%
  dplyr::select(date, adjusted) %>%
  `colnames<-`(., c("Date", "Price"))
head(skx)
```

The next step is to check for any missing values in the dataset.

```
any_na <- anyNA(skx)
any_na
```

Since the value of any_na is FALSE, there are no missing values.

However, there are missing days in the daily data because the stock market is closed on weekends. These dates do not appear in the dataset, so there won't be any NA values.

Nevertheless, we cannot leave it like this. We need to add the weekend dates (to avoid gaps) and then fill in the values for those days using an appropriate method.

```
full_date_sequence <- seq(min(skx$Date), max(skx$Date), by = "days")
complete_dates <- tsibble(Date = full_date_sequence)
skx <- merge(skx, complete_dates, by = "Date", all = TRUE)
head(skx)
```

```
any_na <- anyNA(skx)
any_na
```

Now it is clear that there are missing values.

On non-trading days, we will fill in the adjusted closing price with the last known value.

```
skx <- skx %>% tidyr::fill(Price, .direction= "down")
head(skx)
```

Now, it needs to be converted back into a tsibble.

```
skx <- skx %>%
  as_tsibble(index = Date)
head(skx)
```

```
anyNA(skx)
```

```
summary(skx)
```

Here we can see the lagged values of the closing price: it shows how each data point relates to the previous data points (with different days represented by different colors).

```
skx %>% gg_lag(Price, geom = 'point')
```

Since we obtain a recognizable pattern and a linear trend regardless of the lag considered, we can conclude that there is autocorrelation among the time series values.

A positive autocorrelation is observed, meaning that high values are generally followed by high values, and low values by low values.

Next, we'll examine the trend line using the moving averages method.

One version is the 5-MA, and the other is the 12-MA, where the numbers indicate how many of the most recent data points are averaged by the model.

Both are examples of the Simple Moving Average (SMA).

```
skx <- skx %>%
  mutate(`5-MA` = slider::slide_dbl(Price, mean,
                                  .before = 2, .after = 2, .complete = TRUE))
```

```
skx %>%
  drop_na() %>%
  autoplot(Price) +
  geom_line(aes(y = `5-MA`), colour = '#D550BF') +
  labs(title = "The trend of SKX stock's adjusted closing price between 2020 and 2024 + 5-MA", y = 'Adjusted Closing Price', x = '')
```

```
head(skx)
```

The 5-day moving average shows a short-term smoothed trend, filtering out daily fluctuations, and is sensitive to short-term changes in the time series.

```
skx <- skx %>%
  mutate(`12-MA` = slider::slide_dbl(Price, mean,
                                   .before = 5, .after = 6, .complete = TRUE))
```

```
skx %>%
  drop_na() %>%
  autoplot(Price) +
  geom_line(aes(y = `12-MA`), colour = '#4BA68F') +
  labs(title = "The trend of SKX stock's adjusted closing price between 2020 and 2024 + 12-MA", y = 'Adjusted Closing Price', x = '')
```

```
head(skx)
```

The 12-day moving average shows a slightly longer-term smoothed trend compared to the 5-day moving average, representing a more sustainable trend.

```
skx %>%
  drop_na() %>%
  autoplot(Price) +
  geom_line(aes(y = `5-MA`, color = "5-MA"), linetype = "solid") +
  geom_line(aes(y = `12-MA`, color = "12-MA"), linetype = "solid") +
  labs(title = "The trend of SKX stock's adjusted closing price between 2020 and 2024", y = 'Adjusted Closing Price', x = '') +
  scale_color_manual(values = c("5-MA" = '#D550BF', "12-MA" = '#4BA68F'))
```

We can see from the chart that the 5-MA and 12-MA move closely together, which suggests relative stability.

The curves of the 12-MA are smoother, indicating a more stable trend.

Forecasting with Exponential Smoothing

The first method we can use to forecast the values of the time series is exponential smoothing.

The ETS command selects the appropriate exponential smoothing model based on information criteria.

We will initialize the model and then plot it.

