**Predicting energy consumption at an hourly frequency for San Diego Gas & Electric (SDGE) utility region**

(Springboard Capstone 1 project submitted by Prathamesh Pawar)

Contents

[1. Introduction 2](#_Toc19377012)

[2. Data 3](#_Toc19377013)

[2.1 Energy consumption data 3](#_Toc19377014)

[2.2 Weather data 6](#_Toc19377015)

[2.3 Solar/PV (Photovoltaic) installation data 7](#_Toc19377016)

[3. EDA 8](#_Toc19377017)

[4. Machine learning 15](#_Toc19377018)

[4.1 Introduction to Time series 15](#_Toc19377019)

[4.2 Baseline persistent forecast 18](#_Toc19377020)

[4.3 Simple Linear regression 19](#_Toc19377021)

[4.4 1-hour ahead predictions using lag values 21](#_Toc19377022)

[4.5 Adding Fourier terms to handle multiple seasonality 24](#_Toc19377023)

[4.6 (S)ARIMA(X) model 25](#_Toc19377024)

[4.7 FB Prophet 28](#_Toc19377025)

[4.8 Regression models using Fourier terms 31](#_Toc19377026)

[4.9 XGBoost + FB Prophet 35](#_Toc19377027)

[5. Conclusion 36](#_Toc19377028)

[6. Future work 39](#_Toc19377029)

# Introduction

Electrical utilities need to diligently plan ahead of time the allocation of generating units in their power plants to match their regional energy demand (MW), because if the demand is higher than the generation it can cause several blackouts resulting in a huge loss to the economy; on the other hand if the generation is higher than the demand the extra electricity will be wasted and it can also create an unnecessary load on the transmission lines.

So, it is very important for the utilities to have a forecast of the energy consumption to be able to allocate appropriate resources to meet their demand. A year, month or day ahead forecast can help the utilities plan for a larger time scale but for smoother daily operations an hourly (or even better) forecast can also prove very useful. For example, if the plant operators get a high energy forecast for the next hour, they can ramp up the energy supply by switching on more power plants.

This project will involve analyzing past 5 years of hourly energy consumption data of SDGE utility to find trends in energy consumption around hour of the day, day of the week, season of the year, etc. and to check if factors like outside temperature and solar installations in the region affect the energy consumption.

The developed prediction model can be utilized by the electrical utilities to effectively plan their energy generation operations and balance the demand with appropriate supply. An efficient forecast can prove very useful for the utilities in planning their day to day operations, meeting their customers’ energy demand, and avoiding grid failures or wastage of energy and costs of under or over cutting.

A quick look at the process flowchart followed in this project is given in Figure 1.

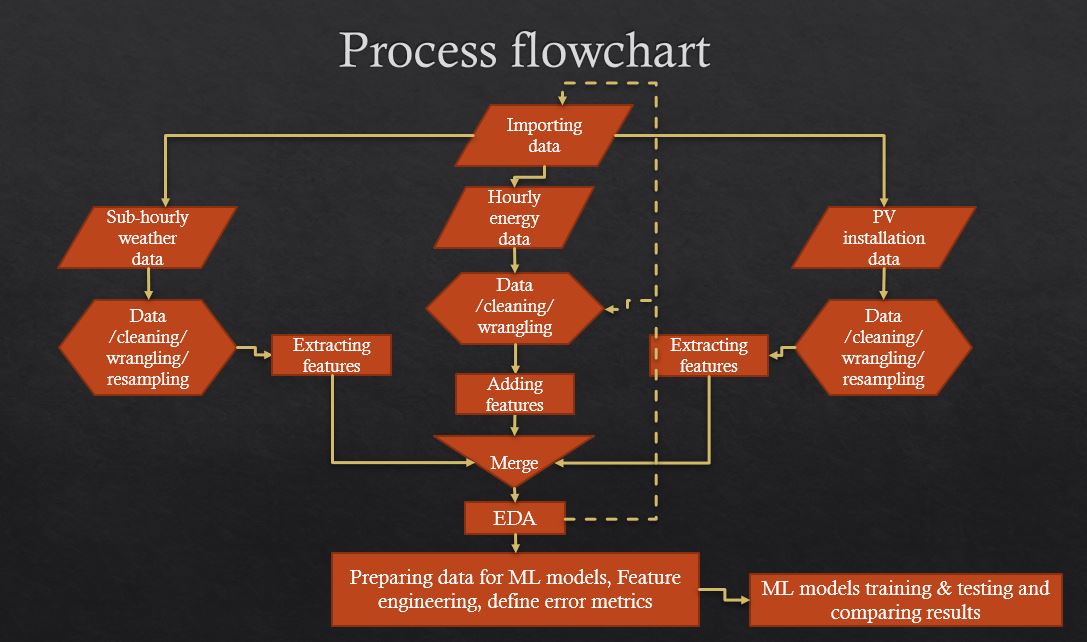


Figure Process flowchart

# Data

## 2.1 Energy consumption data

The hourly energy consumption data for the 4 utilities- PGE (Pacific Gas and Electric), SCE (Southern California Edison), SDGE (San Diego Gas and Electric) and VEA (Valley Electric Association, which actually covers some parts of Nevada)- under the CAISO (California Independent System Operator) is available on CAISO’s website in the form of .csv files. The most recent data for past 3 months and 2018 is available [here](http://www.caiso.com/planning/Pages/ReliabilityRequirements/Default.aspx) and previous data up to 2014 is available in the archive folder [here](http://www.caiso.com/Pages/documentsbygroup.aspx?GroupID=8879C382-6EA8-4357-B752-D4F571388958).

Python’s requests and html packages were used to extract the .csv files directly from the websites and stored in dictionaries before concatenating them into a single data frame.

Below is the map of CA and surrounding states showing the regional divisions in the CAISO- regions served by the utilities PGE, SCE, SDGE and VEA.



Figure CA region map showing areas served by different electrical utilities

After initial exploratory data analysis (EDA), some of the problems that were observed with the data were:

* Dates columns of 2017 and 2018 years were a bit different than the other years' Dates column.
* Also, the last few entries in the Dates column of the year 2018 had 2017 as the year instead of 2018.
* The number of rows in each year; ideally, they should be 8760 for each year except 2016 because it was a leap year so the number of rows in 2014, 2015, 2017 and 2018 should be 8760 each and that in 2016 should be 8784.



Figure Checking if data is clean

It looks like the data in 2018 has some 24 rows labeled as 2017. The plausible explanation to this is that somewhere in the data, 24 hours of year 2018 weren't recorded and the data from 2017 was used to fill in the missing values. This left the Dates column with the 2017 instead of 2018. 2017s value in the 'Dates' column of the 2018 data were replaced with 2018.

Data was checked for null values and outliers and other integrity checks were also done. Focusing only on one utility for now-SDGE (San Diego Gas & Electric); selecting smaller region will make it easier in the future to import the weather data and use it as a forecasting parameter (the red region in Figure 2).

The cleaned data set for SDGE is shown in Figure 4 as a time series plot of energy consumption.

Time dependent variables like year, hour, weekday, month, season and holidays were added to the data frame. Based on SDGE’s website, months from June to October are denoted as 'summer' and months from November to May as 'winter'. A new column named “non\_working” was created which is 1 for weekends and/or holidays and 0 for working days.

Figure 5 shows the cleaned data frame with added time dependent variables.

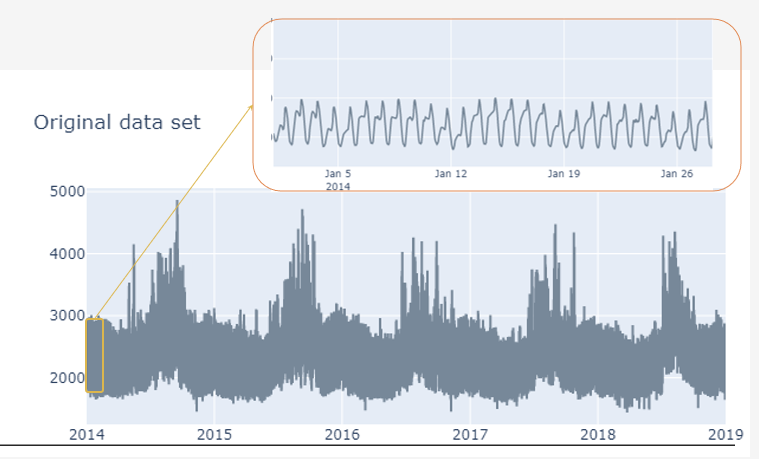


Figure Hourly Energy consumption in MWH from 2014-18 for SDGE region



Figure Cleaned dataframe with added time variables

Some basics about the energy/load forecasting:

* Load -> defined as Demand (in W) or Energy (in Wh)
* For hourly data they are the same
* The peak or maximum load decides the capacity of power plant

## 2.2 Weather data

To see the impact of weather (specifically the temperature) on the energy consumption, we'll import the 7 minute interval temperature data from NOAA’s [site](https://www.ncei.noaa.gov/metadata/geoportal/rest/metadata/item/gov.noaa.ncdc:C00684/html).

There are two or more weather stations in San Diego (Figure 6) at which the weather data has been recorded. Since weather data at airports is more accurate and not biased by nearby buildings and industries, we'll use the San Diego International airport data for this project.

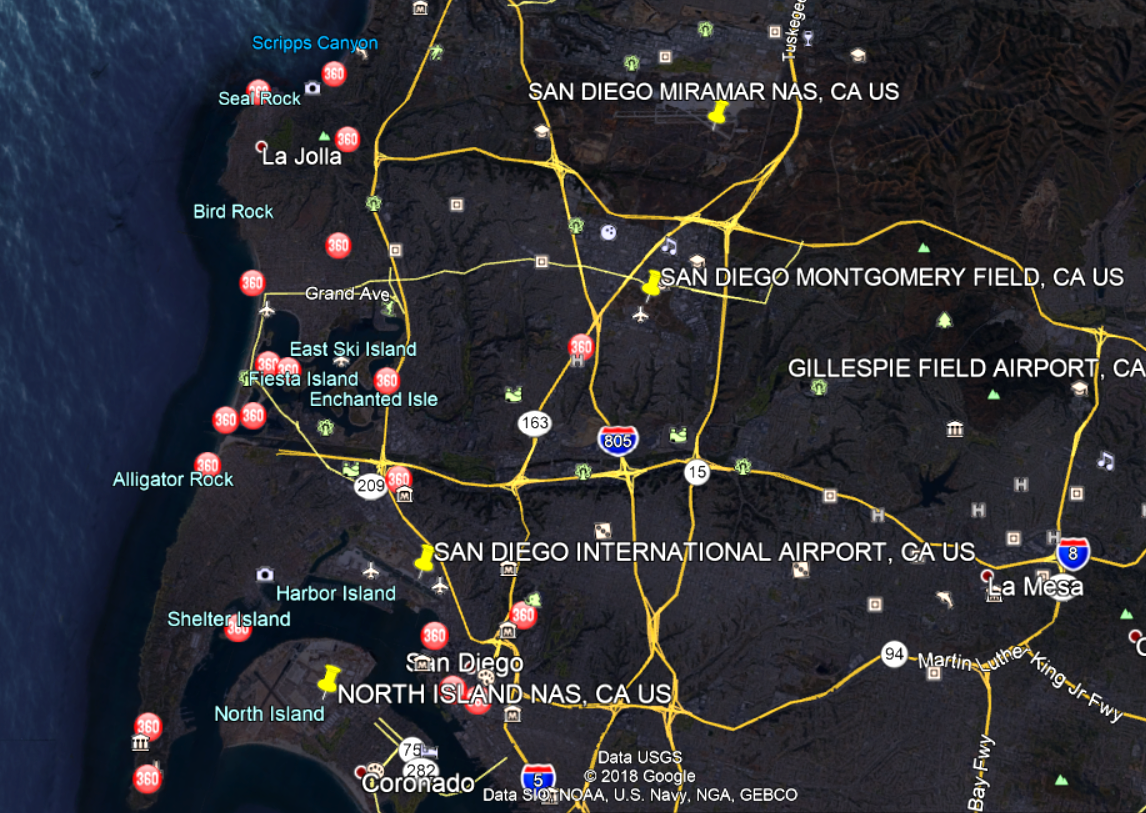


Figure Weather stations in San Diego area (yellow pins)

The 7-minute interval weather data was resampled in hourly intervals the average temperature for each hour was calculated. Only the temperature information was extracted from the weather for this project though it contains much more information like degree heating and cooling days, humidity, pressure, etc., because temperature is assumed to impact the energy the most.

