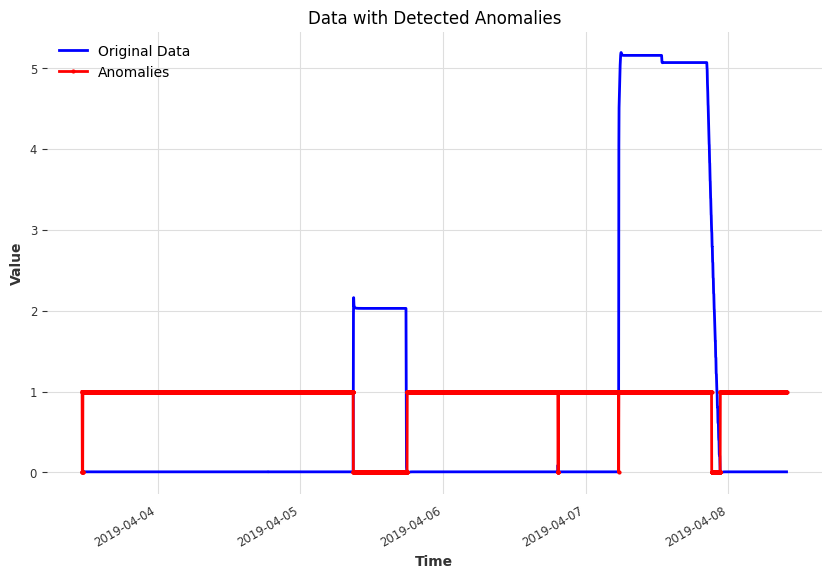
**Journal Report 25**

**03/08/24 & 03/11/24**

I was not here on Friday, but I spent time fine-tuning the threshold detector for a specific variable. It requires a lot of guesswork and checking how it compares to the original data. I found the hardest part to be determining the threshold as I am not sure how much variance is considered anomalous. I spent 1 more hour researching mathematical formulas that could limit the guesswork involved for future variables and automate this process.

Further, I spent the class time researching and analyzing the code for my threshold detector. I was able to implement the threshold detector for multiple variables. However, this threshold detector does not utilize unsupervised machine learning and requires manual setting of the lower and upper thresholds. An example of the threshold detector on one variable is shown below.



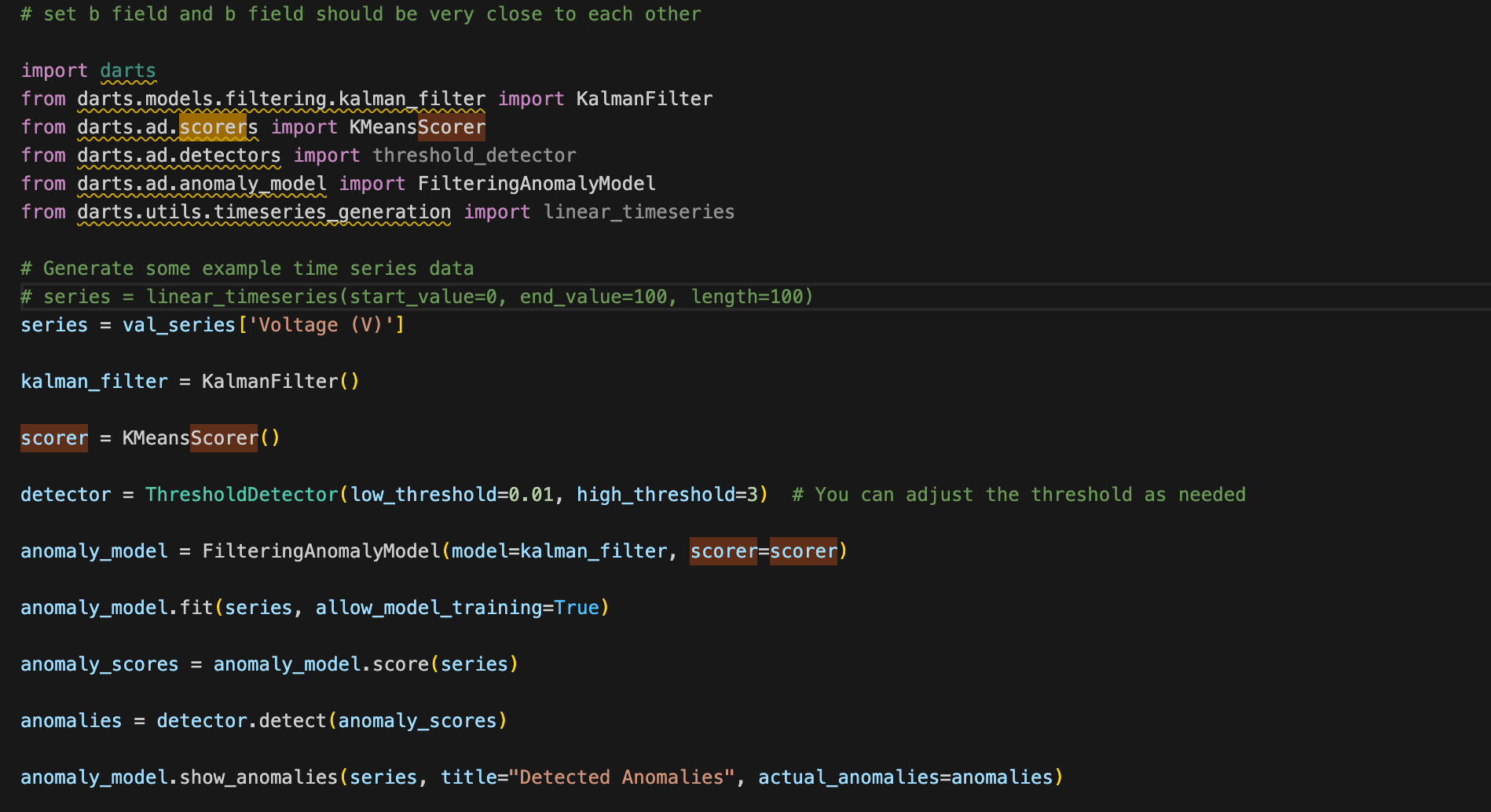
The one problem with this implementation is that it uses the same threshold throughout the time series rather than segmenting the data and adapting over time. This prevents anomaly detection from working over instances where set points and other such variables change past the threshold.

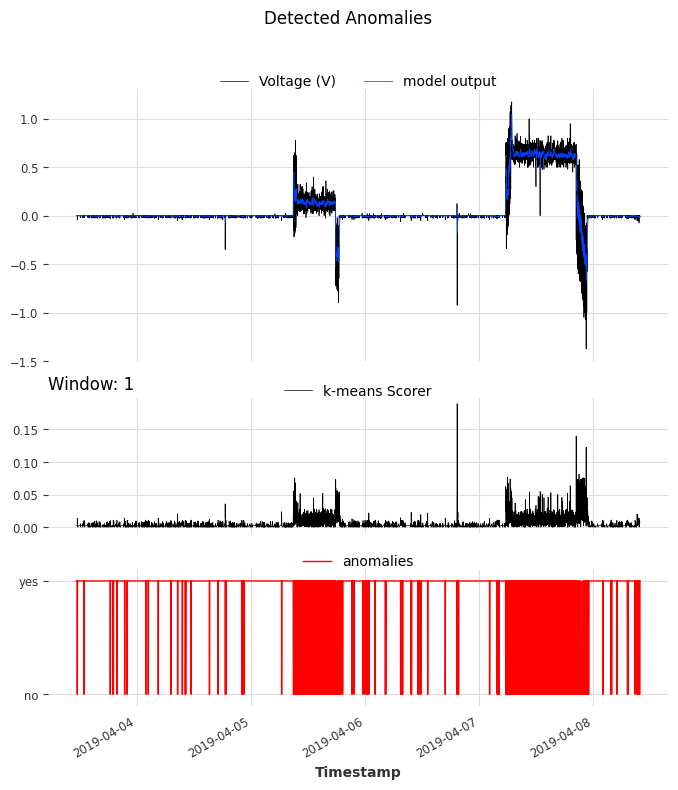
The library for the implementation of the threshold detector was derived from this website:

<https://unit8co.github.io/darts/generated_api/darts.ad.detectors.threshold_detector.html>

**03/13/24**

I spend the first 20 minutes of class researching unsupervised learning methods for threshold detection. I came across something called a filtering anomaly model, which essentially trains a filtering model and then overlays multiple anomaly scorers to determine the variation in comparison to the whole history of the time series. I spent the remainder of the class implementing this model. As shown below, the FilteringAnomalyModel wraps around the Kalman\_filter model and utilizes the K-means algorithm to compute an anomaly score. It then compares how these calculated scores deviate from the model’s predictions (filtered series). As a result, I have now created a pipeline for my anomaly detection: data input, forecast data for a set amount of time, and then utilize the filtering model.





Currently, my main objective is to improve the performance of my filtering model as it is still very sensitive and inaccurate. I am planning to use more scorers and train the model so that it identifies the relationships between different variables to offset the scorers. I am very happy with these results as I now have a minimum viable product, but there is still plenty of work to do in terms of increasing the accuracy of the model.