Project2

2023-04-29

```
library("ggplot2")
```

```
## Warning: package 'ggplot2' was built under R version 4.2.2
```

```
library("forecast")
```

```
## Warning: package 'forecast' was built under R version 4.2.3
```

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

```
library("prophet")
```

```
## Warning: package 'prophet' was built under R version 4.2.3
```

```
## Loading required package: Rcpp
```

```
## Loading required package: rlang
```

```
## Warning: package 'rlang' was built under R version 4.2.2
```

```
library("lubridate")
```

```
## Warning: package 'lubridate' was built under R version 4.2.3
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

1. You will be using a data set accessed via the link https://www.kaggle.com/competitions/store-sales-time-series-forecasting/data (https://www.kaggle.com/competitions/store-sales-time-series-forecasting/data) . The data is contained in the oil.csv file.

```
my_data <- read.csv("D:/Semester/Spring2023/SDM2/Project/Project 2/oil.csv")
```

```
colnames(my_data)
```

```
## [1] "date"       "dcoilwtico"
```

```
dim(my_data)
```

```
## [1] 1218    2
```

```
head(my_data, 10)
```

```
##          date dcoilwtico
## 1  2013-01-01         NA
## 2  2013-01-02      93.14
## 3  2013-01-03      92.97
## 4  2013-01-04      93.12
## 5  2013-01-07      93.20
## 6  2013-01-08      93.21
## 7  2013-01-09      93.08
## 8  2013-01-10      93.81
## 9  2013-01-11      93.60
## 10 2013-01-14      94.27
```

```
sum(is.na(my_data))
```

```
## [1] 43
```

2.Plot the time series as is

In the time series plot the x-axis typically represents the time, and the y-axis typically represents the value of the variable. So, converting the date column to date type

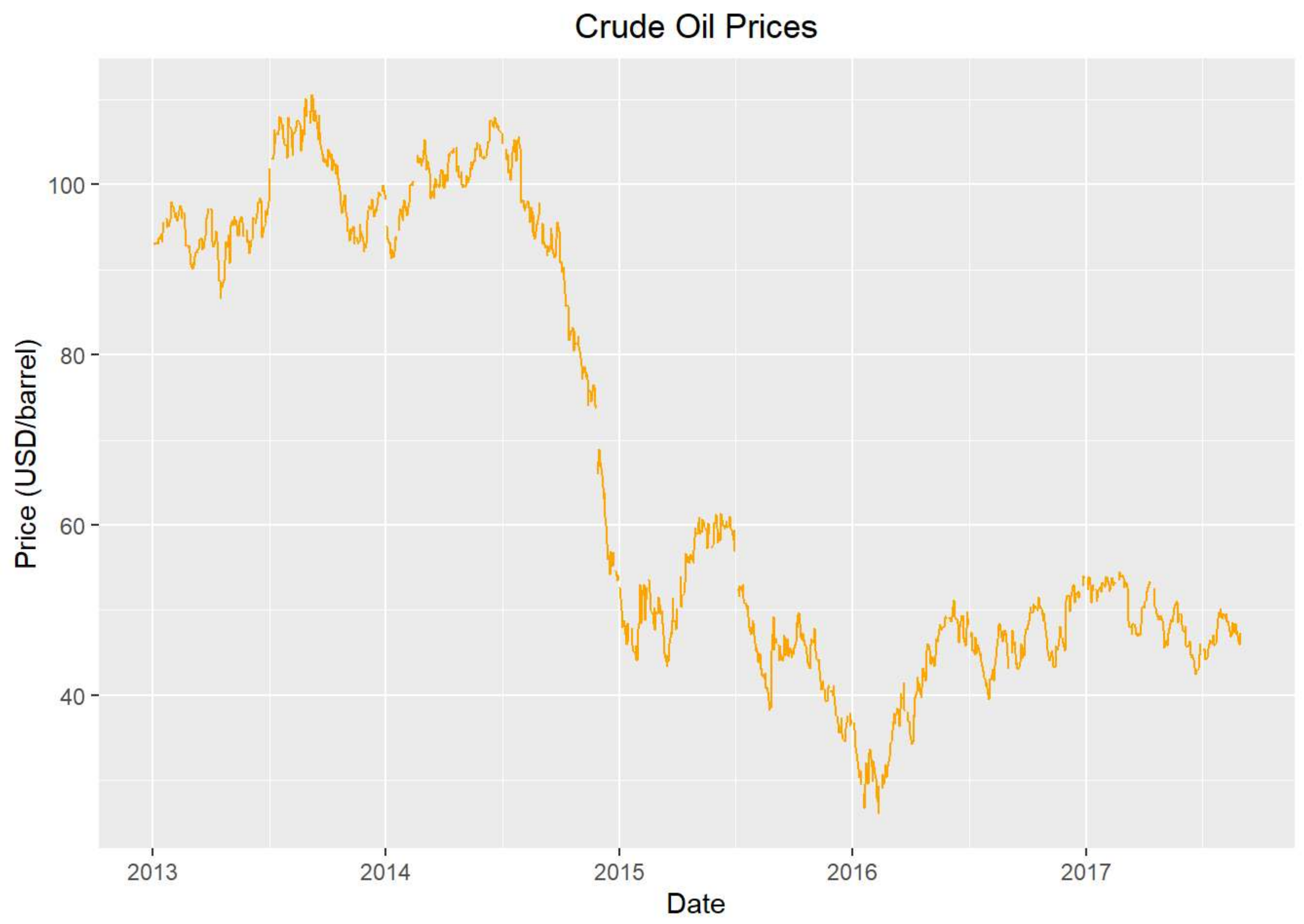
Convert date column to date format