After some cleaning and formatting the temperature data was merged with the previous energy data frame.

## 2.3 Solar/PV (Photovoltaic) installation data

After some initial data visualizations on the energy data it was observed that the average energy consumption and peak demand (the highest demand for 1-hour interval over the entire period) have decreased over the years from 2014 to 2018. Now, this can be due to stricter energy efficiency measures being mandated or increase of incentives for different categories of ratepayers (residential, commercial, agriculture, etc.) to install self-generation resources like solar, wind, etc. and/or energy storage in their premises to avoid peak loads and decrease overall energy consumption.

It was observed that the energy consumption decrease over the years was more dominant over the daylight hours, so we can test the theory that maybe solar capacity installed in the utility region has increased over the years and thus more and more customers are using less energy when the sun is out causing a total reduction in energy consumption. We can test this theory by importing the solar installations data in SDGE territory.

The solar installations data was imported from CA.gov [here](https://www.californiadgstats.ca.gov/downloads/) -> 'NEM Currently Interconnected Data Set’. The dataset has many columns, but we’ll focus on:

* 'approval date’ : the date when the system was connected to the grid.
* 'System Size AC’ : the total kW power in AC of the solar panels installed at a site.

Calculating the cumulative installed capacity till each hour because we want to know how much PV system capacity (operational) has been installed till that hour, because that is what will affect the energy consumption at that point.

Many NaN values were observed since not all days had solar installations. Since the cumulative installed solar system size should remain same until the next non-NaN value is encountered, forward fill method was used to fill the 'cum\_AC\_kW' column (the cumulative PV installations column giving out the cumulative installed capacity in kW). After some cleaning the data was merged with the energy and weather data frame.

# EDA

Checking average hourly load profile of the energy consumption from 2014-18 in Figure 7. Plotting average monthly load profile over the entire period 2014-18 in Figure 8.

From Figure 7 we can observe how the load remains low over the night and then starts increasing as the region wakes up, and continues increasing during the office hours and peaks in the evening when everyone returns home and turns on the electrical appliances in their house.

From Figure 8, we can observe that as expected the monthly load profile peaks in the summer due to high cooling (air-conditioning) load caused by high temperatures.

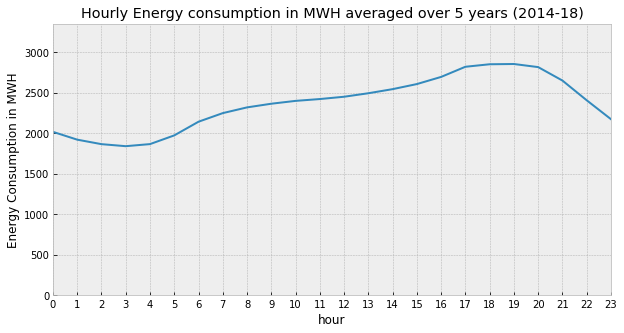


Figure Average energy consumption per hour from 2014-18

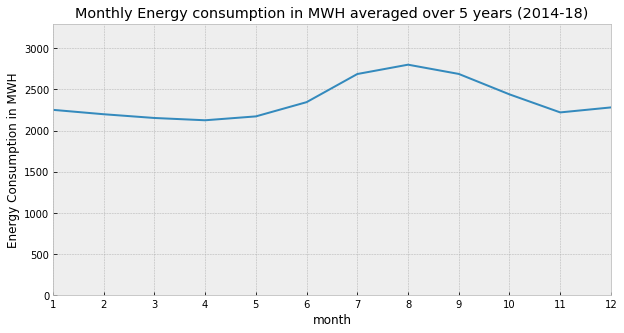


Figure Average energy consumption per month from 2014-18

To check how the energy consumption varies across an average day and week on the same chart, a heat map was plotted with weekday energy consumption on the X axis and hourly consumption on the Y axis, refer Figure 9.

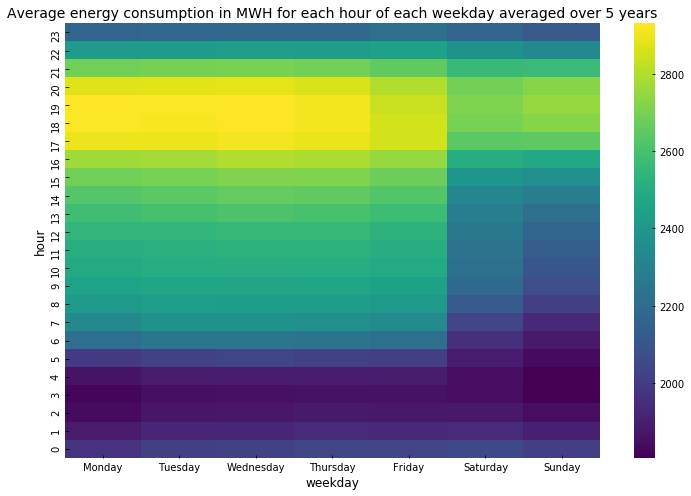


Figure Average energy consumption in MWH for each hour of each weekday averaged over 5 years

* It can be seen that the average consumption from Monday to Friday is below 2000 in the night and it increases during the day and is at peak (>2800) during the evening time before sliding down again for the night. And on the weekends the same pattern can be observed but the overall consumption seems lower on weekends than weekdays as expected because most of the commercial buildings don't operate on weekends (also maybe because people go out on weekends and are not usually at home?)
* And, in fact, to keep the load on the electrical grid lower during this time, SDGE (and other utilities) try to discourage people from using more electricity during these hours. SDG&E began transitioning residential customers to Time-of-Use (TOU) pricing plans in early 2019; a TOU plan is one wherein the utility applies different rates to the customers for different time slots during the day. And in fact, SDGE applies highest rate (peak rate) during 4pm to 9pm to its customers as compared to other hours during the day. From SDGE's website: "If customers can shift some of their energy use to lower-cost time periods outside 4 p.m. to 9 p.m., they can lower their electricity bills and make better use of cleaner, renewable energy sources, like wind and solar, when they are more available."

Plotting the energy distribution using box plot and dividing the columns of the plot by weekdays and adding a hue for working and non-working days, see Figure 10.

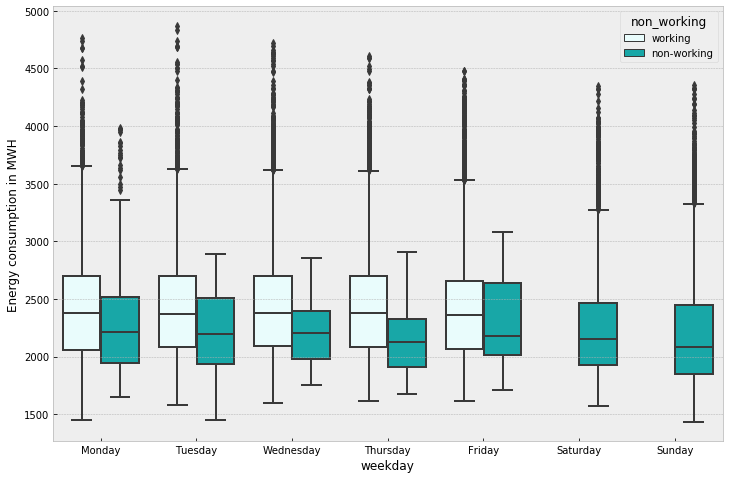


Figure Average energy consumption on working days vs. non-working days

It can be seen that the median energy consumption on working days remains fairly same from Monday to Friday and drops on the weekend as seen before in the heatmap. Also, if a particular day is a holiday or non-working day (indicated in dark green) the energy consumption is much lower than if the same day was a working day as expected.

Visualizing the distribution of energy values for different years in Figure 11. For all the years the distribution is kind of bi-modal and the mode values for energy consumption consistently shift towards left (towards lower energy load) each year from 2014 to 2018. Potential reasons for this shift can be increasing renewable energy installations at customer sites like PV (called as behind the meter), increase in electric efficiency standards of appliances or industry equipment, battery installations at customer sites, etc.

The same distribution is plotted for each hour and one of the hours is shown in Figure 12. From this figure and other similar figures for daylight hours (not shown here), it can be observed that the left shift is more dominant during the daylight hours of 8am to 5pm. One of the reasons for this shift during daylight hours can be the addition of more renewables like solar and wind behind the customer meters. We will see later that the solar installations in the SDGE have increased considerably over the last 5 years.

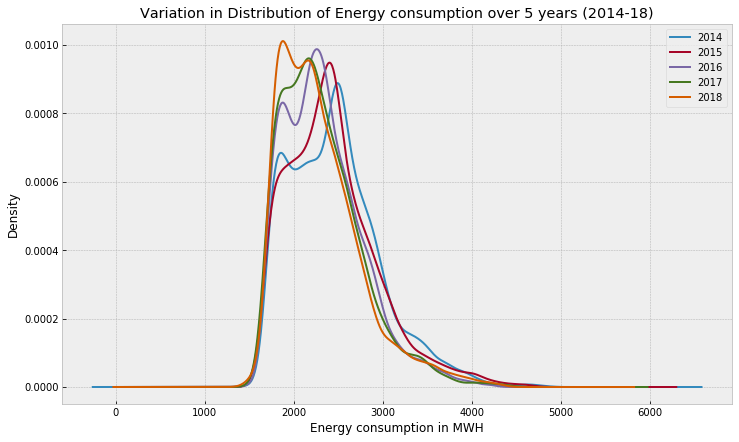


Figure Distribution of energy consumption across 2014-18



Figure Hourly energy consumption distribution for 12th hour of the day for years 2014-18

The temperature and energy time series are explored together in the Figure 13. As suspected before, we can see that the energy consumption and temperature do kind of follow each other and seem to have some level of correlation between them. Note: Rolling mean averages of both variables over a window of 30 days have been used for plotting to achieve some smoothing.

