Any real world data, in principle, is generally unclean, incomplete, unscaled, and overall messy. The pre-processing stage is essential to the machine learning pipeline, in order to prepare the data before further analysis and model building.

A pre-processing generally involves some data transformation, filling missing observations with suitable values, scaling various features to a standard scale, among other things.

The following operations were performed on the raw data set during this phase –

1. Metadata transformation – The raw data set provided to us has got column numbers instead of the actual column names. Given the number of features, at later stages, it would be next to impossible to identify the features. To overcome that, we have assigned the actual column names to the features instead of the numbers. The list of attributes is available Aegean data set web page. In the data set provided to us, the frame.time\_epoch and frame.time\_relative are excluded, so the feature names have been assigned accordingly.
2. Data cleaning – Next we checked the data for missing values. For each feature, we checked if there are any missing observations, and filled them with the median of that feature. The reason for choosing the median, instead of mean is that, in case the overall mean of a feature is very large, it might introduce unnecessary bias in the data, while a median would just assign the most common observation, which should be fair enough.
3. Data transformation – One of the crucial steps in this phase is to ensure that all features are numeric, since ML models generally work with numbers, and Python, unlike R, cannot implicitly handle the categorical features. To do that, first we identified the categorical features, and then used LabelEncoder to assign numbers to the observations. Then, using OneHotEncoder, we transformed those observations into dummy variables. This gives a binary variable for each of the categories.
4. Standardization – The last step we performed on this phase is to standardize the data. The features in any data set are likely to have very different variance with respect to each other, and feeding them to an ML model without scaling them might unfairly tip it in favour of few specific features with greater variance.

StandardScaler and MinMaxScaler are two of the most popular scalers available. However, both these scalers are sensitive to outliers. In this data set, we did not handle the outliers explicitly because we do not possess enough domain knowledge to determine whether a particular outlier is a human error, or a genuinely useful observation. Also, since these are all signal data, and we trying to detect unusual activities through this model, some of the outliers might contain the most useful observation. Therefore, we did not touch the outliers. However, while scaling, we need to keep the outliers in mind as well. For that purpose, we used a different scaler called the RobustScaler. Unlike StandardScaler, which removes the mean and scales to unit variance, RobustScaler removes the median and scales according to the quantile range. Therefore, it is more robust to outliers than any of the other scalers.

A couple of things to keep in mind here,

* The data set provided to us is quite clean, as we couldn’t find any missing values in any of the features. However, in our code, we still handled the missing observation condition as a contingency.
* There were no categorical variables either. There were binary features, but those are most likely signal values. Again, we handled the conversion of categorical to dummy variables, just in case.