```
my_data$date <- as.Date(my_data$date)
```

Create a time series plot

```
ggplot(my_data, aes(x = date, y = dcoilwtico)) +
  geom_line(color = "orange") +
  labs(title = "Crude Oil Prices",
       x = "Date",
       y = "Price (USD/barrel)") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Removed 1 row containing missing values (`geom_line()`).
```



The price of a barrel of oil has been on a downward trend for several years. The peak price was reached between 2013 and 2014, and the price has been falling ever since. The price fell sharply in early 2016, but it has since recovered somewhat

3.Read the literature and find out how to fill the missing data. Impute the data using your preferred method.

There are multiple ways to fill in the missing data.

Mean filling

This method replaces missing values with the mean of the non-missing values. This is a simple and easy method to implement, but it can be biased if the data is not normally distributed.

Median filling

This method replaces missing values with the median of the non-missing values. This method is less biased than mean imputation, but it can be less efficient.

Mode filling

This method replaces missing values with the mode of the non-missing values. This method is the least biased, but it can be inefficient.

KNN filling

This method replaces missing values with the values of the k nearest neighbors. This method is more efficient than the other methods, but it can be more biased.

For our problem, we are using mean filling to impute the missing data

Steps performed:

1. Finding the means for our variables
2. Filling the date column with nulls with mean_date
3. Filling the dcoilwtico column with nulls with mean_price

Finding the means for our variables

```
mean_date <- mean(my_data$date, na.rm = TRUE)
mean_price <- mean(my_data$dcoilwtico, na.rm = TRUE)
```

filling the date column with nulls with mean_date

```
my_data$date[is.na(my_data$date)] <- mean_date
```

filling the dcoilwtico column with nulls with mean_price

```
my_data$dcoilwtico[is.na(my_data$dcoilwtico)] <- mean_price
```

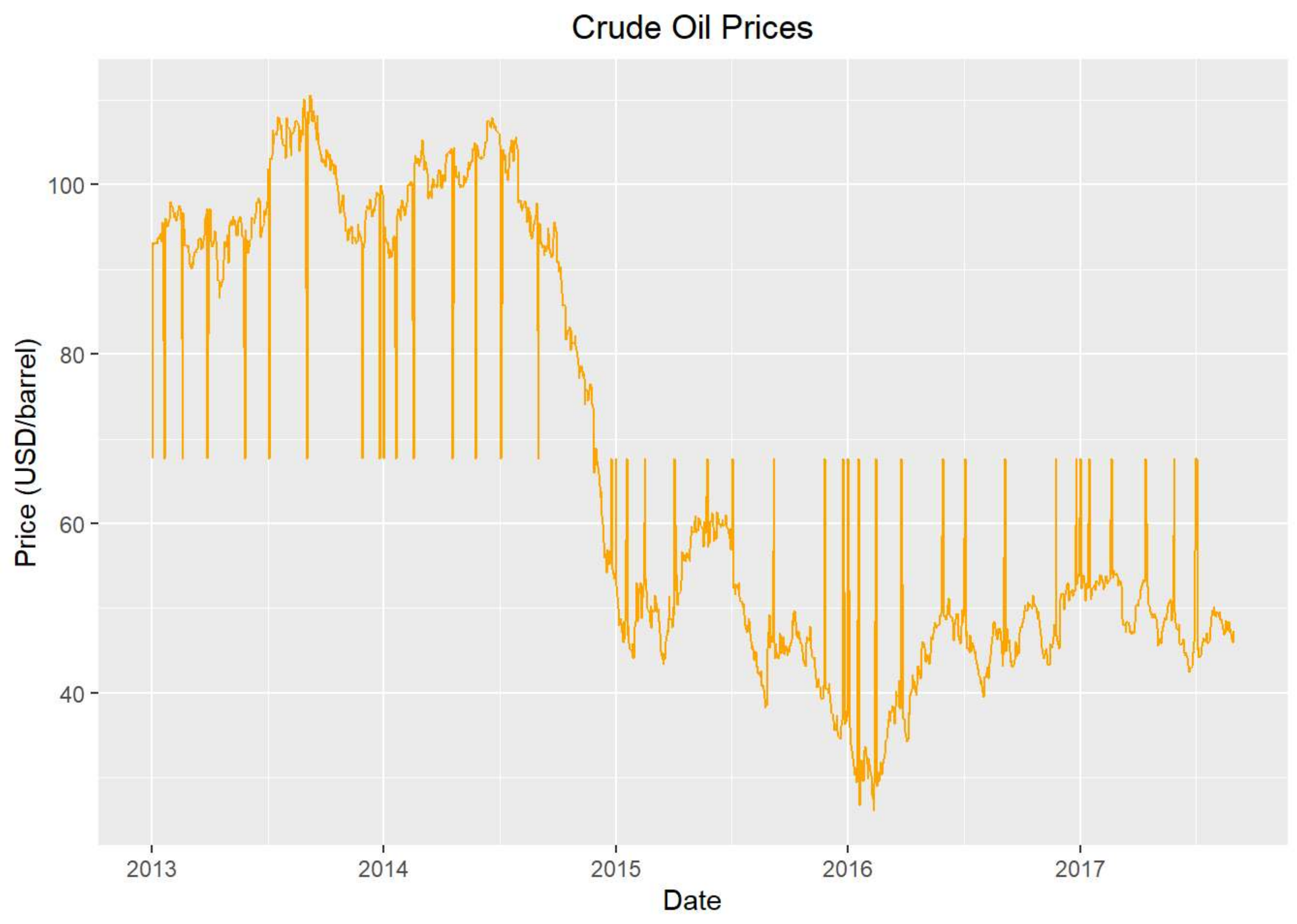
checking if any nulls are present

```
sum(is.na(my_data))
```

```
## [1] 0
```

4. Plot the time series with imputed data. Do you see a trend and/or seasonality in the data?

