Data Wrangling Steps

Introduction:

This document explains the steps taken to clean and prepare the first capstone project data for the analysis and modelling phase.

Source Data:

Display and examine few lines from the top and the bottom of the source data files to prepare for loading the data into Pandas dataframes.

This will be helpful in deciding how to load the data, for example if there are column headers, comments lines at the start or end of the file, empty rows or columns etc. It will also help in identifying field's separator and potential data types.

Load Data:

Taking the above findings into consideration, load the data from source file into Pandas dataframe, assign columns names, and remove empty or unwanted columns.

Run some Pandas dataframe methods to get more information about the loaded data. This includes dataframe .describe(), .head(), .tail(), .dtypes etc. For example, .describe() method will reveal some insights about numeric column data distribution e.g. columns with one constant value that could be excluded from further steps in the pipeline.

Missing Values:

Check for null (NaN) values by running Pandas dataframe method .null().sum(), which gives the total of missing values for each data column. Accordingly, some columns may be excluded from the data if they have high number of missing values.

Alternatively, missing values could be replaced by some statistical measure like mean, median, mode, or some interpolated value based on domain knowledge.

Fortunately, the data set of this project is clean since it mainly contains sensors and machine generated data.

Outliers Detection:

For outlier detection, run Pandas dataframe .quantile() method, SciPy stats.zscore, or manually by identifying the data points above or below [mean ± n standard deviation], where n can take the value 2 or 3 assuming normal distribution.

Pandas dataframe filtering methods could then be used to remove/replace outliers.

Other Data Manipulation:

Join or merge dataframes based on index or common key. For example, labels for test data were in separate source data file. This could be done by using Pandas .concat() and .merge() methods.

Regression and classification labels for training data were created as follows:

- Regression: Time-To-Failure TTF (no. of remaining cycle before failure) for each cycle/engine is the number of cycles between that cycle and the last cycle of the same engine.
- Binary Classification: if the remaining cycles is less than specific number of cycles (e.g. period = 30), the engine will fail in this period, otherwise the engine is fine.
- Multiclass Classification: by segmenting the TTF into cycle bands (e.g. periods: 0-15, 16-30, 30+), we could identify in which period will the engine fail.

For test data, TTF is provided in a separate truth data file. These two files were merged and then classification labels for test data were created the in the same way described above.

Feature extraction is also applied to the training and test data by introducing additional two columns for each of the 21 sensor columns: rolling mean and rolling standard deviation. This smoothing to the sensors measurements over time would improve the performance of some machine learning algorithms.