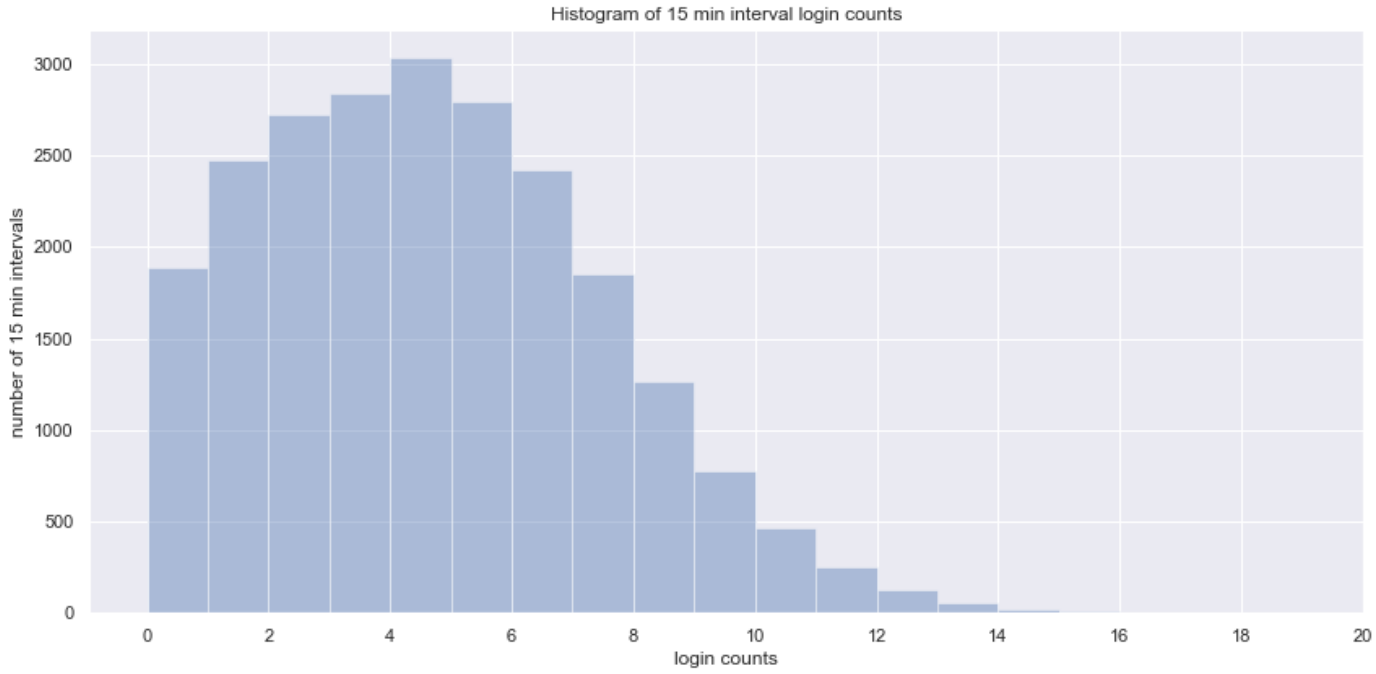
Part 1

1. Preliminary analysis - finding and visualizing patterns

The provided data contains timestamps of logins for about 8 months from January 1, 2010 to August 28, 2010. There are 240 distinct days in the data set, each of which has at least (minimum) 169 logins and at most (maximum) 542. Following the directions in the exercise description, I aggregated the data on 15 minute intervals and analyzed patterns/trends as follows.

