Global Forecasting Competition 2012 Project

By Sambit Nath

2-1-2021

About the business use case:

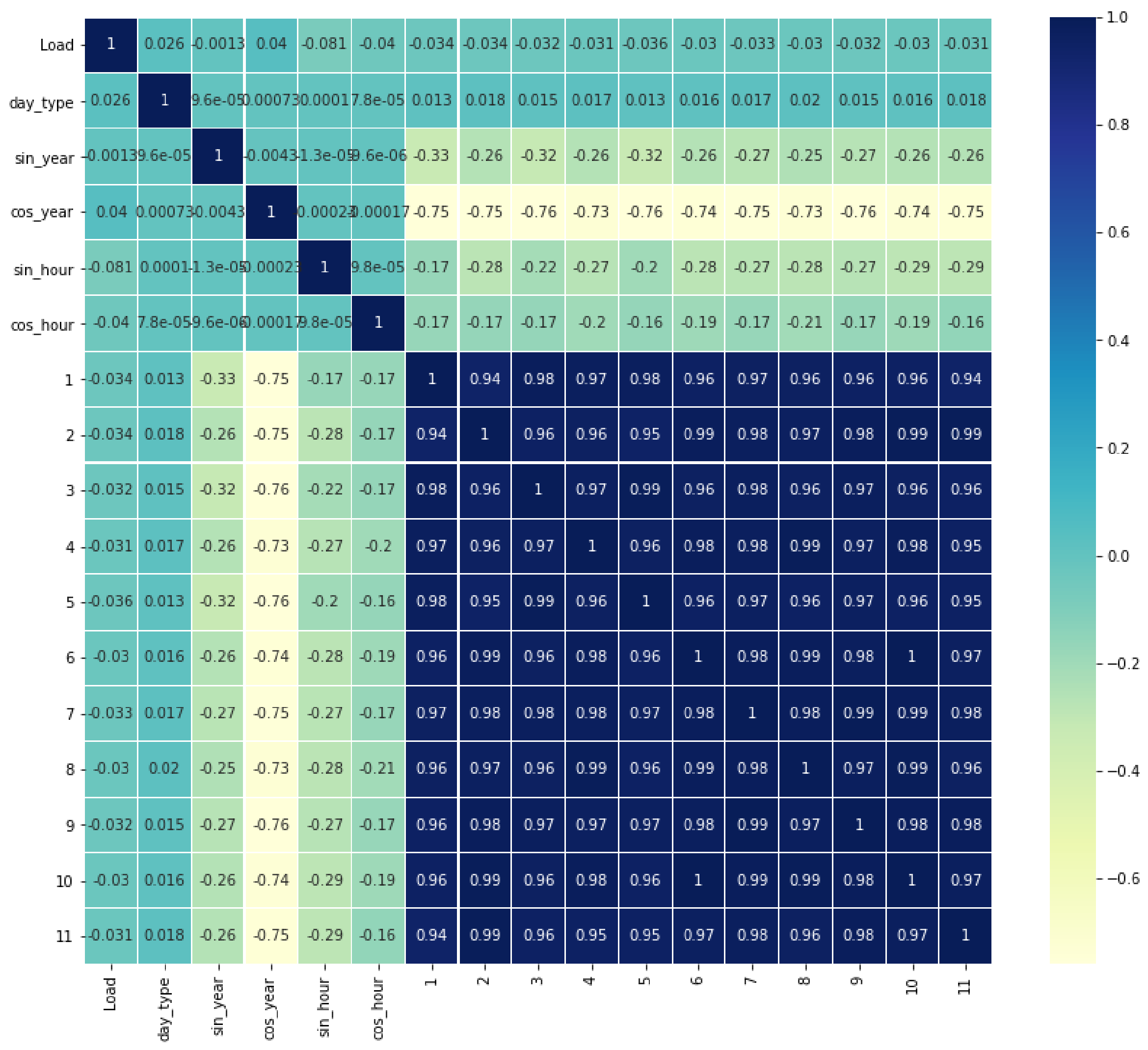
The use case has been selected from Kaggle site. Here we need to forecast/backcast hourly loads (in kW) for a US utility with 20 zones. We need to backcast and forecast at both zonal level (20 series) and system (sum of the 20 zonal level series) level, totally 21 series.

Further details regarding this use case has been described over this link : https://www.kaggle.com/c/global-energy-forecasting-competition-2012-load-forecasting/overview/description.