```
ggplot(my_data, aes(x = date, y = dcoilwtico)) +
  geom_line(color = "orange") +
  labs(title = "Crude Oil Prices",
       x = "Date",
       y = "Price (USD/barrel)") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
temp <-my_data
```

Overall, the time series curve shows a general upward trend, but there are some sharp drops in sales in 2015 and 2017. The reasons for these drops are not clear, but they could be due to several factors, such as a change in the product's price, a change in the product's marketing, or a change in the economy.

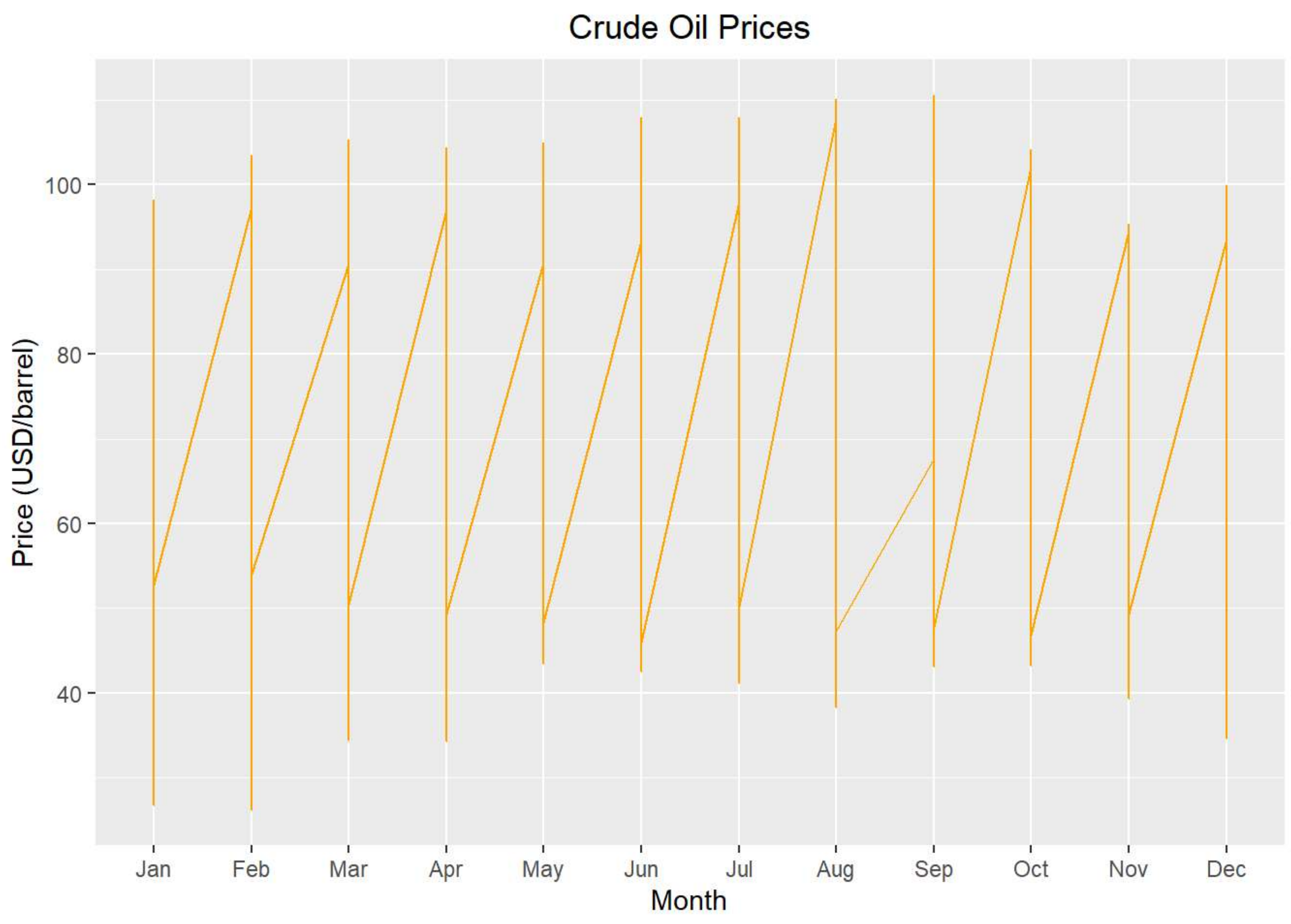
Checking at month level for seasonality

Extract month from date variable

```
temp$month <- month(temp$date, label = TRUE)
```

Plot with month-level seasonality

```
ggplot(temp, aes(x = month, y = dcoilwtico, group = 1)) +
  geom_line(color = "orange") +
  labs(title = "Crude Oil Prices",
       x = "Month",
       y = "Price (USD/barrel)") +
  theme(plot.title = element_text(hjust = 0.5))
```



The data does not exhibit any seasonal patterns. There are both upward and downward trends in the months, with no clear pattern.

5. Learn about the ETS models and about Holt-Winters models (provide all relevant specifics with respect to theoretical aspects and running them). This will expand your toolkit of the candidate models.

ETS models

ETS models are a type of exponential smoothing model that can be used to forecast time series data with trend and seasonality. The acronym ETS stands for “Error, Trend, Seasonality.” The error component of an ETS model represents the random variation in the data. The trend component represents the long-term direction of the data. The seasonality component represents the regular fluctuations in the data that occur over a fixed period.

There are three different types of ETS models:

• Additive ETS models use additive errors, trends, and seasonality. This means that the errors, trends, and seasonality are added together to create the forecast.

• Multiplicative ETS models use multiplicative errors, trends, and seasonality. This means that the errors, trends, and seasonality are multiplied together to create the forecast.

• Damped ETS models are a type of ETS model that uses a dampened trend. This means that the trend is gradually reduced over time.

Holt-Winters models

Holt-Winters models are a type of exponential smoothing model that can be used to forecast time series data with trend and seasonality. The acronym Holt-Winters stands for “Holt” and “Winters,” who were the two scientists who developed the method.

Holt-Winters models are more complex than ETS models, but they can provide more accurate forecasts. There are three different types of Holt-Winters models:

• Simple exponential smoothing is a type of Holt-Winters model that only uses an error component.

• Double exponential smoothing is a type of Holt-Winters model that uses an error and trend component.

• Triple exponential smoothing is a type of Holt-Winters model that uses an error, trend, and seasonality component.

Running ETS and Holt-Winters models

ETS and Holt-Winters models can be run in a variety of software packages, including R, Python, and Excel. The specific steps involved in running the models will vary depending on the software package that is being used.

ETS Model

Fit an ETS model to the data