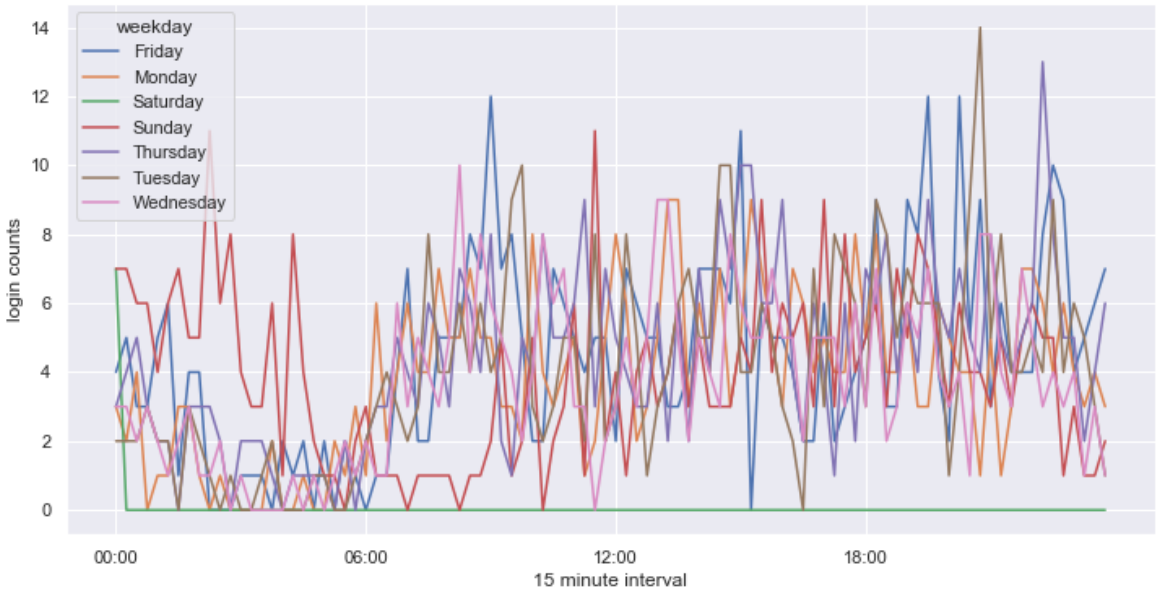
First, to understand the distribution of login counts for the 15 minute interval, the histogram is shown below.



The mean of login counts during the 15 minute interval is 4.2, standard deviation is 2.8, minimum 0 and maximum 19. Based on these basic statistics alone, the most naive login count estimates for prediction point and interval would be 4.2 (historical avg) and [1.4, 7.0] (one standard deviation away from the mean). This would be poor estimates as it does not take into account any underlying patterns. Part of the goal is to come up with a more robust way to estimate/predict login counts with accuracy and probabilistic confidence through statistical modeling and machine learning.

Taking a closer look at one week

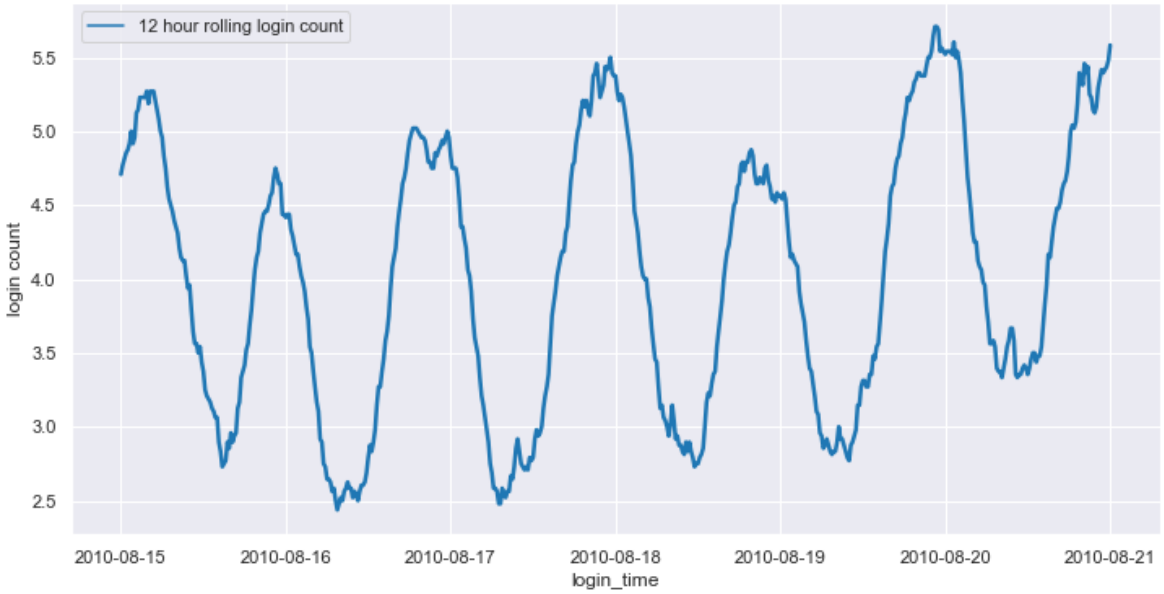
The time series plot below shows one particular week (week #34 from August 15 to August 21) with login counts broken down by day of the week. The graph indicates that there are hours of low login counts from midnight to 6am except for Sunday and increasing trends during 6am - 9am, again, except for Sunday. There tend to be periods of high volatility during 11am - 1pm, 3pm - 4pm and 7pm - 9pm on weekdays. These observations imply daily and weekly frequencies of seasonality.