The project is mainly divided into 5 sections namely:

1. EDA and data preparation.
2. Training the model.
3. Backcast predictions
4. Forecast predictions.
5. Final formatting

1. **EDA and data preparation:** It is considered as a backbone of data analysis as EDA could impact the accuracy of the model which we are making. So, firstly the historical load data is imported and restructured into “long” format so that it can be arranged in the form of an input dataframe. There are no missing values to be handled here. The only values missing are the ones which we are required to predict. We are not given the historical load data for system level (Zone 21) which is the sum of the loads of the other 20 zones. I needed to group all the loads of all the zones taking the sum and append it to the pre-existing dataframe. We added a column called ‘day_type’ which consists of the weekend/weekday information of a particular day.
   We also input the historical temperature data and merge it with the existing load data. We also add some of the time cycle features to the input dataframe like ‘sin_hour’, ‘cos_hour’, ‘sin_year’ and ‘cos_year’. Using the concept of Fourier transformation, a reasonably continuous and periodic function can be expressed as a sum of sin or cos terms. In this case our load can be considered the same.
   Now after assembling these features, we plot a correlation heat map (of the temperature, load and the independent features) as follows:



Here we can observe that the temperature data (represented by the numbers 1, 2 ,3 etc. for various weather stations) are highly correlated. There is no point in taking all of them as input in our model. So, we pick the temperature data for the first station and drop the remaining ones.

2. **Training the model:** After completing the EDA and pre-processing, we save the data into our local for the training step. Here, our goals are primarily two things, first we need to check the performance metrics of the model which we will create using the MAPE calculation and second, to create serialized versions of our trained models and store them so that we can later use them for predictions. Of course, hyperparameter tuning is indeed considered as one of the most major sections of model training, but here I have used only the default parameters in my model.
   For this business case we will be using machine learning technique for creating the predictions. So, unlike classical time series models such as AR, MA, ARMA etc., we need not check the seasonality, trend, stationarity here. We treat this use case like any other ML regression problem now. Here, we have also added some more time categorical features such as day of the week and week of the year, which would help in capturing the time component in a better way. There are also other ways such as introducing lags as independent feature in the model.

3. **Backcast Predictions:** Creating the backcast predictions was easy as I had all the input features ready in the required format. The prediction dataframe which I had created in the 1st step had the backcast dates in them. So, I just imported the pre-processed data from step 1 and filtered out the dates which were needed to be backcasted.
   The serialized models from step 2 were 21 in numbers (i.e one for each zone) which were saved locally. So, I created a loop which was taking filtered prediction dataframe based on zones and creating the required predictions. I will be discussing the code details in the later section of the article.

4. **Forecast Predictions:** This was comparatively more subtle than the backcast predictions because unlike the backcast predictions case, we do not have the temperature data given for the forecast date range. I used the historical temperature data to forecast the temperatures for the dates for which we need to forecast the load.
   In order to predict the temperatures, I used classical time series model ARIMA. The temperature data was found to be stationary, so I did not have to perform any further processing to convert it into stationary. After that I assembled the forecast dataframe and gathered all the remaining input features for the model and generated the predictions.

5. **Final formatting:** In this section, we combine both our forecast and backcast results and restructure them into the required format as per the Kaggle template.

Code flow details:

I have divided the project into 5 scripts for each section as discussed earlier:

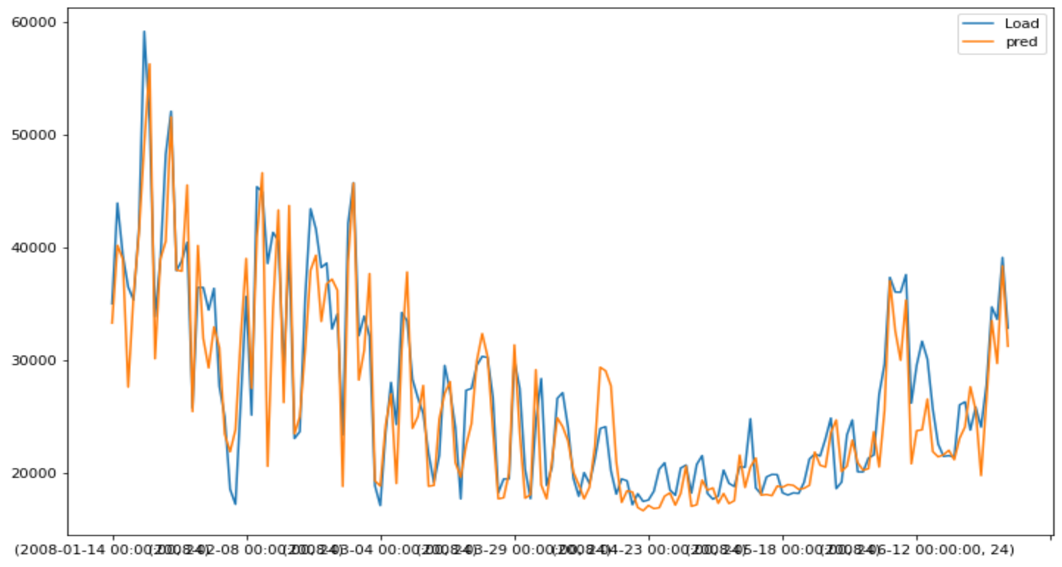
1. EDA and training data prep.py
2. training.py
3. backcast_pred.py
4. forecast_pred.py
5. final_formating.py
6. feature_assemble_functions.py