More importantly we can see that the highest energy consumption values occur at highest temperatures. This, as mentioned before, is the result of higher air conditioning loads at higher temperatures.

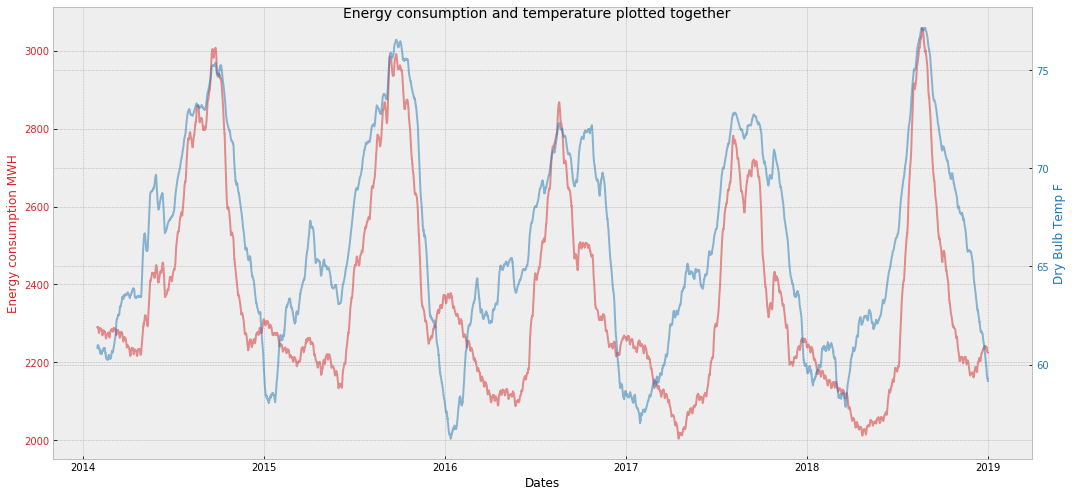


Figure Energy consumption and temperature plotted together (rolling mean average over a window of 30 days)

Plotting a linear regression plot between the energy consumption and temperature (Figure 14), we observe a pretty decent correlation between the energy consumption and the temperature, as suspected before, which is expected because high heat calls out for more air conditioning loads. This correlation gets better if we plot only summer months.

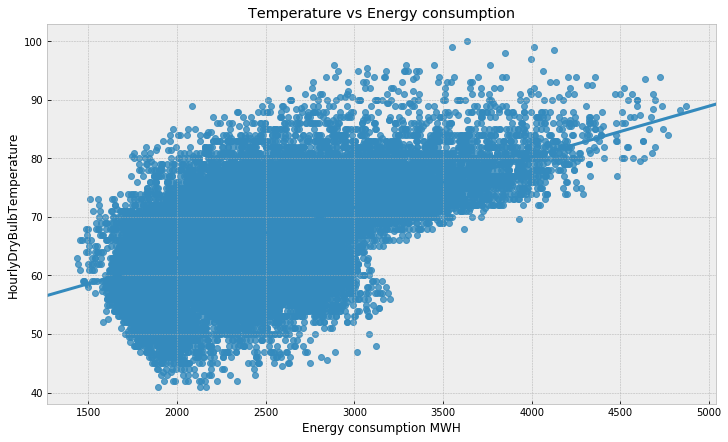


Figure Energy consumption vs Temperature

We can also see the correlations using the *scipy.stats.pearsonr* function. The *scipy.stats.pearsonr* function returns the *pearsonr* coefficient and the pvalue of observing such coefficient if we were to assume that there was no correlation between the x and y data sets. Let's select a significance level of 5%, and so if the pvalue is <5%, we will assume that the correlation coefficient returned is significant.

pearson correlation coefficient and pvalue for winter 0.32 0.0

pearson correlation coefficient and pvalue for summer 0.65 0.0

So, the correlation between the energy consumption and the temperature is positive and also since the pvalue is almost 0, we can say that the correlation is not a result of random chance. And the correlation coefficient for summer is higher than that for the winter (almost twice), which makes sense because high temperatures cause high demand in summer more than the winter months.

Exploring energy and PV installations data together:

As observed in section 3.1, the energy consumption for daylight hours seems to have decreased from 2014 to 2018. Now, this can be due to stricter energy efficiency measures being mandated or increase in incentives for different categories of ratepayers (residential, commercial, agriculture, etc.) to install self-generation resources like solar, wind, etc. and/or energy storage in their premises to avoid the higher energy demand. It was observed that the energy consumption decrease over the years is more dominant over the daylight hours, so we can test the theory that maybe solar capacity installed in the utility region has increased over the years, which it has as we will see below in Figure 15.

The cumulative PV installation capacity has increased considerably from 2014-2018 and the maximum energy demand seems to have decreased at the same time. Pearson correlation coefficients were calculated between the energy consumption values between 10am to 4pm over the years and the cumulative PV installed capacity and it was observed that:

Energy consumption vs PV installed capacity: pearson correlation coefficient and pvalue for winter -0.52 0.0

Energy consumption vs PV installed capacity: pearson correlation coefficient and pvalue for summer -0.34 0.0

Now, as discussed earlier there can be other reasons for this reduction but the correlation coefficients were tested for other hours of the day and no strong correlation was observed for other hours compared to the daylight hours. Thus, we can say with some confidence that the increasing PV capacity has indeed helped the energy demand to go down (keeping other things constant).

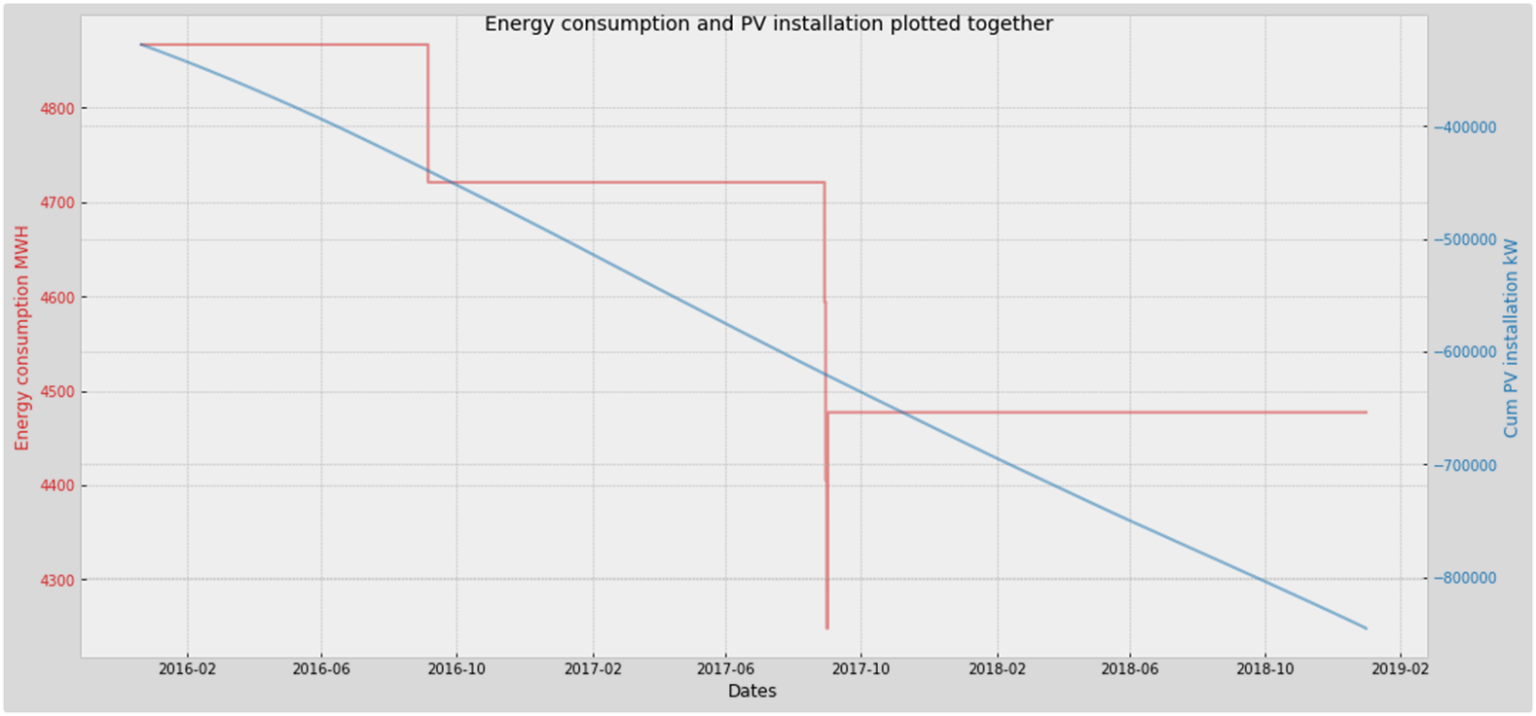


Figure Max energy demand in MW (with a rolling window of 6 months) plotted with the cumulative PV installations in kW

EDA conclusions:

* Our data has a strong seasonal pattern. It looks like it has multiple seasonality – daily, weekly, yearly.
* The temperature and cumulative PV installations are useful independent parameters for our dependent energy variable.
* Other temporal variables like hour of the day, weekday, month, non\_working/working also affect the energy consumption deeply so they can be used as independent variables too.

Since we have our data ready and EDA has been done, let’s move on onto the ML section.

# Machine learning

In this project I have used the basic time series models like ARIMA and FB-prophet and then have extended my approach to include linear regression, random forests and XGBoost to see whether or not these linear and non-linear approaches can model our time series accurately.

## 4.1 Introduction to Time series

**Time series (TS)** is a sequence of observations taken sequentially in time.

TS components:

* Level: The baseline value for the series if it were a straight line.
* Trend: The optional and often linear increasing or decreasing behavior of the series over time.
* Seasonality: The optional repeating patterns or cycles of behavior over time.
* Noise: The optional variability in the observations that cannot be explained by the model.

In addition, TS also exhibits Auto-correlation which simply means that observations close together in time tend to be correlated (serially dependent).

Forecasting windows used in this project are 1-hour ahead, 1-week ahead and long term (1 month to 1 year).

The error metrics used in this project to test our models on our energy TS prediction are:

* R2 score
* MAE (mean absolute error)
* RMSE (root mean squared error)
* MAPE (mean absolute percentage error)

**Cross-validation** for time series is a bit different because time series have this temporal structure and one cannot randomly mix values in a fold while preserving this structure. With randomization, all time dependencies between observations will be lost.

The idea is rather simple -- we train our model on a small segment of the time series from the beginning until some t, make predictions for the next t+n steps, and calculate an error. Then, we expand our training sample to t+n value, make predictions from t+n until t+2∗n, and continue moving our test segment of the time series until we hit the last available observation. As a result, we have as many folds as n will fit between the initial training sample and the last observation. This can be established using the sklearn.model\_selection's TimeSeriesSplit module. The approach looks like the one given in Figure 16.

