**CUNY SPS MS Data Science**

**Comparing Performance of Single Predictor Models vs Multi- Dimensional Models in Predicting Time-Series Counts**

**Abstract**

Time-series models such as the Auto Regressive Integrated Moving Average (ARIMA) model are limited by the fact that they are one dimensional and are unable to use the plethora of information which might inform predictions. Further, not all models are conducive to count data. This paper compares the performance of a single-dimension ARIMA model against that of models which use a multi-dimensional feature space. Specifically, the author compares the predictive power of the ARIMA model against that of lagged Poisson regression and Random Forest regression models. The author chose these models for a variety of reasons. The Poisson regressor is conducive to modeling rare discrete counts, while the Random Forest model can inherently detect interactions. It should also be noted that the Poisson regression model is a parametric model, while the Random Forest regressor is not. Standard practices of model testing such as out of sample (OOS) cross validation with root mean squared error (RMSE) and possibly pinball loss are used in determining model performance. Finally, the author looks at the Facebook Prophet model which uses trends, Fourier series and automatic change point detection to model time-series.

**Introduction**

Timeseries prediction is valuable for many industries and can solve a variety of problems. The Autoregressive Moving Average model (ARMA) is one of the most popular models, which then developed into the Autoregressive Integrated Moving Average model (ARIMA). However, these models are essentially univariate timeseries models. The body of work surrounding univariate models seems to be larger than that of multivariate models, which inspired this paper’s intention: to look into the applicability of multivariate timeseries models to solve business problems. The author decided to explore both parametric and non-parametric models, including the new Facebook Prophet model, which relies on change point detection and Fourier series, along with the Random Forest model. The multivariate models can use additional weather information available in our data sources.

**Data**

The data used to answer this research question is concerning bike rentals in San Francisco for the period 2017 to 2020 from Lyft. This data provides a level of granularity that is useful for modeling seasonality. Further, this data is in the form of discrete counts: Each rental can be modeled as an arrival at a particular period of time: <https://www.lyft.com/bikes/bay-wheels/system-data>.

**Literature Review of Models**

**Auto-Regressive Integrated Moving Average (ARIMA) Model:**

ARIMA models are particularly conducive to data which are serially correlated. Tse (cited in Kumar & Anand, 2014) determined that the following questions should be answered before choosing an ARIMA model: (1) Is the data random? (2) Does the data have trends? A random series ensures that the correlation between previous and subsequent terms is small (close to zero). If the observations of time series are statistically dependent on each other, then the ARIMA is appropriate for the time series analysis.

**Poisson Regression**

In Poisson regression, linear predictors represent the log of the rate parameter for the Poisson probability mass function. The model will yield the probability of seeing the observed output given the linear parameters. The model will tune the coefficients attached to each feature in order to maximize the likelihood estimate (MLE) for the output variable.

Poisson regression is a useful substitute for linear regression when some of the assumptions of linear regression don’t hold. In linear regression, we assume that the conditional mean of the dependent variable is some function of the independent variables. The model also assumes that the variance is the same for all values of the dependent variable (Moksony and Hegedus, 2014). In Poisson regression, one assumption is that the mean must equal the variance (a characteristic of the Poisson random variable). This indicates that as the conditional mean of the dependent variable rises, so does the variance. Therefore, if data demonstrates that it is Poisson distributed, it violates an assumption of linear regression (constant variance across levels of output). Given that rare count data is often Poisson distributed, it is more suitable to be modeled as a Poisson random variable.

**Random Forest Regression:**

Random Forest regression is a brute force decision-tree method which uses values in the featurespace to split the outcome data; into bins that minimize variance at each split. As a non-parametric model, it will often provide different results to generalized linear models. This paper hopes to capture those differences. Random Forests are particularly robust to overfitting (Breiman, 2001), so we can expect to see consistent results between our training and test sets. We can also monitor out-of-bag error as we train data (Hastie, et al., 2009).

While it was pointed out that Random Forest regression is robust to overfitting, it does assume a similar number of samples per class or range (Marti, et al., 2009). If this is not the case, bias can infiltrate the model (in favor of the majority class/range). We assume this will not be a problem with timeseries data.

**Facebook Prophet:**

Another model we are can introduce is the Facebook Prophet timeseries model. Facebook Prophet does not use auto-regressive elements. Rather, it relies on Fourier Series to capture the dominant components of a signal as well as a trend component. The trend component isn’t influential here, given that the series is already stationary. Facebook works best when multiple human scale seasonalities are present (Taylor & Latham, 2017). Another advantage of Facebook Prophet is that it does not need to be regularly spaced. Further, the fit time for Facebook Prophet is very fast. Finally, Facebook Prophet has automatic changepoint detection capability.

**Limitations of Regression Models:**

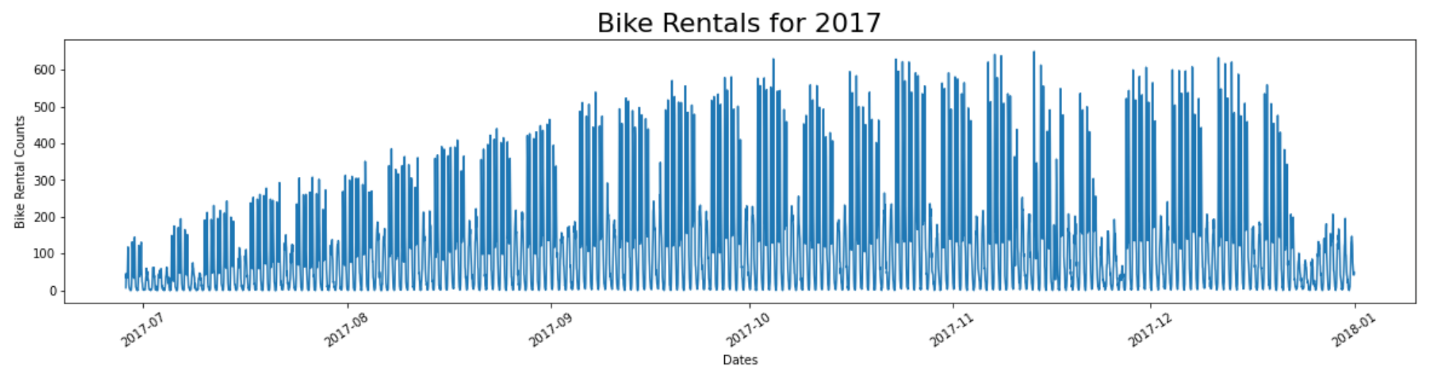
Random Forests are bound to fail predicting values for future dates. If we use time-series data as a feature, the model will split the feature at points where the variance in the outcome is minimized. However, the model cannot predict values outside of the range it has seen, so it’ll fail when we want to predict future dates. An example is in this article by Aman Arora:

<https://medium.com/datadriveninvestor/why-wont-time-series-data-and-random-forests-work-very-well-together-3c9f7b271631>

A solution to the problem pointed out above is to use lagged models. Time-series data is also a problem for Poisson regression, if the data are serially correlated (since the model assumes observations are independent). Further, a lagged Poisson model is very limited in its applicability; especially where stationary series are concerned (Brandt and Williams, 1998). Nevertheless, we will test the performance of both the above models against the ARIMA model. If Poisson regression fails, the author intends to explore Poisson exponentially weighted moving average (PEWMA) models.

**Initial Observations**

Our first model metrics will be performed on data for 2017. The data for 2017 is graphed below in figure 1. The data has been taken using aggregate rentals for a given hour. We can see daily and weekly seasonality, as would be expected.



*Fig 1: Bike rentals for 2017. The counts of rentals are aggregated for a given hour*

Our baseline predictor is an ARIMA model. We will compare the results from the ARIMA model to our multi-dimensional models.

**Fitting the ARIMA Model**

The ARIMA model in PMDARIMA (in the Python eco-system) will be used as our ARIMA baseline package. The model offers the option to toggle three main parameters:

p: The order of the auto-regressive (AR) model. The number of lag terms for autoregression.

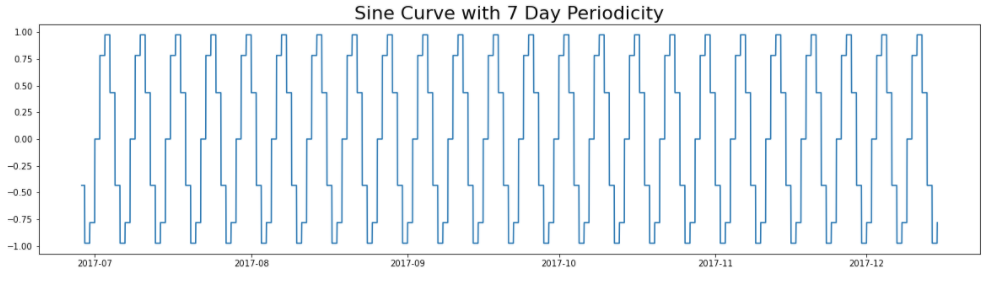
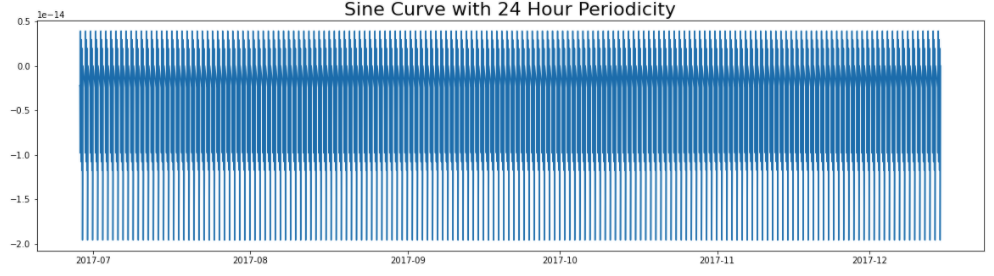
d: The integration order. This determines how many times we difference our data.

q: The order of the moving average (MA) model. The number of lag terms for errors.

