Kumari_Herath_T00655616

Kumari Herath

2023-12-01

```
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     3.4.4
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
## Warning: package 'fabletools' was built under R version 4.3.2
```

```
## -- 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)
```

```
## Warning: package 'forecast' was built under R version 4.3.2
```

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

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

```
## Warning: package 'quantmod' was built under R version 4.3.2
```

```
## 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(fable.prophet)
```

```
## Warning: package 'fable.prophet' was built under R version 4.3.2
```

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

```
library(prophet)
```

```
## Warning: package 'prophet' was built under R version 4.3.2
```

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

```
##
## Attaching package: 'prophet'
```

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

```
library(fabletools)
```

```
# Download data from yahoo Finance!
start.date = '2020-10-01' # starting date of stock
end.date = '2023-11-29' # ending date of stock
# Download the selected stocks from Yahoo finance using `quantmod` package
getSymbols("^DJI", src = "yahoo", from = start.date, to = end.date, auto.assign = TRUE)
```

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

```
# Get close price
Close = DJI$DJI.Close
head(Close)
```

```
##            DJI.Close
## 2020-10-01  27816.90
## 2020-10-02  27682.81
## 2020-10-05  28148.64
## 2020-10-06  27772.76
## 2020-10-07  28303.46
## 2020-10-08  28425.51
```

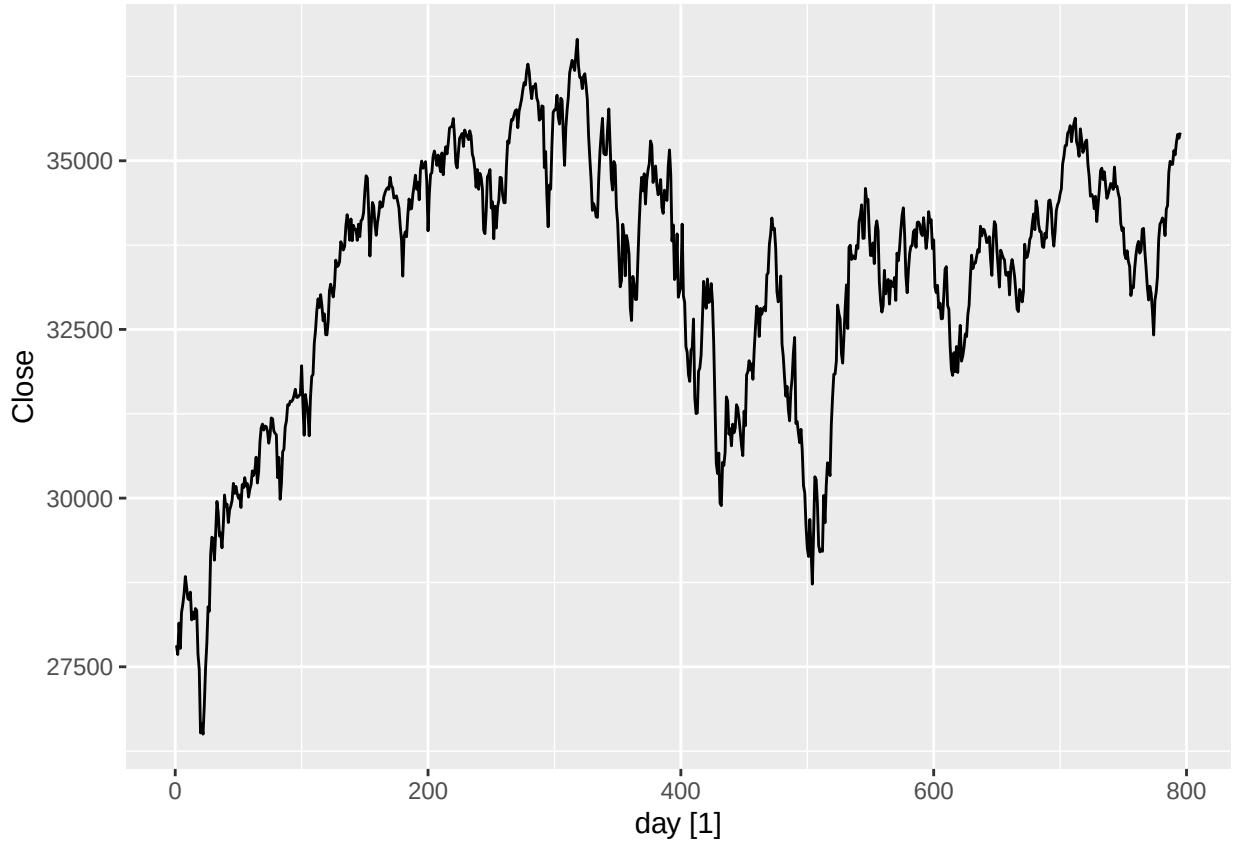
```
# Create date variable
DJI <- zoo::fortify.zoo(DJI)
DJI <- DJI %>% rename(c("Date" = "Index", "Close" = "DJI.Close"))

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

# Reindex by taking care of missing values
```

```r
DJI <- DJI |>
mutate(day = row_number()) |>
update_tsibble(index = day, regular = TRUE)

DJI
```

```
## # A tsibble: 795 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 785 more rows
```

```r
#Create train and test sets for GOOG
DJI_train <- DJI |>  filter(yearmonth(Date) <= yearmonth("2023 Sept"))
DJI_test <- DJI |> filter(yearmonth(Date) > yearmonth("2023 Oct"))
```

```r
#  Make the data stationary
return.DJI = diff(log(Close ))
return.DJI <- na.omit(return.DJI)

DJI |> autoplot(Close)
```




```
summary(DJI)
```

```
##       Date                DJI.Open        DJI.High        DJI.Low
##  Min.   :2020-10-01   Min.   :26481   Min.   :26639   Min.   :26144
##  1st Qu.:2021-07-17   1st Qu.:32162   1st Qu.:32406   1st Qu.:31874
##  Median :2022-04-29   Median :33733   Median :33897   Median :33520
##  Mean   :2022-04-30   Mean   :33229   Mean   :33423   Mean   :33029
##  3rd Qu.:2023-02-13   3rd Qu.:34584   3rd Qu.:34782   3rd Qu.:34438
##  Max.   :2023-11-28   Max.   :36723   Max.   :36953   Max.   :36636
##      Close          DJI.Volume          DJI.Adjusted        day
##  Min.   :26502   Min.   :117040000   Min.   :26502   Min.   :  1.0
##  1st Qu.:32158   1st Qu.:291555000   1st Qu.:32158   1st Qu.:199.5
##  Median :33731   Median :328910000   Median :33731   Median :398.0
##  Mean   :33234   Mean   :344528616   Mean   :33234   Mean   :398.0
##  3rd Qu.:34582   3rd Qu.:381760000   3rd Qu.:34582   3rd Qu.:596.5
##  Max.   :36800   Max.   :811890000   Max.   :36800   Max.   :795.0
```

```
# Fit the models

# Fit Benchmark model
Benchmark_fit <- DJI_train |>
  model( NAIVE(Close),
         MEAN(Close),
         RW(Close ~ drift())
         )
```

```r
# Fit ARIMA model
arima_fit <- DJI_train |>
  model(ARIMA(Close))

# Fit NN model
NN_fit <- DJI_train |>
  model(NNETAR(sqrt(Close)))

report(Benchmark_fit)
```

```
## Warning in report.mdl_df(Benchmark_fit): Model reporting is only supported for
## individual models, so a glance will be shown. To see the report for a specific
## model, use `select()` and `filter()` to identify a single model.
```

```
## # A tibble: 3 x 2
##   .model                sigma2
##   <chr>                  <dbl>
## 1 NAIVE(Close)         100810.
## 2 MEAN(Close)         3951772.
## 3 RW(Close ~ drift())  100810.
```

```r
report(arima_fit)
```

```
## Series: Close
## Model: ARIMA(0,1,0)
##
## sigma^2 estimated as 100734:  log likelihood=-5405.83
## AIC=10813.65   AICc=10813.66   BIC=10818.28
```

```r
report(NN_fit)
```

```
## Series: Close
## Model: NNAR(1,1)
## Transformation: sqrt(Close)
##
## Average of 20 networks, each of which is
## a 1-1-1 network with 4 weights
## options were - linear output units
##
## sigma^2 estimated as 0.7629
```

```r
#Get best fit ARIMA Model
arima_fit <- DJI_train |>
model(
arima010 = ARIMA(Close ~ 1 + pdq(0, 1, 0)),
arima011 = ARIMA(Close ~ 1 + pdq(0, 1, 1)),
arima012 = ARIMA(Close ~ 1 + pdq(0, 1, 2)),
arima013 = ARIMA(Close ~ 1 + pdq(0, 1, 3)),
arima110 = ARIMA(Close ~ 1 + pdq(1, 1, 0)),
arima111 = ARIMA(Close ~ 1 + pdq(1, 1, 1)),
```

```
arima112 = ARIMA(Close ~ 1 + pdq(1, 1, 2)),
arima113 = ARIMA(Close ~ 1 + pdq(1, 1, 3)),
arima210 = ARIMA(Close ~ 1 + pdq(2, 1, 0)),
arima211 = ARIMA(Close ~ 1 + pdq(2, 1, 1)),
arima212 = ARIMA(Close ~ 1 + pdq(2, 1, 2)),
arima213 = ARIMA(Close ~ 1 + pdq(2, 1, 3)),
arima310 = ARIMA(Close ~ 1 + pdq(3, 1, 0)),
arima311 = ARIMA(Close ~ 1 + pdq(3, 1, 1)),
arima312 = ARIMA(Close ~ 1 + pdq(3, 1, 2)),
arima313 = ARIMA(Close ~ 1 + pdq(3, 1, 3))
)
```

```
## Warning in wrap_arima(y, order = c(p, d, q), seasonal = list(order = c(P, :
## possible convergence problem: optim gave code = 1
```

```
## Warning in sqrt(diag(best$var.coef)): NaNs produced
```

```
## Warning in sqrt(diag(best$var.coef)): NaNs produced
```

```
arima_fit|>
  glance() |>
  arrange(AICc) |>
  select(.model, AICc)
```

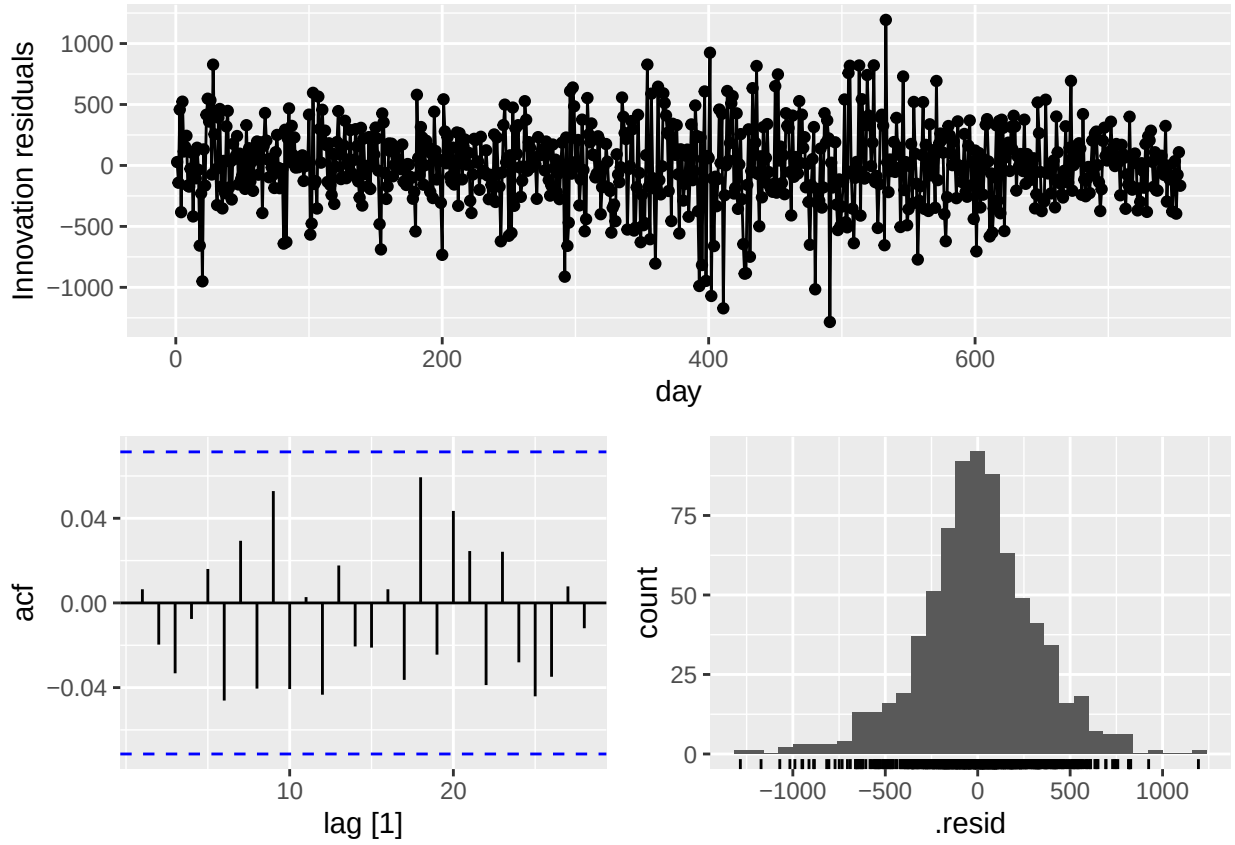
```
## # A tibble: 16 x 2
##    .model     AICc
##    <chr>     <dbl>
##  1 arima010 10815.
##  2 arima111 10816.
##  3 arima011 10817.
##  4 arima110 10817.
##  5 arima212 10818.
##  6 arima211 10818.
##  7 arima210 10819.
##  8 arima012 10819.
##  9 arima213 10820.
## 10 arima013 10820.
## 11 arima310 10820.
## 12 arima113 10820.
## 13 arima112 10820.
## 14 arima313 10820.
## 15 arima311 10822.
## 16 arima312 10824.
```

```
best_arima_fit <- DJI_train|>
model(ARIMA(Close ~ 1 + pdq(0, 1, 0)))

best_arima_fit |> report()
```

```
## Series: Close
## Model: ARIMA(0,1,0) w/ drift
```

```
##
## Coefficients:
##       constant
##         7.5572
## s.e.   11.5629
##
## sigma^2 estimated as 100811:  log likelihood=-5405.61
## AIC=10815.23   AICc=10815.24   BIC=10824.48
```

```
# get the residuals for arima model
best_arima_fit |> gg_tsresiduals()
```




```
augment(best_arima_fit) |> features(.innov, ljung_box, dof = 1, lag = 10)
```

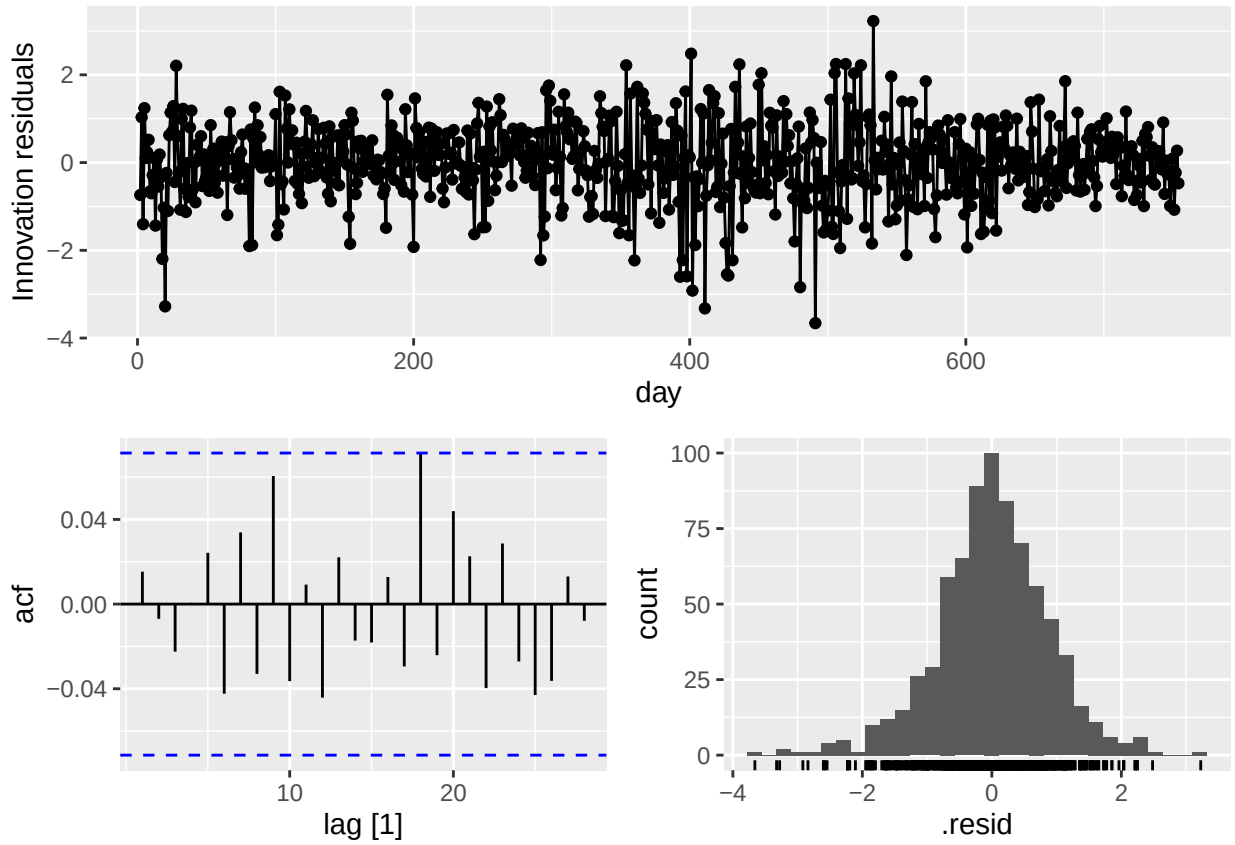
```
## # A tibble: 1 x 3
##   .model                          lb_stat lb_pvalue
##   <chr>                             <dbl>     <dbl>
## 1 ARIMA(Close ~ 1 + pdq(0, 1, 0))    8.34     0.500
```

```
# get the residuals for NN model
NN_fit |> gg_tsresiduals()
```

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

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_bin()`).
```




```
augment(NN_fit) %>% features(.resid, ljung_box, lag=10, dof=0)
```

```
## # A tibble: 1 x 3
##   .model              lb_stat lb_pvalue
##   <chr>                 <dbl>     <dbl>
## 1 NNETAR(sqrt(Close))    7.67     0.661
```

```
# For ARIMA model:
# ACF function shows that the residuals are white noise.
# The residual follows a normal distribution.

# For NN model:
# ACF function shows that the residuals are white noise.
# The residual follows a normal distribution.
```

```
accuracy <- Benchmark_fit |>
  forecast(h = 2) |>
  accuracy(DJI)

accuracy
```

```
## # A tibble: 3 x 10
```

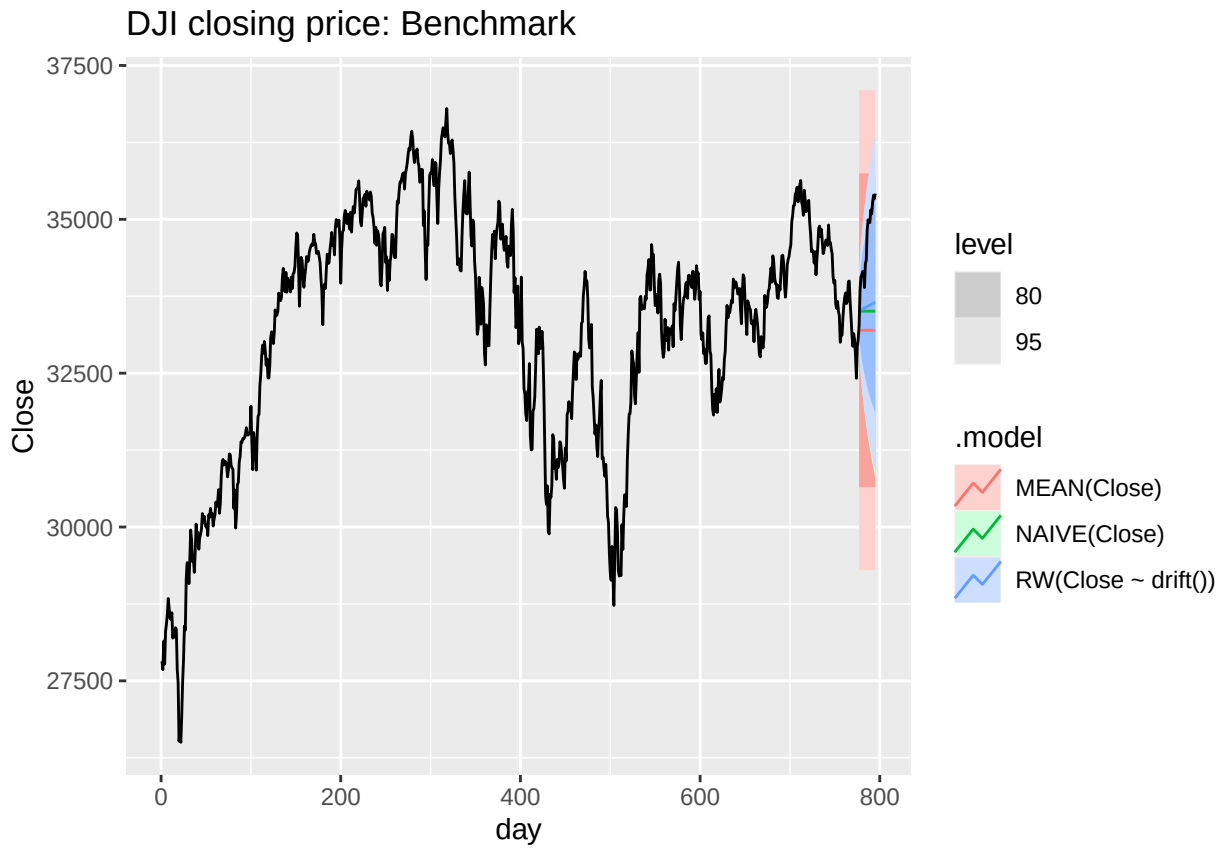
```
##   .model              .type     ME  RMSE   MAE     MPE  MAPE  MASE RMSSE  ACF1
##   <chr>               <chr>  <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 MEAN(Close)         Test    20.7  216.  215.  0.0582 0.648 0.903 0.682  -0.5
## 2 NAIVE(Close)        Test  -290.   361.  290. -0.876  0.876 1.21  1.14   -0.5
## 3 RW(Close ~ drift()) Test  -301.   372.  301. -0.910  0.910 1.26  1.17   -0.5
```

```
accuracy_NN_fit <- NN_fit |>
  forecast(h = 40 ,times = 10) |>
  accuracy(DJI)

accuracy_NN_fit
```

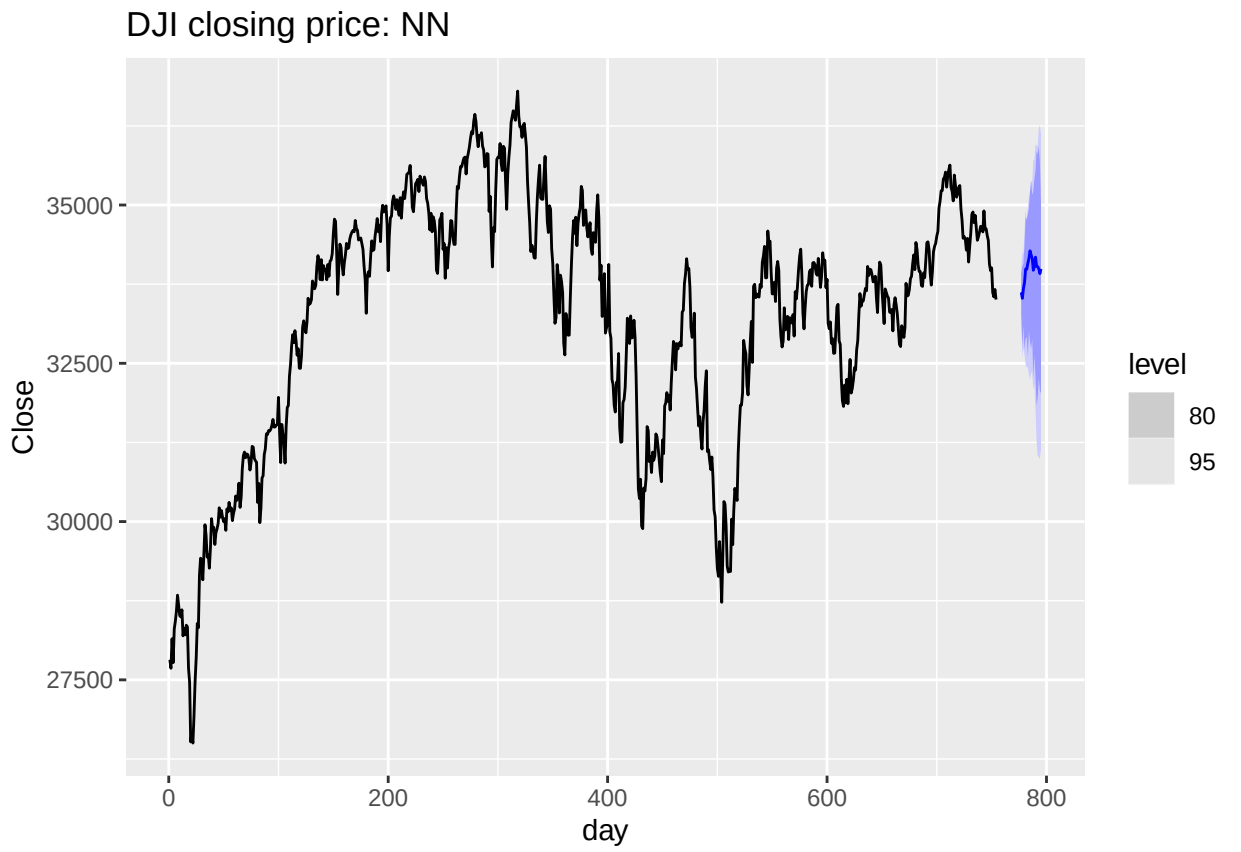
```
## # A tibble: 1 x 10
##   .model              .type     ME  RMSE   MAE     MPE  MAPE  MASE RMSSE  ACF1
##   <chr>               <chr>  <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 NNETAR(sqrt(Close)) Test  -132.   714.  572. -0.437  1.69  2.40  2.25 0.884
```

```
#The MEAN method has the smallest RMSE value among these models.
```

```
# perform forecast on test set

# For Benchmark model
Benchmark_fit |>
  forecast(DJI_test) |>
  autoplot(DJI)+
  labs(title="DJI closing price: Benchmark")
```

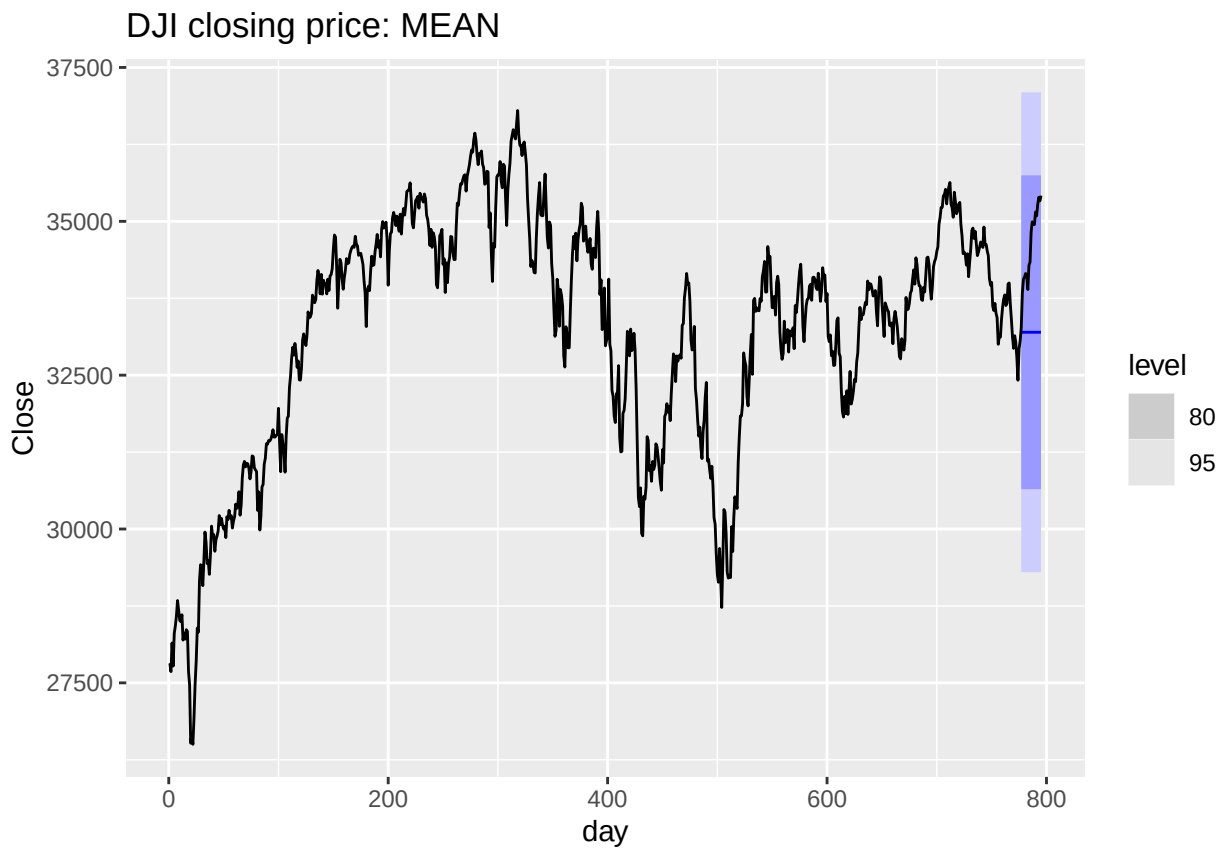


```r
# For NN model
NN_fit |>
  forecast(DJI_test, times= 10) |>
  autoplot(DJI_train) +
  labs(title="DJI closing price: NN")
```

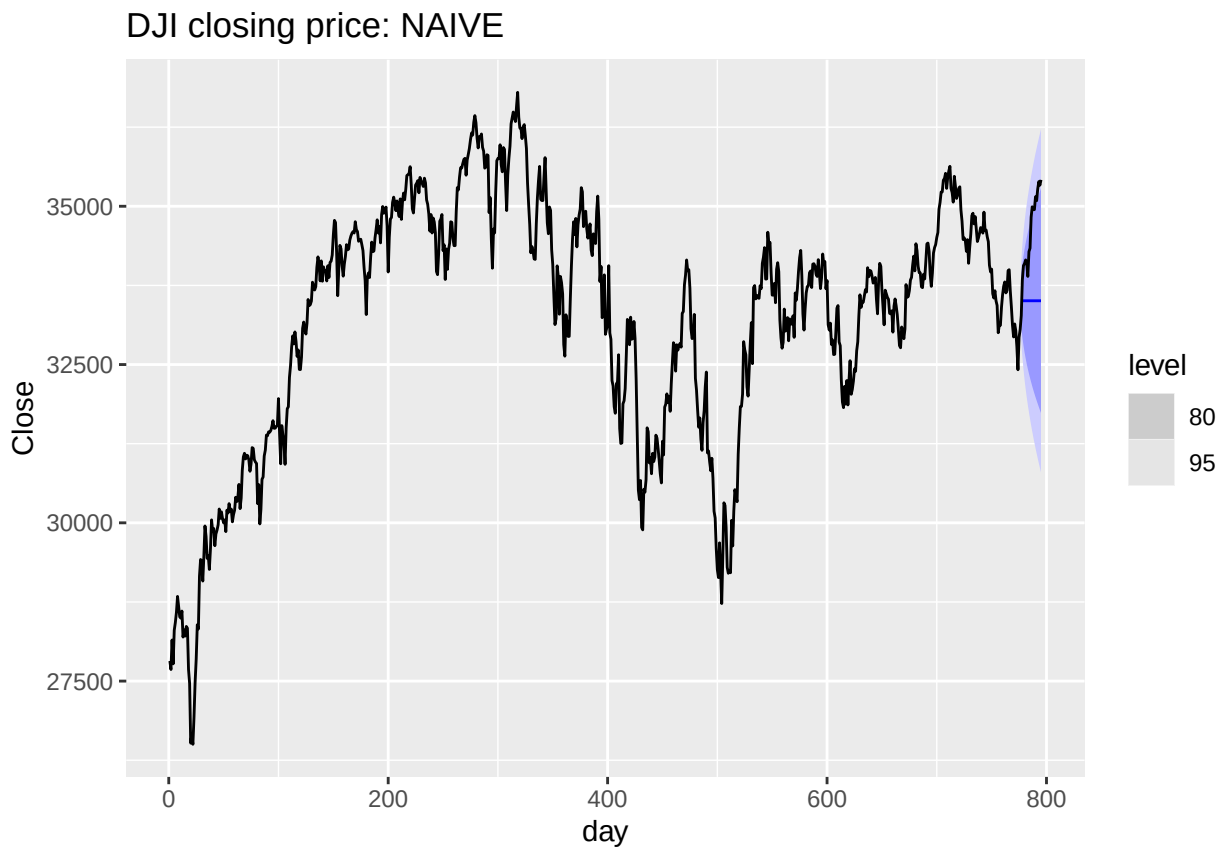



```r
# I did not do a forecast for ARIMA model since the best ARIMA model is (0,1,0).
# Therefore, I ruled out ARIMA model.
```

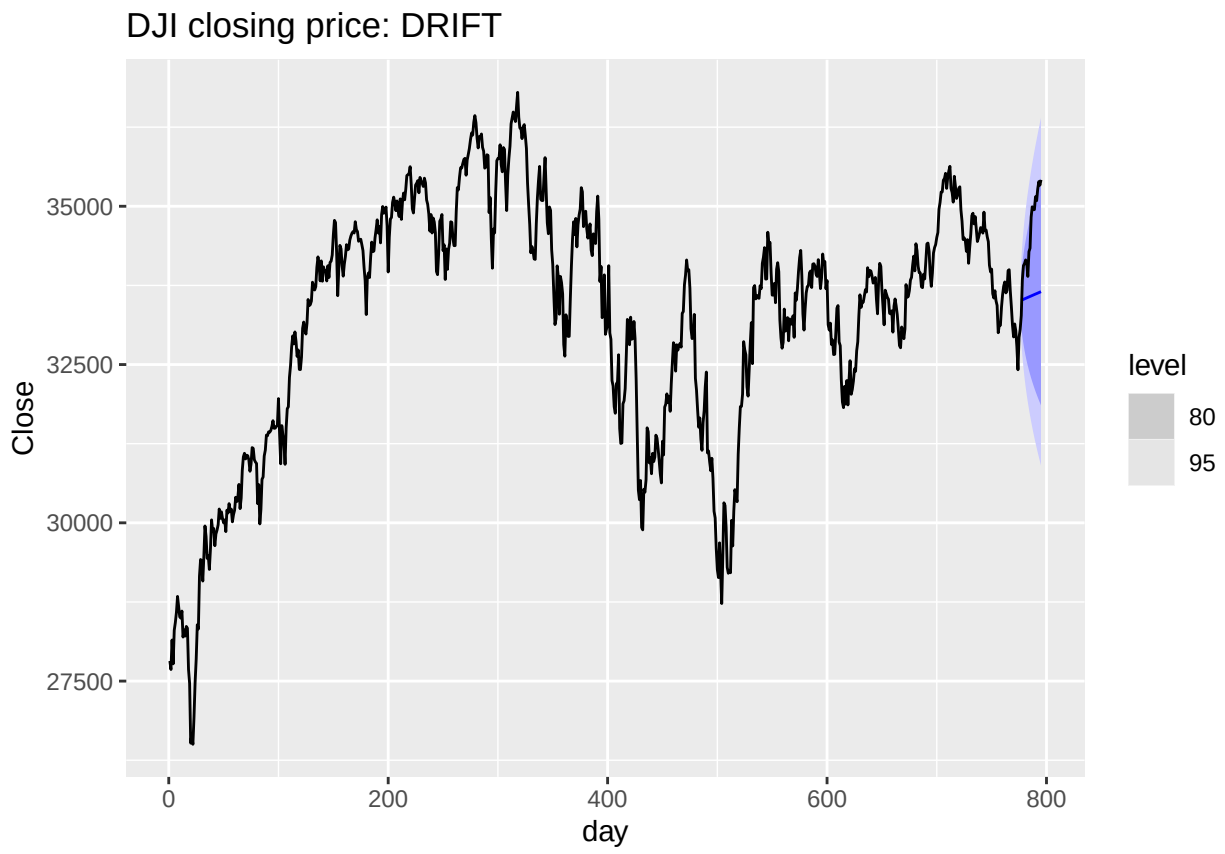
```r
# forecast the models seperately for better visualization
mean_fit_model <- DJI_train |>
 model(MEAN(Close))

naive_fit_model <- DJI_train |>
 model(NAIVE(Close))

drift_fit_model <- DJI_train |>
 model(RW(Close ~ drift()))

mean_fit_model |>
  forecast(DJI_test) |>
  autoplot(DJI) +
  labs(title="DJI closing price: MEAN")
```




```
naive_fit_model |>
  forecast(DJI_test) |>
  autoplot(DJI) +
  labs(title="DJI closing price: NAIVE")
```




```
drift_fit_model |>
  forecast(DJI_test) |>
  autoplot(DJI) +
  labs(title="DJI closing price: DRIFT")
```




```
# Even though MEAN model has the smallest RMSE, the certainty of the forecast is less than other method
# Therefore, I choose NAIVE model as the best model among these model and it has the second smallest RM
# and NAIVE RMSE is not very larger than MEAN RMSE.
```

```
# cross validation
DJI_CV <- DJI|>
  stretch_tsibble(.init = 300, .step = 1) |>
  relocate(Date, .id)
head(DJI_CV)
```

```
## # A tsibble: 6 x 9 [1]
## # Key:       .id [1]
##   Date         .id DJI.Open DJI.High DJI.Low  Close DJI.Volume DJI.Adjusted
##   <date>     <int>    <dbl>    <dbl>   <dbl>  <dbl>      <dbl>        <dbl>
## 1 2020-10-01     1   27941.   28041.  27669. 27817.  373450000       27817.
## 2 2020-10-02     1   27536.   27861.  27383. 27683.  392770000       27683.
## 3 2020-10-05     1   27825.   28163.  27825. 28149.  318210000       28149.
## 4 2020-10-06     1   28214.   28354.  27728. 27773.  435030000       27773.
## 5 2020-10-07     1   27971.   28370.  27971. 28303.  328750000       28303.
## 6 2020-10-08     1   28349.   28459.  28266. 28426.  314750000       28426.
## # i 1 more variable: day <int>
```

```
accuracy_NAIVE <- naive_fit_model |>
  forecast(h = 40) |>
  accuracy(DJI)
```

```
accuracy_NAIVE
```

```
## # A tibble: 1 x 10
##   .model       .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>        <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 NAIVE(Close) Test   368.  875.  688.  1.03  2.01  2.89  2.76 0.909
```