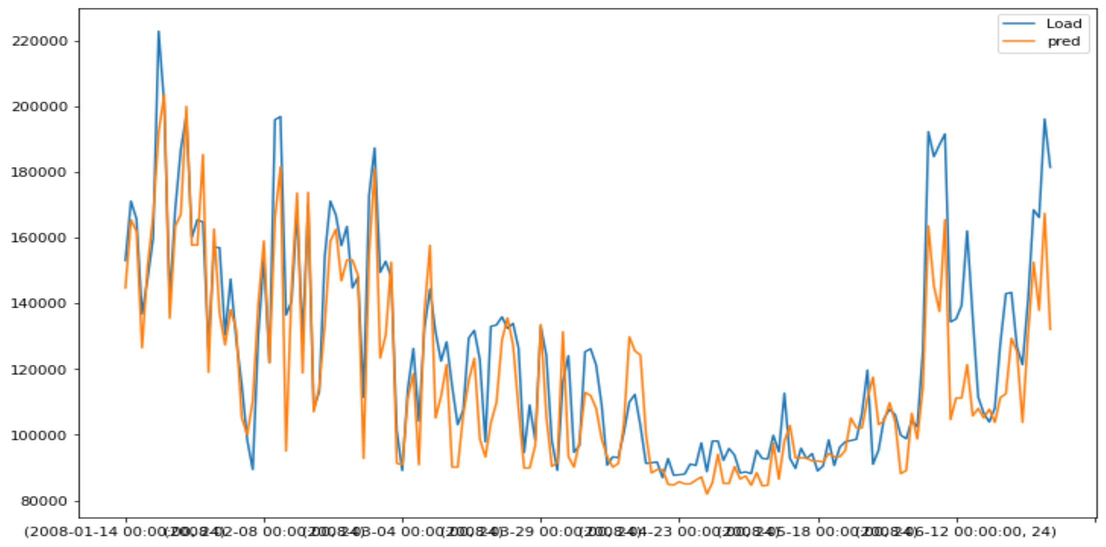
1. **EDA and training data prep.py** : In this script, mainly the pre-processing of the data was done and it was structured in the format in which I needed to fit it in the model. The historical load dataframe was merged with the temperature dataframe and the other features were added. These 'other' features were: the time categorical features such as week of year and day of week, sine and cosine of the year and hour column as discussed earlier and the day type (weekday/weekend) feature. I have combined all these features in a module named feature_assemble_functions.py for the ease of using the functions during training, prediction and backcast dataframe creation.

