STAT 4990 Final Project

Alisa Dmitrieva

2023-12-03

```
library(tsibble)
```

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

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

```
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()
```

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

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

```
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
```

```
library(prophet)
```

```
## Loading required package: Rcpp

## Loading required package: rlang
```

```
library(fabletools)
```

```r
# Download data from yahoo Finance!
#   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

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

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

```r
# Extract 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")
```


```r
# Preparing the training sample for fitting the prophet model
DJI.train <- as.data.frame(training.sample)
```

```r
DJI.train <- cbind(ds = rownames(DJI.train), DJI.train)
rownames(DJI.train ) <- 1:nrow(DJI.train)
colnames(DJI.train ) <- c ("ds", "y")

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
```

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

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

```r
DJI.future <- make_future_dataframe(DJI.prophet, periods = n)
head(DJI.future)
```

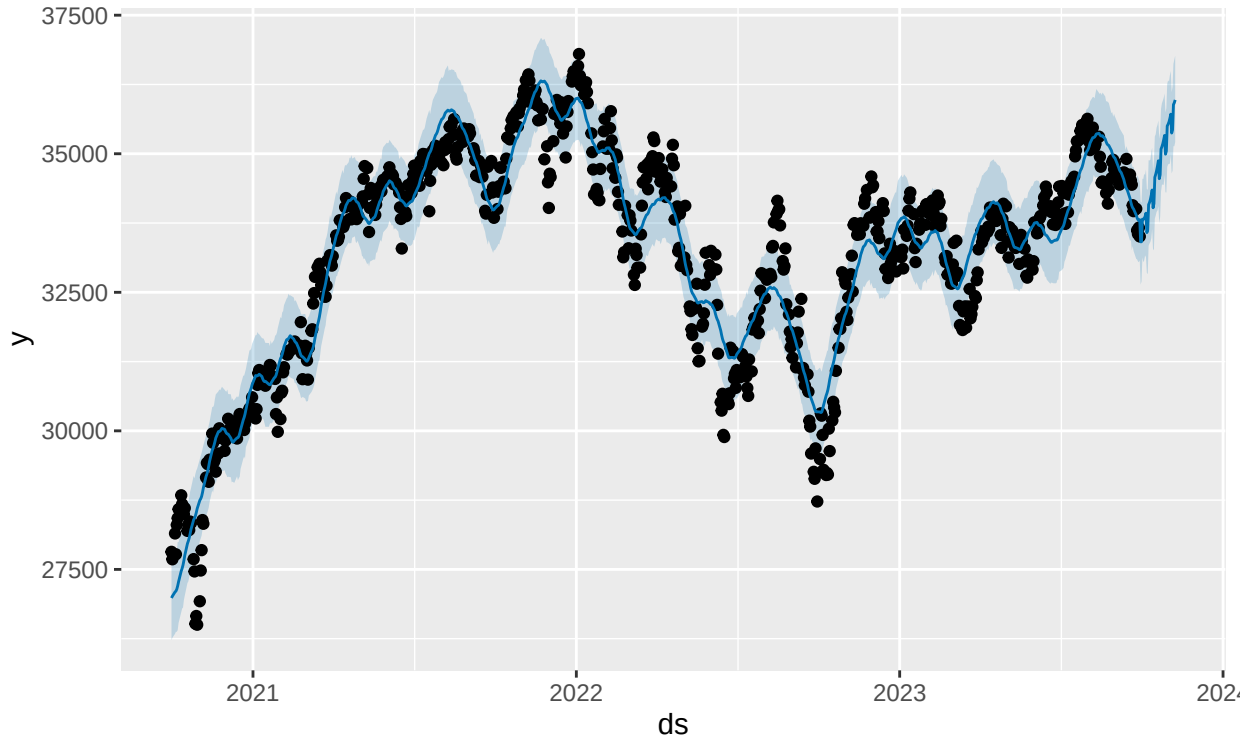
```
##           ds
## 1 2020-10-01
## 2 2020-10-02
## 3 2020-10-05
## 4 2020-10-06
## 5 2020-10-07
## 6 2020-10-08
```

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

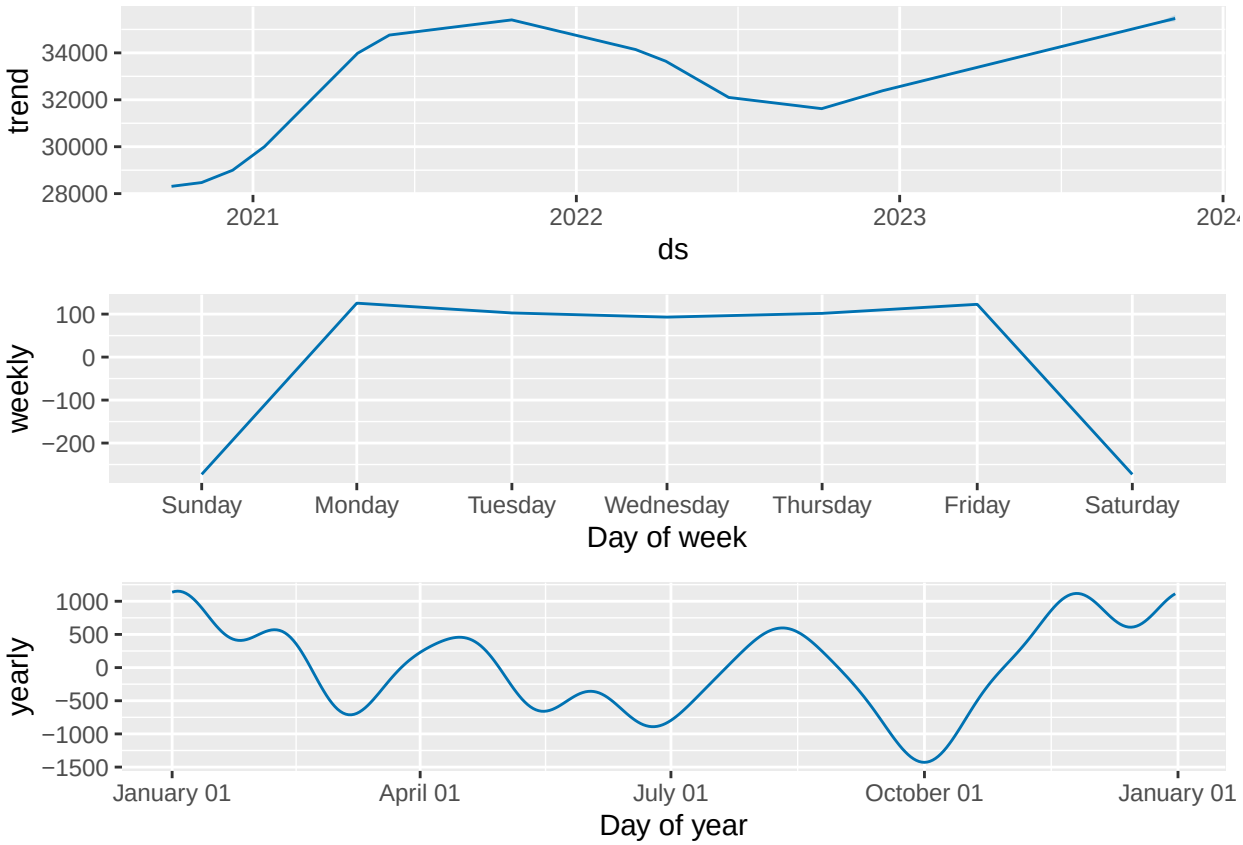
```
##           ds    trend additive_terms additive_terms_lower additive_terms_upper
## 1 2020-10-01 28310.63      -1326.394            -1326.394            -1326.394
## 2 2020-10-02 28315.38      -1299.526            -1299.526            -1299.526
## 3 2020-10-05 28329.65      -1236.659            -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   26221.92   27717.21    28310.63    28310.63 26984.24
## 2   26264.88   27792.94    28315.38    28315.38 27015.86
## 3   26338.32   27795.79    28329.65    28329.65 27092.99
## 4   26374.36   27868.27    28334.41    28334.41 27108.57
## 5   26373.60   27826.74    28339.16    28339.16 27143.33
## 6   26404.82   27994.79    28343.92    28343.92 27202.23
```

```r
# Plotting the forecast for the prophet model
plot(DJI.prophet,forecast.prophet)
```




```r
# Prophet model decomposition
prophet_plot_components(DJI.prophet, forecast.prophet)
```




