STAT 4990 Final Project

Alisa Dmitrieva

2023-12-03

```r
library(tsibble)
```

```
##
## Attaching package: 'tsibble'

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

```r
library(fpp3)
```

```
## -- Attaching packages ---------------------------------------------- fpp3 0.5 --

## v tibble      3.2.1     v tsibbledata 0.4.1
## v dplyr       1.1.3     v feasts      0.3.1
## v tidyr       1.3.0     v fable       0.3.3
## v lubridate   1.9.2     v fabletools  0.3.4
## v ggplot2     3.4.3

## -- Conflicts ------------------------------------------------- fpp3_conflicts --
## x lubridate::date()     masks base::date()
## x dplyr::filter()       masks stats::filter()
## x tsibble::intersect()  masks base::intersect()
## x lubridate::interval() masks tsibble::interval()
## x dplyr::lag()          masks stats::lag()
## x tsibble::setdiff()    masks base::setdiff()
## x tsibble::union()      masks base::union()
```

```r
library(ggplot2)
library(fable)
library(forecast)
```

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

```r
library(tidyr)
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following object is masked from 'package:tsibble':
##
##     index

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## ######################### Warning from 'xts' package ##########################
## #                                                                             #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                            #
## #                                                                             #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                             #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                             #
## ###############################################################################

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last

## Loading required package: TTR
```

```r
library(prophet)
```

```
## Loading required package: Rcpp

## Loading required package: rlang
```

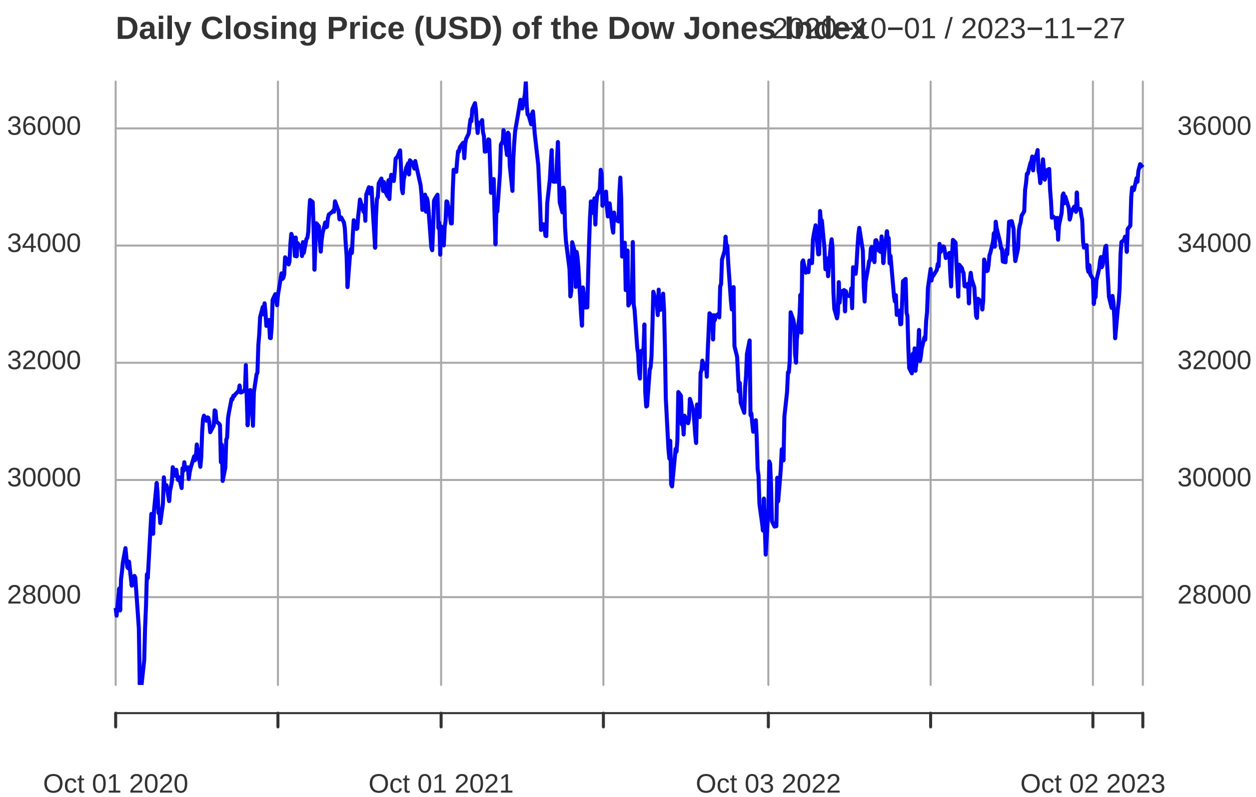
```r
library(fabletools)
```

Downloading and preparing the data