```
expsim <- skx %>%
  model(ETS(Price))
```

```
components(expsim) %>%
  autoplot() +
  labs(title = 'ETS Components')
```

```
report(expsim)
```

ETS(A,N,N) tells us that the model has an additive error term, and it does not account for trend or seasonality.

An alpha value of 0.9343528 is high, indicating that the most recent observations are given significant weight.

The AIC, AUCc, and BIC values provide a basis for model comparison; the smaller these values, the better the model.

Key Statistics:

```
glance(expsim)
```

Examination of Errors:

```
expsim %>% gg_tsresiduals()
```

The distribution of the residuals is normal, with an outlier around the value of 0.

Generating and Plotting the Forecast:

```
expsimf <- expsim %>%
  forecast::forecast(h = 7)
expsimf
```

```
expsimf %>%
  filter(as.Date(Date, format = '%Y.%m.%d') >= as.Date('2023-01-01')) %>%
  autoplot(skx) +
  geom_line(aes(y = Price), color = '#630742') +
  labs(title = "The trend of SKX stock's adjusted closing price between 2020 and 2024", y = 'Adjusted Closing Price', x = '')
```

Examination of Change Points and Structural Breaks

The next task is to examine the change points and structural breaks.

We need to determine if any changes occurred in the time series during the examined period.

First, we will examine changes in growth averages using the PELT method.

```
m_skx <- skx %>%
  dplyr::select(Date, Price) %>%
  mutate(growth = 100 * ((Price / lag(Price)) - 1)) %>%
  drop_na() %>%
  pull(growth) %>%
  cpt.mean(., penalty = 'SIC', method = 'PELT')
```

```
plot(m_skx, type = 'l', col = '#4BA68F', cpt.col = '#630742', xlab = '', ylab = '', cpt.width = 1)
```

```
cpts(m_skx)
```

```
skx %>%
  dplyr::select(Date, Price) %>%
  mutate(growth = 100 * ((Price / lag(Price)) - 1)) %>%
  drop_na() %>%
  slice(cpts(m_skx))
```

In the case of a significant number of observations, changes in the average are detected, indicating that there are significant shifts in the growth pattern at many time points.

Each of these change points represents a structural change in the time series.

Next, we will identify the change points in variance.

```
v_skx <- skx %>%
  dplyr::select(Date, Price) %>%
  mutate(growth = 100 * ((Price / lag(Price)) - 1)) %>%
  drop_na() %>%
  pull(growth) %>%
  cpt.var(., penalty = 'SIC', method = 'PELT')
```

```
plot(v_skx, type = 'l', col = '#4BA68F', cpt.col = '#630742', xlab = '', ylab = '', cpt.width = 1)
```

```
cpts(v_skx)
```

```
skx %>%
  dplyr::select(Date, Price) %>%
  mutate(growth = 100 * ((Price / lag(Price)) - 1)) %>%
  drop_na() %>%
  slice(cpts(v_skx))
```

In the case of variance, there are significantly fewer change points, indicating fewer structural changes.

Lastly, we will conduct a joint examination of both the mean and variance.

```
mv_skx <- skx %>%
  dplyr::select(Date, Price) %>%
  mutate(growth = 100 * ((Price / lag(Price)) - 1)) %>%
  drop_na() %>%
  pull(growth) %>%
  cpt.meanvar(., penalty = 'SIC', method = 'PELT')
```

```
plot(mv_skx, type = 'l', col = '#4BA68F', cpt.col = '#630742', xlab = '', ylab = '', cpt.width = 1)
```

```
cpts(mv_skx)
```

```
skx %>%
  dplyr::select(Date, Price) %>%
  mutate(growth = 100 * ((Price / lag(Price)) - 1)) %>%
  drop_na() %>%
  slice(cpts(mv_skx))
```

The next step is to perform structural break tests, identifying instances where changes occur in the regression coefficients.

Analysis of the structural breaks in the price trend using the Quandt Likelihood Ratio Test (QLR):

The QLR hypothesis framework:

H0: There is no structural break in the time series.

H1: There is a structural break in the time series.