```
ts_data <- ts(my_data$dcoilwtico, frequency = 365)
ets_model <- ets(ts_data)
```

```
## Warning in ets(ts_data): I can't handle data with frequency greater than 24.
## Seasonality will be ignored. Try stlf() if you need seasonal forecasts.
```

Print the summary of the ETS model

```
summary(ets_model)
```

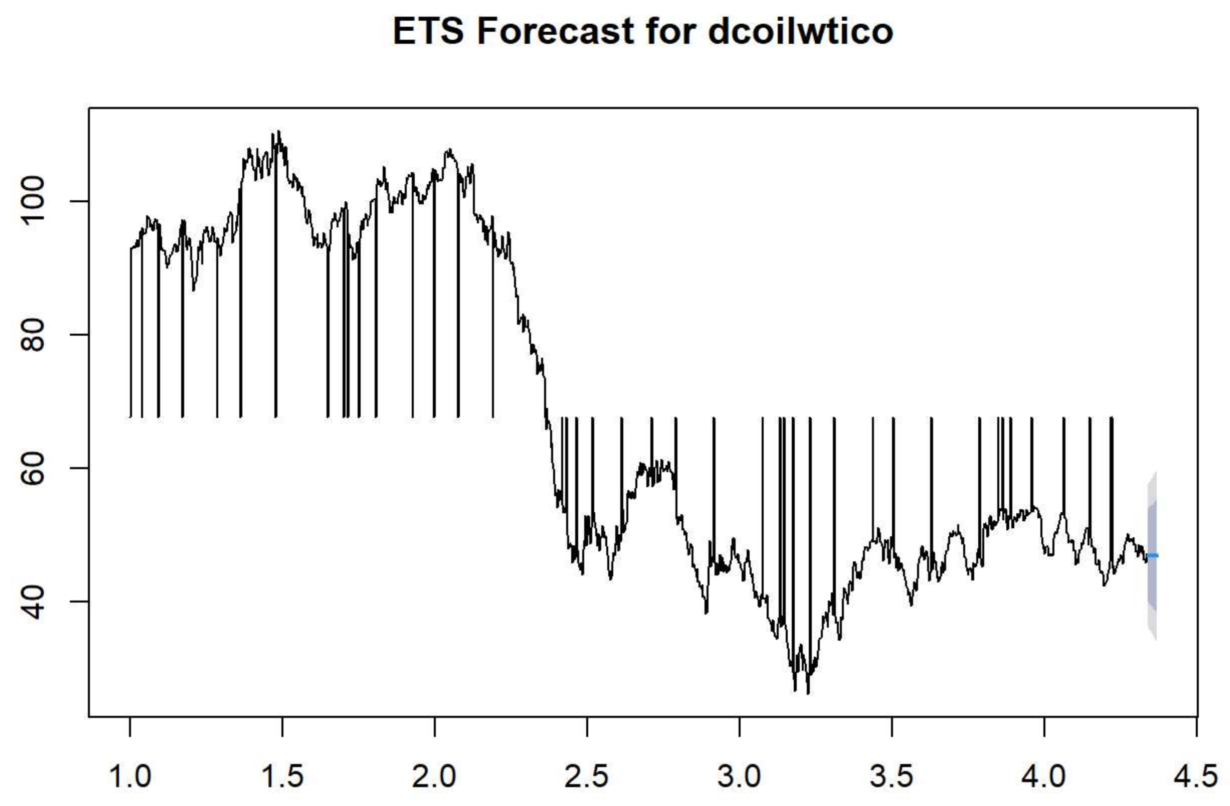
```
## ETS(A,Ad,N)
##
## Call:
##  ets(y = ts_data)
##
##   Smoothing parameters:
##     alpha = 0.185
##     beta  = 0.0098
##     phi   = 0.8
##
##   Initial states:
##     l = 84.2406
##     b = 3.4699
##
##   sigma:  5.3776
##
##      AIC     AICc      BIC
## 12758.76 12758.83 12789.39
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.1875495 5.366514 2.770154 -0.9428757 4.622359 0.1016891
##                   ACF1
## Training set 0.01293332
```

Forecast using the ETS model