```r
# The training set will use 3 years of data
#   and the test set will use approximately 2 moths of data
start.date = '2020-10-01' # starting date of stock
end.date = '2023-11-28' # ending date of stock

# Downloading the Dow Jones Index (DJI) data from Yahoo finance using the `quantmod` package
getSymbols("^DJI", src = "yahoo", from = start.date, to = end.date, auto.assign = TRUE)
```

```
## [1] "DJI"
```

```r
# Extracting the closing price information
DJI.ClosingPrice <- DJI$DJI.Close

# Creating the training and test sample
N <- length(DJI.ClosingPrice)
n <- 40 # 40 days (2 months) is the test sample size
training.sample <- DJI.ClosingPrice[1:(N-n)] # training sample

# Plotting the DJI daily closing data
plot(DJI.ClosingPrice, col = "blue",
     xlab="Date",
     main="Daily Closing Price (USD) of the Dow Jones Index")
```



Forecasting with the Prophet model

```
# Preparing the training sample for fitting the Prophet model
DJI.train <- as.data.frame(training.sample)
DJI.train <- cbind(ds = rownames(DJI.train), DJI.train)
rownames(DJI.train ) <- 1:nrow(DJI.train)
colnames(DJI.train ) <- c ("ds", "y")

# Checking the training sample
head(DJI.train)
```

```
##           ds        y
## 1 2020-10-01 27816.90
## 2 2020-10-02 27682.81
## 3 2020-10-05 28148.64
## 4 2020-10-06 27772.76
## 5 2020-10-07 28303.46
## 6 2020-10-08 28425.51
```

```
# Fitting the Prophet model
DJI.prophet <- prophet(DJI.train)
```

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

```
# Preparing to make Forecasts
DJI.future <- make_future_dataframe(DJI.prophet, periods = n)
head(DJI.future)
```

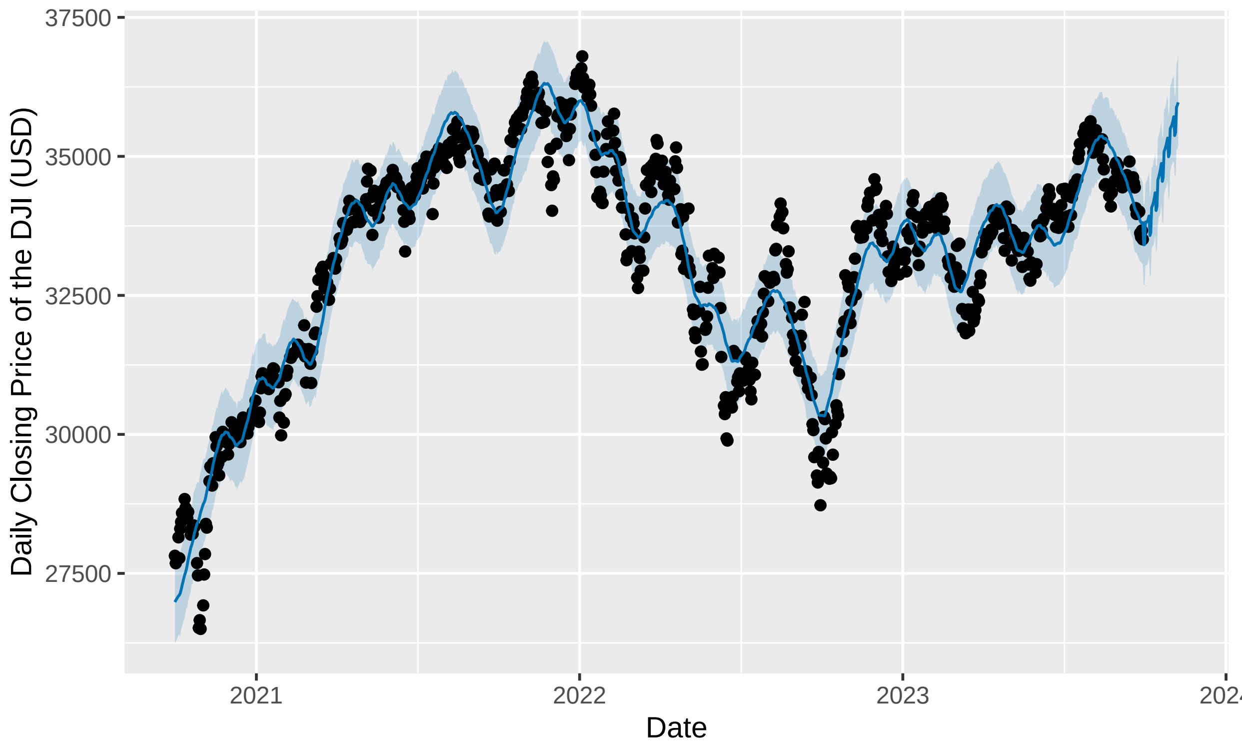
```
##           ds
## 1 2020-10-01
## 2 2020-10-02
## 3 2020-10-05
## 4 2020-10-06
## 5 2020-10-07
## 6 2020-10-08
```