We can add seasonality components by introducing a Fourier series as additional regressors.

**Fourier Series Regressors:**

Since we can see periodicity in the graph above, sine and cosine curves were added (with a cadence of 24 hours, 7 days and a 1 year) with the hopes that this would capture all the seasonality present in the data. Curves to incorporate weekly and daily periodicity are demonstrated below in figure 2:

*Fig 2: Sine curves with a cadence of seven days and 24 hours respectively*

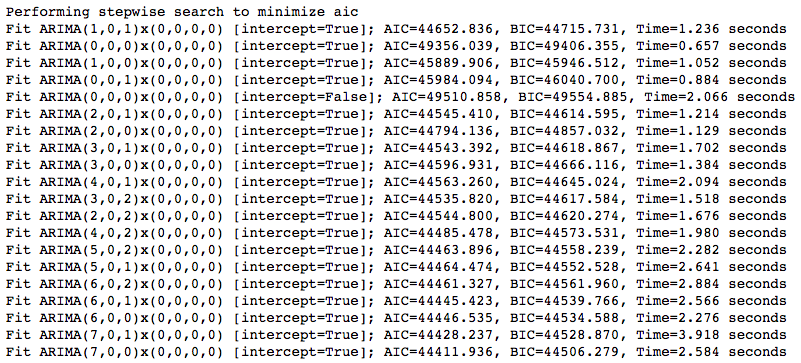
We’ll be setting the seasonal parameter in ARIMA to zero (0,0,0,0) and hope that the Fourier Series captures whatever seasonality there is.

**Dickey-Fuller Test**

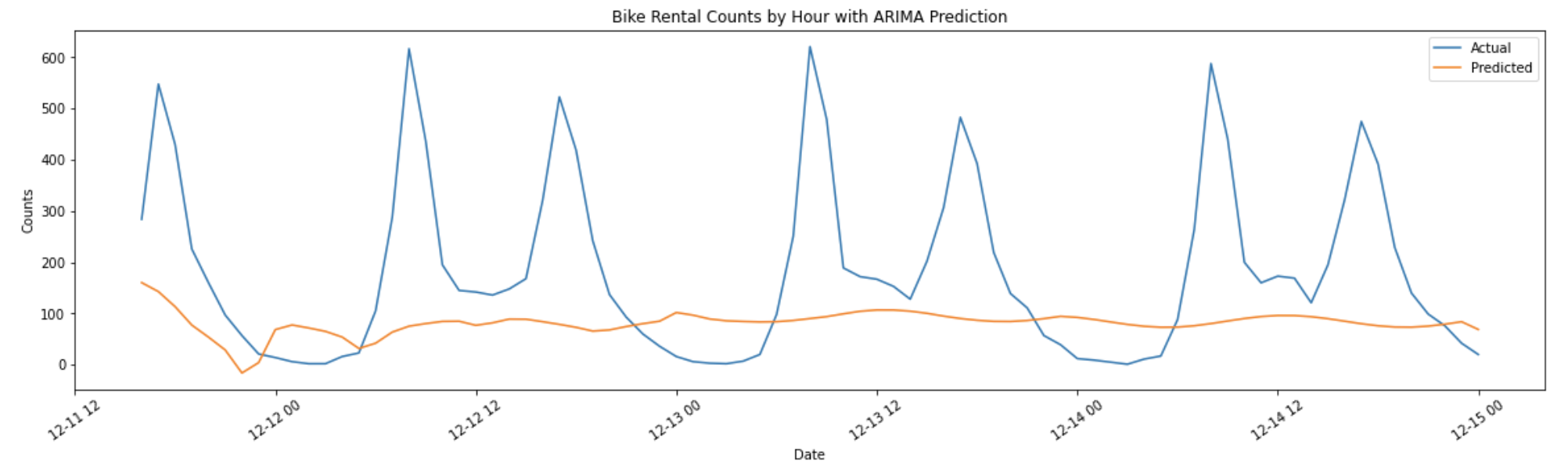
A major concern with ARIMA models is the stationarity of the time-series (Kumar & Anand, 2014), so we will use the Augmented Dickey-Fuller (ADF) to ensure we have a statistically stationary series. For the subset of data used (determined by the cutoff for training the data), the ADF test was performed, determined that the data was sufficiently stationary.

**Using Autoarima to Find Optimal p, d and q Values:**

We used PMDARIMA’s Autoarima module to make a first pass at the right combination of p, d and q parameters. While mean squared error (MSE) is often used in evaluating out of sample error, quite often, there aren’t enough out of sample errors to draw useful conclusions (Hyndman & Khandakar, 2008). For this reason, Akaike's Information Criterion (AIC) is used in Autoariama to find the optimal model. The fitting of the model can be computationally expensive, so we set limits for p and q at seven (7). Further, given that we are using Fourier series to capture the cadence of the signal (peaks and troughs of the timeseries), we have not used the seasonal order parameter in Autoarima. The seasonal order adjusts for p, d and q for seasonality and expects the user to specify a seasonal cadence (7 for a week, for example). However, the caveat is that the user must know the tempo in advance.



Interestingly, the lowest AIC was achieved when the MA order was at zero. More diagnostics are required to determine why this might be the case. An initial pass with Autoarima using the optimal parameters provided the following predictions. The graph does not show promising results. We can see that the ARIMA model failed to capture the pronounced peaks present at regular intervals.



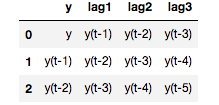
*Fig 2: Autoarima fitted timeseries prediction for 90 periods*

**Poisson and Random Forest Models**

Now we can move onto the multi-dimensional models.

**Feature Engineering**

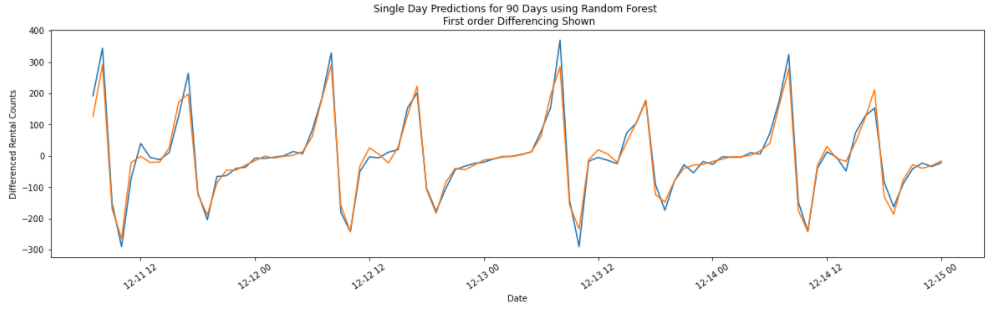
The first step of feature engineering includes differencing the outcome variable. We will then add lagged terms to the featurespace (Yt-1, Yt-2, etc.). Once we predict the outcomes for future dates, we add back the lagged terms (integrate). Hopefully, this will yield the desired results.We will look at weather related data to enhance the featurespace. Also, relevant holidays may have a bearing on time-series data. However, our first pass will only introduce the lagged y variable (output variable) in the featurespace. This is what the data would look like with three (3) lagged terms:



**Challenges with Differencing:**

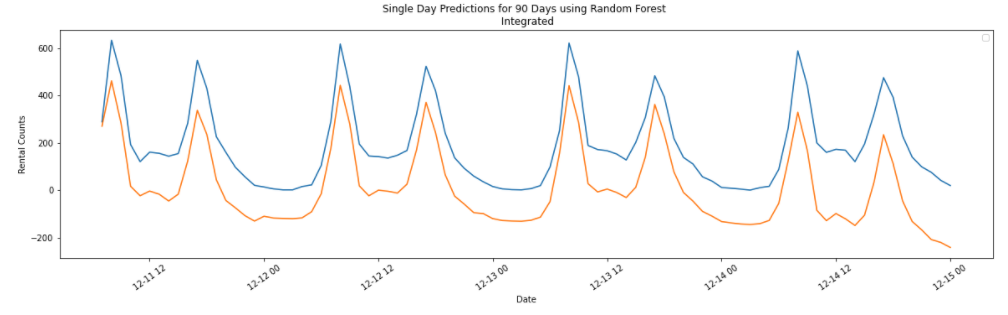
The problem here is that a Poisson regression model cannot estimate negative values, since it is estimating rare counts. Therefore, differencing will not produce desirable results, and should be avoided. Even in situations where a model is able to work with differenced counts, differencing (and integrating) a relatively stationary timeseries can yield unwanted results.

Here’s a look at single day predictions from Random Forest. With five-day lags in the featurespace, we can see that we are getting very accurate predictions. We’d expect this to be the case, given that we a predicting one day at a time.