```
price <- skx %>%
  dplyr::select(Date, Price) %>%
  mutate(growth = 100 * ((Price / lag(Price)) - 1), grow_lag = lag(growth)) %>%
  drop_na()
head(price)
```

Growth represents the value of growth, while grow_lag is its lagged value by one period.

```
price_qlr <- Fstats(growth ~ grow_lag, data = price)
breakpoints(price_qlr)
```

```
skx %>%
  dplyr::select(Date, Price) %>%
  mutate(growth = 100 * ((Price / lag(Price)) - 1), grow_lag = lag(growth)) %>%
  drop_na() %>%
  slice(price_qlr$breakpoint)
```

A breakpoint occurs at observation 238.

```
sctest(price_qlr, type = 'supF')
```

The p-value for the structural break is 0.9309, which is greater than the 0.05 significance level. Therefore, we cannot reject the null hypothesis, meaning we cannot prove that there is a structural break in the time series.

```
plot(price_qlr)
```

We can see this on the chart as well; the values do not even approach the drawn red boundary line.

Analysis of the structural breaks in the price trend using the Bai and Perron structural break test (BP):

```
price_bp <- breakpoints(growth ~ grow_lag, data = price, breaks = 5)
summary(price_bp)
```

```
skx %>%
  dplyr::select(Date, Price) %>%
  mutate(growth = 100 * ((Price / lag(Price)) - 1), grow_lag = lag(growth)) %>%
  drop_na() %>%
  slice(price_bp$breakpoint)
```

By examining up to 5 breakpoints, we can see above the indices of the observations where breakpoints are detected.

```
plot(price_bp, breaks = 5)
```

Since the BIC does not decrease at all (its minimum value is at 0), we can conclude that no structural break can be determined according to this test either.

Analysis of the structural breaks in the price trend using the CUSUM test:

```
price_cusum <- efp(growth ~ grow_lag, data = price, type = 'OLS-CUSUM')
```

```
skx %>%
  dplyr::select(Date, Price) %>%
  mutate(growth = 100 * ((Price / lag(Price)) - 1), grow_lag = lag(growth)) %>%
  drop_na() %>%
  slice(price_cusum$datastp)
```

```
plot(price_cusum)
```

According to the CUSUM test, there are no breaks in the data that exceed the critical values marked by the red lines, so this test also fails to identify any structural breaks.

Stationarity Analysis and Autocorrelation Testing

Next, we will examine whether the time series is stationary or not.

```
skx %>% gg_tsdisplay(Price)
```

Analysis of ACF and PACF:

```
skx %>%
  ACF(Price, lag_max = 20)
```

High ACF values indicate strong autocorrelation and a strong dependence between different time points.

Since the autocorrelations for the lags are large and positive, we can conclude that the data has a trend component.

```
p1 <- skx %>%
  ACF(Price, lag_max = 20) %>% autoplot()

p2 <- skx %>%
  PACF(Price, lag_max = 20) %>% autoplot()

grid.arrange(p1, p2, nrow = 1)

skx %>%
  plot_acf_diagnostics(Date, Price, .lags = 15, .interactive = F)
```

According to the ACF plot, autocorrelation decreases linearly as the number of lags increases, with observations that are temporally distant still being correlated with each other.

In the PACF plot, the autocorrelation between observations is very strong at the first lag, but it suddenly drops for subsequent lags, indicating a significant initial effect that diminishes quickly.

We test the values of the autocorrelation function using the Ljung-Box test developed by Ljung and Box.

The hypothesis framework for the Ljung-Box test is:

H0: There is no autocorrelation in the time series.

H1: There is autocorrelation in the time series.