```
# Creating forecasts using the Prophet model
forecast.prophet <- predict(DJI.prophet, DJI.future)
head(forecast.prophet)
```

```
##           ds    trend additive_terms additive_terms_lower additive_terms_upper
## 1 2020-10-01 28310.63      -1326.394            -1326.394            -1326.394
## 2 2020-10-02 28315.38      -1299.526            -1299.526            -1299.526
## 3 2020-10-05 28329.65      -1236.659            -1236.659            -1236.659
## 4 2020-10-06 28334.41      -1225.837            -1225.837            -1225.837
## 5 2020-10-07 28339.16      -1195.833            -1195.833            -1195.833
## 6 2020-10-08 28343.92      -1141.690            -1141.690            -1141.690
##     weekly weekly_lower weekly_upper    yearly yearly_lower yearly_upper
## 1 101.7164     101.7164     101.7164 -1428.110    -1428.110    -1428.110
## 2 122.7541     122.7541     122.7541 -1422.280    -1422.280    -1422.280
```

```
## 3 125.4581     125.4581     125.4581 -1362.117    -1362.117    -1362.117
## 4 102.7054     102.7054     102.7054 -1328.543    -1328.543    -1328.543
## 5  92.9756      92.9756      92.9756 -1288.808    -1288.808    -1288.808
## 6 101.7164     101.7164     101.7164 -1243.407    -1243.407    -1243.407
##   multiplicative_terms multiplicative_terms_lower multiplicative_terms_upper
## 1                    0                          0                          0
## 2                    0                          0                          0
## 3                    0                          0                          0
## 4                    0                          0                          0
## 5                    0                          0                          0
## 6                    0                          0                          0
##   yhat_lower yhat_upper trend_lower trend_upper     yhat
## 1   26244.90   27710.27    28310.63    28310.63 26984.24
## 2   26272.13   27730.53    28315.38    28315.38 27015.86
## 3   26385.49   27881.22    28329.65    28329.65 27092.99
## 4   26406.39   27818.20    28334.41    28334.41 27108.57
## 5   26364.46   27877.80    28339.16    28339.16 27143.33
## 6   26492.29   27949.31    28343.92    28343.92 27202.23
```