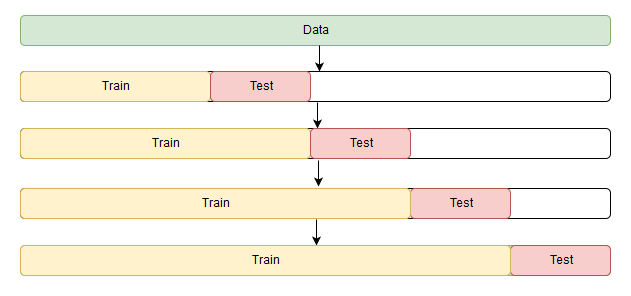


Figure Time series cross validation (tscv)

**TS stationarity:**

For a TS to be stationary it shouldn’t change its statistical properties over time, namely its mean and variance. The covariance function does not depend on time; it should only depend on the distance between observations.

**Why is stationarity so important?**

* Easy to make predictions since we can assume that the future statistical properties will not be different from those currently observed.
* Most of the time-series models try to predict those properties (mean or variance, for example).
* Future predictions would be wrong if the original series were not stationary.

A common way of testing the stationarity is to check the TS visually for any trends and seasonality. Another more powerful and common approach is Dicky Fuller test which tests the TS for the presence of a unit root.

**How to handle stationarity?**

To remove stationarity, usually the TS is differenced once or more (*TS.diff() OR TS – TS.shift(1)*). To remove the seasonal patterns the TS can be differenced with the period of seasonality. For example, if daily seasonality is to be removed in an hourly TS, then the we can use *TS.diff(24) OR TS – TS.shift(24)*).

**Is the SDGE energy TS stationary?**

Visually looking at the data before we observed that the SDGE energy TS has a slightly downward trend and has multiple seasonality patterns – daily, weekly and yearly. So, the visual inspection tells us that our TS is not stationary. Let’s try the Dicky Fuller test on the energy TS.

The test was performed on three version of our energy TS: original TS, single differenced TS, single differenced TS + 24 hours differenced TS.

Based on the p-values of the Dicky Fuller tests it was concluded that even our original dataset is stationary but after checking the t-stat values it was obvious that the stationarity gets more significant with differencing. The original dataset was declared stationary by the test maybe because the trend in our data is very weak (as evident from the decomposed plot in Figure 17).So, we can either use the original dataset as it is with the time series models or to be more robust, we can use the single differencing to remove the trend and fit our models on the detrended data.

**Decomposing the SDGE energy TS:**

The TS was decomposed using sm.tsa.seasonal\_decompose module of statsmodels.api. The module was informed that our TS has a yearly seasonality by passing in the argument frequency=24\*365. Figure 17shows the decomposed plot.

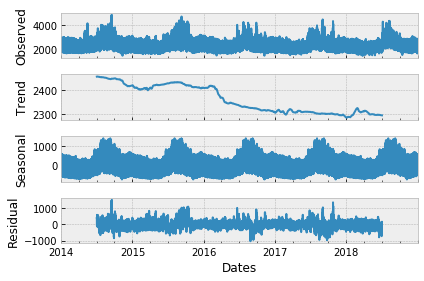


Figure SDGE energy TS decomposed components (frequency = 24\*365)

We can clearly see the weak trend and strong seasonality in our data. In addition to the yearly seasonality, our data also has weekly and daily seasonality patterns which need to be handled by our ML models.

## 4.2 Baseline persistent forecast

A baseline persistent forecast is defined as a forecast whose predictions for future values are equal to the past values for the same time period. For example, for long term baseline persistent forecast, whose test set was defined as a period from April 2018 to Dec 2018, the baseline will predict the energy consumption for this period by simply taking values from April 2017 to Dec 2017. Similarly, for week ahead and 1 hour ahead forecasts. Here are the error values observed on test set for these baseline persistent models:

Baseline persistent long term: MAPE 9.23%

Baseline persistent 1 week ahead: MAPE 20.27%

Baseline persistent 1 hour ahead: 4.21%

Our time series models will be tested against these baseline models. i.e. the models should beat at least these MAPE values to qualify as decent models.

## 4.3 Simple Linear regression

Starting off with simple linear regression models to see how they perform on our time series dataset. And the train test split of our data was carried out such that the last 15% of the energy TS, which is roughly a period from April to Dec 2018, was used as test set.

**Simple linear regression with L2 regularization (Ridge)**

Features: year, month, hour, season and non\_working as categorical and Hourly Dry bulb temperature and cum\_AC\_kW as numerical. Numerical features have been scaled using Standard Scaler.

Figure 18 shows the coefficient plot from the model fit.

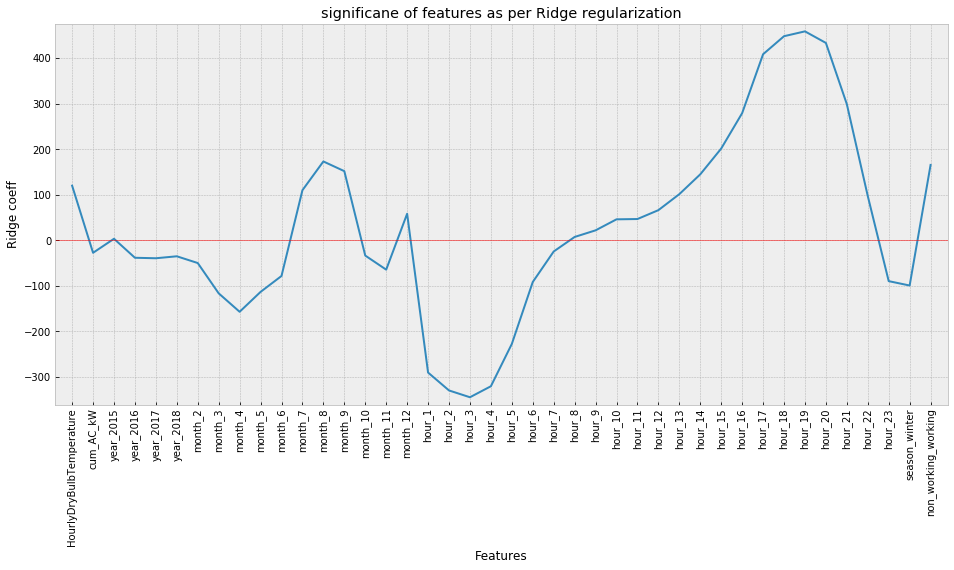


Figure Ridge regression coefficient plot

The coefficient plot basically says that the energy values increase with temperature, are higher in summer months than winter months, are higher for working days than non-working days and decrease over the years from 2014-2018. The hour variable is the most significant coefficient where the energy increases significantly over the evening hours.

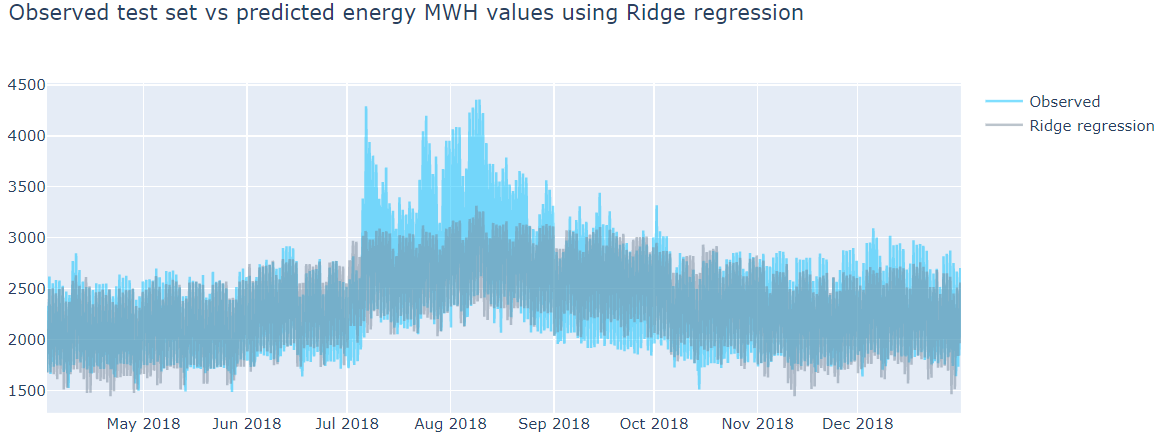


Figure Observed test set vs Ridge regression predictions on test set

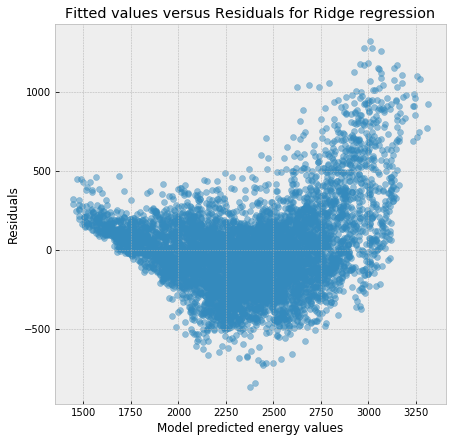


Figure Residual errors vs prediction for Ridge regression

The model predicted the daily trend and seasonality well, but the model failed in capturing the extreme energy consumption values. But another important aspect of this model was that the model is stable and isn't overfitting.

Ridge regression on training set

Variance score: 0.74

Mean Absolute Percentage Error: 7.34 %

Ridge regression on test set

Variance score: 0.72

Mean Absolute Percentage Error: 8.18 %

For a simple linear regression model without any feature engineering that is a decent performance. But we need to keep in mind that in the above models we have used hot encoded variables like hour\_1, hour\_2,..., month\_2, month\_3,.. etc. This results in a loss of information because the model assumes that 23rd hour is far away from the 0th hour (and same for months, month\_12 is far away from month\_1) which is not the case because the time series is periodic and the 0th hour is as much closer to the 23rd hour as it is to the 1st hour. To avoid this, we’ll need to handle the periodic terms differently; we’ll see it in later models.

## 4.4 1-hour ahead predictions using lag values

What makes the time series different from other datasets is that the current y value depends on the previous N values in time depending on the correlation of the data with its lagged version. For example, today's energy consumption can depend on yesterday's energy consumption or maybe depend on the last 5 days of energy consumption values. So, we will add the lagged values of energy consumption as the X parameters and check if we can predict better using the past values.

We try Elastic net regression and Random Forest on the lagged data to predict 1 hour ahead energy consumption. And in addition to the temperature and PV installations the lagged energy values were also scaled using Standard Scaler.

**Elastic net regression with lags:**

From the coefficient plot in Figure 21it can be seen how the 1st, 23rd and 24th lags are the most important predictors.