*Fig 3: Single day predictions for the Random Forest model differenced (first order differencing)*

However, bear in mind that the series above is differenced. What happens when then the series is integrated? See below:



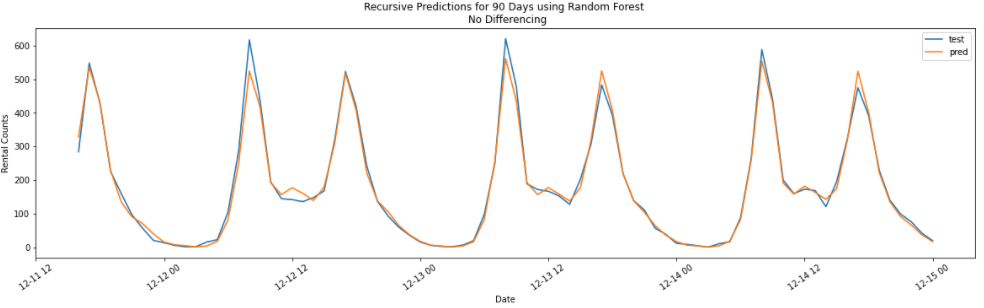
*Fig 4: Single day predictions for the Random Forest model differenced (integrated model)*

This indicates that while differencing a stationary series is not necessary, doing so can yield worse results that predictions on the original series

**Recursive Predictions**

The real test is to see how well we do when we recursively predict out for multiple dates. Given that both the Poisson model and the Random Forest model in Python’s SK-Learn library predict a single outcome at a time, we need to predict future outcomes recursively based on past predictions. This is not dissimilar to the methodology used in the ARIMA model, except that the multi-dimensional models are using the past predictions located in the feature-space (X variables), and not the predicted variable (Y variable).

The method used was to predict out for ninety days from using the preceding timeseries for training. We can see that when using a featurespace of twenty (20) lags, the model seems to perform exceptionally well (see figure 5). However, we’ll need to see the performance on different sections on the timeseries.

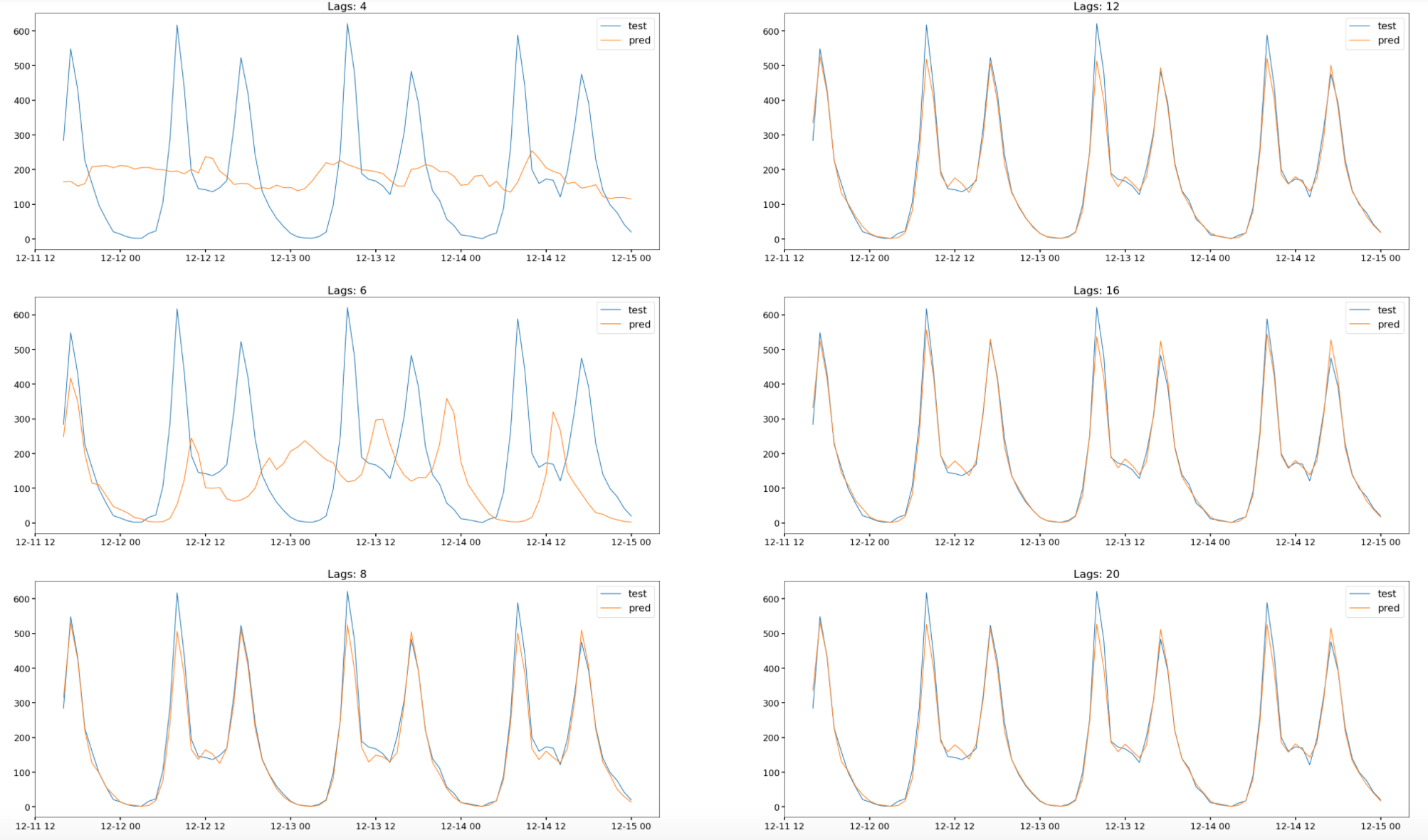


*Fig 5: Single day predictions for the Random Forest model differenced (first order differencing)*

**Grid-Search on Number of Lags**

Further, the number of lagged terms used was toggled. The number of lagged terms used ranged from four (4) to twenty (20). Once we reached a series of lagged terms of eight (8), the model began performing well.

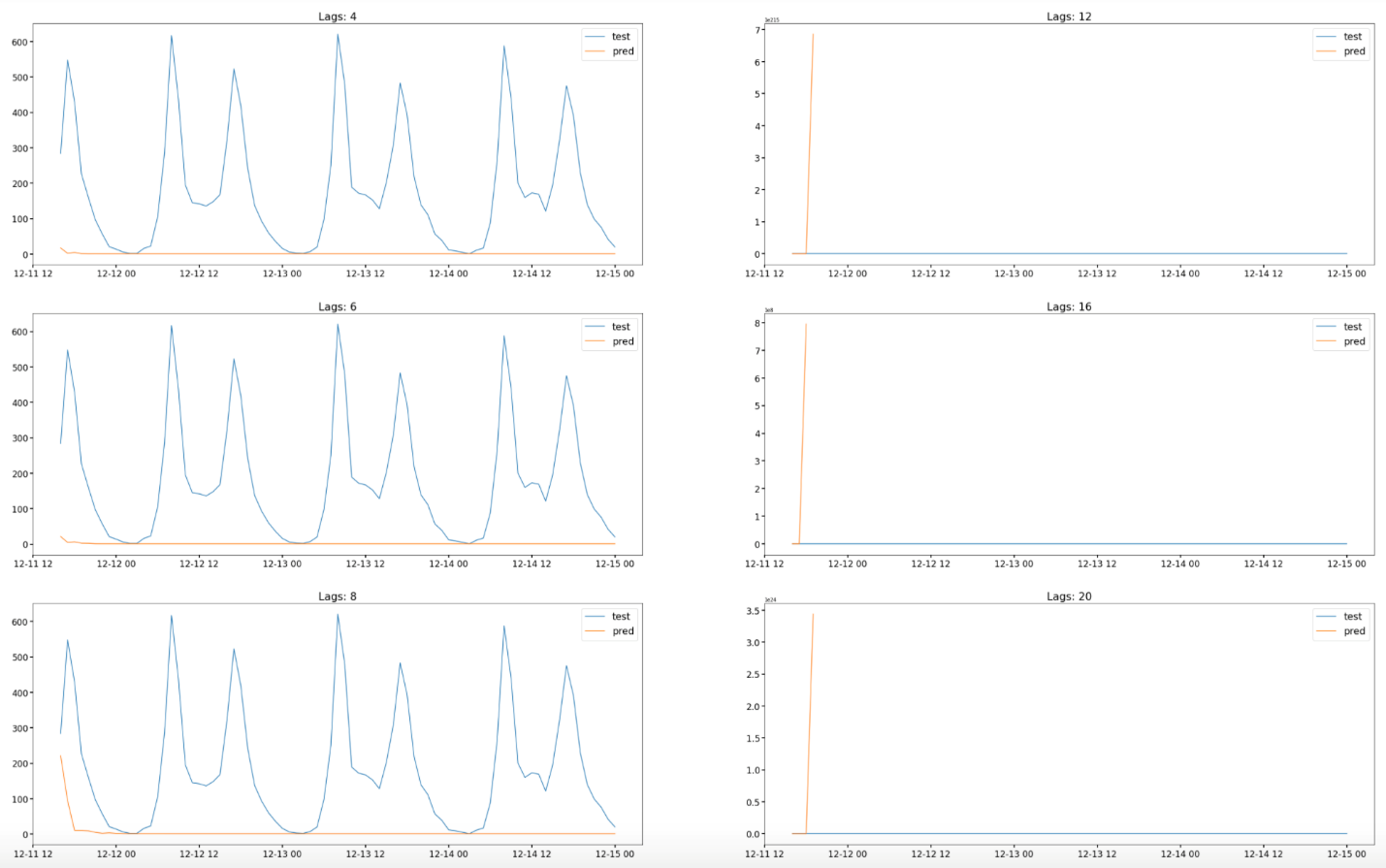
Grid Search on Ideal Lag Terms for Random Forest



*Fig 6: Grid-Search on Lag Terms for Random Forest model. Predictions in yellow.*

A similar experiment was run with Poisson regression. The Poisson regressor seems to be performing a lot worse than the Random Forest regressor. The reason seemed to be that beyond a certain point, the predictions tended to infinity. Once we have an entry for infinity in the featurespace, the model yields a null output. After that, the predictions tend toward null values.