```
ets_forecast <- forecast(ets_model, h = 12)
```

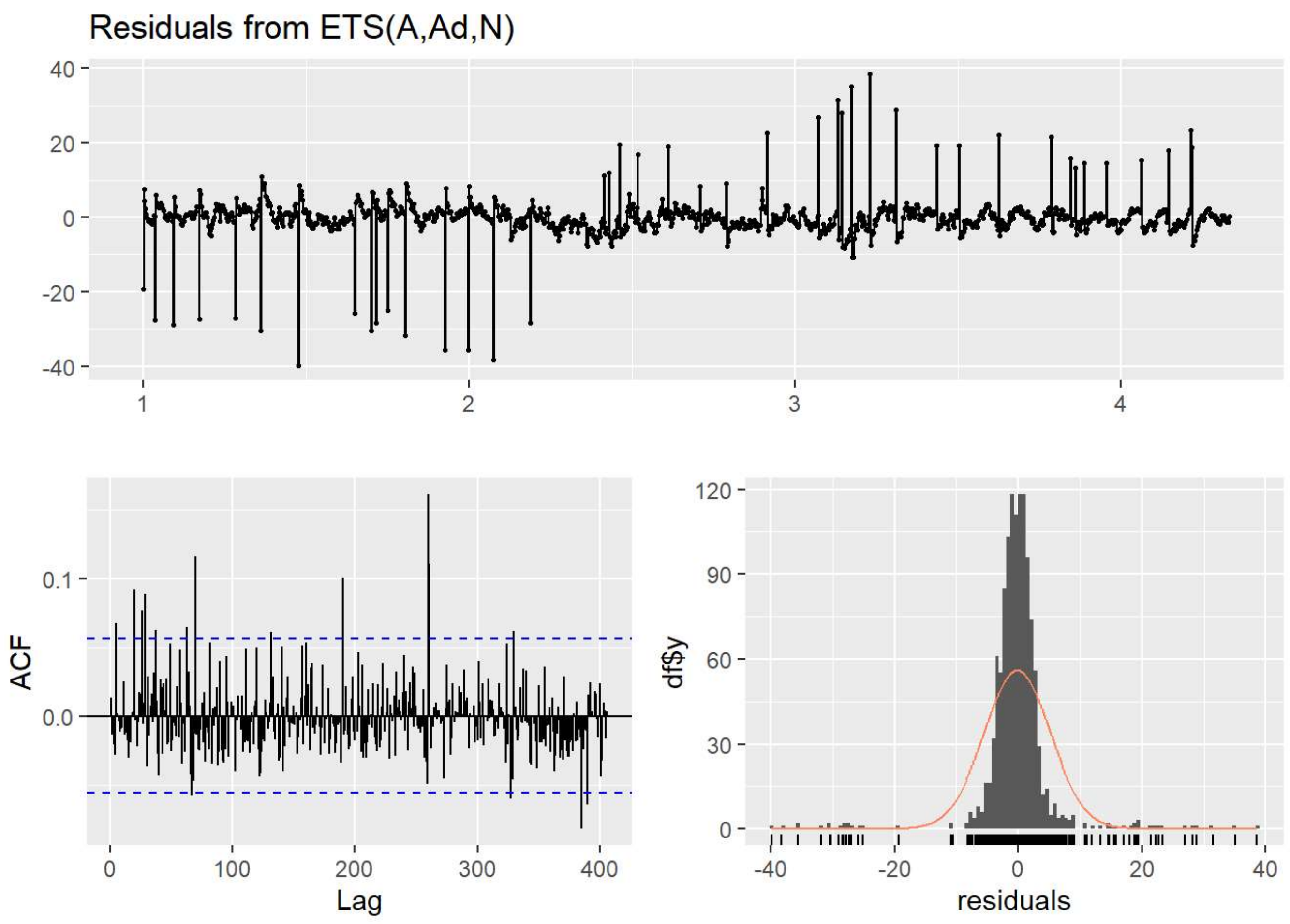
Print the forecasted values

```
plot(ets_forecast, main = "ETS Forecast for dcoilwtico")
```



Residuals for ETS model

```
checkresiduals(ets_model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,Ad,N)
## Q* = 243.72, df = 244, p-value = 0.493
##
## Model df: 0.   Total lags used: 244
```

Holt-Winters Model

Fit a Holt-Winters model to the data

```
ts_data <- ts(my_data$dcoilwtico, frequency = 7)
hw_model <- hw(ts_data, seasonal = "additive")
```

Print the summary of the Holt-Winters model

```
summary(hw_model)
```

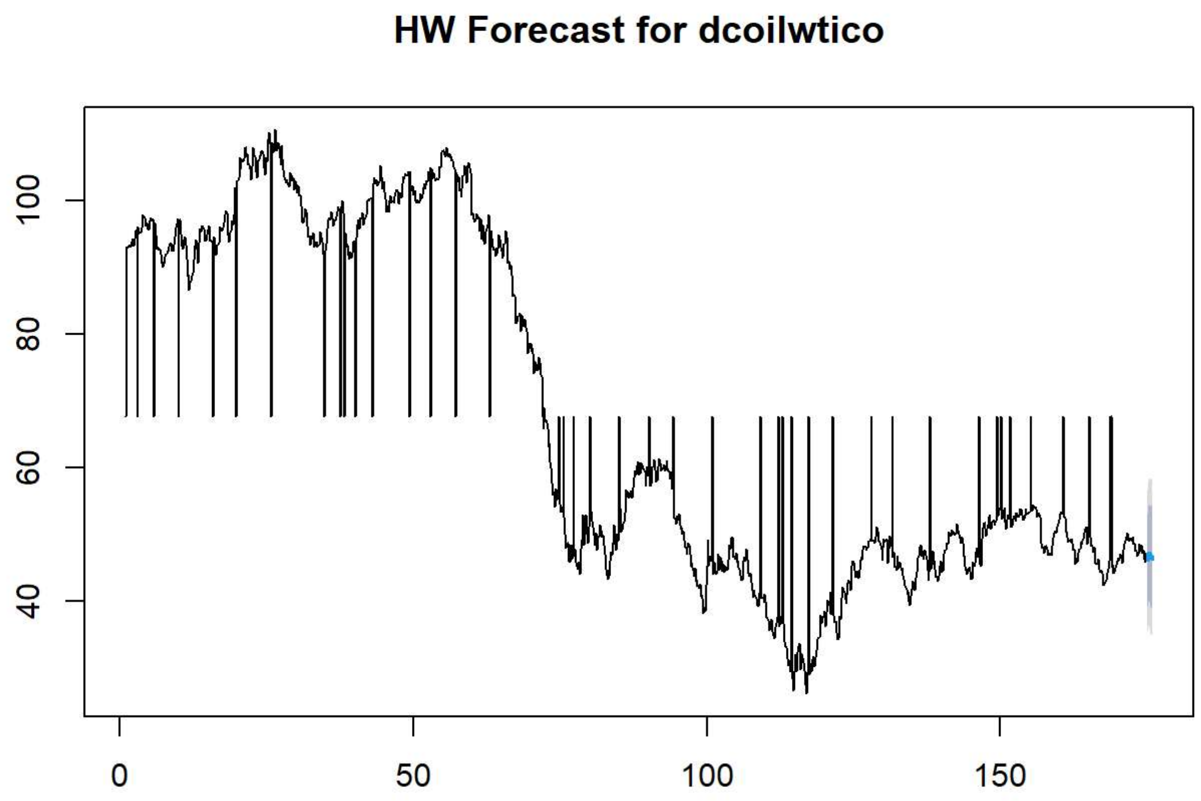
```
##
## Forecast method: Holt-Winters' additive method
##
## Model Information:
## Holt-Winters' additive method
##
## Call:
##  hw(y = ts_data, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.2115
##     beta  = 1e-04
##     gamma = 1e-04
##
##   Initial states:
##     l = 93.728
##     b = -0.033
##     s = -0.2739 0.0256 -0.0638 0.4388 0.2018 0.4113
##            -0.7398
##
##   sigma:  5.3976
##
##      AIC     AICc      BIC
## 12773.79 12774.05 12835.05
##
## Error measures:
##                       ME    RMSE      MAE        MPE     MAPE      MASE
## Training set -0.0132346 5.37319 2.797417 -0.6405998 4.633677 0.7137828
##                       ACF1
## Training set -0.003650463
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 175.0000       46.16388 39.24656 53.08121 35.58475 56.74302
## 175.1429       47.27916 40.20862 54.34971 36.46570 58.09263
## 175.2857       47.03347 39.81281 54.25412 35.99043 58.07650
## 175.4286       47.23618 39.86834 54.60402 35.96804 58.50432
## 175.5714       46.70125 39.18897 54.21354 35.21221 58.19030
## 175.7143       46.75681 39.10268 54.41094 35.05083 58.46279
## 175.8571       46.41932 38.62579 54.21284 34.50014 58.33849
## 176.0000       45.92181 37.99108 53.85255 33.79280 58.05082
## 176.1429       47.03709 38.97148 55.10270 34.70181 59.37237
## 176.2857       46.79139 38.59301 54.98978 34.25305 59.32974
## 176.4286       46.99411 38.66494 55.32328 34.25575 59.73247
## 176.5714       46.45918 38.00113 54.91723 33.52371 59.39465
## 176.7143       46.51474 37.92962 55.09985 33.38494 59.64454
## 176.8571       46.17724 37.46680 54.88769 32.85577 59.49872
```

Forecast using the Holt-Winters model