MAPE: 1.58%[[1]](#footnote-1)

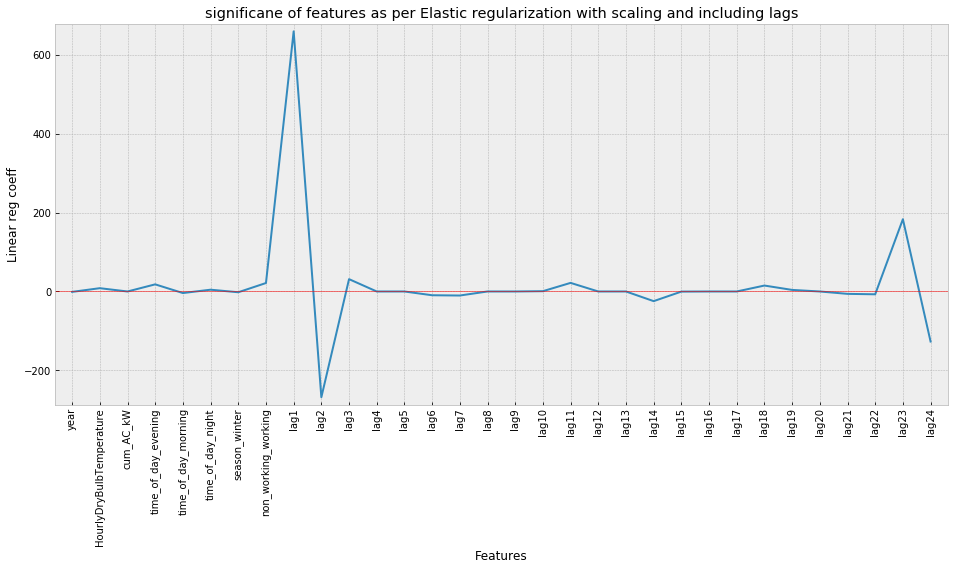


Figure Coefficient plot for Elastic net regression with all lags up to 24 hours

**Random Forest (RF) regression with all lags:**

Similar results as that the elastic net regression were obtained using RF on data with all lags for predicting energy consumption 1 hour ahead.

MAPE: 2.73%

Both Elastic net and Random Forest models give excellent performance by using all lags up to past 24 hours.

These models perform very well but they come with a limitation- we can use it to predict only the next hour value. That is, the maximum time window it can predict accurately is 1 hour. So, if that is the application case then the elastic net model or the RF model with previous lag values should be used.

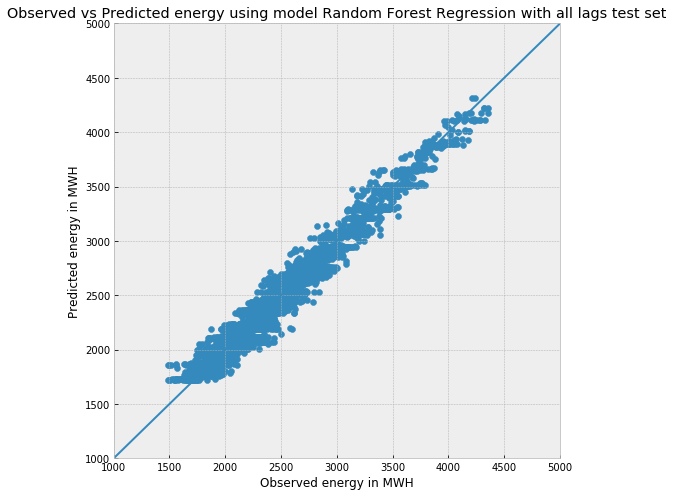


Figure Observed test set vs RF predictions on test set using all lags (1 hour ahead forecast window)

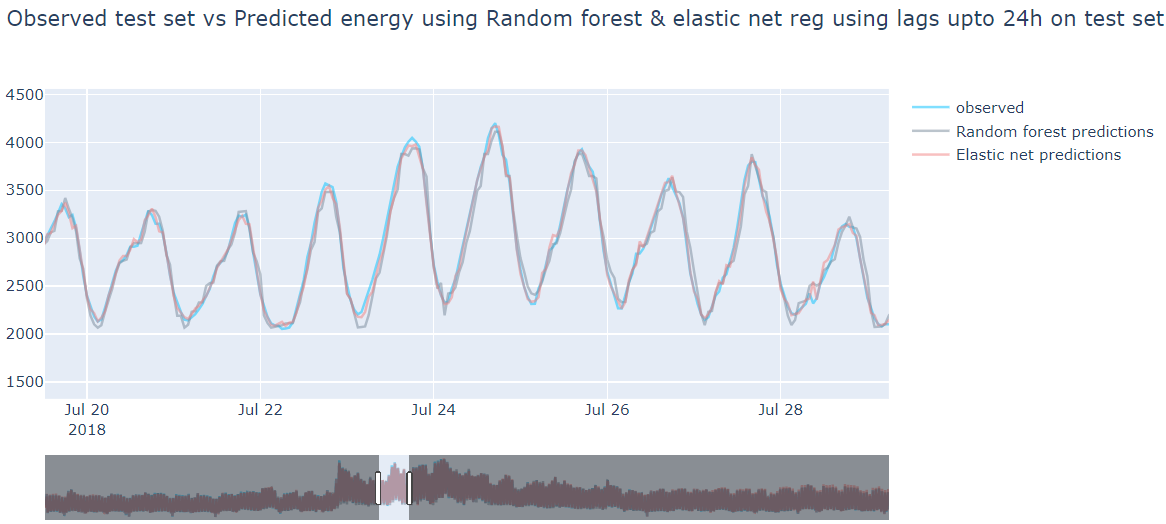


Figure Observed test vs 1 hour ahead predictions using Elastic net and RF on data with lags up to past 24 hours

## 4.5 Adding Fourier terms to handle multiple seasonality

There are two interesting time series forecasting methods called BATS and TBATS that are capable of modeling time series with multiple seasonality patterns. Unfortunately, BATS and TBATS capabilities do not come for free. The method is very generic. Under the hood it builds and evaluates many model candidates. This results in slowness of the computation. And SARIMAX models with Fourier series handling the multiple seasonality patterns can perform as good as the TBATS model, so we will opt for the simpler model here i.e. SARIMAX. We will need to create some extra features here to model the multiple seasonality patterns.

Adding Fourier terms to mimic the daily, weekly and yearly seasonal patterns.



* m & n can be discrete.
* Choosing m=n=5 based on some trial and test on the training set
* Daily -> T = 24
* Yearly -> T = 365.25
* Weekly -> 7

We will be adding these new 30 columns (5 sin and 5 cos terms for each season) as our X columns for predictions.

Just to give a visual on how these Fourier terms look like here is a weekly seasonal plot with the sum[[2]](#footnote-2) of all the 10 (5 sin and 5 cos) terms.

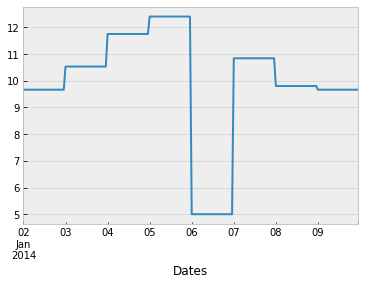


Figure Sum of the weekly Fourier terms

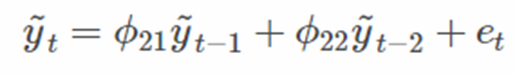
## 4.6 (S)ARIMA(X) model

Before going into details about SARIMAX let’s get a brief overview about the ACF and PACF plots.

**ACF:** Correlation of the time series observations calculated with values of the same series at previous times, is called an autocorrelation (ACF). It is used to determine the moving average (MA or q) term of the SARIMAX (p,d,q) models.



**PACF:** A partial autocorrelation (PACF) is a summary of the relationship between an observation in a time series with observations at prior time steps with the relationships of intervening observations removed. It is used to determine the auto regression (AR or p) term of the SARIMAX (p,d,q) models.

****

So, technically the ACF and PACF plots help us to determine the order of the SARIMAX model. Let’s plot these plots for our original TS. The seasonal period of 24 hours can easily be observed from Figure 25.

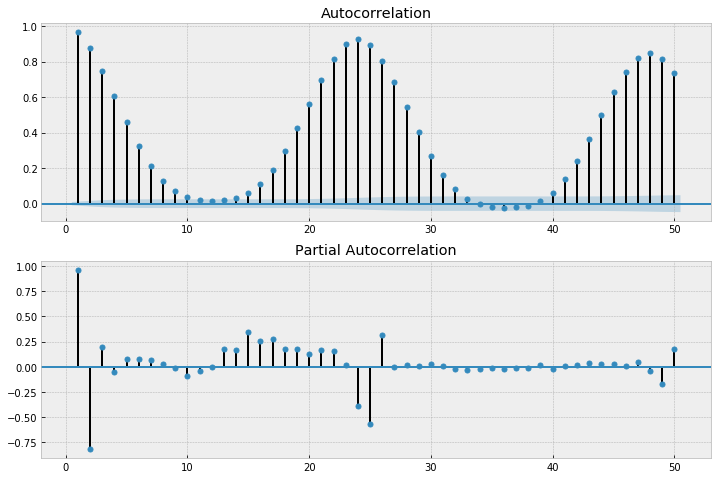


Figure ACF and PACF on our original TS

Usually, if the ACF plot tapers down smoothly and the PACF has no significant lags (falls below the confidence band) after a certain lag then q=0 and p=that lag, is determined as the order. And if PACF plot tapers down smoothly and ACF plot is insignificant after a certain lag then q=that lag and p=0 is determined as the order for the ARIMA model. But from our plots don’t give our any clear indication of which p or q values should be used.

Similar plots were plotted for a differenced version of the dataset (TS\_SDGE.diff(). dropna().diff(24).dropna() .diff(24\*365).dropna(), but still , no clear pattern to infer the p or q terms were observed.

**SARIMAX (p,d,q)x(P,D,Q,S)**

* ARIMA- Auto Regressive Integrated Moving Average Model
* S – Seasonality
* X – Exogenous variables like Temperature, PV installations, non\_working, etc.
* AR – concept of PACF; MA – concept of ACF
* But SARIMAX can handle only 1 seasonality and our energy time series has at least 3 evident seasonality patterns.
* We can handle this by creating seasonal X variables using Fourier series which can be passed into the model as exogenous variables (which we have already done in the previous section).

Since the ACF and PACF plots were inconclusive the pm.auto\_arima module was used to find the optimal order for the SARIMAX model. Running pm.auto\_arima with some parameters for tuning gave out the best model as SARIMAX(2,1,1)x(1,0,1,24). The auto regressive term p=2 means two values from the past (1 and 2 hours behind) will be used and moving average term of q=1 means 1 past term will be used as the moving average term. d=1 term means the energy series will be differenced once. Seasonal period m of 24 hours here and P=Q=1 means both the auto regressive term (P) and the moving average (Q) of exactly 1\*24 hours behind will be used.

Figure 26 shows a diagnostics plot for the above model. We can that the residuals are more skewed than the actual normal distribution and also they perform worse on the extreme ends. Also, the Correlogram shouldn’t have any significant lags indicating any kind of auto correlation between the current residual and its lag terms but it does show some correlation.

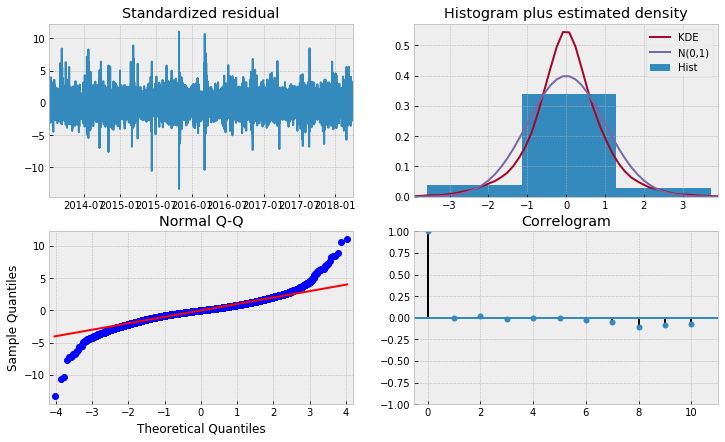


Figure Diagnostic plots for SARIMAX model (2,1,1)x(1,0,1,24)

Using the model to predict the 1st week of the test set we get a MAPE of 5.33% but the same error for the training set is 16.74% indicating that the model is not stable.