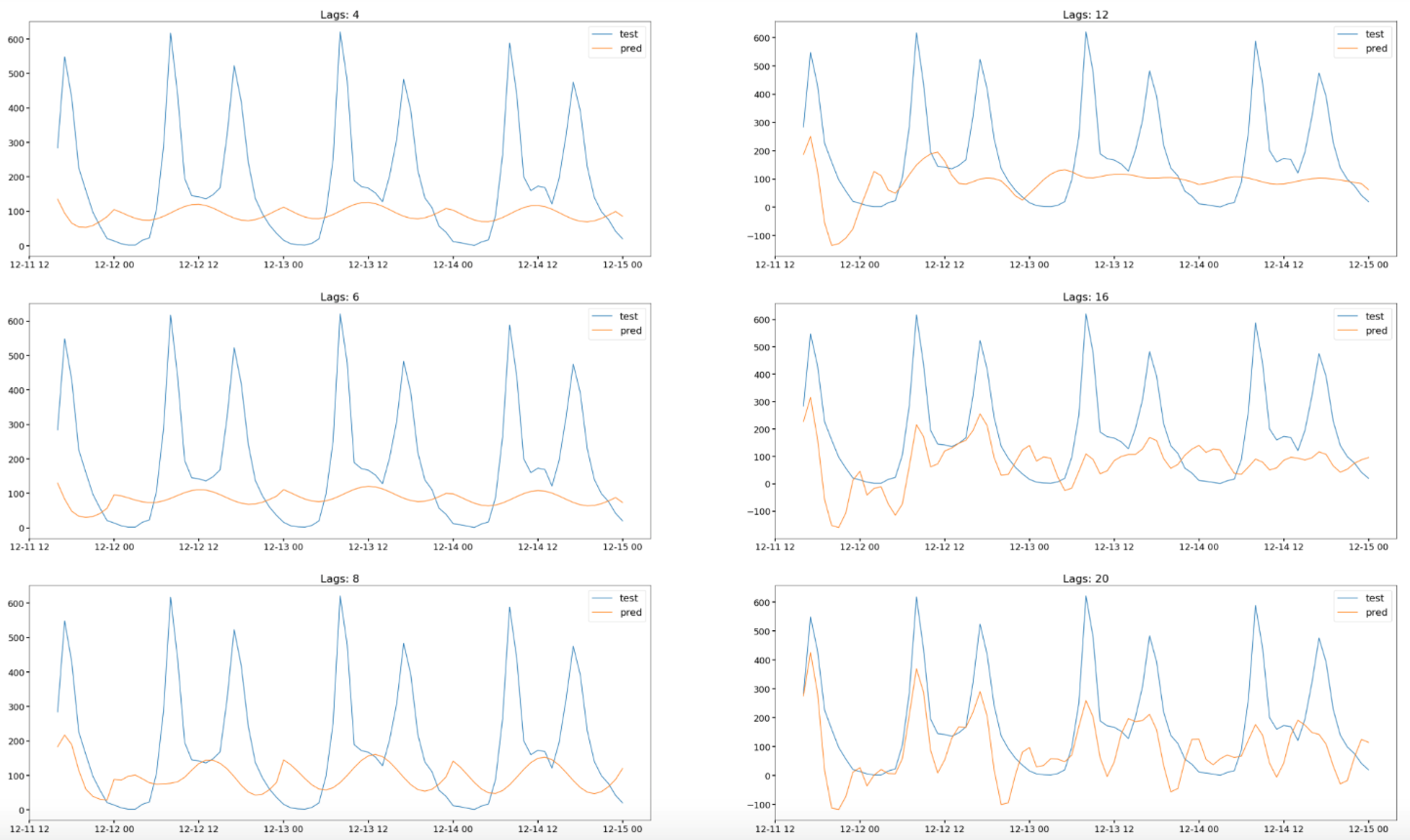
Grid-Search on Lag Terms for Poisson Regression



*Fig 7: Grid-Search on Lag Terms for Poisson Regression. Predictions with higher lags tend to infinity*

Finally, we toggled the number of lags with the ARIMA model. We already know that we achieve better results with no differencing (for this particular series). We also know from our experiments with Autoarima that the model prefers no moving average (MA) weights. As can be seen from the graphs below, the fit looks poor. It seems that irrespective of the number of autoregressive lags used, the model is failing on this particular segment of the timeseries.

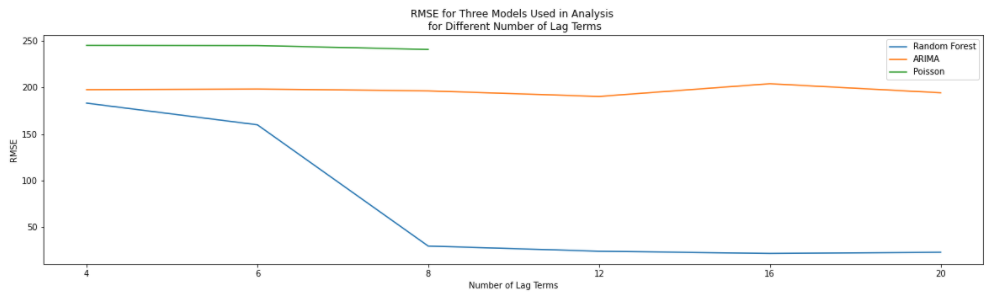
Grid-Search on Ideal Lag Terms for ARIMA (predictions in yellow)



*Fig 8: Grid-Search on Ideal Lag Terms for ARIMA*

**Error Metrics**

We’ll use both Root Mean Square Error (RMSE) as well as pinball loss to determine how well our data has predicted unseen data. Finding confidence intervals for pinball loss is complicated and, therefore, will be derived at a later time. As can be seen from the error metrics, The Random Forest model seems to outperform all others.



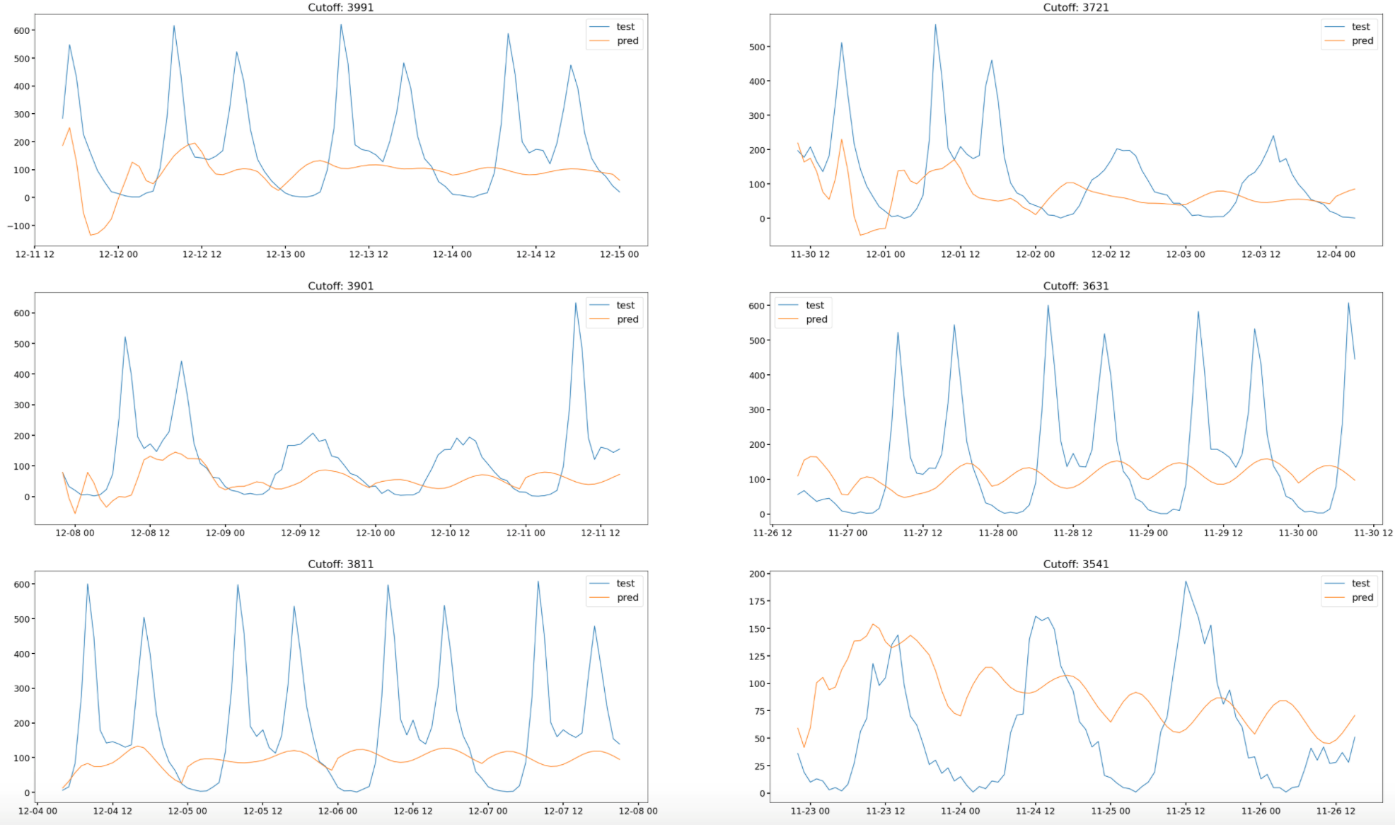
*Fig 9: RMSE for Random Forest, ARIMA and Poisson models. After 12 lags, the Poisson model’s iterative predictions tend to infinity, and can no longer be plotted.*

However, we should note that we performed these metrics on a single portion of the timeseries. There is no guarantee that we will have the same performance on other portions of the timeseries as well. Therefore, let’s move onto performing a grid-search on the different portions of the timeseries.