```
skx %>%
  pull(Price) %>%
  Box.test(lag = 12, type = 'Ljung-Box')
```

The p-value is less than the 0.05 significance level, so we reject the null hypothesis, indicating that there is significant autocorrelation present.

Next, we will perform unit root tests to determine whether the time series is stationary or not (based on the plot and autocorrelation values, it is already possible that the series may be non-stationary).

The hypothesis framework for the ADF and PP unit root tests is:

H0: The time series is non-stationary.

H1: The time series is stationary.

The hypothesis framework for the KPSS unit root test is:

H0: The time series is stationary.

H1: The time series is non-stationary.

Augmented Dickey-Fuller Test (ADF):

```
skx %>%
  pull(Price) %>%
  aTSA::adf.test()
```

The p-value is greater than 0.05, so we cannot reject the null hypothesis. Therefore, according to the ADF test, the time series is non-stationary and has a unit root.

Phillips-Perron Test (PP):

```
skx %>%
  pull(Price) %>%
  aTSA::pp.test()
```

```
skx %>%
  features(Price, unitroot_pp)
```

The p-value is greater than 0.05, so we cannot reject the null hypothesis. Therefore, according to the PP test, the time series is non-stationary and has a unit root.

KPSS Test: For this test, the null and alternative hypotheses are reversed compared to other tests.

```
skx %>%
  features(Price, unitroot_kpss)
```

The p-value is less than 0.05, so we can reject the null hypothesis. Therefore, according to the KPSS test, the time series is non-stationary and has a unit root.

There is also another test, the Zivot-Andrews test:

```
za_skx <- skx %>%
  pull(Price) %>%
  ur.za(., model = 'both', lag = 1)
summary(za_skx)
```

The test statistic value is greater than the critical values, so we accept the null hypothesis. This means the time series is non-stationary and has a unit root.

A potential breakpoint is indicated at observation 458, so we will examine that point.

```
skx %>% slice(458)
```

```
plot(za_skx)
```

According to all four conducted unit root tests, the time series is non-stationary.

This implies that the time series needs to be transformed to achieve stationarity.

One method to do this is by differencing the time series.

```
skx %>%
  features(Price, unitroot_ndiffs)
```

One differencing is required to make the time series stationary.

```
skx <- skx %>%
  mutate(dif_price = difference(Price))
head(skx)
```

```
a1 <- ggplot(skx, aes(x = Date, y = Price)) +
  geom_line(color = 'deepskyblue') +
  labs(title = "The trend of SKX stock's adjusted closing price between 2020 and 2024", y = 'Adjusted Closing Price', x = '')

a2 <- ggplot(skx, aes(x = Date, y = dif_price)) +
  geom_line(color = 'deepskyblue') +
  labs(title = 'The trend of the differenced adjusted closing price of SKX stock between 2020 and 2024', y = 'Adjusted Closing Price', x

ggarrange(a1, a2, ncol = 1, nrow = 2)
```

Now that we have made the time series stationary, we need to perform the same tests as before to verify this.

ACF and PACF:

```
skx %>%
  gg_tsdisplay(dif_price)
```

```
p3 <- skx %>%
  ACF(dif_price, lag_max = 20) %>%
  autoplot()

p4 <- skx %>%
  PACF(dif_price, lag_max = 20) %>%
  autoplot()

grid.arrange(p3, p4, nrow = 1)

skx %>%
  plot_acf_diagnostics(Date, dif_price, .lags = 15, .interactive = F)
```

Testing for Autocorrelation:

```
skx %>%
  pull(dif_price) %>%
  Box.test(lag = 12, type = 'Ljung-Box')
```

The p-value is greater than 0.05, so we retain the null hypothesis. There is no autocorrelation in the differenced time series.

ADF Test:

```
skx %>%
  pull(dif_price) %>%
  aTSA::adf.test()
```

The p-value is less than 0.05, so we can reject the null hypothesis. Therefore, the time series is stationary and no longer has a unit root.

PP Test:

```
skx %>%
  pull(dif_price) %>%
  aTSA::pp.test()
```

```
skx %>%
  features(dif_price, unitroot_pp)
```

The p-value is less than 0.05, so we can reject the null hypothesis. This means the time series is stationary and no longer has a unit root.

KPSS Test:

```
skx %>%
  features(dif_price, unitroot_kpss)
```

The p-value is greater than 0.05, so we cannot reject the null hypothesis. Therefore, the time series is stationary and no longer has a unit root.

ZA Test:

```
za_skx2 <- skx %>%
  pull(dif_price) %>%
  ur.za(., model = 'both', lag = 1)

summary(za_skx2)
```

The test statistic value is smaller than the critical values, so we reject the null hypothesis. Therefore, the time series is stationary and does not have a unit root.

According to all four tests, our time series is now stationary.

Next, we will test another forecasting method, starting with forecasting the price using a simple moving average.

Forecasting with Simple Moving Average

Creating Training and Test Sets:

```
rate <- initial_time_split(skx, prop = 0.8)
```

```
train <- training(rate)
dim(train)
```

```
test <- testing(rate)
dim(test)
```

Next, we will set up the moving average models whose results we wish to examine.

We will use the ARIMA model; however, ARIMA(0,0,q) is equivalent to the MA(q) model.

```
models_ma <- train %>%
  model(ma = ARIMA(Price ~ pdq(p = 0, d = 0, q = 1:3)),
        masearch = ARIMA(Price ~ pdq(p = 0, d = 0), stepwise = FALSE, approximation = FALSE))
```

```
glance(models_ma)
```

```
models_ma %>% select(ma) %>% report()
```

```
models_ma %>% select(masearch) %>% report()
```

```
models_ma %>% pivot_longer(everything(), names_to = 'Model name')
```

```
glance(models_ma) %>% arrange(AICc) %>% select(.model:BIC)
```

```
models_ma %>% select(ma) %>% coef()
```

The equation for MA(3) / SARIMA(0,0,3)(2,0,0)[7] is:

$y(t)=\epsilon(t)+0.9434\cdot\epsilon(t-1)+0.8267\cdot\epsilon(t-2)+0.4356\cdot\epsilon(t-3)+0.6035\cdot y(t-7)+0.2495\cdot y(t-14)+6.3013$

Where:

- $\epsilon(t)$ represents the error term at time t,
- $\epsilon(t-1)$, $\epsilon(t-2)$, and $\epsilon(t-3)$ are the lagged error terms,
- $y(t-7)$ and $y(t-14)$ represent the values of the series at lags 7 and 14 respectively,
- 6.3013 is the constant term.

```
models_ma %>% select(masearch) %>% coef()
```

The equation for MA(4)[7] / SARIMA(0,0,4)(2,0,0)[7] is:

$y(t)=\epsilon(t)+0.9889\cdot\epsilon(t-1)+0.7911\cdot\epsilon(t-2)+0.6031\cdot\epsilon(t-3)+0.3237\cdot\epsilon(t-4)+0.5128\cdot y(t-7)+0.3151\cdot y(t-14)+7.3822$

Where:

- ϵ(t) represents the error term at time t,
- ϵ(t−1), $\epsilon(t-2)$, $\epsilon(t-3)$, and $\epsilon(t-4)$are the lagged error terms,
- y(t−7) and $y(t-14)$ represent the values of the series at lags 7 and 14 respectively,
- 7.3822 is the constant term.

Testing on the Test Dataset:

```
models_ma %>%
  fabletools::forecast(h=5) %>%
  forecast::accuracy(test) %>%
  dplyr::select(.model, RMSE:MAPE)
```

The model to choose is the one with the smallest errors (RMSE and MAPE), which is the masearch model, specifically the SARIMA(0,0,4)(2,0,0)[7] model.