```r
# Plotting the forecasts from the Prophet model
plot(DJI.prophet, forecast.prophet,
     xlab="Date",
     ylab="Daily Closing Price of the DJI (USD)")
```

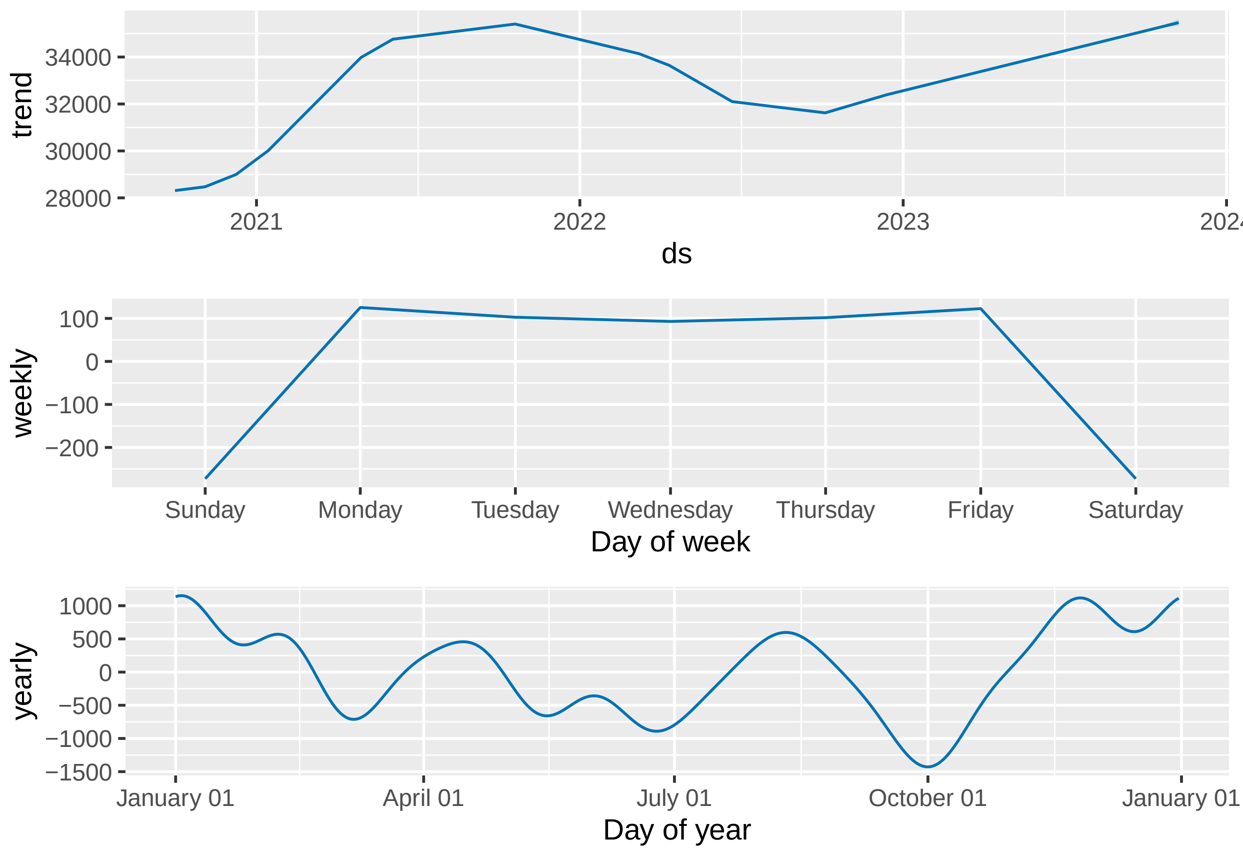



```r
###########################################################################
## Forecasting the Daily Closing Price of the DJI Using the Prophet Model ##
###########################################################################
```

Comments on the plot of the Prophet model forecasts:

The Prophet model appears to be a reasonable fit for the data. The forecasts produced by the Prophet model appear to capture the trend of the data in the test set (positive trend). Additionally, the forecasts produced by the Prophet model appear to have relatively low variance.

```r
# Extracting and plotting the Prophet model components
prophet_plot_components(DJI.prophet, forecast.prophet)
```




```r
##################################
## Prophet model decomposition ##
##################################
```

Comments on the Prophet model decomposition:

The Prophet model identified an overall positive trend, a weekly trend, and a yearly trend.

The overall trend shows that the value of the Dow Jones Index (DJI) has been generally increasing during the past 3 years, with the exception of a dip in the middle of 2022. This trend predicts that the value of the DJI will continue increasing over time. This is probably true - the value of the DJI will likely continue to increase along with rising inflation and increasing market caps of the stocks included in the DJI. However, it is always possible that the DJI will decrease due to war, natural disasters or other unforseen events. The trend identified by the Prophet model has no way of accounting for this.

The weekly trend shows that the closing price of the DJI is highest on weekdays and lowest on the weekends. This is due to the fact that the markets are closed on weekends and this trend is likely to continue indefinitely.

The yearly trend is the least appropriate of all trends identified by the Prophet model. There is no way of predicting that the value of the DJI will continue to increase or decrease in certain months as it has in the past.

```r
# Calculating the sign correlation of the the daily closing price of the DJI

# Sign correlation function
rho.cal<-function(X)
{
  rho.hat<-cor(sign(X-mean(X)), X-mean(X))
  return(rho.hat)
}

# Calculating the sign correlation
rho_cal<-apply(as.matrix(DJI.ClosingPrice), MARGIN=2, FUN=rho.cal)

# Sign correlation value
rho_cal
```

```
## DJI.Close
## 0.8067395
```

Comments on the sign correlation value:

The sign correlation value indicates that the daily closing price of the DJI during the dates specified above may follow a normal distribution.

Forecasting with an EWMA model