```
# Calculating the sign correlation of the the daily closing price of 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)
rho_cal
```

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

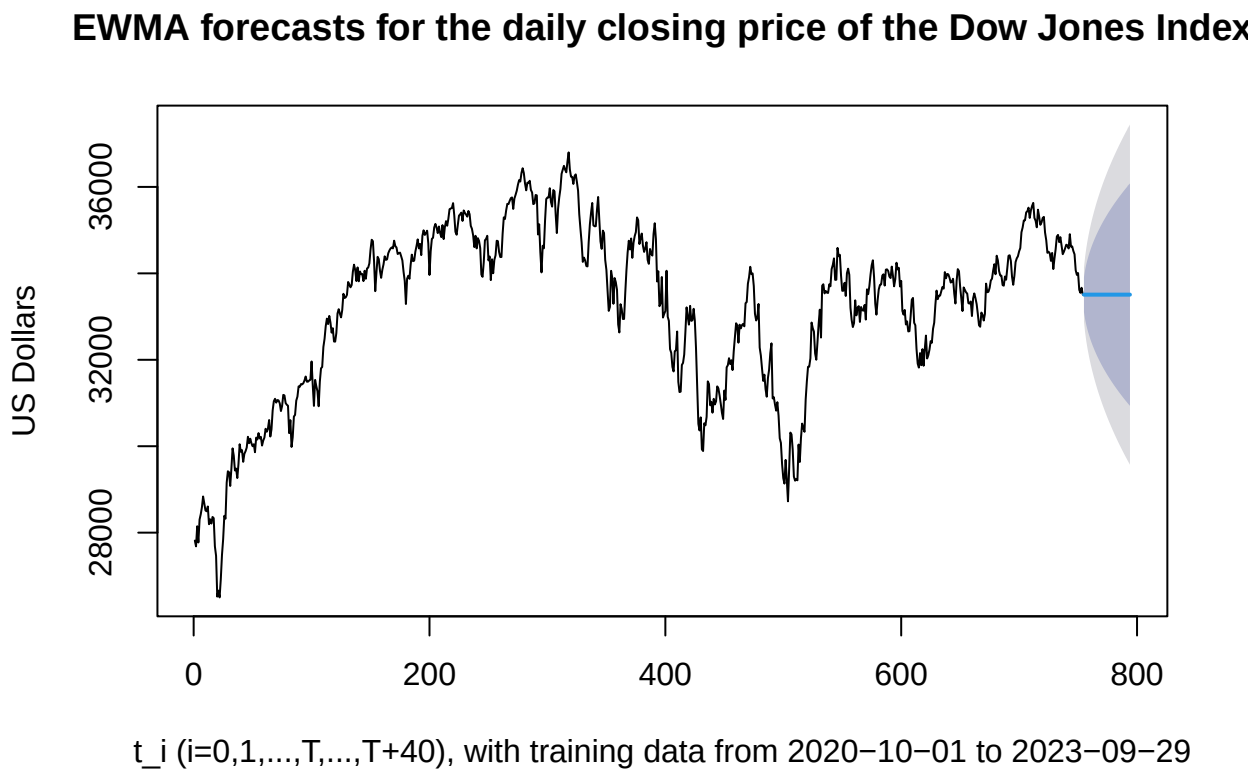
The sign correlation value indicates that the daily closing price of the Dow Jones Index follows a normal distribution

```
# 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
```

```
# Plotting the EWMA model forecasts for the next 2 months
plot(forecast(DJI.ets, h=n),
  xlab = "t_i (i=0,1,...,T,...,T+40), with training data from 2020-10-01 to 2023-09-29",
  ylab = "US Dollars",
  main = "EWMA forecasts for the daily closing price of the Dow Jones Index")
```



From a visual inspection, the prophet forecasts have smaller variance and appear to be a closer match to the plot of the real data than the EWMA model forecasts. The EWMA forecasts do not capture the trend of the Dow Jones Index data, but the prophet model forecasts appear to capture the trend,

```
# Comparing the accuracy of the EWMA and Prophet model forecasts

# 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)

DJI
```

```
## # A tsibble: 794 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
##  7 2020-10-09   28534.   28676.  28441. 28587.  324050000       28587.     7
##  8 2020-10-12   28671.   28958.  28660. 28838.  493680000       28838.     8
##  9 2020-10-13   28765.   28809.  28604. 28680.  526110000       28680.     9
## 10 2020-10-14   28731.   28793.  28462. 28514   370800000       28514     10
## # i 784 more rows
```