```
hw_forecast <- forecast(hw_model, h = 7)
```

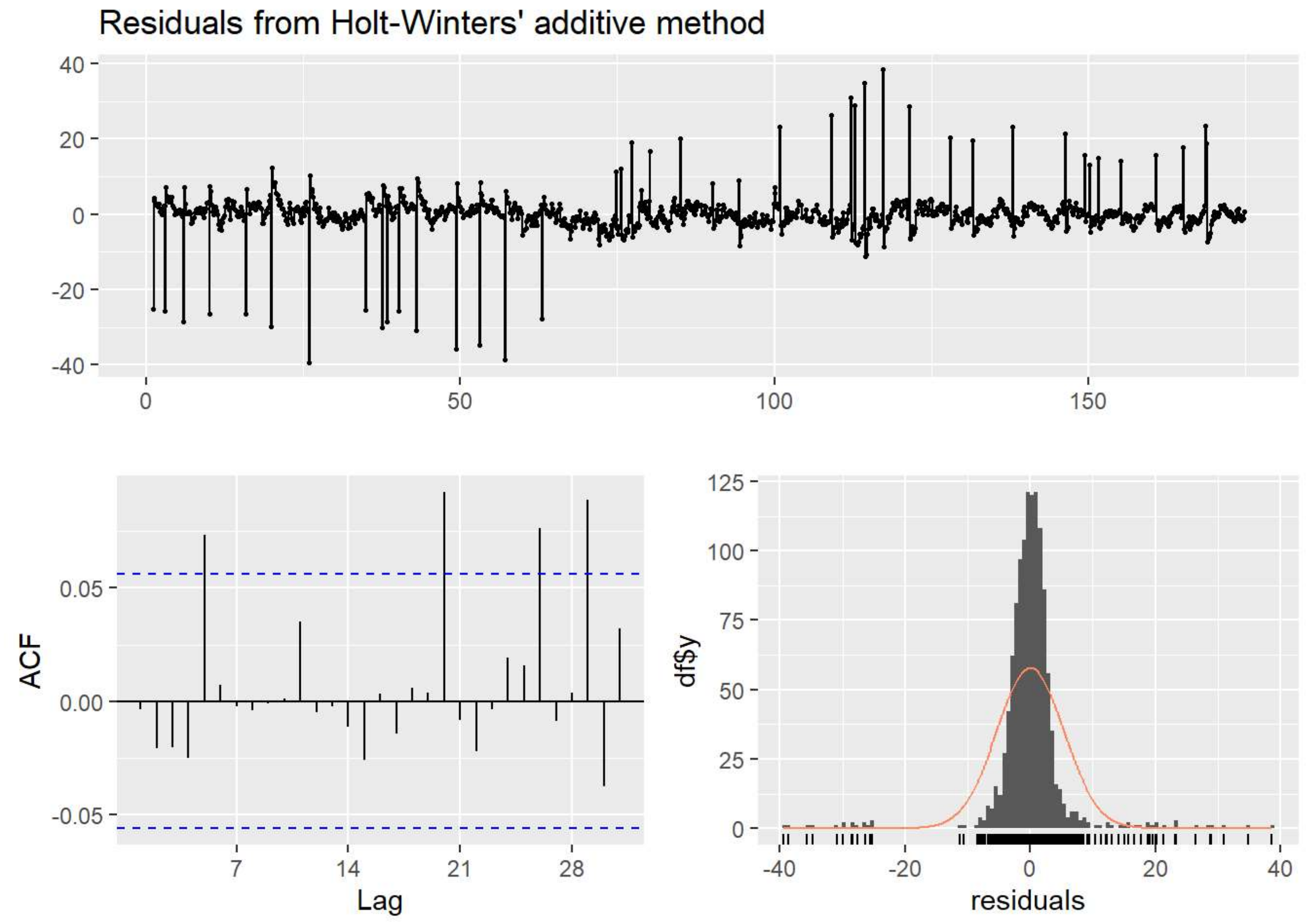
Print the forecasted values

```
plot(hw_forecast, main = "HW Forecast for dcoilwtico")
```



Residuals for Holt-Winters model