The way to identify daily and weekly seasonality - moving (rolling) average

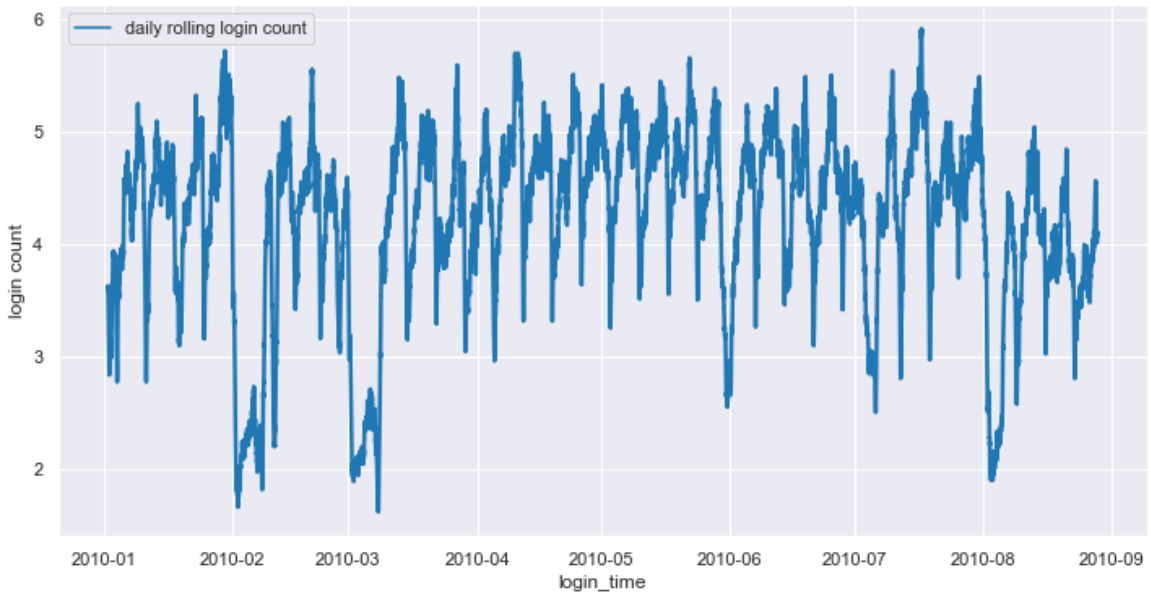
By smoothing out the time series data with one unit-smaller frequency of suspected seasonality, it visually produces a clear pattern when seasonality is in fact present. This also implies that the removal of one type of seasonality reveals another type of seasonality. Unless the plot depicts a stationary series, we need to keep looking into the next layer of seasonality. That said, when the data is aggregated at frequency of 15 minutes, monthly and annual types of seasonality would be a little too hard to make sense of. In other words, it's not so helpful to compare 15 minute time windows that are one month or even one year apart. For this reason, I only examine daily and weekly seasonality.

The 12 hour (half a day) moving average (mean of rolling 48 fifteen-minute intervals) below shows a clear pattern of daily seasonality for the week #34. It contrasts the low activities in the morning and high activity in the evening.