**Cross-Validation on Expanding / Diminishing Window**

We pointed out earlier that regular cross-validation is not appropriate for timeseries data. Bergmeir, et al., (2008) point out that out of sample (OOS) cross-validation is better suited for this purpose. We will use an expanding / diminishing window for cross-validation. This is essentially cutting off the training sample at specified increments / decrements and forecasting out for a fixed window. We have forecasted out for ninety (90) days. The results below are for cross-validation on the ARIMA model. As we can see, depending on where the cutoff resides, the quality of the forecast changes.

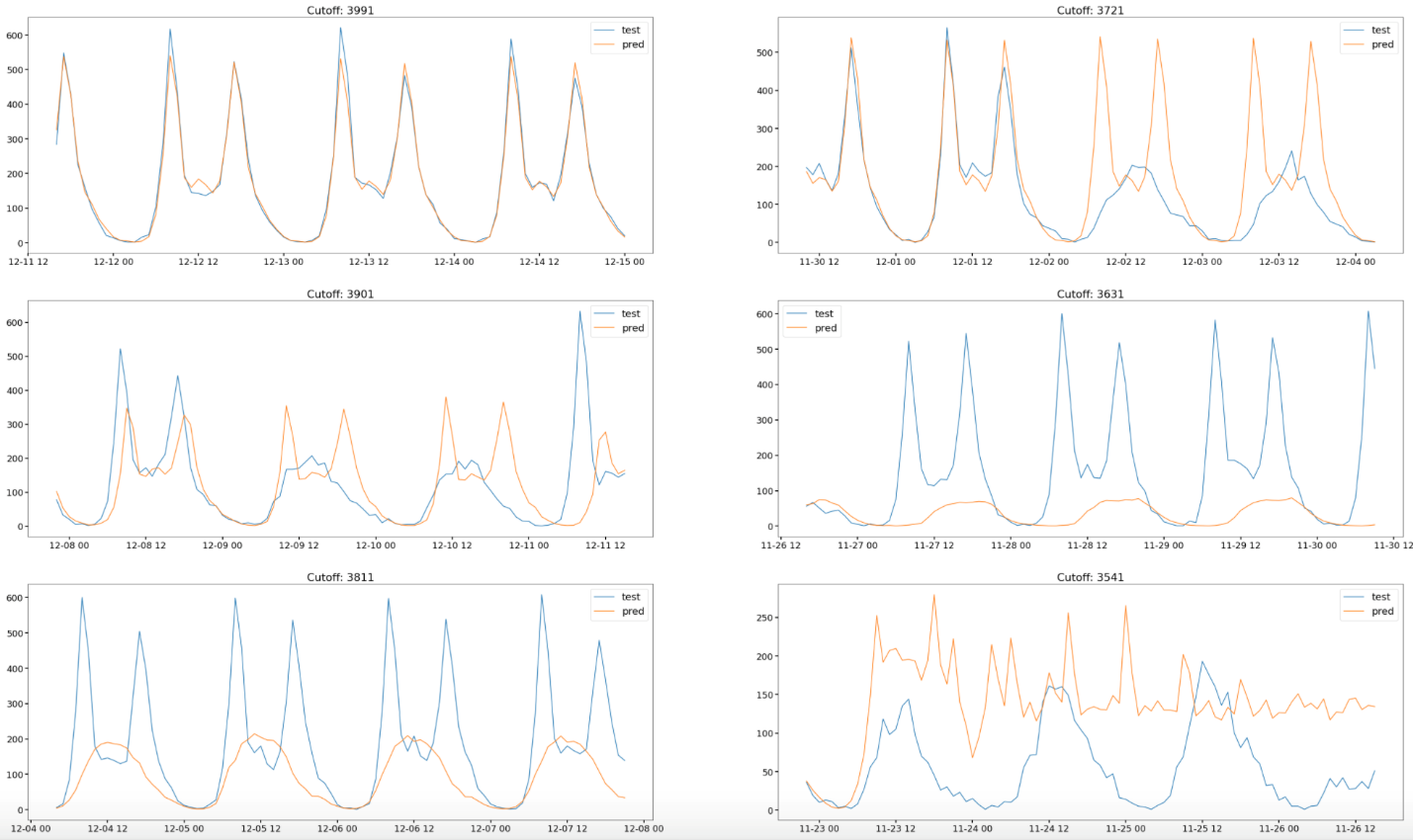
OOS Cross-Validation on the ARIMA Model (predictions in yellow)



*Fig 10: OOS Cross-Validation on the ARIMA Model (predictions in yellow)*

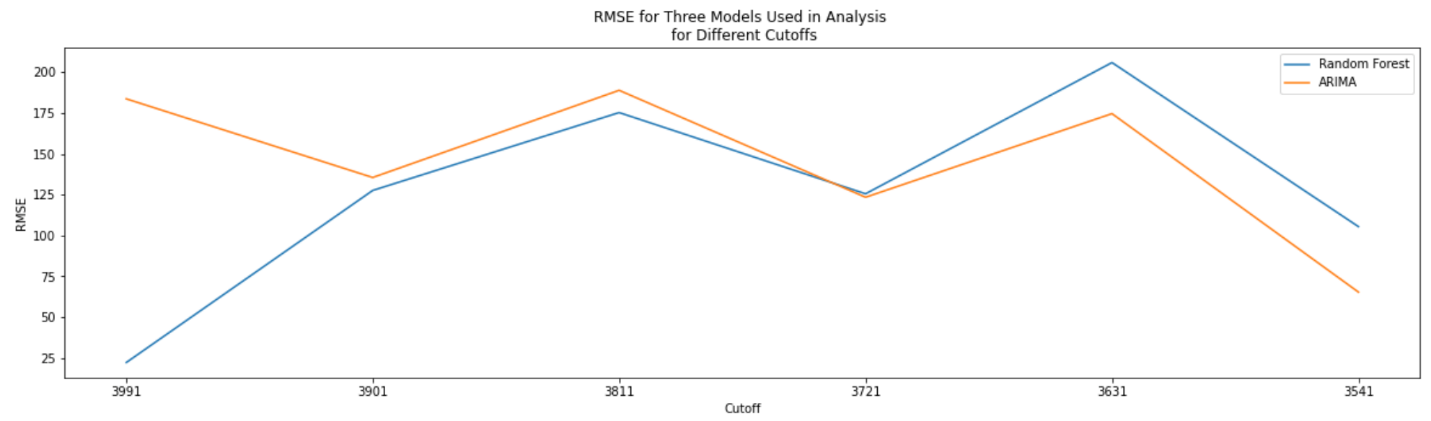
We discarded the Poisson regression model for the time-being and decided to concentrate on the Random Forest model as our multi-dimensional model. There are some visual clues to the predictions; which are that Random Forest does capture the peaks better than the ARIMA model. Since Random Forest is not a parametric model, it does not need to find coefficients, but rather splits in the features-pace.

OOS Cross-Validation on the Random Forest Model with 20 Lags (predictions in yellow)



*Fig 11: OOS Cross-Validation on the Random Forest Model (predictions in yellow). The training sample uses an expanding window. The Random Forest model uses 20 lags.*

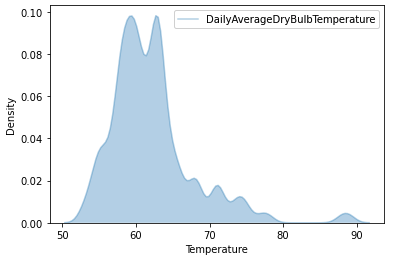
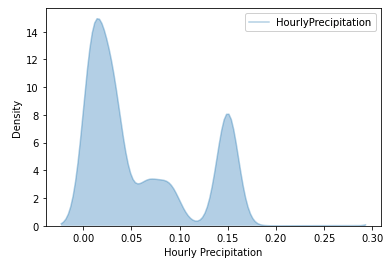
If we look at the RMSE at different cutoffs, it is not clear which model actually performs better. However, as noted previously the non-parametric model (Random Forest) seems to capture extreme values a lot better than the ARIMA model.



*Fig 12: RMSE for Random Forest, and ARIMA models based on the expanding window method.*

**Adding Weather Information:**

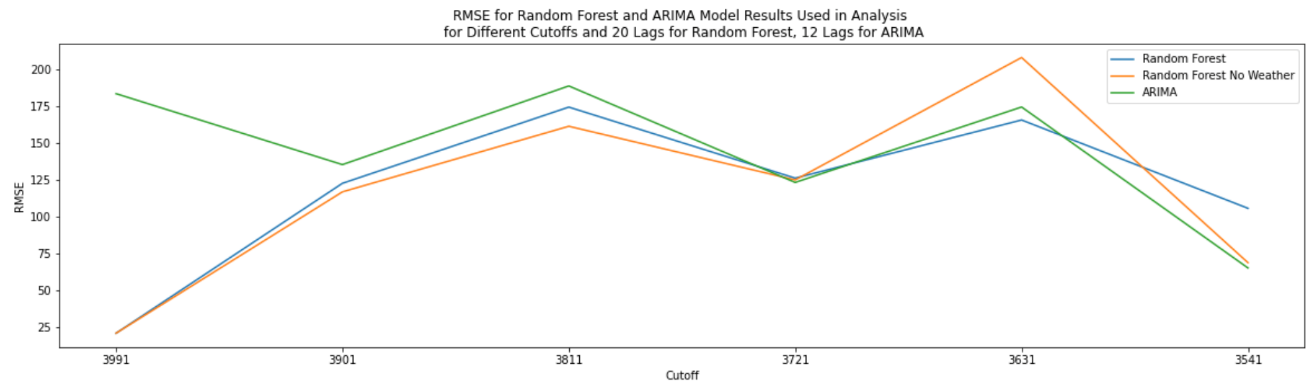
We added information pertaining to weather to our dataset and observed how the predictions for Random Forest changed. The weather information considered included daily average temperature as well as hourly precipitation. The distributions of precipitation and temperature are given below (figure 13).

*Fig 13: Daily temperature and hourly precipitation information for 2017. Precipitation shows bimodal nature.*

Both distributions are very much left skewed. The distribution for hourly precipitation in particular shows bimodality. Using a linear model to predict a bi-modal feature is a problem but is less of an issue for the ARIMA and Random Forest models.

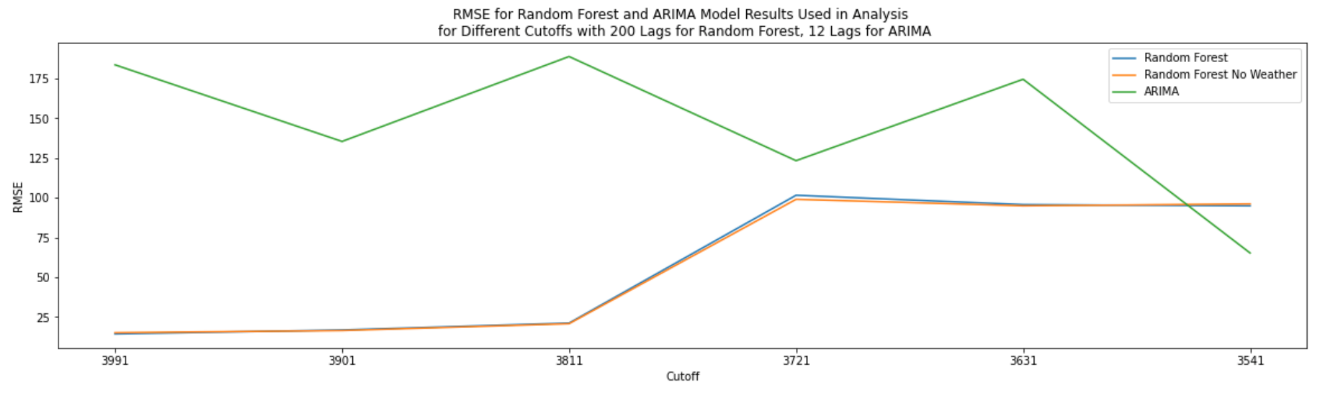
We compared how the model performed against the baseline ARIMA model, as well as a Random Forest model without weather information. As you can see, with 20 lags, there is no discernible difference in the three models.



*Fig 14: RMSE for Random Forest and ARIMA models with and expanding window of training data. Tests were run separately on the Random Forest model, with and without weather related information.*

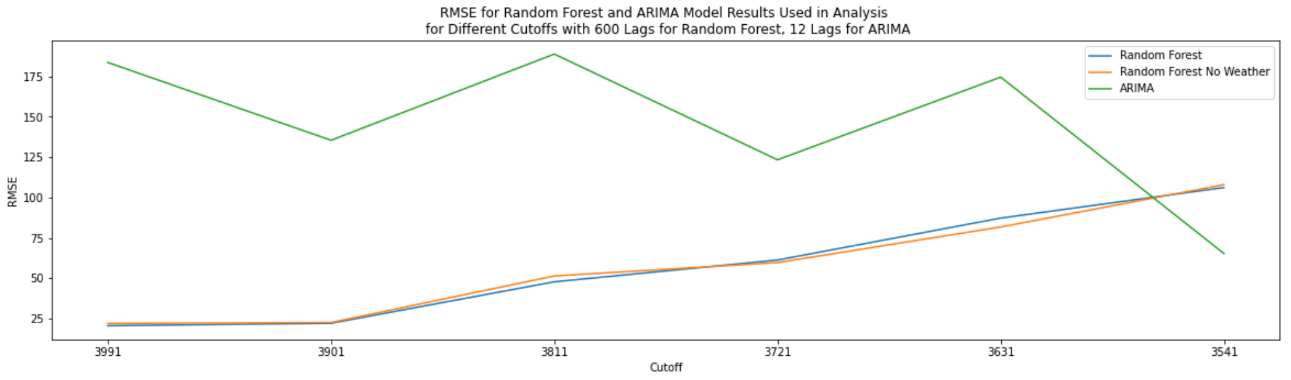
**Increasing the Number of Lag Terms:**

Let’s make large leaps in the lag terms used. Once we increase the number of lag terms to 200, we see a noticeable difference in the ARIMA model and the Random forest model. However, we can’t see that the weather information is providing any useful information for the Random Forest model.

****

*Fig 15: RMSE for Random Forest model fitted with 200 lags, compared with RMSE for ARIMA. Tests were run separately on the Random Forest model, with and without weather related information.*

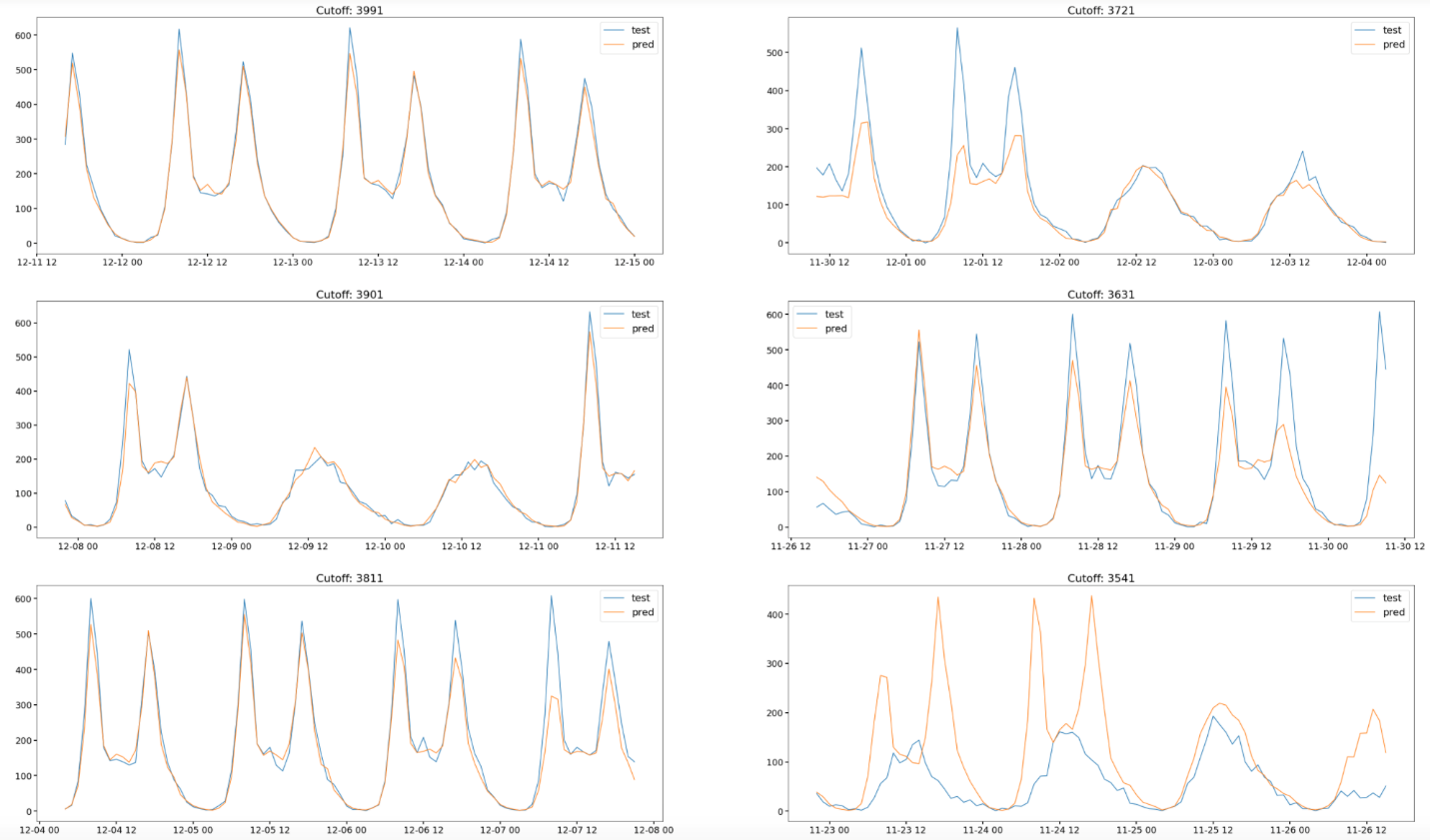
As an additional exercise, let’s increase the number of lags used in Random Forest to 600. At this level, the difference between the Random Forest model and the ARIMA model becomes more pronounced. Once again, we can tell that adding weather related information does not provide anything useful.

****

*Fig 16: RMSE for Random Forest model fitted with 600 lags, compared with that for ARIMA. Tests were run separately on the Random Forest model, with and without weather related information.*

From figure 17 below we can see that with 600 lags, the Random Forest model generally performs very well, except for a specific part of the time-series. This particular portion of the time-series may not be a pattern that is present anywhere else.

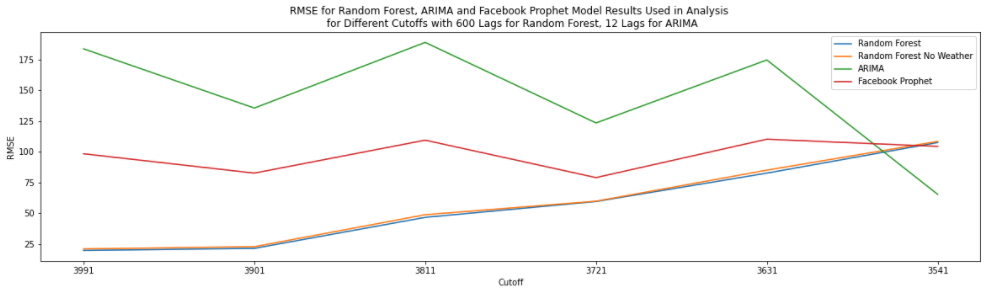
OOS Cross-Validation on the Random Forest Model with 600 Lags (predictions in yellow)

****

*Fig 17: OOS on Random Forest model with 600 lags*

**Facebook Prophet:**

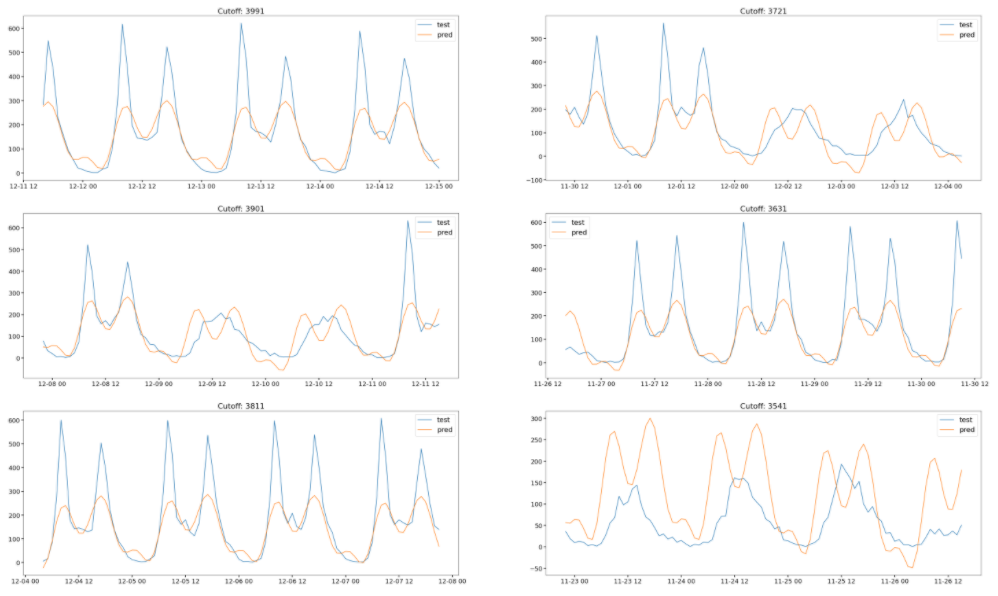
Now we come to running experiments with Facebook Prophet. We went straight to OOS cross-validation using daily and weekly seasonality. The results are on figure 18 below:

****

*Fig 18: RMSE for Facebook Prophet model, compared with that for Random Forest model fitted with 600 lags and that of ARIMA. Tests were run separately on the Random Forest model, with and without weather related information.*

Looking at Facebook Prophet, we can see that it does not fit the data as well as the Random Forest does. However, it does seem to do a better job than the ARIMA model. What is important, though, is that it is matching the cadence of the spikes in rentals. This is impressive for a parametric model.

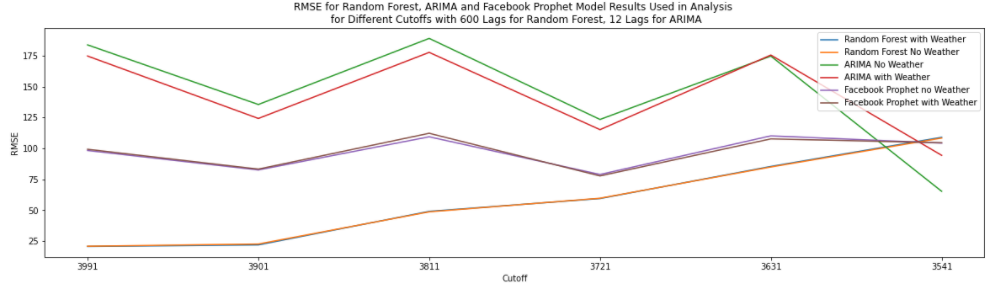
OOS cross-validation on the Facebook Prophet model (predictions in yellow)

****

*Fig 19: OOS cross-validation on the Facebook Prophet model (predictions in yellow). The training sample uses an expanding window. The model fit looks to be better than the ARIMA model.*

**Facebook Prophet and ARIMA models with weather information as additional regressors:**

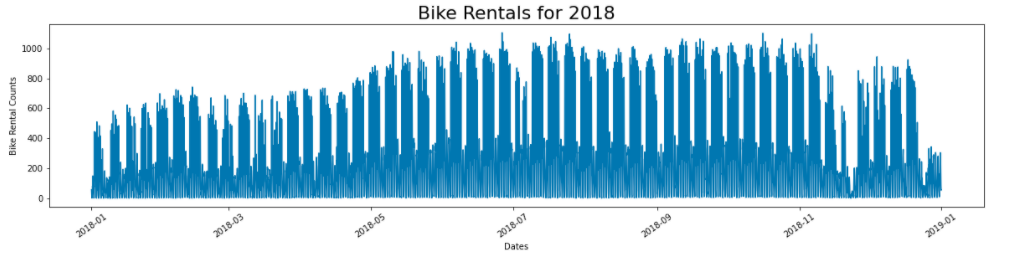
We also attempted to add weather related information as additional regressors to the ARIMA and Facebook Prophet models. Since these are regressors, they are treated differently than weather related information in the Random Forest model. As we can see from the plot below, adding weather related information as regressors has done very little to change the performance of each model.

****

*Fig 20: RMSE for Facebook Prophet model, compared with that for Random Forest model fitted with 600 lags and that of ARIMA. Tests were run separately on all models with and without weather related information.*

**2018 Results:**

The author also experimented with predictions in 2018 to determine how consistent the models were across year, and the data is given below. The data seems to follow a similar trend to that of 2017, with some irregular cycles toward the end of 2018.

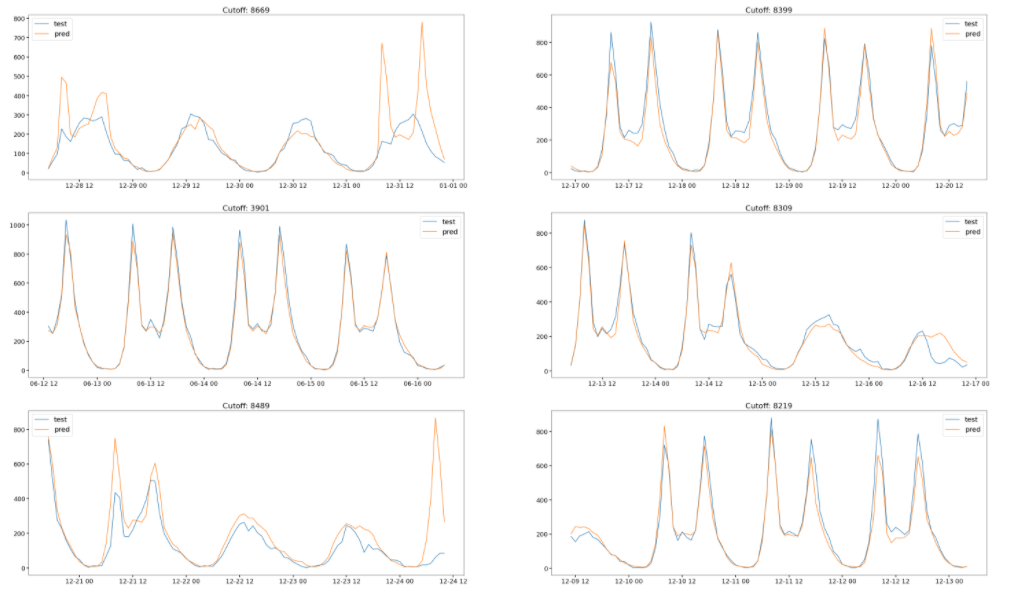
****

*Fig 21: Bike rentals for 2018. The counts of rentals are aggregated for a given hour.*

**2018 Results for the Random Forest model with Weather Information:**

The author persisted with the best model which used 600 lags, 40 estimators and a minimum sample split of 2. We added weather information, the same way he did for the 2017 dataset. The results are below and show an even better fit than for the 2017 data.

