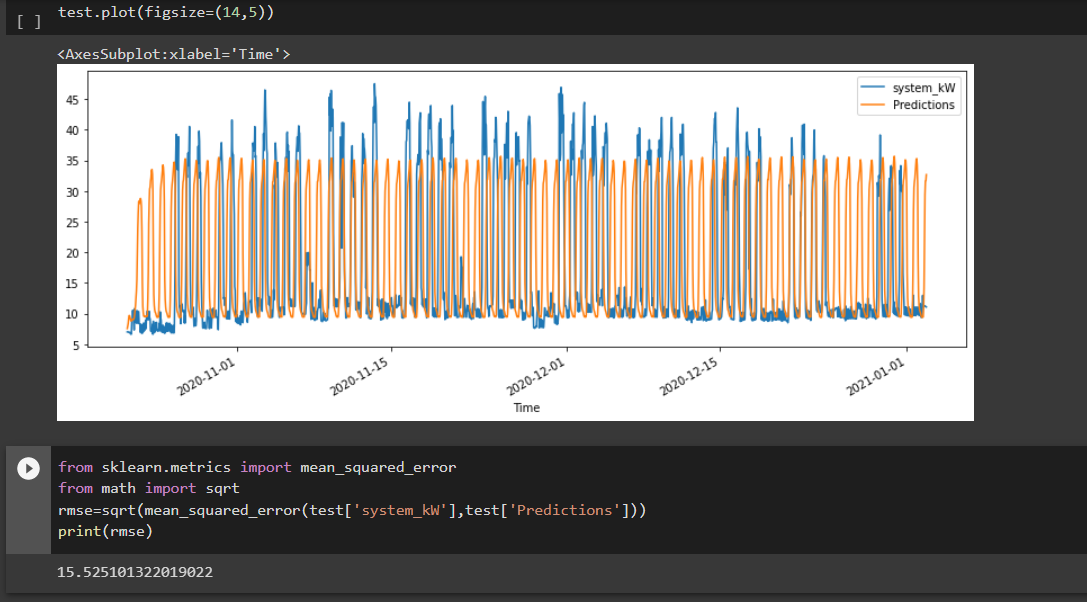
**First Iteration: Using LSTM (Univeriate) (**[**script**](https://colab.research.google.com/drive/1GNeCcNah9DdBAy4DZpyVbpLV7MU_V2_t#scrollTo=upCctYtUVmS9)**)**  
LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) architecture used in machine learning. It was designed to address the problem of vanishing gradients in traditional RNNs, where the gradient used to update the model parameters becomes increasingly small as it is backpropagated through time, making it difficult to train models that depend on long-term dependencies.

LSTM networks use a memory cell, which can store information over a longer period of time, and three gates that control the flow of information into and out of the cell: the input gate, the forget gate, and the output gate. The input gate controls how much new information is allowed into the cell, the forget gate controls how much information is discarded from the cell, and the output gate controls how much information is allowed to leave the cell.

***Result***



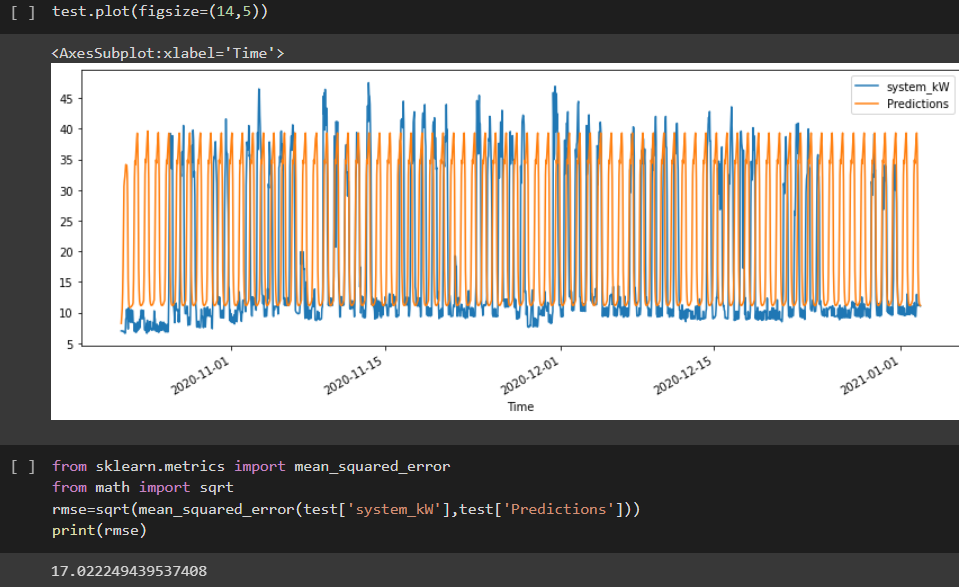
The model gives an RMSE score of 15.52  
  
Note the idea behind RMSE is that the lower the value the better the mode  
This model might be used to model only working periods

*RMSE (Root Mean Square Error) is a commonly used evaluation metric in machine learning problems. It measures the average deviation of the predictions from the actual values. Mathematically, RMSE is defined as the square root of the mean of the squared differences between the predicted and actual values:*

*RMSE = sqrt(mean((predicted - actual)^2))*

*where predicted is the predicted value, actual is the actual value, and mean is the arithmetic mean. RMSE is expressed in the same units as the target variable, and lower values indicate better performance of the model.*

**Iteration 2: Using LSTM (Univeriate) (**[**script**](https://colab.research.google.com/drive/1xVBLxe151nuibFVwS-L8qAJDEr3RUExZ)**)**



The model gives an RMSE score of 17.02, meaning this is a poorer model compared to the above. And from the graphs we still see a lot of off predictions

**Iteration 3: Facebook Prophet (**[**script**](https://colab.research.google.com/drive/1z92yc9l_uLAqSslkMf5KUkvV7cNaYWdg#scrollTo=oFa7S3GojL-l)**)**

Facebook Prophet is a time series forecasting tool developed by Facebook's Core Data Science team. It is an open-source library for Python and R programming languages that is designed to make it easier for analysts and data scientists to build accurate time series forecasting models.

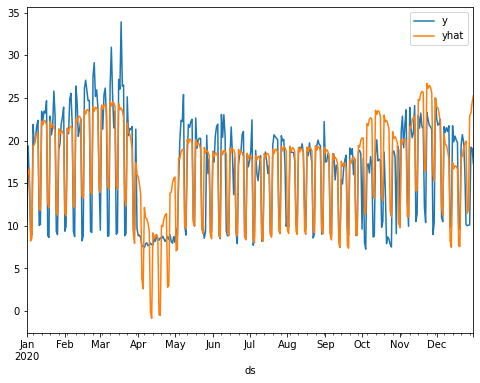
Prophet is built on the decomposable time series model, which is a generalized additive model that consists of three main components: trend, seasonality, and holidays. The model also incorporates additional regressors, such as weather data, economic indicators, or other relevant factors that can help improve the forecast accuracy.

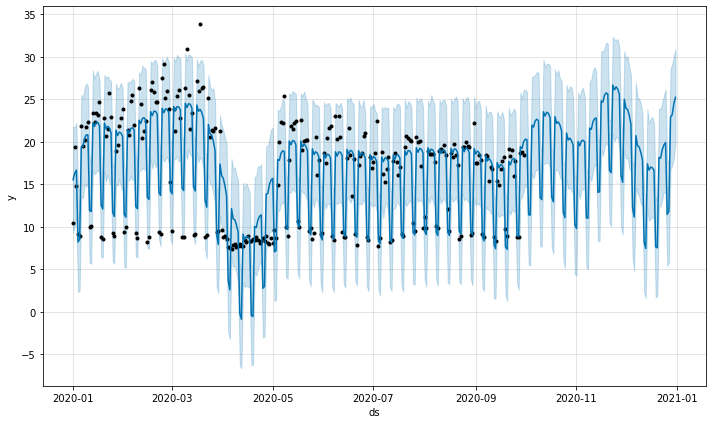
Prophet has several benefits, including:

* User-friendly interface: Prophet has a simple and intuitive API that is easy to use, even for users without a strong background in time series modeling.
* Flexibility: The library allows users to customize the model's components, such as the seasonality and holiday effects.
* Scalability: Prophet can handle large datasets and is designed to work with multiple cores, making it faster than some other time series forecasting tools.
* Automatic outlier detection: Prophet has built-in outlier detection that can identify and adjust for anomalies in the data.
* Probabilistic forecasting: Prophet provides uncertainty intervals for the forecast, allowing users to assess the risk associated with each prediction.
* Prophet has been used in a variety of applications, including demand forecasting, sales forecasting, and financial forecasting.

Note: One could use this when trying to make anomaly detection has this is said to be a good algorithm for that, and this has been noted from the outputs in the scripts

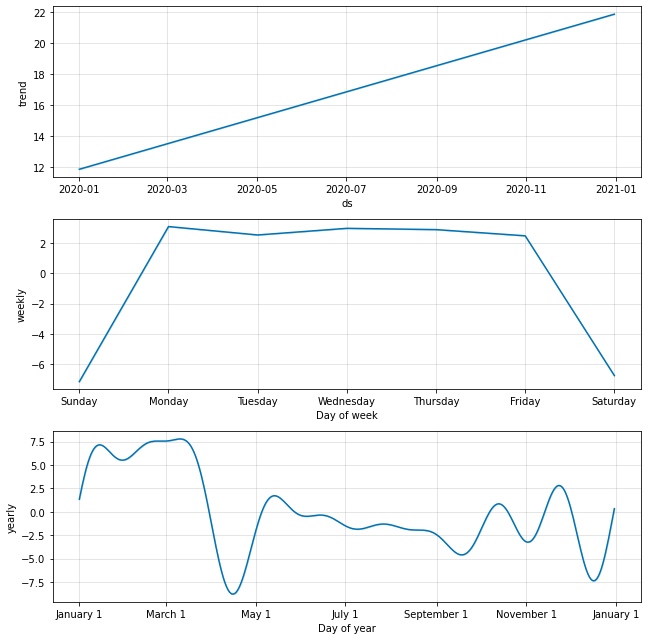
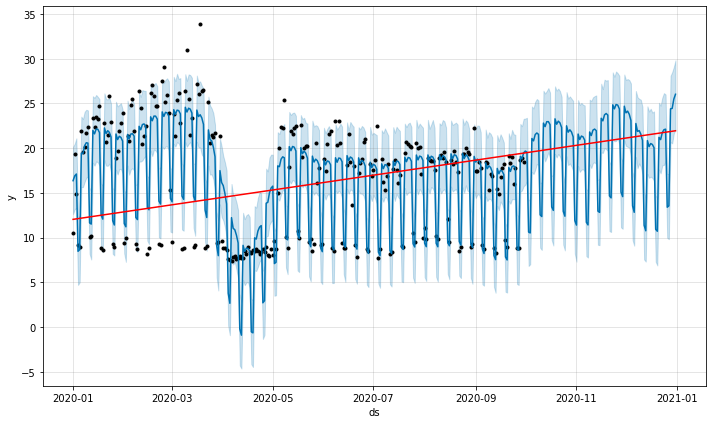
Result of the prediction chart:



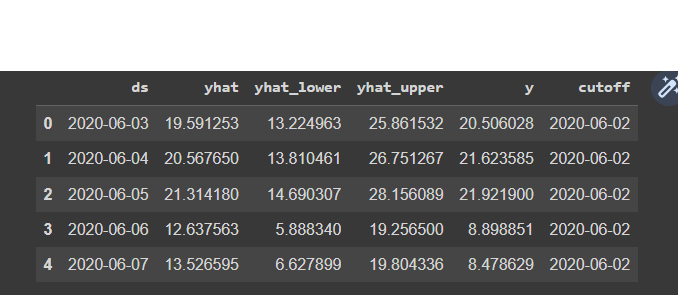


The above chart shows a prediction boundary using a 95% confidence interval. Points outside this boundary are classified as anomaly. Reasons why Facebook prophet can be used in building a anormally detector in this case.

**Iteration 4: Facebook Prophet (Factoring Day) (**[**script**](https://colab.research.google.com/drive/1z92yc9l_uLAqSslkMf5KUkvV7cNaYWdg#scrollTo=oFa7S3GojL-l)**)**



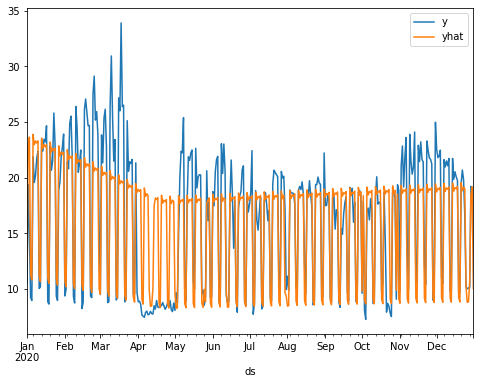
This iteration is modelled to consider what day it is.

***Output of Predicted Vs Actual***  


***Result***

Yielded an RMSE score of approximately 12, which can be seen to be better compared to the previous approaches.

**Iteration 5: Facebook Prophet (Factoring Holidays) (**[**script**](https://colab.research.google.com/drive/1z92yc9l_uLAqSslkMf5KUkvV7cNaYWdg#scrollTo=oFa7S3GojL-l)**)**

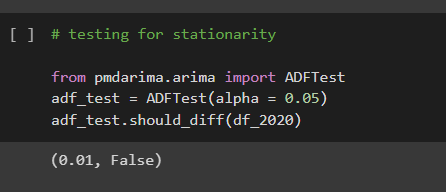


From the plot, its evident that this will have no good accuracy score. But one way to go about using the holiday feature is when we have like 5 years worth of data and indicating the holiday dates in those years, meaning there would be at list a form of repeated holiday.   
  
  
**Iteration 6: Auto ARIMA ML (**[**script**](https://colab.research.google.com/drive/1TpCk8U1UJ-hRT0LFdnqC0psrMVWUmFvp#scrollTo=smHQxnsd7wPB)**)**

AUTO ARIMA (Automated ARIMA) is a statistical model used in time series analysis to automatically select the optimal parameters (p,d,q) for an ARIMA model. ARIMA stands for Autoregressive Integrated Moving Average, which is a type of model used for forecasting time series data. AUTO ARIMA works by searching through a large number of potential parameter combinations to find the combination that produces the best fit for the data. The model considers different types of seasonal patterns, trends, and error structures to automatically select the best parameters. It is a powerful tool for time series analysis and forecasting, and can be used for a wide range of applications such as sales forecasting, stock market prediction, and energy demand forecasting.

For this Model to be valid, a stationarity test has to be conducted   
Stationarity is an important concept in time-series and any time-series data should undergo a stationarity test before proceeding with a model.

We use the ‘Augmented Dickey-Fuller Test’ to check whether the data is stationary or not which is available in the ‘pmdarima’ package.



From the above, we can conclude that the data is non-stationary. Hence, we would need to use the “Integrated (I)” concept, denoted by value ‘d’ in time series to make the data stationary while building the Auto ARIMA model.  
  
  
*When a dataset is non-stationary, it means that its statistical properties change over time. In other words, the mean, variance, and/or covariance of the data points are not constant over time. This can make it difficult to accurately model and predict future values, since the patterns observed in the data may not continue into the future. Non-stationarity can be caused by a variety of factors, such as trends, seasonality, and changes in the underlying process generating the data. In time series analysis, it is important to identify and account for non-stationarity in order to build accurate models and make reliable predictions.*

*To account for non-stationarity in order to build accurate models, we can use several techniques, including:*

*Differencing: We can take the difference between consecutive observations to make the series stationary. This can be done once or multiple times until we achieve stationarity.*

*Transformation: We can transform the data by taking the logarithm or square root of the series. This can help stabilize the variance and make the series more stationary.*

*Seasonality Removal: We can remove the seasonal component of the time series, either by seasonal differencing or using a seasonal decomposition technique.*

*Trend Removal: We can remove the trend component of the time series, either by differencing or using a smoothing technique like moving averages.*

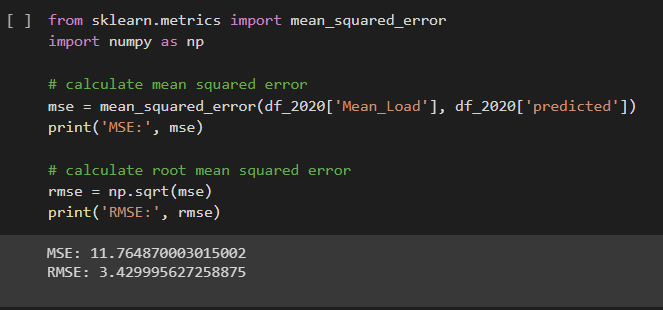
*ARIMA Modeling: We can use ARIMA modeling, which can incorporate differencing, seasonality removal, and trend removal as part of the model. The ARIMA model can be used to forecast future values of the time series.*

*Other techniques: Other techniques like wavelet transforms, singular spectrum analysis, and empirical mode decomposition can also be used to decompose the time series into components and make it more stationary.*

*The choice of technique depends on the characteristics of the time series and the problem at hand. It is important to visually inspect the data and conduct statistical tests to determine if the data is stationary and if the transformations have been successful in achieving stationarity.*

***Result of ARIMA Modellling***

Overall, the model appears to fit the data reasonably well, but there may be some remaining issues with autocorrelation and non-normality in the residuals. Further analysis and modeling may be necessary to improve the accuracy of predictions.

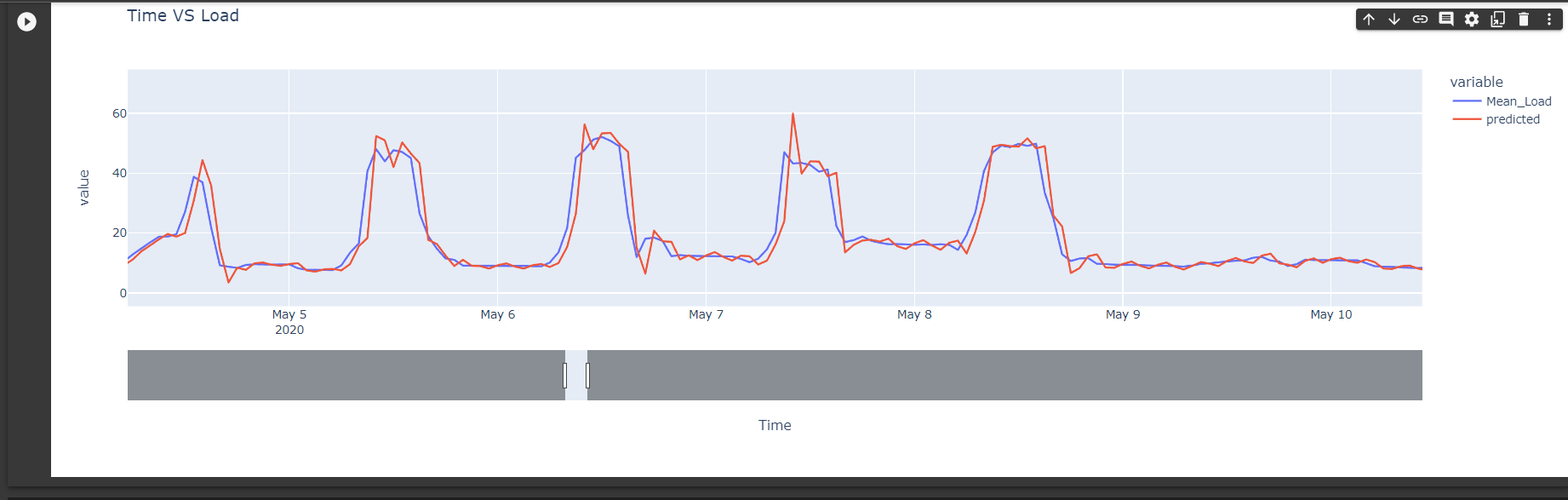


Having an RMSE score of 3. Which is better than the previous Iteration.  
  
Next step would be to PIpleline this model as a machine learning model to make future prediction and see its performance fi there is any form of over fitting and underfitting  
  
Overfitting and underfitting are common problems in machine learning modeling:

*Overfitting occurs when a model learns the training data too well, resulting in poor performance on new, unseen data. The model has essentially memorized the training data and is not able to generalize to new data. Overfitting can occur when a model is too complex or when it is trained on a small amount of data relative to the number of features.*

*Underfitting occurs when a model is not complex enough to capture the patterns in the data, resulting in poor performance on both the training and testing data. This can occur when a model is too simple or when it is not trained for enough epochs.*

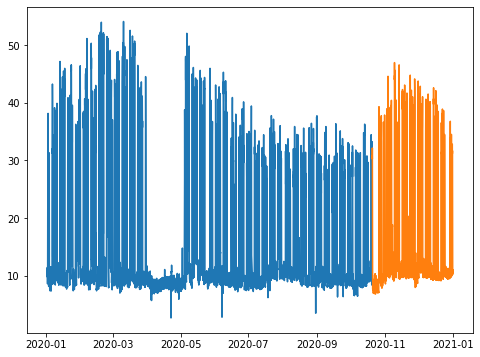
*To avoid overfitting and underfitting, it is important to balance model complexity with the amount of training data available. Regularization techniques, such as L1 and L2 regularization, can also be used to prevent overfitting by adding a penalty term to the loss function. Cross-validation can be used to evaluate model performance on new data and help identify if a model is overfitting or underfitting.*



The plot above shows a well trained model as the actual and predicted seems inline

Current Issue and next ideological steps to improve the model:

* Issues with compute power as the tool used in doing this modeling (colab) is crashes when it reachs a particular RAM limit.
* Pipeline the last iteration (AUTO ARIMA ML) and see its prediction capacity



* The idea behind this plot above is to have the trained data (blue) and use the model to predict for the test data (yellow), then we find the Accuracy performance of the actual test data against the predicted values.
* Build multivariate models that would incurr more data,
  + Energy data of about 3 years
  + Join the temperature data to the energy data and make it a second feature. In previous research we have seen that energy consumption is positively correlated with Temperature
  + Add holiday day, but this time for the 3 years worth of data
* Try other available time series models