```
# Creating the training set for the DJI
DJI.train2 <- DJI |> filter(yearmonth(Date) <= yearmonth("2023 Sept"))
tail(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 2023-09-22   34077.   34156.  33947. 33964.  268760000       33964.   749
## 2 2023-09-25   33908.   34018.  33781. 34007.  229450000       34007.   750
## 3 2023-09-26   33863.   33880.  33570. 33619.  280100000       33619.   751
## 4 2023-09-27   33683.   33732.  33306. 33550.  300330000       33550.   752
## 5 2023-09-28   33519.   33778.  33474. 33666.  275610000       33666.   753
## 6 2023-09-29   33883.   33894.  33407. 33508.  319830000       33508.   754
```

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

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

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

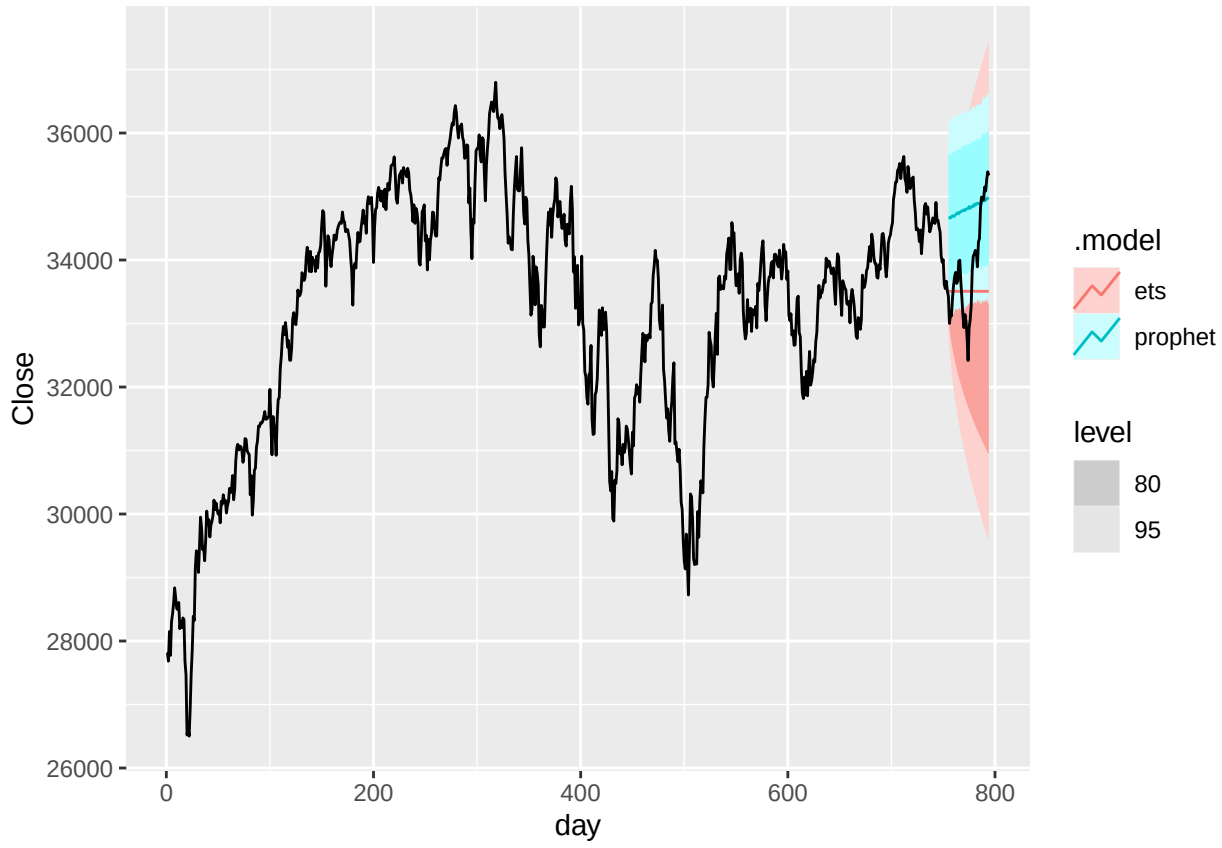
```
# Fitting both the prophet and ETS models using fable
fit <- DJI.train2 |>
  model(
    ets = ETS(Close),
    prophet = prophet(Close)
  )
```

```
# Comparing the accuracy of forecasts from the ets and prophet models
fc <- fit |> forecast(h = n)
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  -941. 1190. 1019. -2.83  3.05  4.27  3.75 0.903
```

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

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


Close
36000
34000
32000
30000
28000
26000
0
200
400
600
800
day
.model
ets
prophet
level
80
95

When comparing both models together on the plot, it is clear the the prophet model produces forecasts with less variance than the EWMA (ets) model. The prophet model forecasts also appear to capture the trend of the data more accurately.

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