We see from Figure 27 that the first week forecast is pretty well but even at the end of first week the forecasting performance decreases and the confidence interval values grow larger beyond the scale of the range of the energy consumption values. Thus, SARIMAX model was not able to capture long term trends but it did well on 1 week ahead forecast.

SARIMA models don't capture multiple seasonality patterns well and are also very time consuming. So, it won't be the first choice if we need both a quick and accurate forecast.

Errors for 1 hour ahead forecasts weren't calculated above for SARIMAX model because we get excellent results using elastic net regression for 1 hour ahead forecasts using the lag variables and it is much faster to fit than SARIMAX. So, if 1 hour ahead forecasting is the goal then elastic net regression should be used anyways.

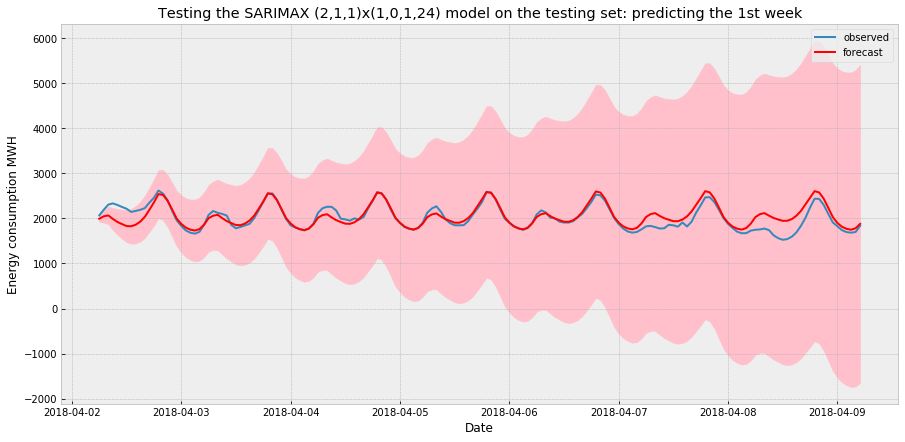


Figure SARIMAX (2,1,1)x(1,0,1,24) prediction on the 1st week of the test set

## 4.7 FB Prophet

Traditional approaches like SARIMAX models often require manual data pre-processing steps (e.g. differencing to make the data stationary) and it’s also hard to explain why these models produce the prediction results to people without forecasting expertise. In addition, these models are not allowed to add additional domain knowledge to improve precision. For solving these problems, Facebook researchers recently released FB Prophet, a time series forecasting tool supporting both Python and R.

FB Prophet provides a decomposition regression model that is extendable and configurable with interpretable parameters. Prophet frames the forecasting problem as a curve-fitting exercise rather than looking explicitly at the time-based dependence of each observation within a time series. Similar to SARIMAX, we can add extra regressor terms like temperature data to the model as well.

At its core, Prophet is an additive model with the following components:

y(t)=g(t)+s(t)+h(t)+ϵₜ

g(t) models trend, which describes long-term increase or decrease in the data. Prophet incorporates two trend models, a saturating growth model and a piecewise linear model, depending on the type of forecasting problem.

s(t) models seasonality with Fourier series, which describes how data is affected by seasonal factors such as the time of the year

h(t) models the effects of holidays or large events that impact business time series (e.g. new product launch, Black Friday, Superbowl, etc.)

ϵₜ represents an irreducible error term

* Using only the 'SDGE', 'HourlyDryBulbTemperature','cum\_AC\_kW', 'non\_working\_working' columns while using Prophet because Prophet, unlike SARIMAX, handles multiple seasonality patterns well. So, we don't need to pass in the Fourier terms separately.
* FB Prophet can be passed with a holiday feature, but since we have already captured the holidays and weekends in the 'non\_working\_working' column we won't pass a separate holiday list to Prophet.
* We can add the seasonalities separately in the Prophet model by replacing the 'auto' mode of the seasonalities above with 'FALSE'. Then we can add the yearly, weekly, daily, monthly, quarterly, etc. seasonalities using the .add\_seasoanlity feature of the Prophet specifying the period in days for a seasonality along with the Fourier terms to be used and the prior\_scale to set.
* After some trial and error, it was concluded that for this problem adding seasonalities manually doesn't give better results than the 'auto' mode, so keeping the seasonality as 'auto'.
* There are two mode options for any seasonality - 'additive' or 'multiplicative'. Multiplicative should be used if the seasonality affects the trend exponentially.
* External regressor like the temperature column are added to the model by using the 'add regressor' function of the Prophet. There is an option for Standardization while passing the regressors, so Scaler wasn't used here while creating the train and test splits.

Let’s check the performance of the FB Prophet on our SDGE energy dataset (Figure 28).

MAPE: 8.55%

FB Prophet has a feature .plot\_components which can be used to see how the data was decomposed by the model ().

From Figure 29 we can see that the multiple seasonalities were captured very well by the Prophet model. In the bottom most plot, the extra regressor additive terms include the temperature, non\_working\_working and cum\_AC\_kW variables. We see the impact of temperature and working days in the form of wiggles whereas the overall impact of the cum\_AC\_kW is a downward trend in the energy (as was expected since more PV gets installed at customer sites, the lower the demand on the grid).

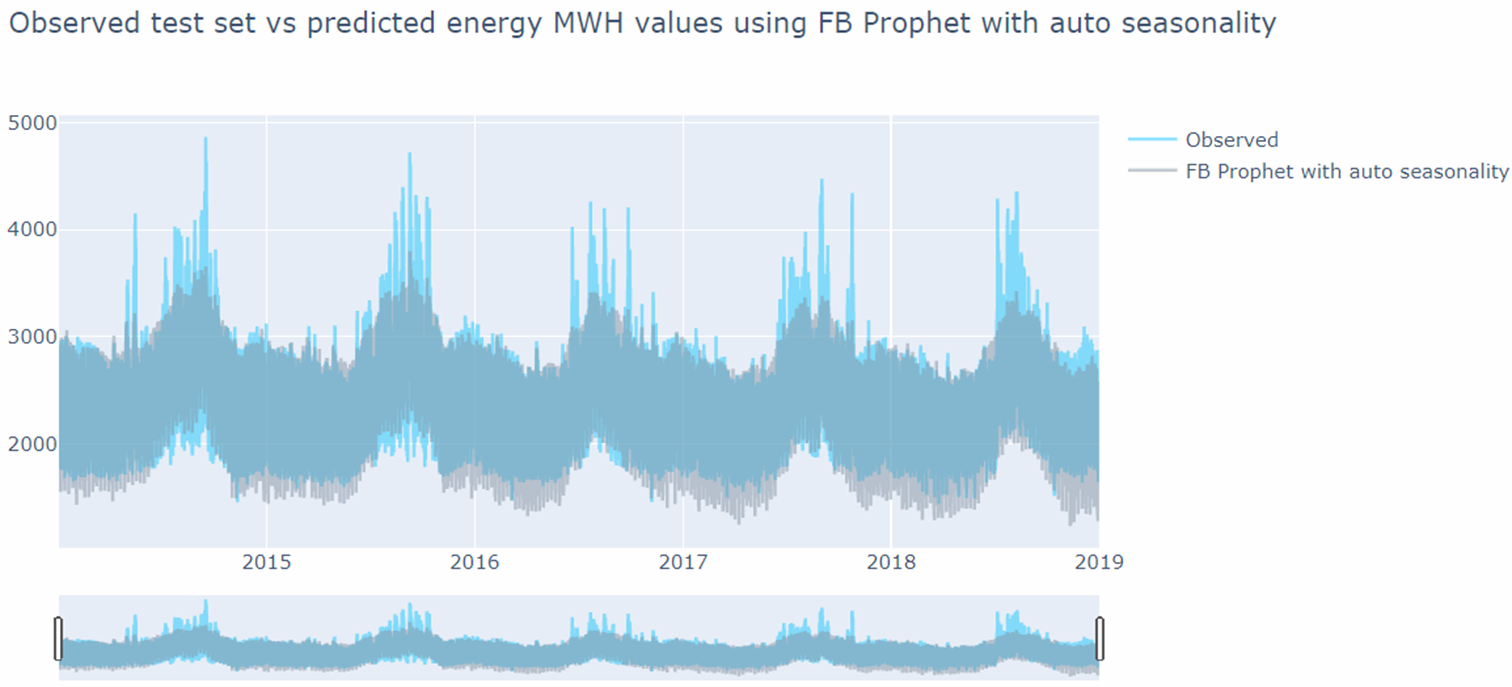


Figure FB Prophet predictions on the test set

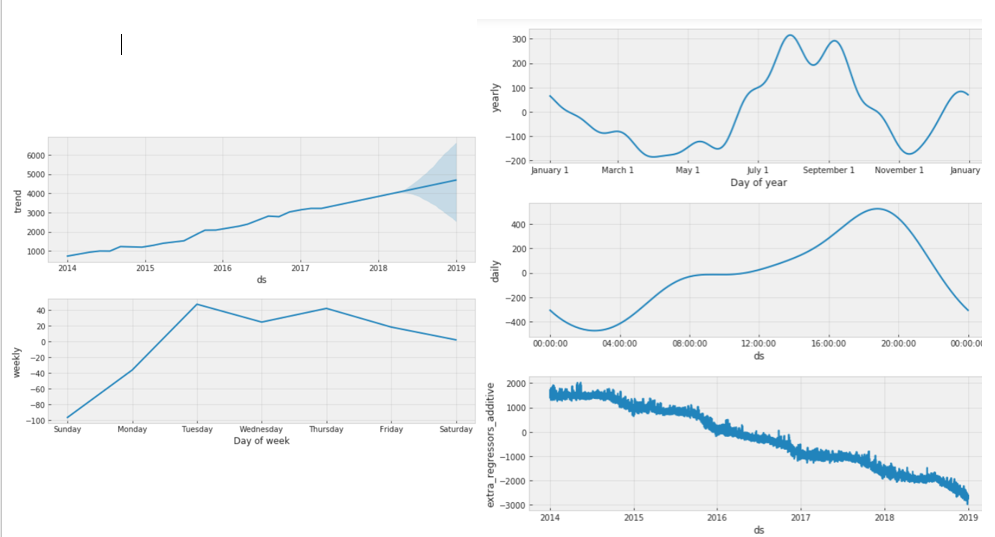


Figure FB Prophet components on the fitted model

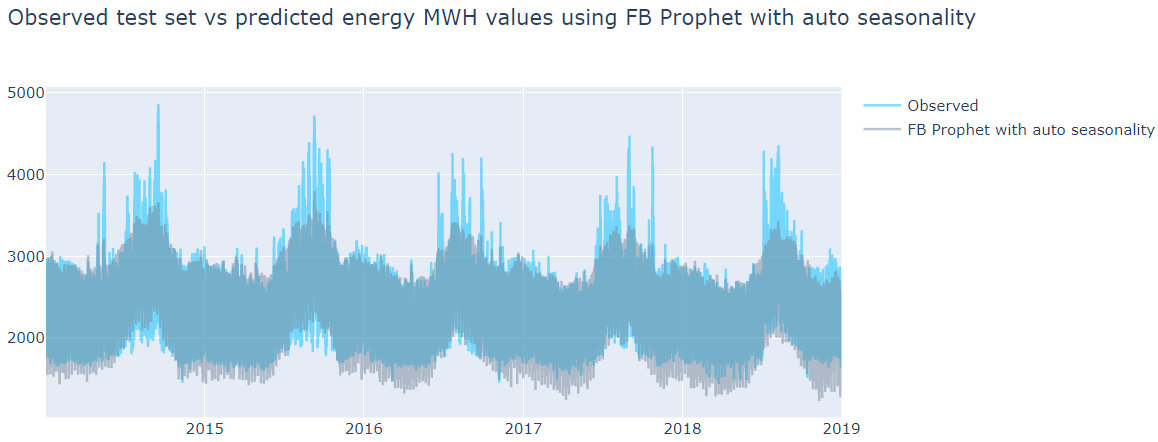


Figure Predictions on test set using FB Prophet

We can conclude that the MAPE of our FB Prophet model is high, but the model has learned the underlying structure of our data very well.