Not as strong as the daily seasonality, the weekly seasonality is also present as shown below. The graph shows the login counts for the entire period available in the data. The 4 dips and 4 peaks appear evenly on a weekly basis within a given month.

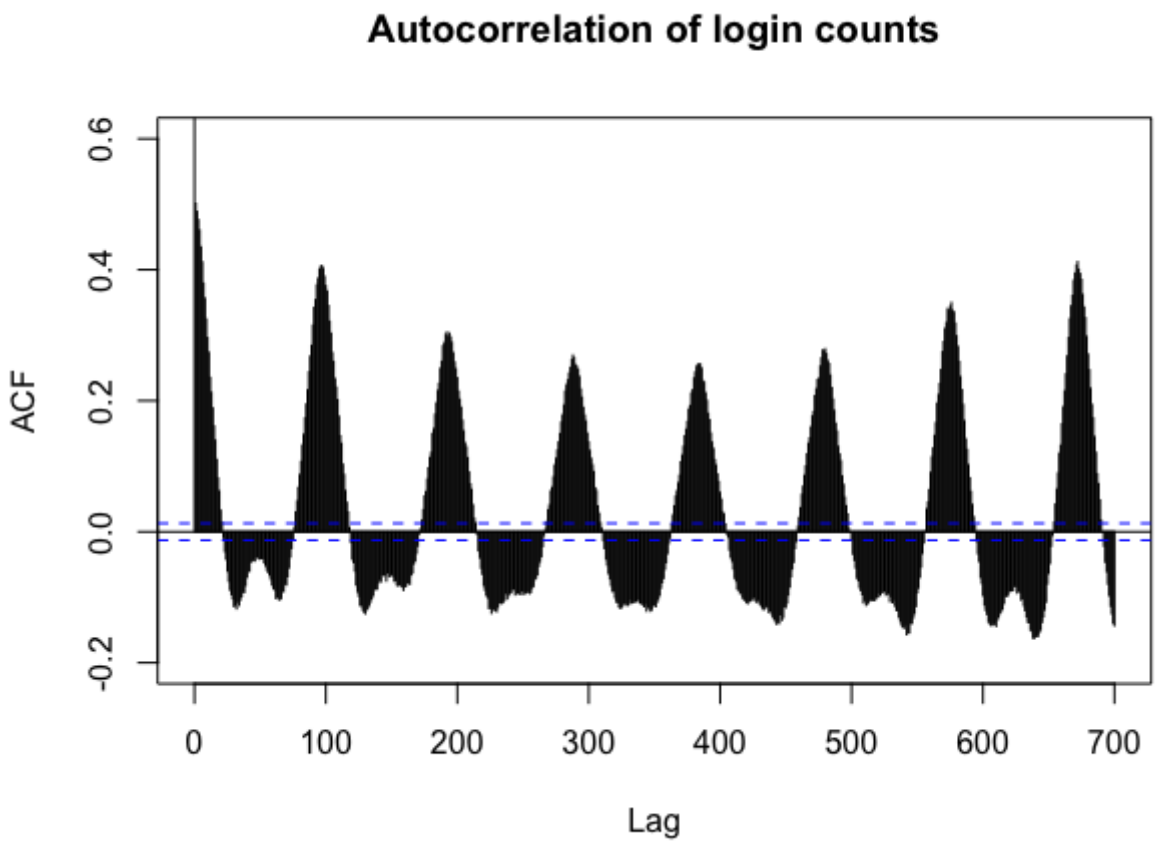
There might be a question of data quality on huge dips in February and March. We might have to investigate any external causes of these dips. If we look farther into the past for more historical data or accumulate login counts in the future, this pattern might be explained away with annual seasonality. But as the data covers only 8 months in this project, this question is out of scope.



Estimate seasonality with autocorrelation

In this particular case, the most powerful and visually clearest way to detect seasonality is to use autocorrelation plot. The graph below shows Pearson correlations with various lags up to 700 (about one week). This plot strongly confirms that:

1. Peaks occur at a multiple of 96 lags (one day) - daily seasonality
2. Positive peaks themselves form another layer of correlation. The peaks closer to one week cycle (those at lags 96,192, 576 and 672) are taller than other peaks - weekly seasonality.


Autocorrelation of login counts
ACF
-0.2
0.0
0.2
0.4
0.6
0
100
200
300
400
500
600
700
Lag

2. Prediction - Fourier Series Regression with ARIMA errors and Gradient Boosting

Now that we know daily and weekly seasonality in the data, one of the most well-known and reliable approaches to time series forecasting would be ARIMA (Autoregressive integrated moving average). However, ARIMA with two frequencies of seasonality is not so well-known. To my knowledge, there are no packages that handle ARIMA with multiple frequencies of seasonalty, at least in R. On top of this, ARIMA is better suited for a model that requires just a few coefficients. For our data with such a high frequency (92 periods for daily and 672 for weekly), a number of potential coefficients is hundreds. Directly applying ARIMA to this data is prohibitive.

Fourier Transformation with ARIMA errors

Because of the strong and multiple frequencies of seasonality, I applied Fourier Transformation to the data and used those Fourier terms as exogenous variables in the regression model. In this case, autocorrelation in the errors are likely to be present and can be channeled further into stationary series through ARIMA. Because both types of seasonality are handled by Fourier terms, there will be no seasonal differencing for regression errors in ARIMA. When this approach is successful, it also take a few coefficients to model the autocorrelated error terms.

The mathematical formulation of this approach is the following:

$$y_t = a + \sum_{i=1}^{M} \sum_{k=1}^{K_i} [\alpha \sin(\frac{2\pi kt}{p_i}) + \beta \cos(\frac{2\pi kt}{p_i})] + N_t$$

Here, *M* is 2 as there are two frequencies of sesoanlity. P_1 is 96 (daily) and P_2 is 672 (weekly). Our remaining task is to estimate K_i for each seasonality. In other words, we estimate how many sin and cos terms are necessary by applying grid search and sliding window time series cross validation.

As this approach involves ARIMA in error terms (N_t in the above), I used adjusted AIC (AICc) to assess the fit of the model with training data. To supplement the decision on the model selection, I also applied root mean squared errors (RMSE) in validating the model with test data in order to see its predictive power.

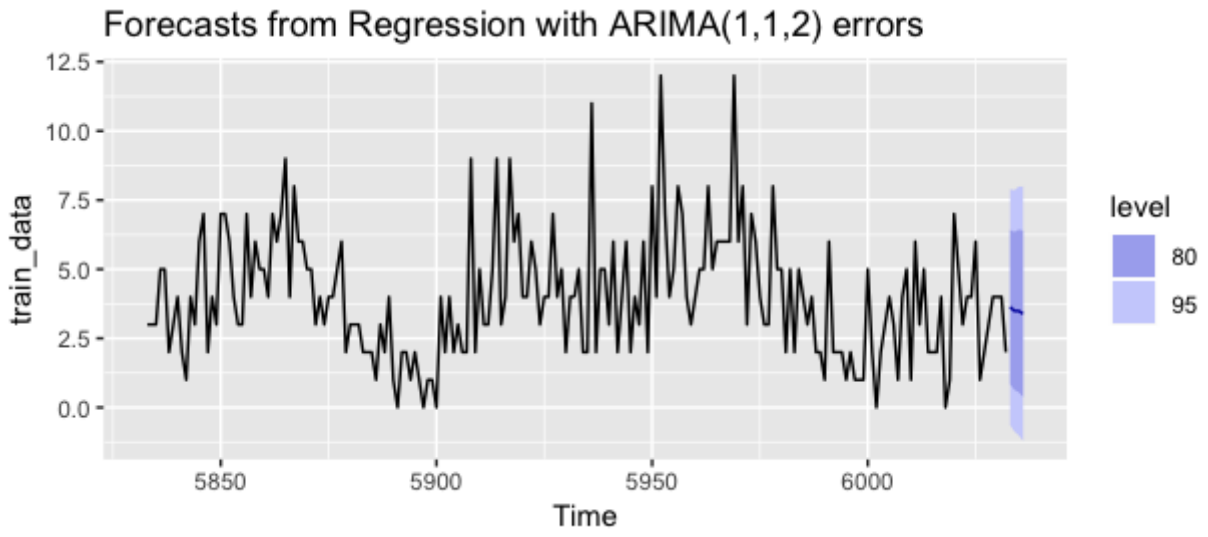
Since the data is a time series, cross validation needs to take the time sequence into account. As there is enough data in this project, I used the sliding window of 3 months as one period for training. In splitting the data, I also arranged each training data to end at 2:45 pm on 4th Saturday of the month and used the next 24 hours as the test data (except for August). The purpose of this arrangement is to simulate forecasting 4th Saturday of the month since the goal of this project is to predict the next one hour from 3pm on August 28 (the end of the provided data)

The results of the cross validation are shown below. As it turns out, forecast RMSE's are noticeably different from training period to training period. This is somewhat expected as we have seen varying degrees of volatility of login counts in different months. As we're targeting the forecast for August 28th and not necessarily fit of residuals, I focused on the predictive power (lowest RMSE) as long as AICc's are reasonably similar. In addition, when there are periodic variations in the time series data, we also need to focus on the most relevant part of the data, particularly in the case where all the possible patterns such as weather and events are not factored into the model.

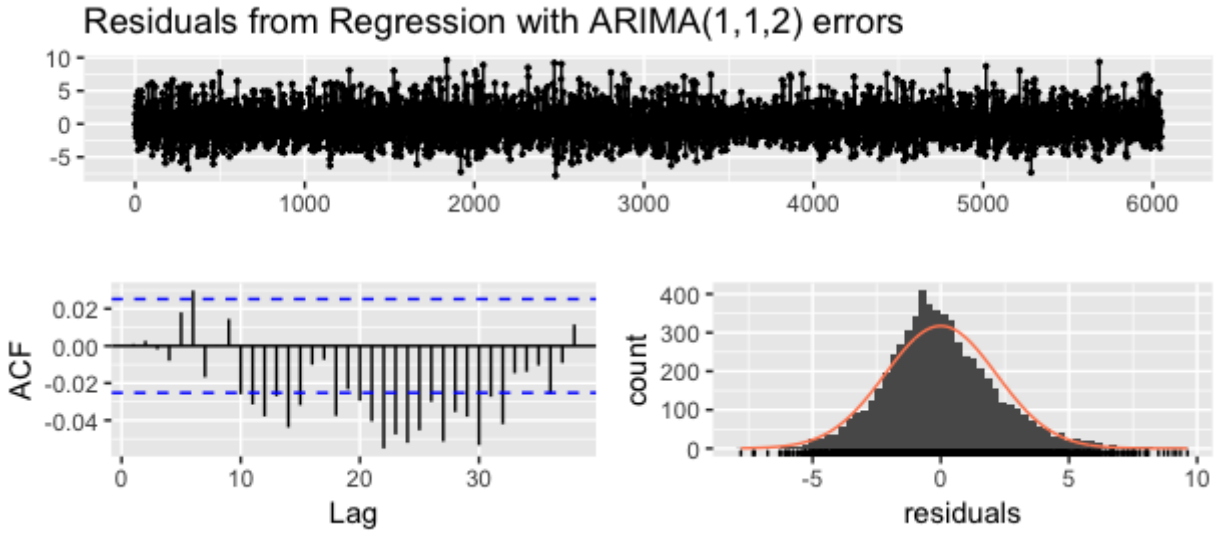
# of Fourier Terms for freq 96	# of Fourier Terms for freq 672	AICc	forecast RMSE	Training Period
2	4	35573	3.49	Jan. - Mar.
1	5	38617	1.97	Apr. - Jun.
1	6	23175	2.26	Jul. - Aug.

With 1 Fourier term at freq 96 and 6 terms at freq 672 and ARIMA(1,1,2) errors, I forecast for the one hour from 3pm on August 28th. The result is below:

time	login count mean forecast	Lo 80	Hi 80	Lo 95	Hi 95
Aug. 28 3:00pm	3.63	0.84	6.41	0 (-0.63)	7.89
Aug. 28 3:15pm	3.49	0.63	6.34	0 (-0.87)	7.85
Aug. 28 3:30pm	3.49	0.55	6.43	0 (-1.00)	7.98
Aug. 28 3:30pm	3.38	0.37	6.39	0 (-1.21)	7.98



The prediction interval is based on the forecast variance of residuals. They are assumed to be normally distributed and have zero autocorrelation. The variance also needs to be homoscedastic. To see whether these assumptions are met, the time series plot, ACF and histogram (with KDE) of residuals are shown below.



It seems that normality assumption is satisfied while homoscedasticity and (negative) correlation appear to be in question. The presence of the correlation in the errors could tighten interval more than necessary. The homoscedasticity of the variance could widen or tighten the interval, depending on the volatility of the period. Despite these rather minor deviations from the assumptions, the model was fit reasonably to produce relatively accurate forecast, at least in terms of test RMSE in cv.

Gradient Boosting

Another approach to forecasting time series data is to use machine learning algorithms, in particular, tree-based Gradient boosting Regression. This can be done successfully through feature engineering that captures both daily and weekly seasonality. The reason for the choice of gradient boosting was based on the number of features for this model. It does not seem to have that many features to take advantage from randomness as in random forest. The following is the list of features:

- **day_numeric**: numeric value from 0 to 1 in a day. Here, we still stick to 15 minute interval and use the center of the interval for numeric transformation. So, for example, 9:37am falls into the interval betwee 9:30am and 9:45am, the center of which is 9:37:30am. Therefore, it is expressed as (9+0.625)/24 = 0.4010
- **day_cos**: apply cosine to day_numeric (ie $\cos(2\pi$*day_numeric)). This feature captures the transition of the day. Say Monday 11:35pm and Tuesday 12:25 am are close to each other. since $\sin(2\pi$0.98438) = 0.9952 and $\sin(2\pi$*0.0260) = 0.9952
- **day_sin**: apply sine to day_numeric (ie $\sin(2\pi$*day_numeric)). It captures the daily pattern. It highlights the low activities early in the morning and high activities during the evening.
- **week_numeric**: numeric value from 0 to 1 in a week. It starts with 0 at Monday 0am and ends with 1 at Sunday midnight. For example, Tuesday 9:37am is expressed as (1+(9+0.625)/24)/7 = 0.2001
- **week_cos**: apply cosine to week_numeric.
- **week_sin**: apply sine to week_numeric
- **weekend**: dummy variable of 0 and 1. If weekend, 1. Otherwise, 0

Once the features were engineered, gradient boosting regression was fit and trained through grid search and cross validation. After the hyperparameters were tuned, the model was again fit with data to see test RMSE.

model	test RMSE
GBM	2.11
Fourier	2.26

GBM outperformed the Fourier ARIMA approach with a narrow margin. The forecast based on GMB is the following. The Fourier forecast is only shown for reference.

time	GBM forecast	Fourier forecast
Aug. 28 3:00pm	**4.80**	3.63
Aug. 28 3:15pm	**5.52**	3.49
Aug. 28 3:30pm	**4.89**	3.49
Aug. 28 3:45pm	**4.68**	3.38

The two forecasts are close. Both models predict a decreasing trend from 3pm to 4pm, except for one upward forecast at 3:15pm from GBM.

Prediction Interval of Gradient Boosting Regression Tree

Finally, prediction intervals of gradient boosting regression Tree can be estimated through the use of quantile loss function in the GBM model. By specifying the quantile, say 2.5 percentile and 97.5 percentile, 95% percent prediction interval could be estimated. As quantile (and percentile) does not depend on any parametric distribution, this approach has an advantage for accuracy when the assumptions of parametric approaches such as ARIMA are not met.

Following the directions of this project, I did not perform prediction interval estimates for GBM.

In Conclusion

Both Fourier Series with ARIMA errors and Gradient Boosting Regression Tree have almost the same predictive power. Both models outperformed the naive estimate since test RMSE is smaller than the standard deviation of 15 minute login counts. These models served the purpose of predicting the future demands better.

Next Step

Both models can be further improved with the addition of new exogenous variables such as weather, events, location (longitude, latitude). These variables could explain some of the erratic changes that we've seen earlier.

In [ ]: