Final Course Project: Forecasting Bike Demand

16.05.2021

by Tom K. Walter

1. Dataset and Goal

For my project, I have chosen the Capital Bikesharing Dataset, which is hosted by the UCI Machine Learning Repository, and used in Fanaee-T's and Gama's paper *Event labeling combining ensemble detectors and background knowledge*.[1] Like many bike sharing companies, Capital Bikesharing is a bike-rentals company where the whole process from membership, rental and return is facilitated through the internet rather than a physical storefront. Through this system, users are enabled to rent a bike from any available pick-up location and return it at any other. The research goal associated with the dataset is to accurately forecast the demand for bikes, i.e. how many bikes will be rented at any given point in the future. Underestimating the number of bikes will cause the company to miss out on revenue or even lose members in the long-term, while overestimating the amount may also cause financial loss in terms of tied up capital and maintenance costs. Therefore, I plan to forecast the demand for bicycles at three levels of granularity, namely the daily, weekly, and monthly demand.

The Capital Bikesharing Dataset contains 17,379 rows corresponding to the hourly observation of 16 attributes over the period of two full years (2011 to 2012). The target variable CNT lists the total number of rental bikes used per hour. This target is the sum of two other variables REGISTERED and CASUAL, which list the number of bikes rented by registered and casual users respectively. The remaining attributes list various information about the time or the weather. For more details on these variables see Table 1.

Table 1

Name	Description	Type
INSTANT	record index	(useless for ML)
DTEDAY	date in YYYY-MM-DD format	datetime
SEASON	season (1:winter, 2:spring, 3:summer, 4:fall)	categorical
YR	year (0: 2011, 1:2012)	categorical
MNTH	month (1 to 12, from January to December)	categorical
HR	hour (0 to 23)	categorical
HOLIDAY	holiday: weather day is national holiday or not	categorical
WEEKDAY	day of the week (0 to 6, from Monday to Sunday)	categorical
WORKINGDAY	workingday, if day is neither weekend nor holiday is 1, otherwise is 0	categorical
WEATHERSIT	weather situation: 1: clear, few clouds, partly cloudy, partly cloudy 2: mist + cloudy, mist + broken clouds, mist + few clouds, mist	ordinal

[1] For the paper, see Fanaee-T, Hadi, and Joao Gama. "Event labeling combining ensemble detectors and background knowledge." *Progress in Artificial Intelligence* 2, no. 2 (2014): 113-127. For the dataset, see https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset.

	3: light snow, light rain + thunderstorm + scattered clouds, light rain + scattered clouds 4: heavy rain + ice pellets + thunderstorm + mist, snow + fog	
TEMP	normalized temperature in Celsius; derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)	continuous
ATEMP	normalized feeling temperature in Celsius; derived via (t-t_min)/(t_max-t_min), t_min=-16, t_max=+50 (only in hourly scale)	continuous
HUM	normalized humidity; values are divided to 100 (max)	continuous
WINDSPEED	normalized wind speed; values are divided to 67 (max)	continuous
CASUAL	count of casual users	continuous
REGISTERED	count of registered users	continuous
CNT	count of total rental bikes including both casual and registered	target

While the variables about weather and time may hold valuable information for prediction, I will only focus on univariate forecasting of CNT itself (at three levels of granularity). In order to do so, I will model the data using Facebook's Prophet module as well as a simple RNN from the Keras library. However, before modelling, I will address the data cleaning and exploration steps that are particular to time series data given here. These are important to uncover time series components such as trend and seasonality, which will help improve the forecasting accuracy of the models. Furthermore, I will discuss the hyperparameters used for building the models and visualize their results. Finally, I will evaluate and compare their performances.

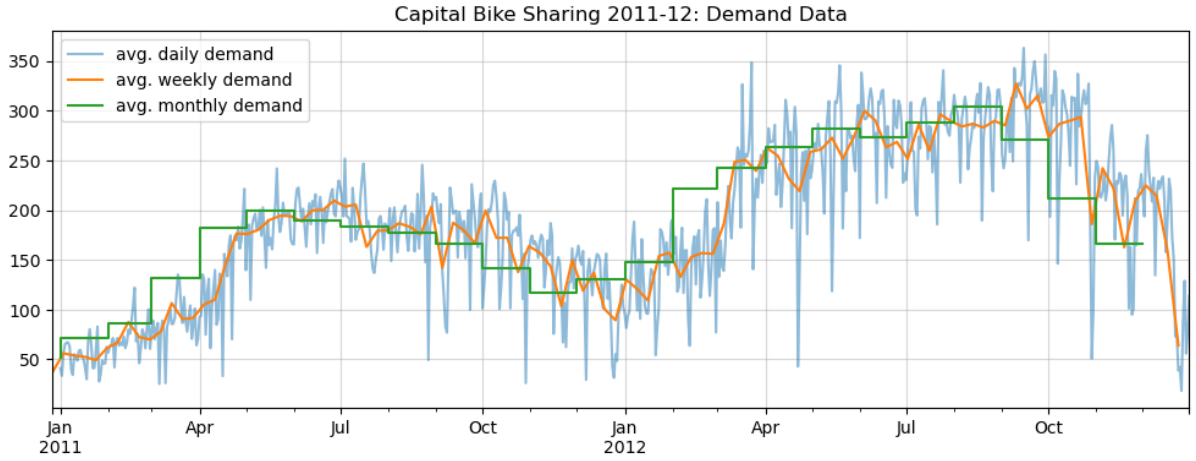
2. Time Series Cleaning

To prepare the dataset for machine learning analysis, it must usually be cleaned, checked for duplicate and missing values, and if appropriate encoded and normalized. But since this is a time series dataset collected at hourly intervals, my first step was to create a datetime index from the DTEDAY and HR variables with Pandas. Then, I used Pandas's *date_range* function to create a new datetime index from the very first and very last value of the previous index at an hourly frequency, i.e. from '2011-01-01 00:00:00' to '2012-12-31 23:00:00'. Because 2012 is a leap year, for 365 days in 2011 plus 366 days in 2012 times 24 hours per day, the result will be equal to 17,544 new index values. The original dataset has 17,379 rows but the new datetime index has 17,544 rows, leading to a discrepancy of 165 missing values. While the missing values for all time-related variables such as YR, MNTH, HR, WEEKDAY, etc. can simply be extracted again from the new datetime index, the missing value for the continuous variables such as HUM, TEMP, CNT, etc. have been filled with Pandas' *interpolate* function. Given that only a tiny fraction (<0.01%) is missing from the dataset, this will not affect the overall distribution and characteristics of the continuous variables. Although I will not use most of these variables for modelling, especially the time-related variables will be important for the time series-specific exploration and crucially the target must not have missing values.

3. Time Series EDA

In this section, I will explore the target variable CNT, resample it for the three levels of granularity, and investigate its time series components such as trend and seasonality. My first step is to resample CNT at a daily, weekly, and monthly level and to aggregate by the mean.

Figure 1:



The average daily demand, shown in blue in Figure 1, exhibits many strong spikes indicating a non-constant variance and very strong autocorrelation (even after 30 lags). Shown in orange in Figure 1 is the average weekly demand, which already seems to smooth out the spikes from the daily demand but still follows it closely. The weekly demand has no significant autocorrelation after lags=3 and no partial autocorrelation after lag=2. Figure 1's green step-line is the average monthly demand and has the fewest observations (i.e. 24) but also the least autocorrelation. The weekly and monthly demand line seem to indicate that the time series is composed of an upward trend and an annual seasonality in the form of an additive model.

In order to investigate these components further and discover additional types of seasonality, I have split the CNT series by the year and then compared the daily, weekly, and monthly level again. Figure 2 shows the average demand by hour of the day but split by 2011 and 2012. Both plots reveal another seasonal pattern that corresponds to rush hour traffic with high demands in the morning and evening when most people commute to and from work. Moreover, the trend component is confirmed as in 2012 the average demand during daytime hours has increased compared to 2011. From a business perspective, it is also interesting to note that between 10:00 am and 07:00 pm the average number of casual users is higher.

Figure 2:

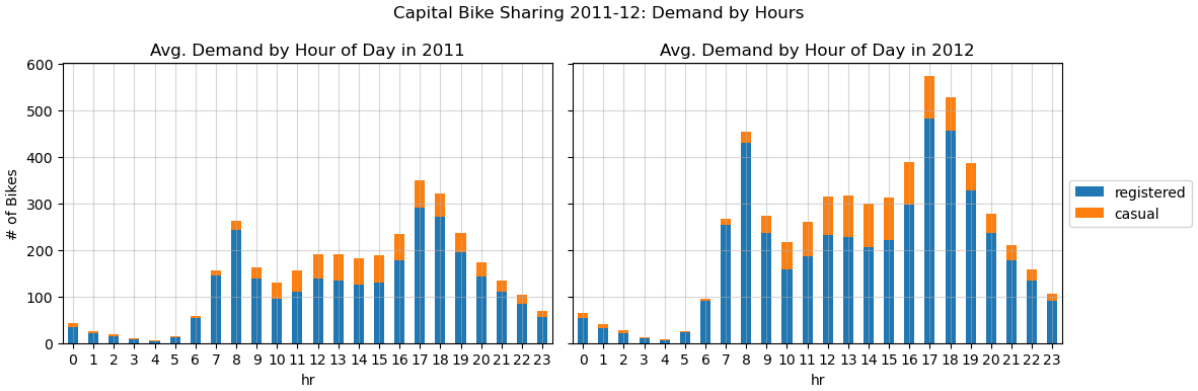
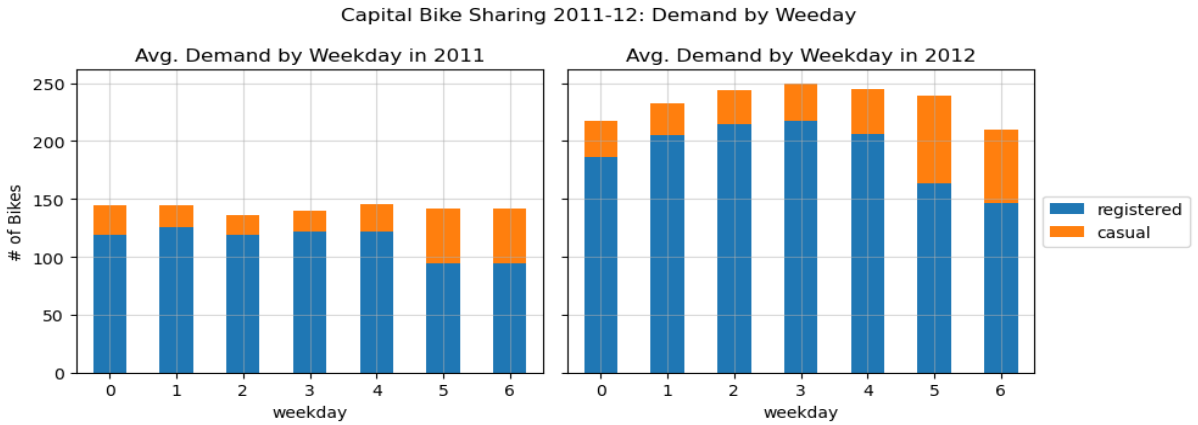


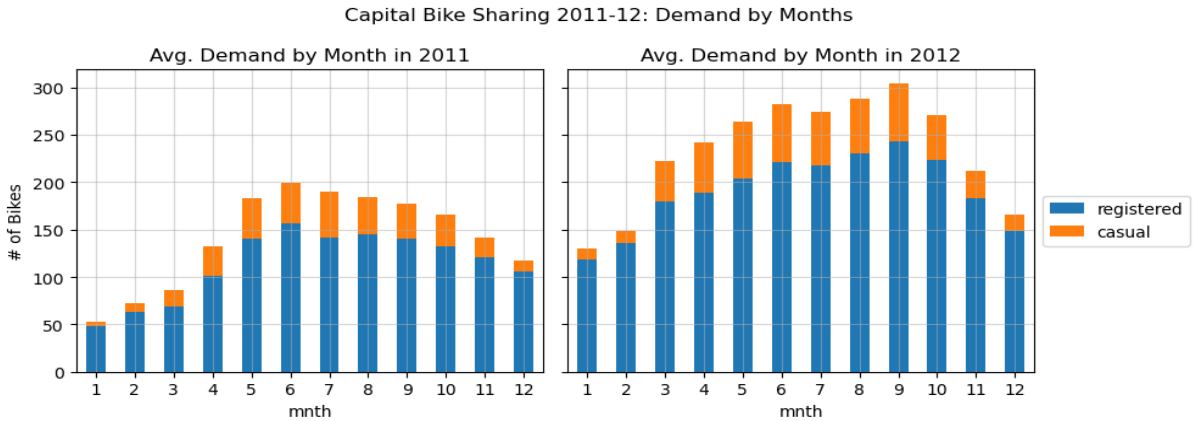
Figure 3 shows the average demand by weekday, split by 2011 and 2012. While the upward trend component is exhibited as well from 2011 to 2012, there is no clear weekly seasonal component in both years. On average, there are twice as many casual users on the weekend.

Figure 3:



The average monthly demand by registered and casual users is shown in Figure 4. It exhibits a clear seasonal pattern that there are more users during spring, summer, and fall as well as that there is a clear upward trend from 2011 to 2012.

Figure 4:



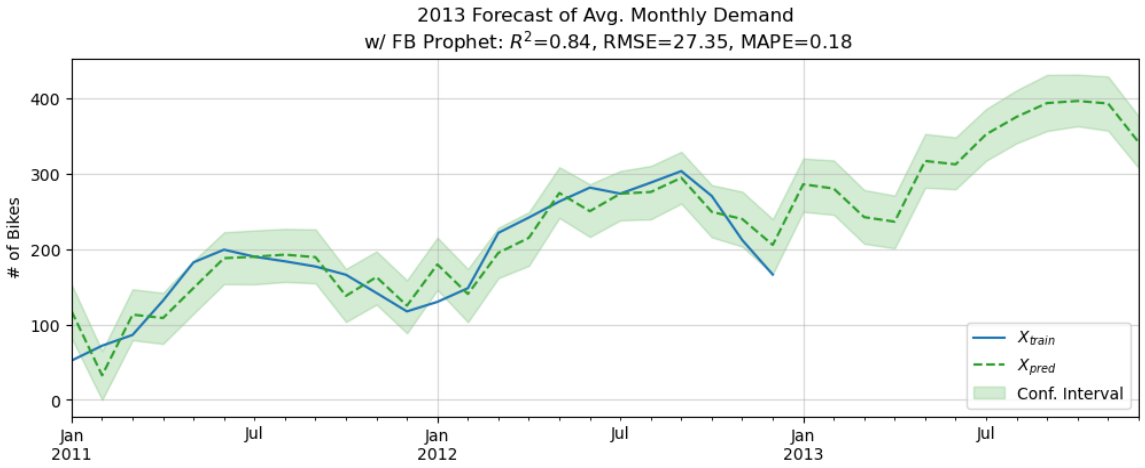
For autocorrelation and partial autocorrelation plots at each level, confer to the appendix.

4. Facebook Prophet Models

Facebook's open-source time series forecasting module called "Prophet" will be used here for the non-deep-learning models because it was developed to forecast business data with many seasonal components and without preprocessing.[2] Particularly, Prophet works with the statistical assumption of an underlying additive model, which fits well with my observation that the bike demand is probably additive. Although Prophet will try to find different levels of seasonality automatically, sometimes it can attain better results when the seasonality is set manually. Therefore, I have optimized the seasonality hyperparameter with knowledge of the time series exploration from before. Since forecasting is easy with Prophet, I have also added the predictions for one year into the future, i.e. until the end of 2013. For all Prophet and Keras models, I will use the R-squared, root mean squared error (RMSE), and mean absolute percentage error (MAPE) as evaluation metrics (see Table 2 for full comparison).

Figure 5 shows the forecast of the average monthly demand with training data in blue and the function fit by Prophet in green. Setting the seasonal hyperparameter either to yearly or monthly, led to very similar results (although the setting yearly may cause overfitting in the future). This model has a very high R-squared as well as a low RMSE and MAPE, which means its forecast for the future will likely be very accurate. The monthly model clearly picks up on the seasonality of higher demand in middle and lower demand at the beginning and end of the year. Likewise, both the upward trend and seasonality are forecast into 2013, but the model anticipates the highest demand at the end of 2013, during the colder months which is unlikely given our knowledge about past data.

Figure 5:



Next, Figure 6 shows the average weekly demand with training data in blue and the model fit by Prophet in green. According to the error metrics, this weekly model fits even better than the monthly one. However, in order to achieve that I had to set seasonal hyperparameter for yearly as well as disable monthly and weekly seasonality (although this

[2] For Facebook Prophet, see https://facebook.github.io/prophet/.

seems counterintuitive). Compared to Figure 4, the model in Figure 6 also seems to fit better seasonality patterns better even if some spikes of the training data are outside the confidence interval. Similarly, trend and seasonality are projected into 2013 better than the previous one, as here the highest demand is expected in summer and not in winter.

Figure 6:

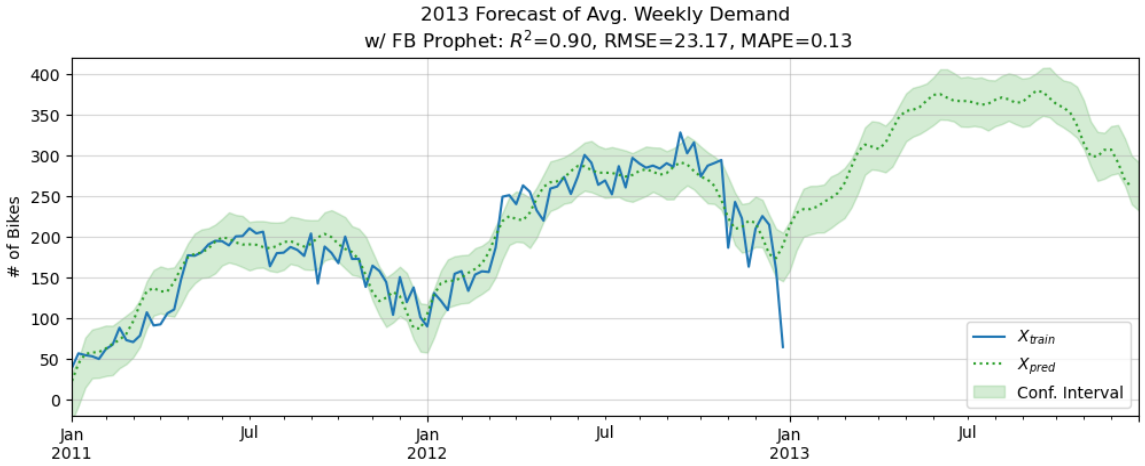
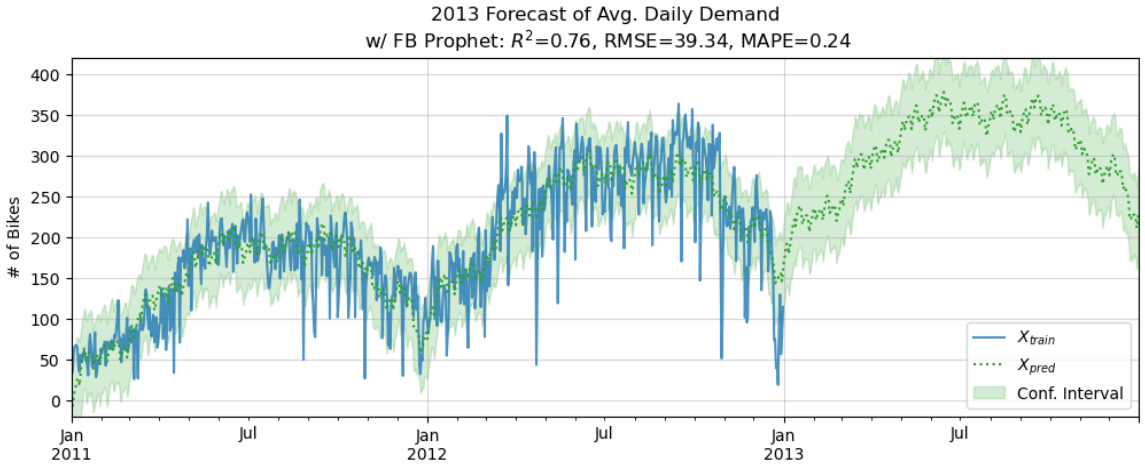


Figure 7 shows the Prophet model for the average daily demand in green and the training data in blue. This level of granularity had the highest autocorrelation and variance. Accordingly it was the most difficult for Prophet to find the underlying function and has the worst performance scores among all Prophet models. I have set the seasonality to monthly with 30.5 days per month. Nonetheless, the forecast for the average daily demand does not appear too inaccurate in itself as the trend and seasonality are still found by the model. Having another look at this level of granularity also serves as a reminder that no model so far has the ability to predict days with extremely low demand, which are not uncommon with this data.

Figure 7:



Being able to predict such a period of extremely low demand may be a key element for Capital Bikesharing's business to schedule maintenance. Deep learning models may be able to find such complex patterns and deal with associated autocorrelation.

5. Keras RNN Models

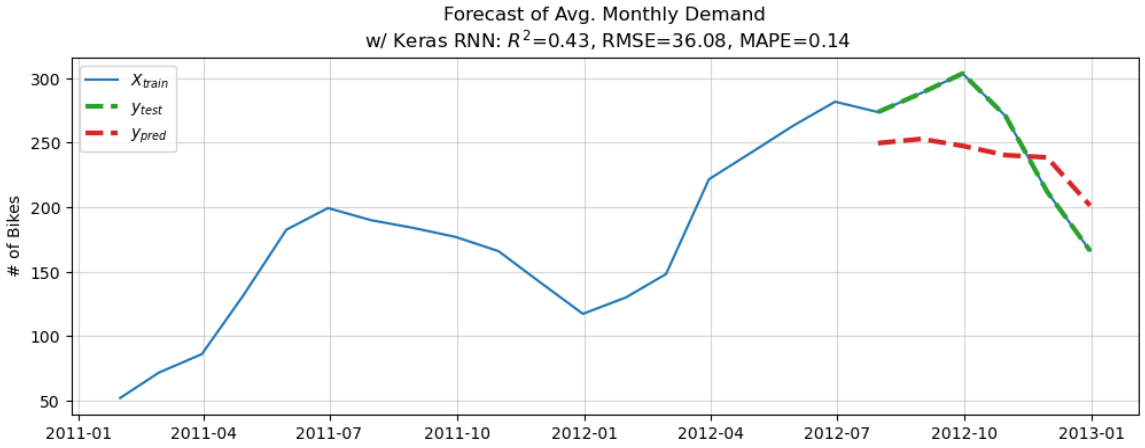
All deep learning models have been built with the Keras module using the Tensorflow backend and all DL models are recurrent neural networks (RNNs).[3] Moreover, they all share a same architecture:

- two layers of RNN cells,
- two layers of dense neurons,
- optimizer = adam,
- loss = MSE.

While the Prophet model worked with the demand data on the original scale, the time series is min-max-scaled for the RNNs. Lastly, the demand data has been arranged into sequential samples expected by the RNNs and split into 75% training and 25% testing data.

Figure 8 shows the first RNN model which has been trained on the average monthly demand data. Although the RMSE and MAPE are indicative of a good fit, the R-squared is bad. This is because neural networks need large amounts of training data in order to accurately fit the weights of their neurons but the monthly demand data only has 24 observations. Thus, the RNN model is unable to identify the underlying trend or seasonality even after 200 training epochs.

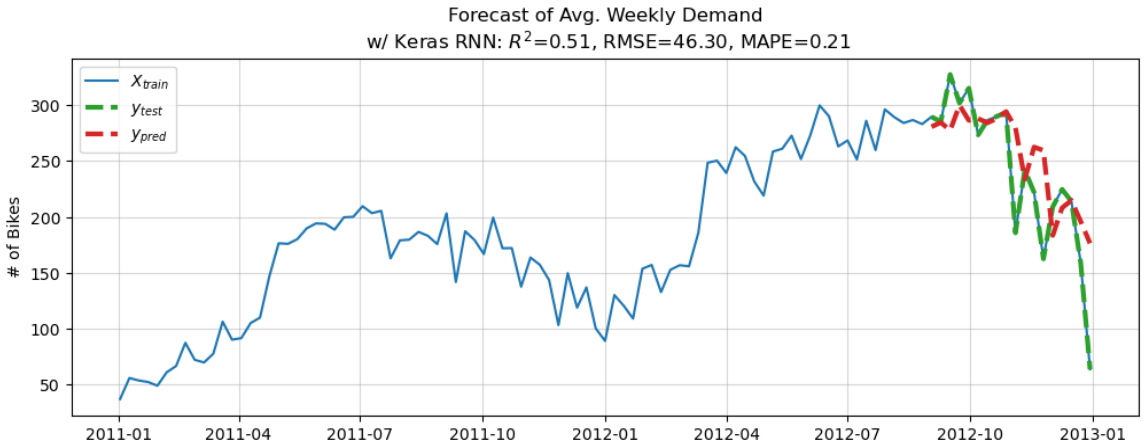
Figure 8:



The second RNN model has been trained on the average weekly demand data. Although visually this model seems to predict the testing data much better as shown in Figure 9, even down to some monthly fluctuations, this is not reflected in the performance scores. RMSE and MAPE have become worse and R-squared has improved only slightly. Again the weekly demand dataset was small with only 105 samples in total. Therefore, I would argue that the metrics may be misleading on NNs when the training dataset is small as the Figure 9 clearly confirms that the model is able to find a complex pattern after 200 epochs.

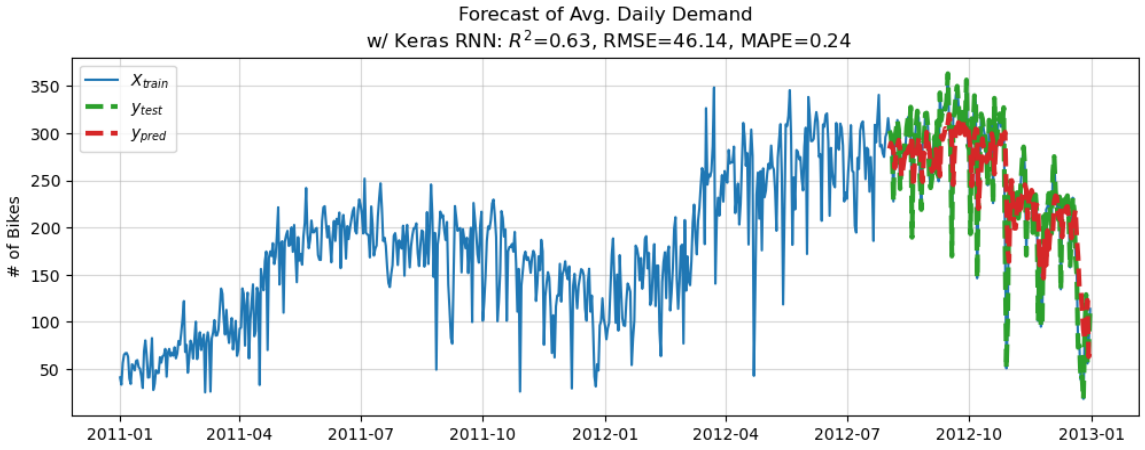
[3] For the Keras API, see https://keras.io/api/.

Figure 9:



The last RNN probably has the best chance of identifying complex patterns of seasonality and trend as the average daily demand data has 731 samples in total. Yet again, there is a discrepancy between the visual model and its quantitative score in Figure 10. In fact, visually, the model seems to fit the data the best and is able to identify days of extremely low demand, which was not possible with Prophet models. However, neither R-squared nor the RMSE are as good as the ones for the Prophet model. This highlights the importance of visualizing the results of time series forecasting.

Figure 10:



6. Model Evaluation and Conclusion

Looking at the evaluation table below and the figures above, there are two pertinent conclusions I draw from forecasting the rental bike demand for Capital Bikesharing. The first conclusion is that the Prophet models are more accurate when it comes to forecasting mid-term to long-term demand (see Table 2). In fact, the Prophet models identified the upward trend as well as monthly and annual seasonality patterns. However, deep learning models will be better for forecasting a short time horizon as they can predict spikes in the daily demand data. The second conclusion is that this distinction may not have become obvious without the visualization of Prophet and Keras models. By the metrics alone, it would have seemed that the Prophet models are more accurate on every level of granularity.

Table 2:

	Prophet			**RNN**		
Level	*Monthly*	*Weekly*	*Daily*	*Monthly*	*Weekly*	*Daily*
R^2	0.8445	0.9012	0.7602	0.4312	0.5117	0.6262
RMSE	27.3485	23.1713	39.3394	36.0818	46.2957	46.1434
MAPE	0.182	0.1274	0.2415	0.1404	0.215	0.2407

For future analysis of the Capital Bikesharing Dataset, it will be useful to consider more feature variables when predicting the demand. For Prophet, especially the variable HOLIDAY would be interesting to test. Whereas for both machine and deep learning models, the use of further features about the weather will probably yield even more accurate models. Depending on the level of granularity that Capital Bikesharing wants to use for its demand forecasting, RNNs and perhaps even Long Short Term Memory models may be trained on hourly data for short-term forecasting, and Prophet models for long-term forecasting.

7. Appendix

Figure 11:

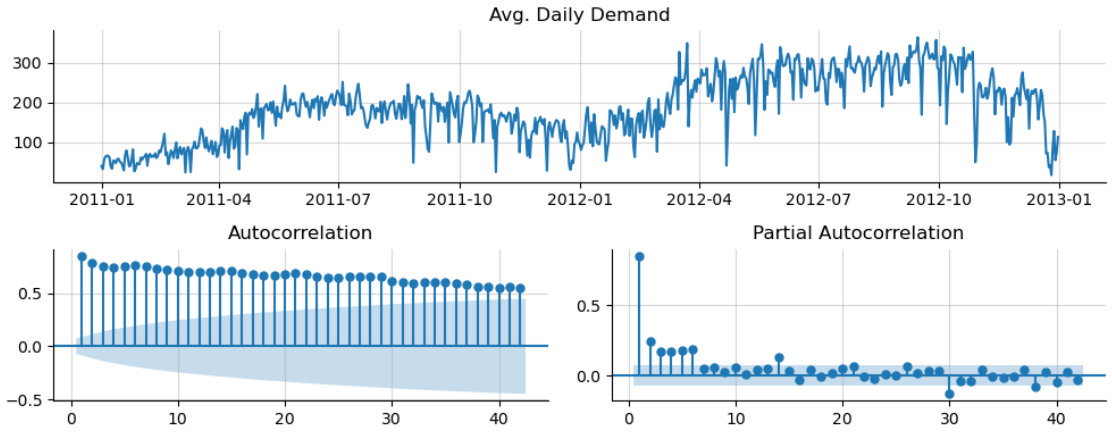


Figure 12:

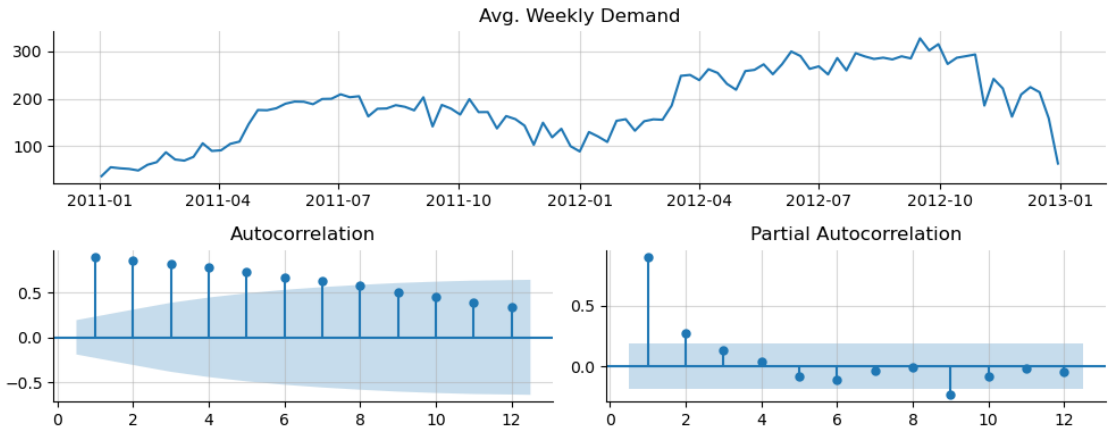


Figure 13: