Forecasting Competition Report

Rachael Stephan & Rosie Wu

Github Repository

Introduction

This report contains the final deliverable for the forecasting competition for ENV 797. The objective of this competition was to produce the best time series forecast of a daily load using various models and exogenous factors of humidity and temperature. This report contains the workings for the 5 top performing models.

Data Source

Data was retrieved from the Kaggle competition site. All data was provided by instructor Luana Lima.

Wrangling

The hourly load data frame were uploaded into R and wrangled as follows to produced both daily and hourly data.

Daily Data:

- Format date columns as date.
- Aggregate data frame by row with `rowwise()`
- Calculate the daily load as the mean of every hour.
- Ungroup the data frame from `rowwise()`
- Drop unnecessary columns (i.e., hourly data and meter id)

```r
#load data and create a daily average data frame
dailyload <- readxl::read_xlsx("./Data/Raw/load.xlsx") %>%
    mutate(date = as.Date(date, format = "%Y-%m-%d")) %>%
    rowwise() %>%
    mutate(daily_average = mean(c_across(h1:h24), na.rm = TRUE)) %>%
    ungroup() %>%
    select(-c(h1:h24), -meter_id)
```

Hourly Data:

- Format date columns as date.
- Calculate the daily load as the mean of every hour.
- pivot data frame longer to put the hour into one column and the hourly load into another.
- Extract hour integer and reformat hour column as integer data.

- Drop unnecessary columns (i.e., meter id)

```r
#load data and create an hourly average data frame
hourlyload <- readxl::read_xlsx("./Data/Raw/load.xlsx") %>%
    mutate(date = as.Date(date, format = "%Y-%m-%d")) %>%
    pivot_longer(cols = starts_with("h"),
                 names_to = "hour",
                 values_to = "load")%>%
    mutate(hour = as.integer(substring(hour,2))) %>%
    select(-meter_id)
```

Section Data

Data was wrangled into training (n = 2141) and test (n = 50) observations. These were used to evaluate our models before uploading to Kaggle.

```r
dailyload_train <- head(dailyload,
                        (length(dailyload$daily_average)-50))

dailyload_test <- tail(dailyload, 50)
```

Time Series Objects

The timeseries objects were created with a weekly and yearly seasonality in the `msts`. The `ts` object was created with a frequency of a year. Representative code of this process is shown below with the daily load data frame.

```r
#create time series objects
load_daily_msts <- msts(dailyload$daily_average,
                            seasonal.periods =c(7,365.25),
                            start=c(2005,1,1))

load_daily_ts <- ts(dailyload$daily_average,
                        frequency = 365.25,
                        start=c(2005,1,1))
```

Forecasting Methods

To evaluate our own code before uploading onto kaggle, we designated a test period as the last 50 observations in the load data set and the training period as all observations but the last 50. These test forecasts were evaluated based on their MAPE.

We ran multiple types of models, including arima, sarima, neural network, tbats, state space, and ETS. Some models were run on the hourly data, which were averaged into daily loads after forecasting. However, these models had a very long processing time. Therefore, we were unable to use these models to forecast for both the full and training data sets. We ran these models only on the full data set if our computers were able to process them. Thus, they are not included in our top 5 models in this report but they are uploaded onto Kaggle.

The code for some of the best performing models for the full data set are included below, in no particular order.

Neural Network Model

```r
# Set max K values to test
max_K1 <- 3  # For weekly seasonality
max_K2 <- 5  # For yearly seasonality

results <- expand.grid(K1 = 1:max_K1, K2 = 1:max_K2)
results$AICc <- NA

# Loop over all K1-K2 combinations
for (i in 1:nrow(results)) {
  Kvals <- c(results$K1[i], results$K2[i])
  xreg <- fourier(load_daily_msts, K = Kvals)

  fit <- auto.arima(load_daily_msts, xreg = xreg, seasonal = FALSE)
  results$AICc[i] <- fit$aicc
}

# Find best K1-K2 pair
best_row <- results[which.min(results$AICc), ]
cat("Best K values: K1 =", best_row$K1, ", K2 =", best_row$K2, "\n")

# Training Data Set
fit_nn_daily_train <- nnetar(as.numeric(load_daily_train_msts))

forecast_daily_nn_train <- forecast(fit_nn_daily_train,
                   h=59)

#write into csv
write.csv(forecast_daily_nn_train,
          file = "../Forecasts/Training/nn2.csv",
          row.names = FALSE)

fit_nn_daily <- nnetar(as.numeric(load_daily_msts))

forecast_daily_nn <- forecast(fit_nn_daily,
                   h=59)
```

TBATS

```r
daily_tbats_fit <- tbats(load_daily_msts)

#forecasting test data
forecast_daily_tbats <- forecast(daily_tbats_fit,
                           h = 59)
```

SARIMAX

```r
#autofit arimax
temp_arimax_fit <- auto.arima(load_all_ts,
                         xreg = temp_all_ts)

#create forecast
temp_arimax_forecast <- forecast(temp_arimax_fit,
                          xreg = tail(temp_all_msts, 59),
                          h = 59)
```

ARIMAX with Fourier

```r
#autofit the arima model
arima_fourier_autofit <- auto.arima(load_all_msts,
                                    seasonal=FALSE,
                                    lambda=0,
                                    xreg=fourier(temp_all_msts,
                                                 K=c(1,3)))

#create the arima forecast with the autofit
forecast_arima_fourier <- forecast(object = arima_fourier_autofit,
                                   xreg = fourier(tail(temp_all_msts, 59),
                                                  K = c(1, 3),
                                                  h = 59),
                                   h = 59)
```

Neural Network

```r
# Fit the neural net model
nnetar_fit <- nnetar(load_all_msts,
              xreg = temp_all_msts,
              repeats = 10)

#create forecast
forecast_nnetar <- forecast(nnetar_fit,
                          xreg = tail(temp_all_msts, 59),
                          h = 59)
```

Performance Evaluation

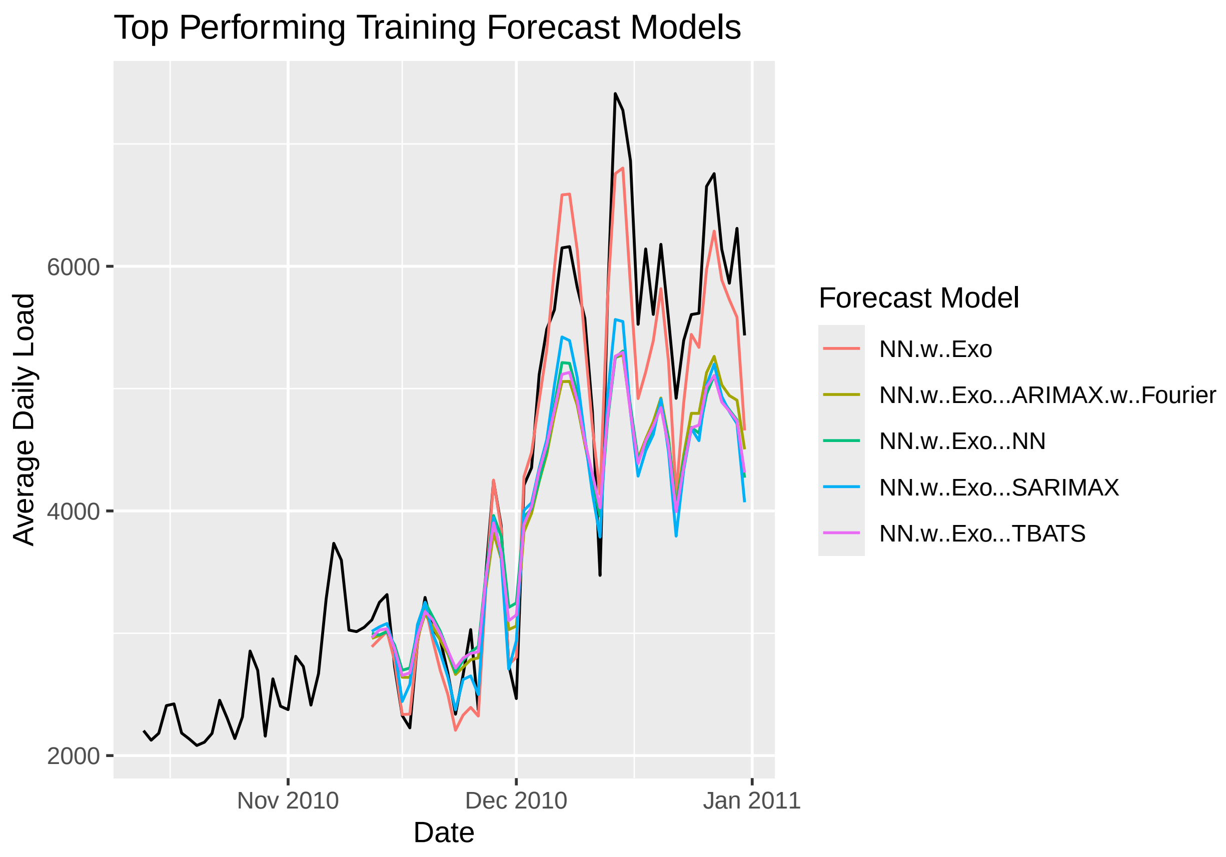
The top 5 models based on training and their test statistics were as follows

Table 1: The top five training forecasts based on MAPE

Model	ME	RSME	MAE	MPE	MAPE
nnetarExo	199.4329	410.8597	314.8748	3.799985	6.602462
nnetarExo__sarimaxTemp	639.7234	911.3139	708.1218	10.463332	13.114187
arimaxFourierTemp__nnetarExo	612.4622	921.5348	744.3415	9.222852	14.374418

Model	ME	RSME	MAE	MPE	MAPE
nnetarExo_tbats	612.1412	951.1645	766.8715	8.872108	14.891570
nn_nnetarExo	596.3962	949.0191	764.2989	8.400495	14.952596

The model forecasts of the top performing models are shown below.



We had some issues regarding the hourly data sets and uploading our averaged data. Our issues with hourly data sets are described above. The averaging and results ran fine, but we had issues uploading to Kaggle. Our code output produced almost 10,000 averaged model combinations. We were not able to upload all of these. Instead, the top 10 averaged models identified by MAPE on the training set were uploaded. Please note that this will not include any of the averages including hourly data forecasts.

1. Neural Network Model (Daily dataset, k= 1, 1, 3)
2. model 2
3. model 3
4. model 4
5. model 5

Conclusions

A short paragraph about which were the best models, how well they aligned with