```
model_final_ma <- skx %>%
  model(ARIMA(Price ~ pdq(0,0,4) + PDQ(2,0,0)))
```

```
report(model_final_ma) %>% coef()
```

The final model equation is:

The equation for the MA(4)[7] / SARIMA(0,0,4)(2,0,0)[7] model is:

$y(t)=\epsilon(t)+0.9719\cdot\epsilon(t-1)+0.8021\cdot\epsilon(t-2)+0.5821\cdot\epsilon(t-3)+0.2987\cdot\epsilon(t-4)+0.5519\cdot y(t-7)+0.3493\cdot y(t-14)+4.4797$

Where:

- ϵ(t) represents the error term at time t,
- ϵ(t−1), $\epsilon(t-2)$, $\epsilon(t-3)$, and $\epsilon(t-4)$ are the lagged error terms,
- y(t−7) and $y(t-14)$ represent the values of the series at lags 7 and 14 respectively,
- 4.4797 is the constant term.

Analysis of Residuals:

Hypothesis Testing:

H0:The error term is white noise.

H1: The error term is not white noise.

```
model_final_ma %>%
  gg_tsresiduals()
```

According to the null hypothesis, the residuals are white noise—ideally, we want the p-value to be greater than 0.05.

```
augment(model_final_ma) %>%
  features(.resid, ljung_box, lag = 14, dof = 1)
```

The p-value < 0.05, so we reject the null hypothesis; the residuals are not white noise.

Forecasting:

```
model_final_ma %>%
  forecast::forecast() %>%
  autoplot(skx) +
  geom_line(aes(y = .fitted), col = 'deeppink', data = augment(model_final_ma)) +
  labs(title = "The trend of SKX stock's adjusted closing price between 2020 and 2024", y = 'Adjusted Closing Price', x = '')
```

```
model_final_ma %>%
  forecast::forecast(h=7)
```

Forecasting with the ARIMA model

Finally, we will also test using the ARIMA model.

```
models_arima <- train %>%
  model(arimaauto = ARIMA(Price),
        arima = ARIMA(Price ~ pdq(p = 0:3, d = 1, q = 0:3)),
        arimasearch = ARIMA(Price ~ pdq(d = 1), stepwise = FALSE, approximation = FALSE))
```

```
glance(models_arima)
```

```
models_arima %>% select(arimaauto) %>% report()
```

```
models_arima %>% select(arima) %>% report()
```

```
models_arima %>% select(arimasearch) %>% report()
```

```
models_arima %>% pivot_longer(everything(), names_to = 'Model name')
```

```
glance(models_arima) %>% arrange(AICc) %>% select(.model:BIC)
```

```
models_arima %>% select(arimasearch) %>% coef()
```

ARIMA(2,1,2) Equation:

$\Delta y(t)=-0.5956\cdot\Delta y(t-1)-0.9302\cdot\Delta y(t-2)+\epsilon(t)+0.5627\cdot\epsilon(t-1)+0.9047\cdot\epsilon(t-2)$

Where:

- $\Delta y(t)$ is the first difference of the series at time t.
- $\epsilon(t)$ represents the white noise error term at time t.

```
models_arima %>% select(arimaauto) %>% coef()
```

```
models_arima %>% select(arima) %>% coef()
```

SARIMA(2,1,2)(1,0,1)[7] Equation:

$\Delta y(t)=-0.6023\cdot\Delta y(t-1)-0.9268\cdot\Delta y(t-2)+\epsilon(t)+0.5697\cdot\epsilon(t-1)+0.8974\cdot\epsilon(t-2)-0.7064\cdot y(t-7)+0.6868\cdot\epsilon(t-7)$

Where:

- $\Delta y(t)$ is the first difference of the series at time t.
- $\epsilon(t)$ represents the white noise error term at time t.
- $y(t-7)$ is the value of the series at lag 7.
- $\epsilon(t-7)$ is the white noise error term at lag 7.

Testing on the Test Dataset:

```
models_arima %>%
  fabletools::forecast(h=5) %>%
  forecast::accuracy(test) %>%
  dplyr::select(.model, RMSE:MAPE)
```

The model to choose is the one with the smallest errors (RMSE: MAPE), which is the ARIMA/auto, the SARIMA(2,1,2)(1,0,1)[7] model.

```
model_final_arima <- skx %>%
  model(ARIMA(Price ~ pdq(2,1,2) + PDQ(1,0,1)))
```

```
report(model_final_arima) %>% coef()
```

Final Model Equation:

SARIMA(2,1,2)(1,0,1)[7] Equation:

$\Delta y(t)=0.2754\cdot\Delta y(t-1)+0.3455\cdot\Delta y(t-2)+\epsilon(t)-0.3402\cdot\epsilon(t-1)-0.3389\cdot\epsilon(t-2)+0.6985\cdot y(t-7)-0.7181\cdot\epsilon(t-7)$

Where:

- $\Delta y(t)$ is the differenced value of the time series at time t.
- $\Delta y(t-1)$ and $\Delta y(t-2)$ are the differenced values of the time series at lag 1 and lag 2, respectively.
- $\epsilon(t)$ is the white noise error term at time t.

- ϵ(t−1) and ϵ(t−2) are the white noise error terms at lags 1 and 2, respectively.
- y(t−7) is the value of the time series at lag 7.
- ϵ(t−7) is the white noise error term at lag 7.

Analysis of Residuals:

Hypothesis Testing:

H0:The error term is white noise.

H1: The error term is not white noise.

```
model_final_arima %>%
  gg_tsresiduals()
```

According to the null hypothesis, the residuals are white noise—we would want the p-value to be greater than 0.05.

```
augment(model_final_arima) %>%
  features(.resid, ljung_box, lag = 14, dof = 1)
```

The p-value is greater than 0.05, so we retain the null hypothesis. The residuals are white noise, meaning that a GARCH model is not needed.

Forecasting:

```
model_final_arima %>%
  forecast::forecast() %>%
  autoplot(skx) +
  geom_line(aes(y = .fitted), col = 'deeppink', data = augment(model_final_arima)) +
  labs(title = 'The adjusted closing price trend of SKX stock from 2020 to 2024', y = 'Adjusted Closing Price', x = '')
```

```
model_final_arima %>%
  forecast::forecast(h=7)
```

Comparison of Models

```
skx %>%
  model(ets = ETS(Price), arima = ARIMA(Price ~ pdq(2, 1, 2) + PDQ(1, 0, 1)), ma = ARIMA(Price ~ pdq(0, 0, 4) + PDQ(2, 0, 0))) %>%
  forecast::accuracy()
```

Based on the table above, the ETS model appears to be the best model.

The final step is to compare the estimated values with the actual values.

Exponential Smoothing

```
expsimf
```

Moving Average

```
model_final_ma %>%
  forecast::forecast(h=7)
```

ARIMA

```
model_final_arima %>%
  forecast::forecast(h=7)
```

Actual Values:

- February 12, 2024: 59.34
- February 13, 2024: 59.06
- February 14, 2024: 59.63
- February 15, 2024: 60.33
- February 16, 2024: 60.01

Absolute Errors:

Exponential Smoothing:

- February 12, 2024: 0.4948
- February 13, 2024: 0.2148
- February 14, 2024: 0.7848
- February 15, 2024: 1.4848
- February 16, 2024: 1.1648

Moving Average:

- February 12, 2024: 1.4483
- February 13, 2024: 0.5939
- February 14, 2024: 1.8494
- February 15, 2024: 1.6364
- February 16, 2024: 3.0890

ARIMA:

- February 12, 2024: 0.5023
- February 13, 2024: 0.2032
- February 14, 2024: 0.7353
- February 15, 2024: 1.4890