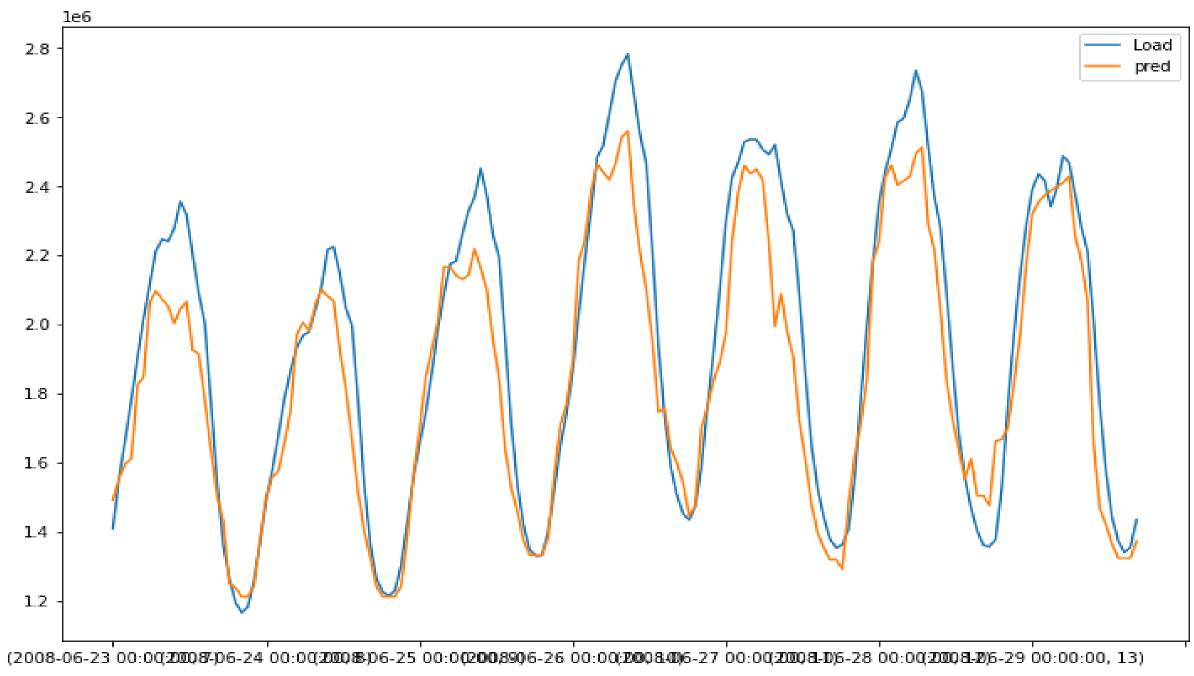
2. **training.py**: In this script I have tested the model using MAPE as the performance metric. After that used the entire historical data available to be the training dataset. As there are 21 zones (including system level), I had to create separate models for each zone and store them as key value pair as dictionary. But I wanted the training operation to be in this script alone. So, I had to store these trained models as serialized versions locally.

For testing the model performance, the MAPE table was created which is attached in the GitHub repo along with the codes.

Here are some of the plots of the predictions and actual load for zone 12 ,16 and 21:

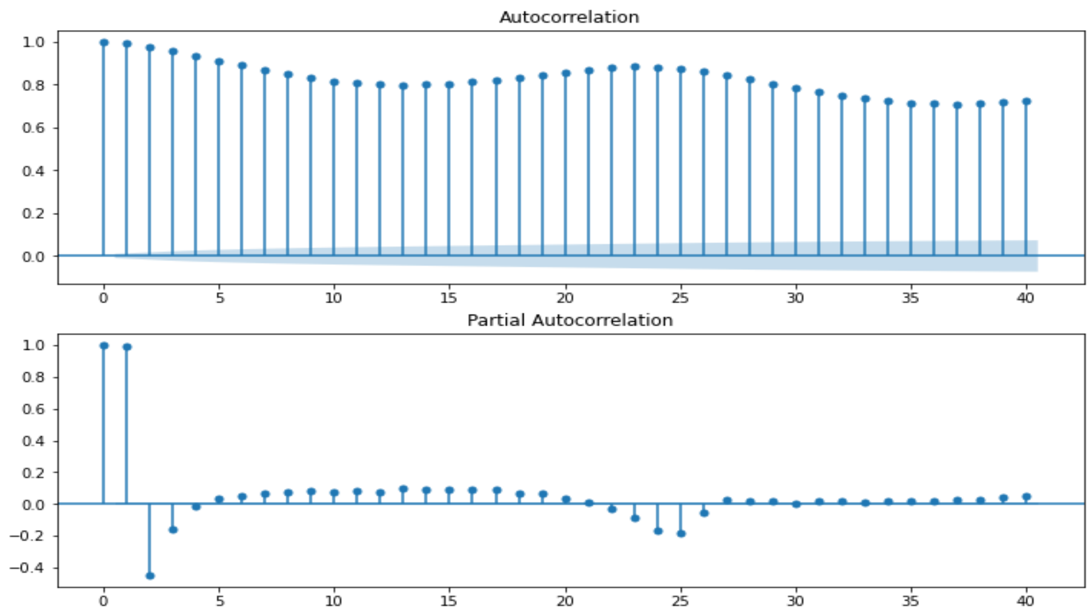






3. **backcast_pred.py:** In this script, the saved models were used to predict the backcast results for each zone.

4. **forecast_pred.py**: As I did not have the temperature values for the dates which I need to forecast the load, I need to generate the temperature for this date range. Here I have used classical time series model ARIMA to generate the temperature for these dates. As temperature series was already stationary, I did not have to apply any further transformation. I plotted the PACF and ACF plots to determine the p, d and q values.



After calculating the temperature values, the other input features were assembled, and the forecasts were predicted.

5. **final_formating**: The final dataframe was created combining the forecast and backcast and restructured as per the submission template.

6. **feature_assemble_functions.py:** This is a module which consists of the essential functions need to gather the input features while assembling the training, forecast and backcast dataframes.