```r
# Fitting an EWMA model using the ETS function
DJI.ets = ets(training.sample$DJI.Close)
summary(DJI.ets)
```

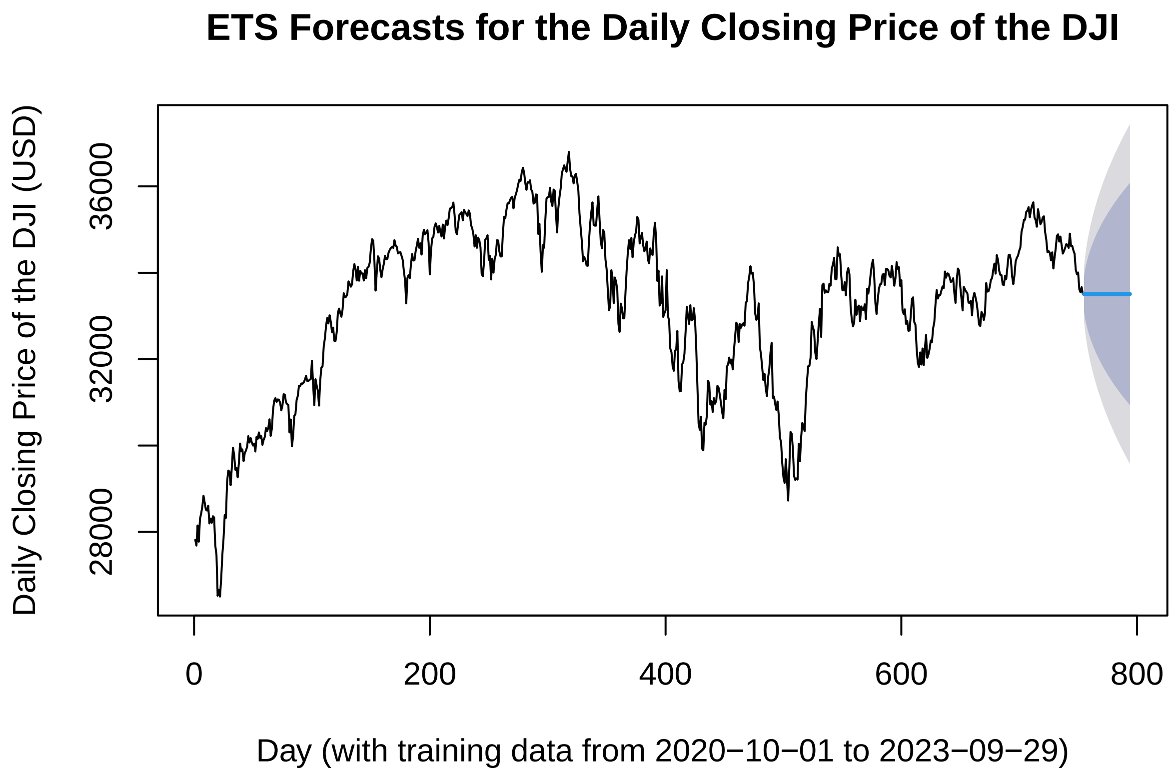
```
## ETS(A,N,N)
##
## Call:
##  ets(y = training.sample$DJI.Close)
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 27820.6948
##
##   sigma:  317.5959
##
##      AIC     AICc      BIC
## 13686.80 13686.83 13700.68
##
## Training set error measures:
##                  ME     RMSE      MAE        MPE      MAPE      MASE       ACF1
## Training set 7.543 317.1744 238.2268 0.01990846 0.7260493 0.9986979 0.00663868
```

Comments on the ETS model:

The optimal alpha identified here is 0.9999.

The MASE value indicates that the ETS model performs marginally better than the naive model.

```r
# Plotting the ETS model forecasts for the next 2 months
plot(forecast(DJI.ets, h=n),
  xlab = "Day (with training data from 2020-10-01 to 2023-09-29)",
  ylab = "Daily Closing Price of the DJI (USD)",
  main = "ETS Forecasts for the Daily Closing Price of the DJI")
```



Comments on the plot of the ETS model forecasts:

The ETS model forecasts appear to predict no trend in the test data and resemble the naive model forecasts. Additionally, The ETS model produces forecasts with a lot of variance.

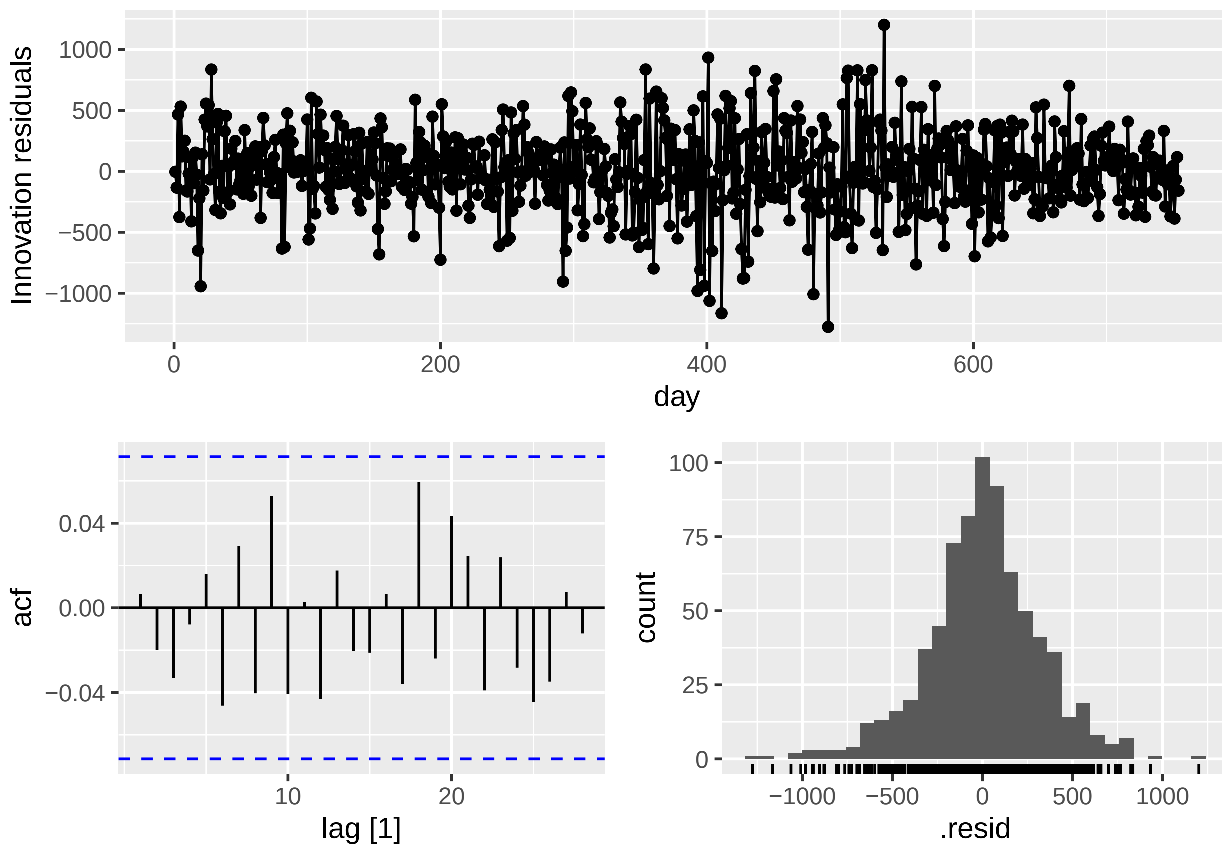
Comparing the Prophet and ETS models

```r
# Preparing the data to compare the Prophet and ETS models using fable

# Creating a date variable
DJI <- zoo::fortify.zoo(DJI)
DJI <- DJI %>% rename(c("Date" = "Index", "Close" = "DJI.Close"))

# Creating a tsibble object
DJI <- as_tsibble(DJI, index = Date)

# Re-indexing to remove the missing values
DJI <- DJI |>
mutate(day = row_number()) |>
update_tsibble(index = day, regular = TRUE)

# Creating the training set for the DJI
DJI.train2 <- DJI |> filter(yearmonth(Date) <= yearmonth("2023 Sept"))

# Checking the training set
head(DJI.train2)
```

```
## # A tsibble: 6 x 8 [1]
##   Date       DJI.Open DJI.High DJI.Low  Close DJI.Volume DJI.Adjusted   day
##   <date>        <dbl>    <dbl>   <dbl>  <dbl>      <dbl>        <dbl> <int>
## 1 2020-10-01   27941.   28041.  27669. 27817.  373450000       27817.     1
## 2 2020-10-02   27536.   27861.  27383. 27683.  392770000       27683.     2
## 3 2020-10-05   27825.   28163.  27825. 28149.  318210000       28149.     3
## 4 2020-10-06   28214.   28354.  27728. 27773.  435030000       27773.     4
## 5 2020-10-07   27971.   28370.  27971. 28303.  328750000       28303.     5
## 6 2020-10-08   28349.   28459.  28266. 28426.  314750000       28426.     6
```

ETS model residual diagnostics

```
# Fitting the ETS model
fit.ets <- DJI.train2 |> model(ETS(Close))

# Plotting the residuals
gg_tsresiduals(fit.ets)
```



The residuals for the ETS model resemble white noise. The ACF plot indicates no significant autocorrelation and the histogram appears to be normally-distributed with a mean of 0.

```
# Ljung-Box test for autocorrelation
augment(fit.ets) |> features(.innov, ljung_box, lag=10)
```

```
## # A tibble: 1 x 3
##   .model     lb_stat lb_pvalue
##   <chr>        <dbl>     <dbl>
## 1 ETS(Close)    8.34     0.595
```

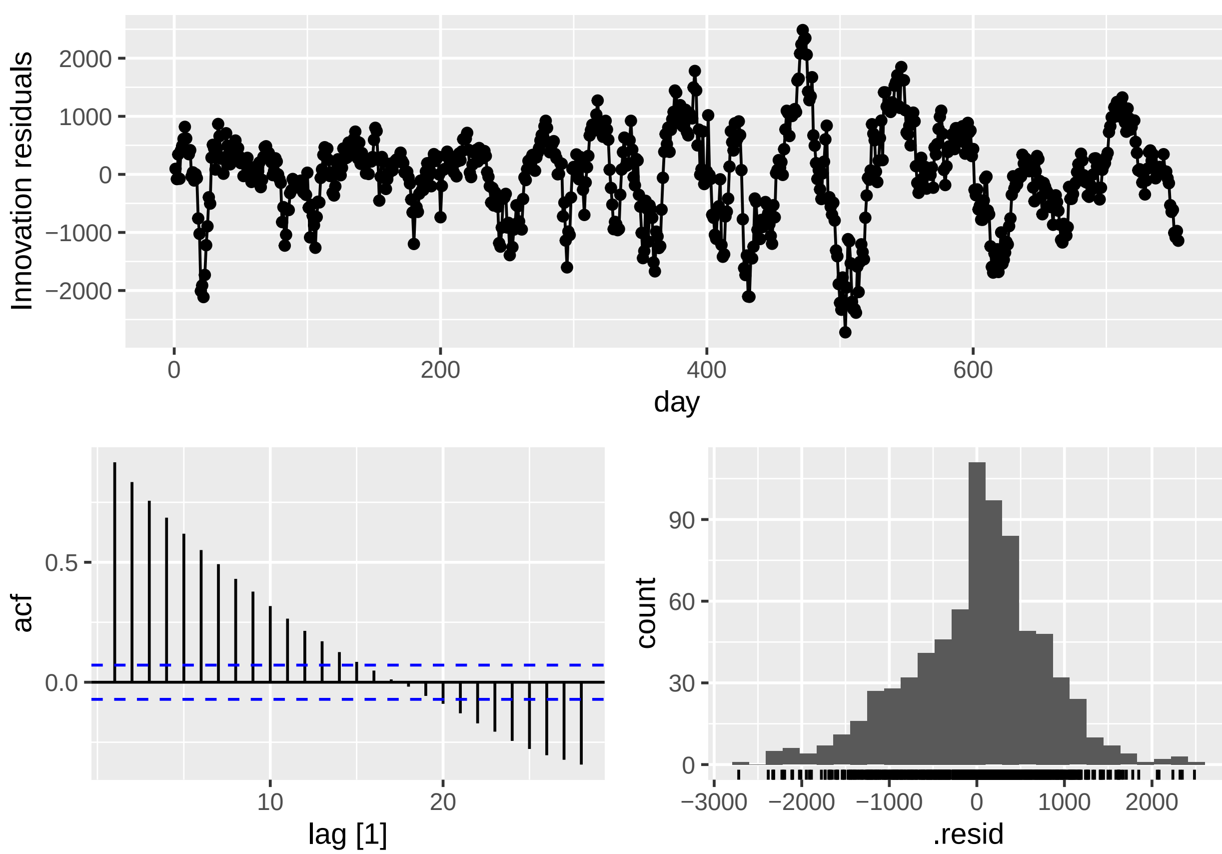
The p-value is quite large, indicating that we fail to reject the null hypothesis and thus we may treat the residuals from the ETS model as white noise.

Residual diagnostics for the Prophet model

```
library(fable.prophet)
```

```
##
## Attaching package: 'fable.prophet'

## The following object is masked from 'package:prophet':
##
##     prophet
```

```
# Fitting the Prophet model
fit.prophet <- DJI.train2 |> model(prophet(Close))

# Plotting the residuals
gg_tsresiduals(fit.prophet)
```




```
# Ljung-Box test for autocorrelation
augment(fit.prophet) |> features(.innov, ljung_box, lag=10)
```

```
## # A tibble: 1 x 3
##   .model         lb_stat lb_pvalue
##   <chr>            <dbl>     <dbl>
## 1 prophet(Close)   2993.         0
```

Not white noise!!

Further comparison of Prophet and ETS models

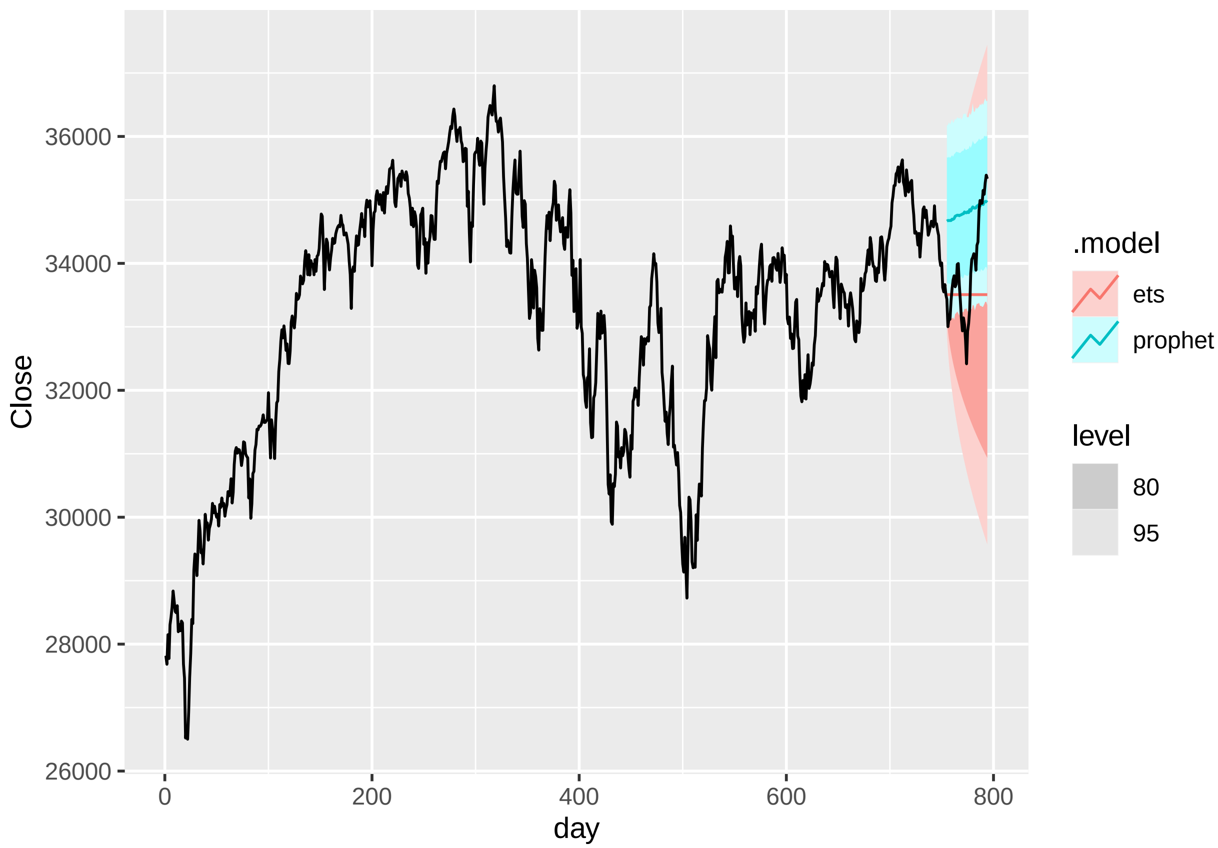
```r
# Fitting both the Prophet and ETS models using fable
DJI.fit <- DJI.train2 |>
  model(
    ets = ETS(Close),
    prophet = prophet(Close)
  )
```

```r
# Comparing the accuracy of forecasts from the ets and prophet
DJI.fc <- DJI.fit |> forecast(h = n)
DJI.fc |> accuracy(DJI)
```

```
## # A tibble: 2 x 10
##   .model  .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>   <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ets     Test   368.  875.  688.  1.03  2.01  2.89  2.76 0.909
## 2 prophet Test  -942. 1188. 1019. -2.83  3.05  4.27  3.74 0.903
```

Here, the ets model appears to be better on all values.

```r
# Comparing the plots of the prophet and ets model forecasts
DJI.fc |> autoplot(DJI)
```



When comparing both models together on the plot, it is clear the the prophet model produces forecasts with less variance than the ETS model. The prophet model forecasts also appear to capture the trend of the data more accurately. However, the residual diagnostics from the Prophet model showed that there was a significant amount of variation which was not captured by the model. The residuals showed a strong pattern and had significant autocorrelation. The ETS model was also better on all measures of accuracy

Overall, I would choose the ETS model to make forecasts over the prophet model.