## 4.8 Regression models using Fourier terms

Regression models – Elastic net, RF and XGBoost were tried on the dataset with Fourier terms.

**Elastic net**

MAPE: 8.39%

**RF**

MAPE: 5.73%

RF did well in capturing the overall trend, multiple seasonal patterns and even many of the high peaks. Figure 32 shows how well the RF performs in predicting even the higher peak values.

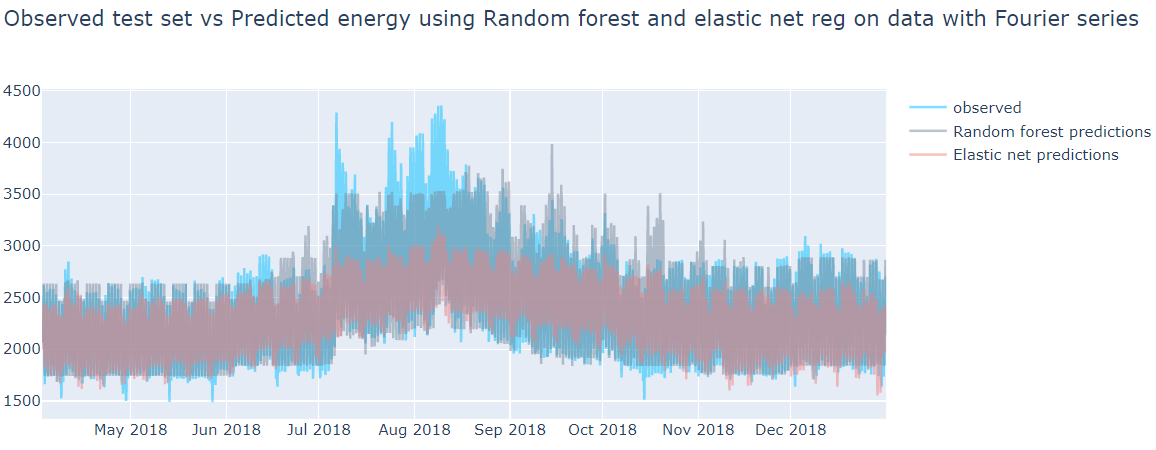


Figure Performance of Elastic net and RF on energy SDGE dataset with Fourier terms

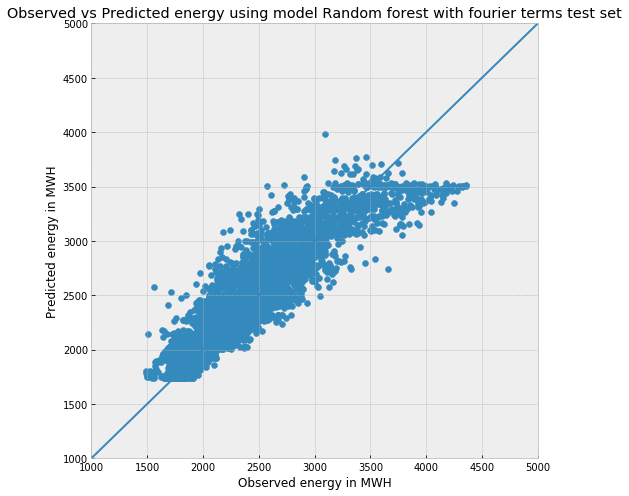


Figure Observed test vs RF predictions on the test dataset with Fourier terms

**XGBoost:**

XGBoost (Extreme Gradient Boosting) belongs to a family of boosting algorithms and uses the gradient boosting (GBM) framework at its core. It is an optimized distributed gradient boosting library.

XGBoost is well known to provide better solutions than other machine learning algorithms. It is not often used for time series, especially if the base used is trees because it is difficult to catch the trend with trees, but since our data doesn't have a very significant trend and also since it has multiple seasonalities and depends significantly on an exogenous variable like temperature, we can try XGboost to see how it performs on the time series data of energy consumption if we pass in the Fourier terms as the exogenous variables as done for the elastic net and RF models above.

With XGBoost the MAPE on test set obtained was 5.08% (after some hyper parameter tuning). The hyperparameters that were tuned are : max\_depth, learning\_rate, subsample, colsample\_bytree, colsample\_bylevel, min\_child\_weight, gamma, n\_estimators.

The performance of the model can be verified with the plots in Figure 33 and Figure 34.

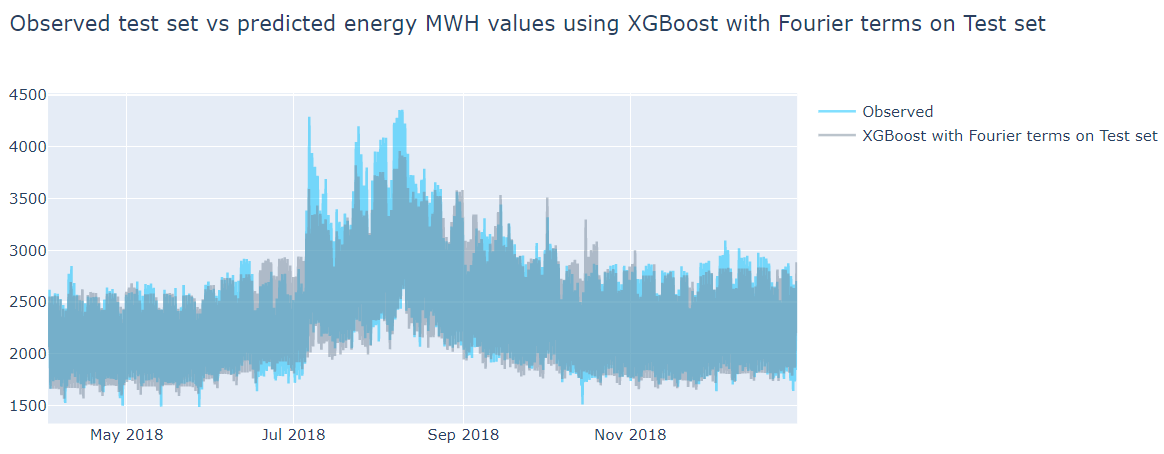


Figure XGBoost predictions on test set plotted with the true values of the test set

The XGBoost model was the fastest to fit, in fact, the entire model was tuned and fit in around 2 minutes.

The feature importance plot of the model in Figure 35 shows the most important features as: temperature, hour\_sin1 (the hour Fourier terms with k=1), PV installations, year\_sin1, hour\_cos1, hour\_sin2, …..more hour and year Fourier terms……, non\_working, and so on. These features were highlighted as important by almost all models we have tested in this project, but adding in the Fourier terms for multiple seasonality helped the tree based RF and XGBoost models to make good predictions overall.

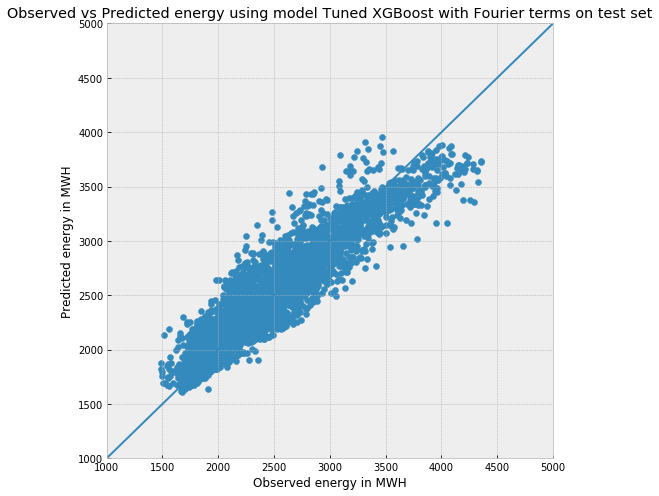


Figure Observed test values vs XGBoost predictions on test set with Fourier terms

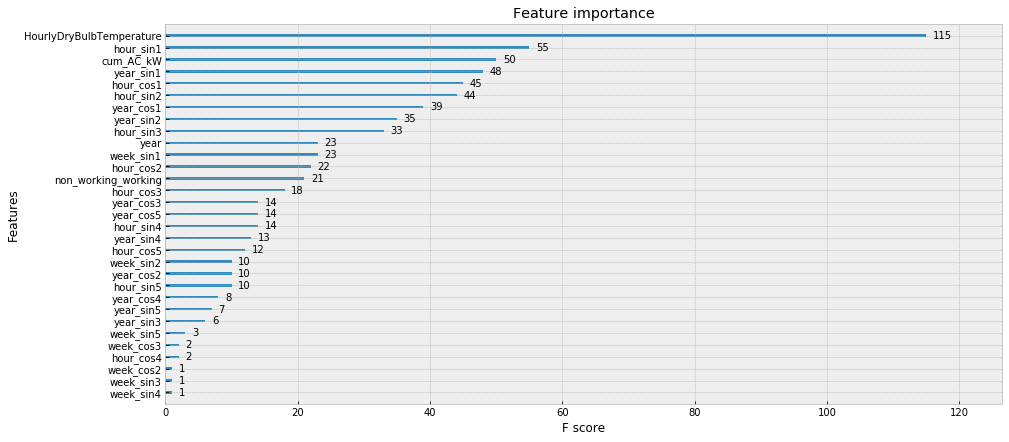


Figure Feature importance plot for the XGBoost model with Fourier terms

## 4.9 XGBoost + FB Prophet

As discussed earlier, to model a time series data it needs to be stationary. So, the ideal case would be to detrend the data and then feed it into the ML models and then add the trend to the forecasted results. Nonetheless good results were obtained above without detrending because the energy consumption data from 2014 to 2018 has a very weak trend and the multiple seasonalities were handled well by the Fourier terms.

Alternatively, the overall data trend and also the effect of cum\_AC\_kW, which is the cumulative PV installation till date, can be modeled using FB Prophet and then merged with the XGBoost's forecast. Any tree-based regression model like XGBoost cannot easily handle the X variables like cum\_AC\_kW because it is an ever increasing variable and the test data will always have higher magnitude values not seen by the model in the training set.

I have extracted the overall trend and the cum\_AC\_kW impact on energy from the FB Prophet model and subtracted these two components from our main energy data frame with all the Fourier terms. Then this detrended energy consumption data was passed onto the XGBoost model and the XGBoost forecast results were added back to the total trend to get the final predictions.

Figure 36 shows the combined overall trend with the effect of cumulative PV installations captured by the FB Prophet model. This combined effect was subtracted from the energy TS (i.e. the y values) of the data frame before feeding into the XGBoost model. This combined effect as added back to the predictions from XGBoost.

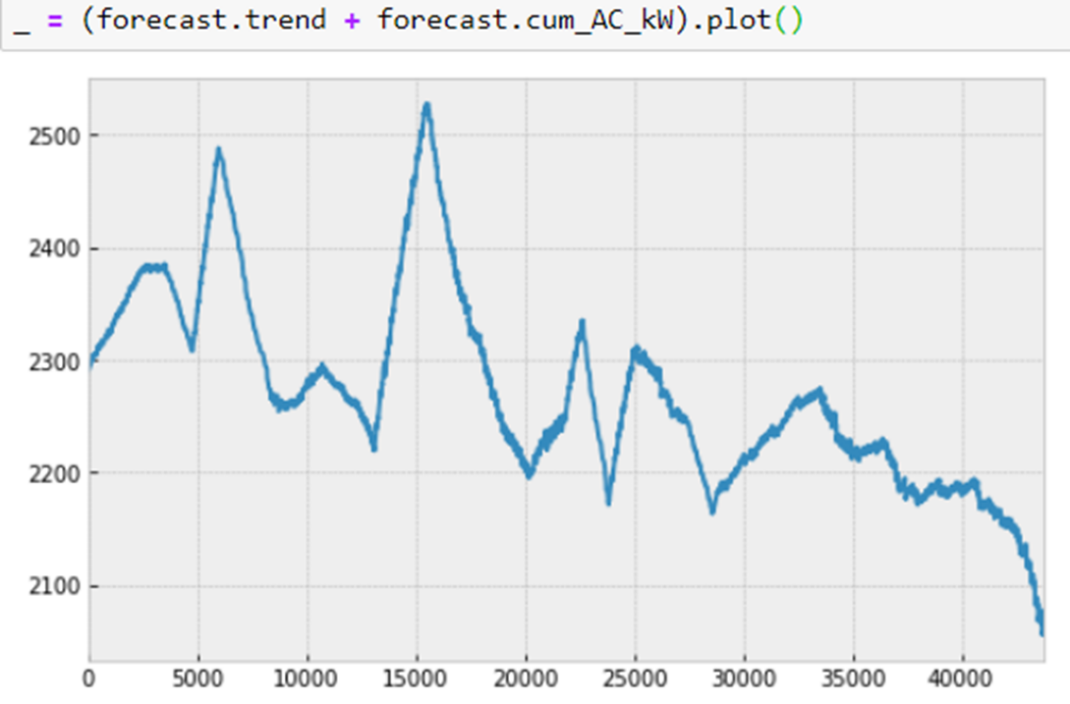


Figure Overall trend plus PV installation effect on energy consumption captured by FB Prophet

MAPE: 7.3%

Predictions were not better than the XGBoost alone (but better than FB Prophet alone)

This is because the combined trend was predicted more negative than actual by the FB Prophet as can be seen from the Figure 36, how the energy values fall very steeply at the end.

Need to check if trying different combinations of variable distribution among the two models gives better results. i.e. predicting the impact of Xn variables using 1 model and that of N-Xn variables using other model where N is the total number of X variables being considered.

# Conclusion

Different models were tried to forecast the energy consumption in MWH of the San Diego Gas and Electric (SDGE) utility region. The energy consumption is highly dependent on the outside temperature and has strong multiple seasonalities - daily, weekly and yearly. Also, the energy consumption has decreased slightly from 2014 to 2018. The increasing PV (photovoltaic) installations in the region (cum\_AC\_kW) seems to have brought the decreasing trend in the energy consumption since more renewable energy at customer's facility means less load on the utility. But there can be other factors causing this decreasing trend such as the energy storage installations at the customer facilities, increase in electric efficiencies of the household and commercial equipment, people becoming more conscious of their usage (morally or through utility incentives), etc.

The best way to capture the trend, which is a combination of all the above factors and maybe more, is to make the model learn the trend over a long period of time (more than 3 years at least). The seasonality is an important part in predicting the energy consumption of a region, so getting that part right was also very crucial for improving the model's performance.

Different models were tried and here is a summary of the error metric for each model including the baseline model where today's energy consumption is equal to the last year's energy consumption at the same hour.

(Note: Model with "all lags" and "1 week" at the end of their names are limited to 1-hour ahead and 1-week ahead forecasts respectively. All other models have a forecast window of roughly 8 months. And model with "daily max" at end compare the forecasted daily max with the actual daily max. Rest of the models compare hour to hour energy prediction.

The long term forecast accuracy will also depend on the forecast accuracy of independent variables like temperature and PV installation capacity. In this notebook we are using a test set of 2018 so we have the actual data for all the independent variables.)



Figure Error metrics for all the models tried on the energy dataset sorted with the lowest MAPE on test set ascendingly

Note: NaN values are for the models which couldn't be tested on either the training set or the test set.

Figure 38 shows only the MAPE error for all the models on train and test sets.

* Based on the MAPE and RMSE scores, the XGBoost model with the Fourier terms has performed the best, predicting for a forecasting window of 8 months ahead. For an hourly data with multiple seasonalities that is a pretty impressive result.
* For long term forecasts- most of the models have performed better than the baseline persistence model and the best model (XGBoost) gives a MAPE of 5.08% compared to the baseline error of 9.23% on the test set. The RMSE, R2 and MAE values are also considerably lower than the baseline model. The difference in RMSE with the baseline model is almost 160 MW which is pretty significant.
* FB Prophet does a very good job in identifying the trend and seasonalities of the data. It can be paired with XGBoost and a more robust long-term forecast can be obtained. The trend and cum\_AC\_kW regression coefficient information was extracted from trained FB Prophet model and it was used it to detrend the training set for XGBoost model. The XGBoost predictions were then combined with the predictions from both the models to give an aggregate forecast. Unfortunately, this didn't work very well as the forecast at the end of our time series seems to drop down at a faster rate than the observed.
* While comparing the models, I am not considering the elastic net and random forest regression with all lags because they are only good for hour ahead forecasts. So, pretty much any of the model given above using lag variables should be good for short term forecasts (95-98% R2 and 1-3% MAPE). Elastic net should be used for short term forecasts, given it had the highest accuracy and also the model training time is very less compared to SARIMAX.
* SARIMA performs terribly bad on long term forecasts and doesn't capture the multiple seasonalities well. Maybe more feature engineering can be done to help SARIMA identify multiple seasonalities but given how time consuming the model training for SARIMA is, it is better to focus the resources on other models.

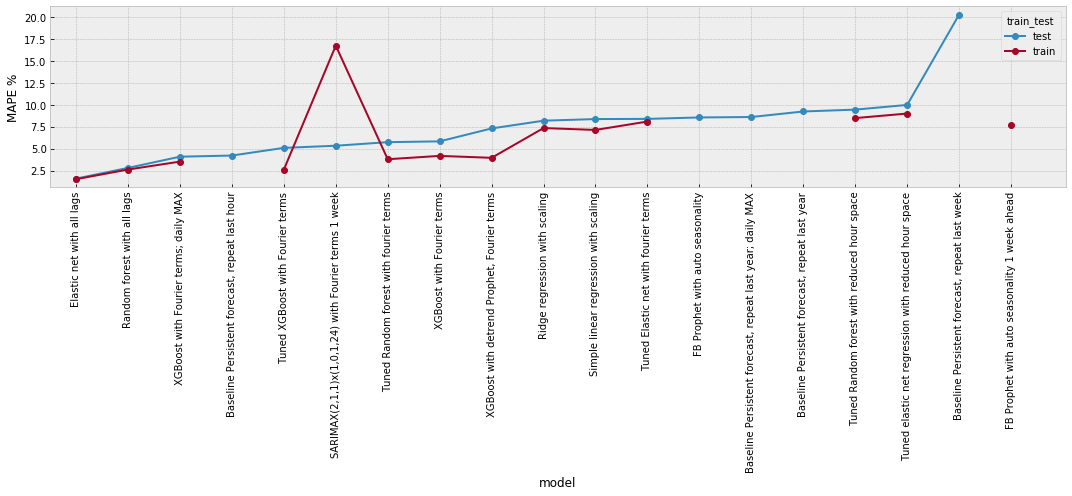


Figure MAPE of al the models on train and test sets

# Future work

* Try more methods where time series models and other traditional ML models could be clubbed together for better performance. Also, for the FB prophet and XGBoost combination, need to check if trying different combinations of variable distribution among the two models gives better results. i.e. predicting the impact of Xn variables using 1 model and that of N−Xn variables using other model where N is the total number of X variables being considered.
* Add std. error regions to the predictions.
* Bring in the new data of 2019 year and use it as a fresh test set.
* Try LSTM and SVR(linear & rbf both)
* Check the effect of other variables on energy consumption such as:
* The battery (energy storage) installation and Electric vehicle (EV) ownership data. A combination of solar panels and batteries usually result in lower dependency of a house or a building on the grid because their combination makes them more self-sufficient. And electric vehicles can add load at different times to the grid, so, battery installation and EV data clubbed with the PV data can help us improve the energy consumption prediction.
* Try some feature engineering by multiplying the PV and energy storage cumulative capacities with the cyclical hour data since their impact on energy consumption of a facility is highly time dependent. For example, solar panels produce electricity only during the day, so we can help the model understand that the PV impact should be accounted for only during the sunlight hours.
* Number of active residential and commercial buildings each month or quarter and overall economic condition of the region also can affect the consumption.

The nbviewer rendered jupyter notebooks for the EDA and the ML parts can be found here:

[EDA](https://nbviewer.jupyter.org/github/pratha19/Springboard_capstone_project_1/blob/master/SDGE_energy_EDA.ipynb#4)

[ML](https://nbviewer.jupyter.org/github/pratha19/Springboard_capstone_project_1/blob/master/SDGE_energy_ML.ipynb#8)

-------------------------------------------------------------------------------------------------------------------------------

1. All MAPE values unless otherwise specified were calculated on the test set. And also, unless otherwise specified (as 1 hour ahead or 1 week ahead) all MAPE values were calculated for long term forecasts (~8 months) [↑](#footnote-ref-1)
2. Though a sum is plotted here for simplicity, in actual there will be 10 different (5 sin and 5 cos) terms for each seasonality and the ML model will fit a coefficient on each of these terms separately depending on which terms are the most important ones. [↑](#footnote-ref-2)