OOS Cross-validation on 2018 data using the Random Forest Model (predictions in yellow)

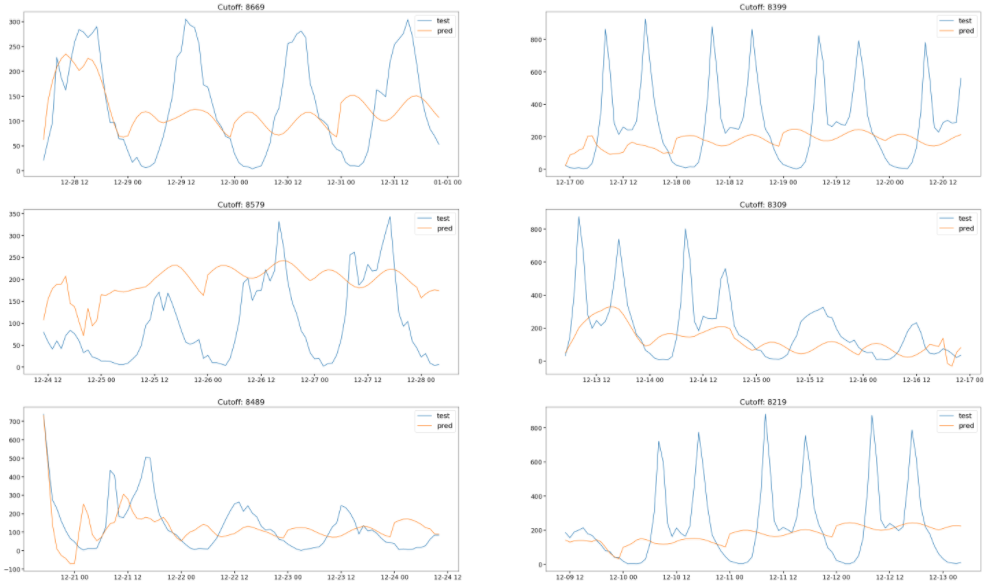
****

*Fig 22: OOS Cross-validation on 2018 data using the Random Forest Model (predictions in yellow). The training samples use an expanding window. The Random Forest model uses 600 lags.*

**2018 results for the ARIMA model (12 lags):**

The author did not conduct a grid-search using Auto ARIMA to determine the best parameters (for p, d and q), given that the dataset for 2018 looks stationary and shows the same characteristics as the data for 2017. Therefore, the author used 12 autoregressive lags again, no differencing (d=0) and no moving average errors (q=0).

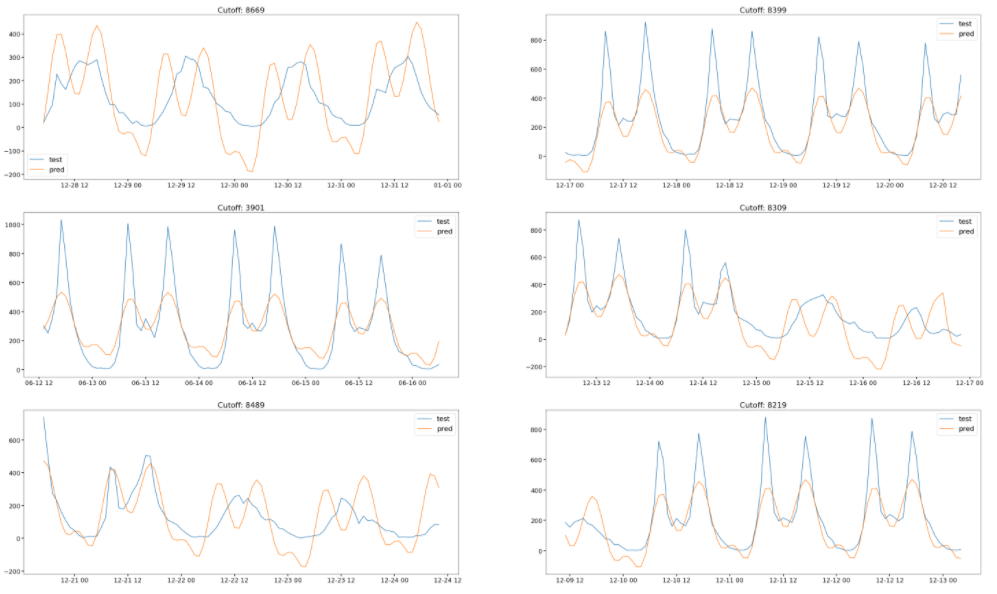
OOS cross-validation on 2018 data using the ARIMA model (predictions in yellow

****

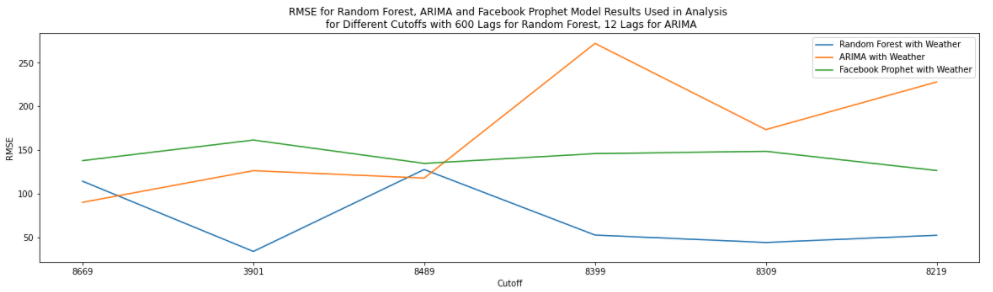
*Fig 23: OOS cross-validation on 2018 data using the ARIMA model (predictions in yellow). The training samples use an expanding window.*

**2018 results for Facebook Prophet:**

Once again, visually, the fit from the Facebook Prophet model seems to be a lot better than for the ARIMA model. It still underperforms in comparison with the Random Forest model.

****

*Fig 24: OOS cross-validation on 2018 data using the Facebook model. The training samples use an expanding window.*

****

**Other Methods Not Used:**

Taylor and Latham (2017) point out that exponential smoothing models on Facebook event time-series data were not very successful, due to the model’s inability to capture seasonalities which were longer than weekly. Therefore, the author has skipped this family of models entirely. The other family of models not used are the PEWMA models. Given the poor performance of the Poisson regression model along with the lack of stable models in the public domain, we decided to forego these models as well.

**Conclusion**

The multivariate Random Forest model turns to have the best predictive power based on our OOS cross-validation. The author found that a higher number of lag terms worked well for the Random Forest model, in spite of a bloated featurespace and the dangers of overfitting. The Facebook Prophet model, which uses Fourier series and trend detection, performs better than the ARIMA autoregressive model. It also has the added benefit of a faster fit time. We also discovered that adding weather related information as additional features or as additional regressors did not add value to the modeling process.

**References**

Moksony, F. and Hegedus, R., (2014). *The Use of Poisson Regression in The Sociological Study of Suicide* Corvinus. Journal of Sociology and Social Policy Vol.5 2, 97–114. Retrieved from:

<https://www.researchgate.net/publication/266140268_The_use_of_Poisson_regression_in_the_sociological_study_of_suicide>

Breiman, L., (2001). *Random Forests*. Statistics Department University of California Berkeley.

Retrieved from: <https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf>

Hastie, T., Tibshirani R. and Friedman, J., (2009). *Elements of Statistical Learning Ed. 2*, p592-593, Springer.

Kumar, M. and Anand, M., (2014). *An Application of Time Series Arima Forecasting Model for Predicting Sugarcane Production in India*. Studies in Business and Economics. 9. 81 - 94.

Retrieved from: <https://www.researchgate.net/publication/263505554_An_Application_Of_Time_Series_Arima_Forecasting_Model_For_Predicting_Sugarcane_Production_In_India>

Garcia-Marti I., Zurita-Milla R. and Swart A., (2019). *Modelling tick bite risk by combining random forests and count data regression models*. PLOS ONE 14(12). Retrieved from: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0216511>

Mathlouthi, W., (2015). *Trees and Random Forests for Nonhomogeneous Count Data Processes.* Retrieved from: <https://biblos.hec.ca/biblio/theses/2015NO21.PDF>

Brandt, P. and Williams, J., (1998). *Dynamic Modeling for Persistent Count Time-Series.* Indiana University. Retrieved from: <https://personal.utdallas.edu/~pbrandt/pests/pewma.pdf>

Hyndman, R., & Khandakar, Y. (2008). Automatic Time Series Forecasting: The forecast Package for R. *Journal of Statistical Software, 27*(3), 1-22. doi: http://dx.doi.org/10.18637/jss.v027.i03

Christoph Bergmeir, Rob J Hyndman, Bonsoo Koo (2018) *Computational Statistics and Data Analysis*, 120, 70-83

Taylor SJ, Letham B. 2017. Forecasting at scale. PeerJ Preprints 5:e3190v2 https://doi.org/10.7287/peerj.preprints.3190v2