TS Competition Final Report

Daniel & Ellie

2025-04-25

Github Link

https://github.com/Dsw52/Shang_Whitehead_TS_Competition

Top Five Models

Import Data and Packages

```
library(lubridate)
```

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

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

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

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

```
library(Kendall)
library(tseries)
library(outliers)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr   1.1.4     v stringr 1.5.1
## v forcats 1.0.0     v tibble  3.2.1
## v purrr   1.0.2     v tidyr   1.3.1
## v readr   2.1.5
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(smooth)
```

```
## Loading required package: greybox
## Package "greybox", v2.0.3 loaded.
##
##
## Attaching package: 'greybox'
##
## The following object is masked from 'package:tidyr':
##
##     spread
##
## The following object is masked from 'package:lubridate':
##
##     hm
##
## This is package "smooth", v4.1.1
## Any thoughts or suggestions about the package? Have you found a bug? File an issue on github: https:/
```

```
library(zoo)
```

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

```
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
##
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
library(readxl)
library(dplyr)
library(cowplot)
```

```
##
## Attaching package: 'cowplot'
##
## The following object is masked from 'package:lubridate':
##
##     stamp
```

```r
# Read in data
load_data <- read_excel(path = path.expand("~/Downloads/load.xlsx"), col_names = TRUE)
relative_humidity_data <- read_excel(path = path.expand("~/Downloads/relative_humidity.xlsx"), col_name
temperature_data <- read_excel(path = path.expand("~/Downloads/temperature.xlsx"), col_names = TRUE)

load_data <- mutate(load_data, date = ymd(date))

# Convert hourly data into daily training data (1/2005-12/2009)
training_load_data <- load_data %>%
  filter(date < as.Date("2010-01-01")) %>%
  select(h1:h24) %>%
  rowMeans(na.rm = TRUE) %>%
  msts(seasonal.periods =c(7,365.25), start = c(2005, 1, 1))

# Without date cutoff
all_load_data <- load_data %>%
  select(h1:h24) %>%
  rowMeans(na.rm = TRUE) %>%
  msts(seasonal.periods =c(7,365.25), start=c(2005, 1))
```

Neural network selection (responsible for 5/5 most accurate submissions)

Neural networks use past values and seasonal patterns to model complex time series like load data. Unlike traditional models like ARIMA, they can capture non-linear relationships in the data. By using lagged inputs and seasonal signals, neural networks handle both short- and long-term patterns, similar to SARIMA models. However, they offer an advantage by learning more complex, non-linear behaviors in the data.

In our original neural network models, the autoregressive parameters were set to p=1 and P=1. This allows the models to learn both short-term dependencies and long-term seasonal structure. Combined with the Fourier terms, this structure enhances the model's ability to capture the complex, non-linear, and multi-seasonal behavior characteristic of ERCOT load data. We manipulated the K parameters, and found that the K=c(1,4) and K=c(2,4) models were the most accurate in terms of accuracy in forecasting.

In the model below, we implemented a function to automatically select the most accurate parameters for autoregressive inputs and Fourier terms. One of the top-performing configurations was p = 4, P = 0, k1 = 1, k2 = 2, which achieved a MAPE of 21.68631—though it resulted in a slightly higher score on the Kaggle submission. This setup showed improved accuracy over our earlier models, which used p = 1, P = 1, highlighting the benefit of including more lagged inputs and removing seasonal lags.

After running multiple trials, we found that models NN1, NN3, and NN4 consistently produced the best results in terms of MAPE. Among these, NN3 was the most accurate with a MAPE of 21.98, followed by NN1 at 22.20, and NN4 at 22.29. For the Kaggle submissions, we also experimented with averaged forecasts, such as nn1nn3nn4 and nn1nn3, aiming to smooth out predictions. While these combinations offered stable forecasts, their accuracy was slightly lower than that of our best individual model, NN3.

```r
best_mape <- Inf
best_params <- list()

for (k1 in 1:3) {
  for (k2 in 1:4) {
    xreg_train <- fourier(training_load_data, K =c(k1, k2))
```

```
    for (p in 1:4) {
      for (P in 0:1) {

        nnfit <- nnetar(training_load_data, p=p, P=P, xreg=xreg_train, repeats=20)
        nnfor <- forecast(nnfit, h=59, xreg=fourier(training_load_data, K =c(k1, k2), h=59))
        acc <- accuracy(nnfor$mean, all_load_data)
        mape <- acc["Test set", "MAPE"]
        if (mape < best_mape) {
          best_mape <- mape
          best_params <- list(p=p, P=P, k1=k1, k2=k2)
        }

      }
      }
  }

}

best_mape
```

```
## [1] 21.46998
```

```
best_params
```

```
## $p
## [1] 3
##
## $P
## [1] 0
##
## $k1
## [1] 2
##
## $k2
## [1] 1
```

```
nnfit <- nnetar(all_load_data,
                p=best_params$p,
                P=best_params$P,
                repeats=20,
                xreg=fourier(all_load_data, K=c(best_params$k1, best_params$k2)))
nnfor <- forecast(nnfit, h=59, xreg=fourier(all_load_data, K=c(best_params$k1, best_params$k2), h=59))

# 1. Formatting the decimal dates as actual dates
start_date_forecast <- as.Date("2011-01-01")
forecast_values <- as.numeric(nnfor$mean)
forecast_dates <- seq(start_date_forecast, by = "day", length.out = length(forecast_values))

# to CSV
formatted_forecast <- data.frame(
  date = forecast_dates,
  load = forecast_values
```

```
)

print(formatted_forecast)
```

```
##          date     load
## 1  2011-01-01 4948.691
## 2  2011-01-02 4866.685
## 3  2011-01-03 4858.659
## 4  2011-01-04 4859.425
## 5  2011-01-05 4784.212
## 6  2011-01-06 4719.900
## 7  2011-01-07 4768.275
## 8  2011-01-08 4799.575
## 9  2011-01-09 4674.295
## 10 2011-01-10 4601.627
## 11 2011-01-11 4664.146
## 12 2011-01-12 4644.599
## 13 2011-01-13 4609.020
## 14 2011-01-14 4697.316
## 15 2011-01-15 4732.779
## 16 2011-01-16 4608.609
## 17 2011-01-17 4549.852
## 18 2011-01-18 4623.640
## 19 2011-01-19 4603.042
## 20 2011-01-20 4577.335
## 21 2011-01-21 4677.710
## 22 2011-01-22 4708.371
## 23 2011-01-23 4584.100
## 24 2011-01-24 4526.848
## 25 2011-01-25 4585.487
## 26 2011-01-26 4552.547
## 27 2011-01-27 4528.669
## 28 2011-01-28 4629.358
## 29 2011-01-29 4652.898
## 30 2011-01-30 4520.212
## 31 2011-01-31 4450.353
## 32 2011-02-01 4492.185
## 33 2011-02-02 4447.475
## 34 2011-02-03 4424.867
## 35 2011-02-04 4521.969
## 36 2011-02-05 4543.632
## 37 2011-02-06 4395.400
## 38 2011-02-07 4292.269
## 39 2011-02-08 4319.686
## 40 2011-02-09 4267.729
## 41 2011-02-10 4249.808
## 42 2011-02-11 4336.996
## 43 2011-02-12 4356.840
## 44 2011-02-13 4205.211
## 45 2011-02-14 4047.475
## 46 2011-02-15 4039.134
## 47 2011-02-16 3996.686
## 48 2011-02-17 3989.478
```

```
## 49 2011-02-18 4046.019
## 50 2011-02-19 4084.616
## 51 2011-02-20 3962.598
## 52 2011-02-21 3745.093
## 53 2011-02-22 3675.029
## 54 2011-02-23 3649.659
## 55 2011-02-24 3656.020
## 56 2011-02-25 3706.757
## 57 2011-02-26 3796.973
## 58 2011-02-27 3712.265
## 59 2011-02-28 3473.284
```

```
write.csv(formatted_forecast, "nn#.csv", row.names=FALSE)
```

Merged NN1 and NN3

This shows the code for the merged NN1 and NN3 data

```
# Read the two CSV files
nn1 <- read_csv("~/Downloads/nn1.csv")
```

```
## Rows: 59 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (1): load
## date (1): date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
nn3 <- read_csv("nn3.csv")
```

```
## Rows: 59 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (1): load
## date (1): date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Merge the datasets by 'date'
mergednn1_3 <- merge(nn1, nn3, by = "date", suffixes = c("_1", "_3"))

# Find the average of the two forecasts
mergednn1_3$load <- rowMeans(mergednn1_3[, c("load_1", "load_3")])

# Keep only the date and averaged load
averagednn1_3 <- mergednn1_3[, c("date", "load")]

# Write the result to a new CSV file
write_csv(averagednn1_3, "nn1+nn3.csv")
```

Merged NN1NN3NN4

This shows the code for the merged NN1, NN3, and NN4 data

```
nn1 <- read_csv("~/Downloads/nn1.csv", col_names = TRUE)
```

```
## Rows: 59 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (1): load
## date (1): date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
nn3 <- read_csv("nn3.csv", col_names = TRUE)
```

```
## Rows: 59 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (1): load
## date (1): date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
nn4 <- read_csv("nn4.csv", col_names = TRUE)
```

```
## Rows: 59 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (1): load
## date (1): date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
mergednn1_3 <- merge(nn1, nn3, by = "date", suffixes = c("_1", "_3"))

mergednn1_3_4 <- merge(mergednn1_3, nn4, by = "date")
names(mergednn1_3_4)[names(mergednn1_3_4) == "load"] <- "load_4"

mergednn1_3_4$load <- rowMeans(mergednn1_3_4[, c("load_1", "load_3", "load_4")])

averagednn1_3_4 <- mergednn1_3_4[, c("date", "load")]

write_csv(averagednn1_3_4, "nn1+nn3+nn4.csv")
```