```
checkresiduals(hw_model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' additive method
## Q* = 10.196, df = 14, p-value = 0.7477
##
## Model df: 0.   Total lags used: 14
```

Advantages of ETS and Holt-Winters models

ETS and Holt-Winters models are relatively simple to understand and implement. They can be used to forecast a wide variety of time series data. ETS and Holt-Winters models are also relatively accurate.

Disadvantages of ETS and Holt-Winters models

ETS and Holt-Winters models can be sensitive to outliers. They can also be computationally expensive for large datasets.

Overall, ETS and Holt-Winters models are a powerful tool for forecasting time series data. They are relatively simple to understand and implement, and they can be used to forecast a wide variety of time series data.

6. Based on your answer to the question 4, suggest suitable model(s) for the data.

ARIMA

This model is a more complex time series model that can capture a wide range of time series patterns, including trends, seasonality, and autoregressive and moving average components.

Prophet

This is a forecasting model developed by Facebook, which is designed to capture seasonal patterns and long-term trends. It is capable of handling missing values and can incorporate external variables, which may be useful for predicting crude oil prices.

7. Run the models and check their adequacy

ARIMA model

Fit ARIMA model

```
arima_model <- auto.arima(ts_data, seasonal=FALSE)
```

Summary of ARIMA model

```
summary(arima_model)
```

```
## Series: ts_data
## ARIMA(1,1,1)
##
## Coefficients:
##          ar1      ma1
##       0.0013  -0.7901
## s.e.  0.0356   0.0208
##
## sigma^2 = 29.04:  log likelihood = -3776.07
## AIC=7558.14   AICc=7558.16   BIC=7573.46
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.1289061 5.381783 2.786541 -0.8841382 4.626618 0.7110078
##                    ACF1
## Training set 0.01010003
```

Forecast with ARIMA model

```
arima_forecast <- forecast(arima_model)
```

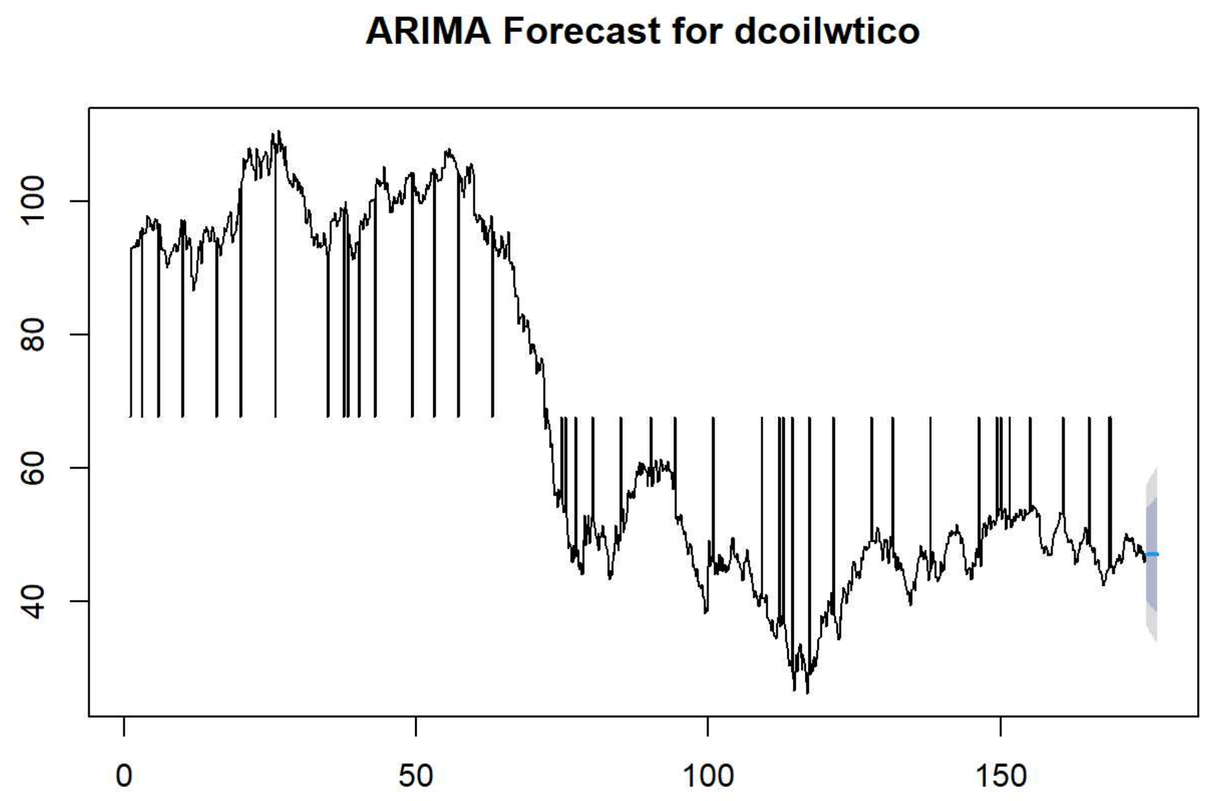
Evaluate performance of ARIMA model

```
accuracy(arima_forecast)
```

```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.1289061 5.381783 2.786541 -0.8841382 4.626618 0.7110078
##                    ACF1
## Training set 0.01010003
```

Plot actual vs forecasted values for ARIMA model

```
plot(arima_forecast, main = "ARIMA Forecast for dcoilwtico")
```



Prophet model

Convert data to Prophet-compatible format

```
prophet_data <- data.frame(ds = my_data$date, y = my_data$dcoilwtico)
```

Fit Prophet model

```
model <- prophet(prophet_data)
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

Cross-validation

```
horizon <- 90 # number of days to forecast
cv <- prophet::cross_validation(model, horizon = horizon, initial = 365, period = 180, units =
"days")
```

```
## Making 7 forecasts with cutoffs between 2014-06-18 and 2017-06-02
```

```
## Warning in prophet::cross_validation(model, horizon = horizon, initial = 365, :
## Seasonality has period of 365.25 days which is larger than initial window.
## Consider increasing initial.
```

Compute performance metrics

```
rmse <- sqrt(mean((cv$yhat - cv$y)^2))
print(rmse)
```

```
## [1] 13.30909
```

```
mape <- mean(abs((cv$y - cv$yhat)/cv$y))
print(mape)
```

```
## [1] 0.2087216
```

```
mae <- mean(abs(cv$y - cv$yhat))
print(mae)
```

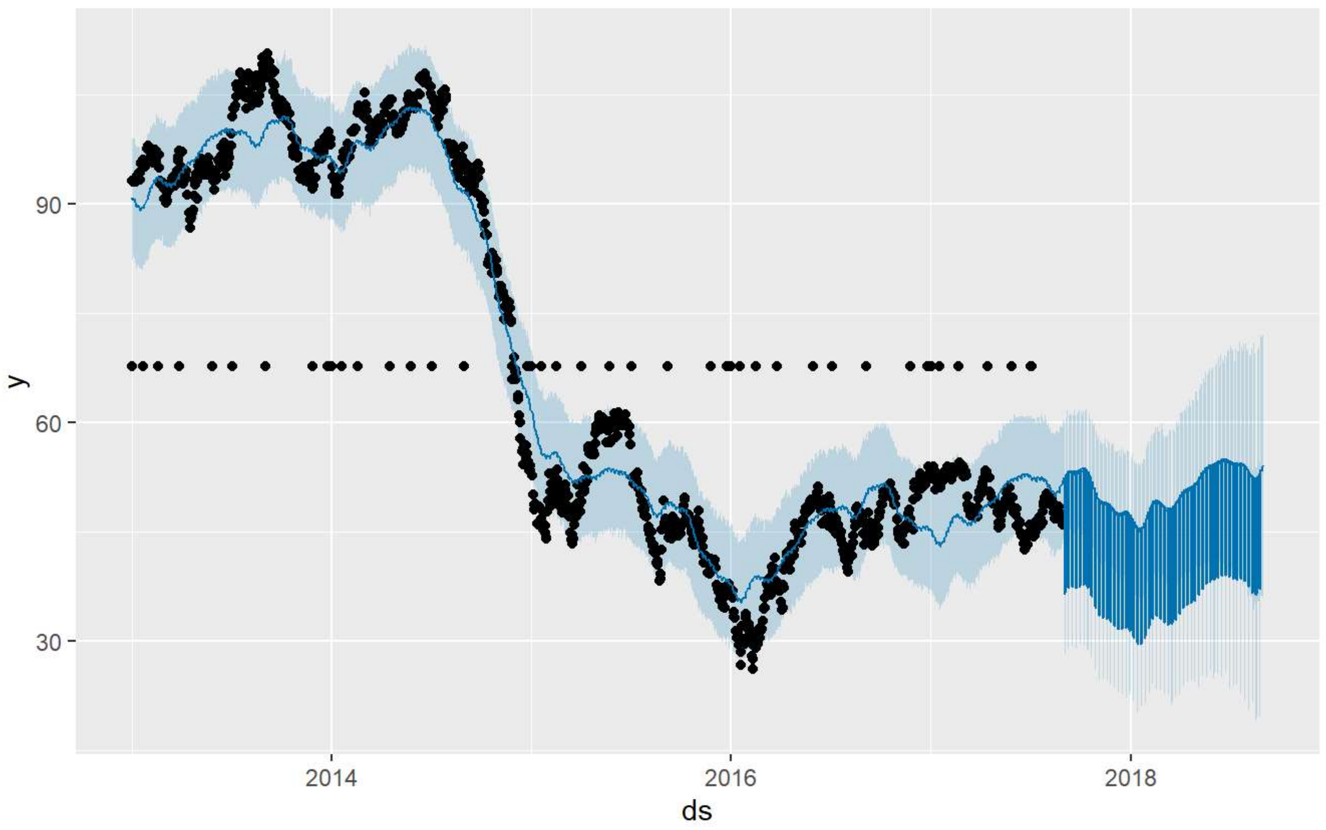
```
## [1] 11.11569
```

Make predictions with the fitted model

```
future_dates <- make_future_dataframe(model, periods = 365)
forecast <- predict(model, future_dates)
```

Plot the forecast

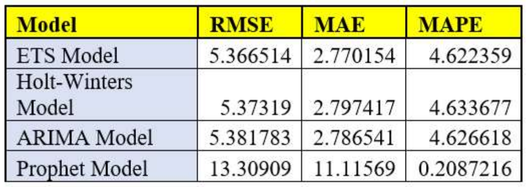
```
plot(model, forecast)
```



8.8. Compare the models’ performance by the metrics that you think are relevant. Try to identify a model with a low RMSE.

```
knitr::include_graphics("C:/Users/tejad/Desktop/performance.png")
```

```
## Warning in knitr::include_graphics("C:/Users/tejad/Desktop/performance.png"): It
## is highly recommended to use relative paths for images. You had absolute paths:
## "C:/Users/tejad/Desktop/performance.png"
```



Model	RMSE	MAE	MAPE
ETS Model	5.366514	2.770154	4.622359
Holt-Winters Model	5.37319	2.797417	4.633677
ARIMA Model	5.381783	2.786541	4.626618
Prophet Model	13.30909	11.11569	0.2087216

Based on the RMSE, MAE, and MAPE values, the ETS model is the best model. It has the lowest RMSE and MAE values, and it is also the closest to the actual value in terms of MAPE. The Holt-Winters model is a close second, but it has a slightly higher RMSE and MAE values. The ARIMA model is the worst model, with the highest RMSE and MAE values. The Prophet model is also not very good, with a very high MAPE value.

In conclusion, based on the RMSE, MAE, and MAPE values provided, the ETS, Holt-Winters, and ARIMA models appear to be the best choices for this particular dataset. However, further analysis and comparison of these models, along with consideration of other factors, would be necessary to make a more definitive conclusion.