Final Report

Corrie M. Meyersick

6/30/2019

Data Science Practicum I MSDS_692

Regis University

Equipment Failure Predictions

Purpose

The purpose of this project is to use the data retrieved from UCI Machine Learning Repository, which consist of readings from sensors used for condition monitoring of a hydraulic system, to predict when the equipment will fail.

Data Collection

The dataset used in the project was retrieved from UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Condition+monitoring+of+hydraulic+systems#) and consists of readings from sensors used for condition monitoring of a hydraulic system including pressure, volume flow and temperatures along with the condition of the hydraulic components (cooler, valve, pump, and accumulator).

The Data

The following information regarding the dataset provides was supplied from UCI Machine Learning Repository for the Condition monitoring of hydraulic systems data set (Helwig, Pignanelli & SchÃtze, 2015).

> The data set was experimentally obtained with a hydraulic test rig. This test rig consists of a primary working and a secondary cooling-filtration circuit which are connected via the oil tank [1], [2]. The system cyclically repeats constant load cycles (duration 60 seconds) and measures process values such as pressures, volume flows and temperatures while the condition of four hydraulic components (cooler, valve, pump and accumulator) is quantitatively varied.

> Attribute Information:
> The data set was experimentally obtained with a hydraulic test rig. This test rig consists of a primary working and a secondary cooling-filtration circuit which are connected via the oil tank [1], [2]. The system cyclically repeats constant load cycles (duration 60 seconds) and measures process values such as pressures, volume flows and temperatures while the condition of four hydraulic components (cooler, valve, pump and accumulator) is quantitatively varied.

> Attribute Information:
> The data set contains raw process sensor data (i.e. without feature extraction) which are structured as matrices (tab-delimited) with the rows representing the cycles and the columns the data points within a cycle. The sensors involved are:

Sensor	Physical quantity	Unit	Sampling rate
PS1	Pressure	bar	100 Hz
PS2	Pressure	bar	100 Hz
PS3	Pressure	bar	100 Hz
PS4	Pressure	bar	100 Hz
PS5	Pressure	bar	100 Hz
PS6	Pressure	bar	100 Hz
EPS1	Motor power	W	100 Hz

FS1	Volume flow	l/min	10 Hz
FS2	Volume flow	l/min	10 Hz
TS1	Temperature	Â°C	1 Hz
TS2	Temperature	Â°C	1 Hz
TS3	Temperature	Â°C	1 Hz
TS4	Temperature	Â°C	1 Hz
VS1	Vibration	mm/s	1 Hz
CE	Cooling efficiency (virtual)	%	1 Hz
CP	Cooling power (virtual)	kW	1 Hz
SE	Efficiency factor	%	1 Hz

The target condition values are cycle-wise annotated in â€˜profile.txtâ€˜ (tab-delimited). As before, the row number represents the cycle number. The columns are

1: Cooler condition / %:
3: close to total failure
20: reduced effifiency
100: full efficiency

2: Valve condition / %:
100: optimal switching behavior
90: small lag
80: severe lag
73: close to total failure

3: Internal pump leakage:
0: no leakage
1: weak leakage
2: severe leakage

4: Hydraulic accumulator / bar:
130: optimal pressure
115: slightly reduced pressure
100: severely reduced pressure
90: close to total failure

5: stable flag:
0: conditions were stable
1: static conditions might not have been reached yet

Data Preparation

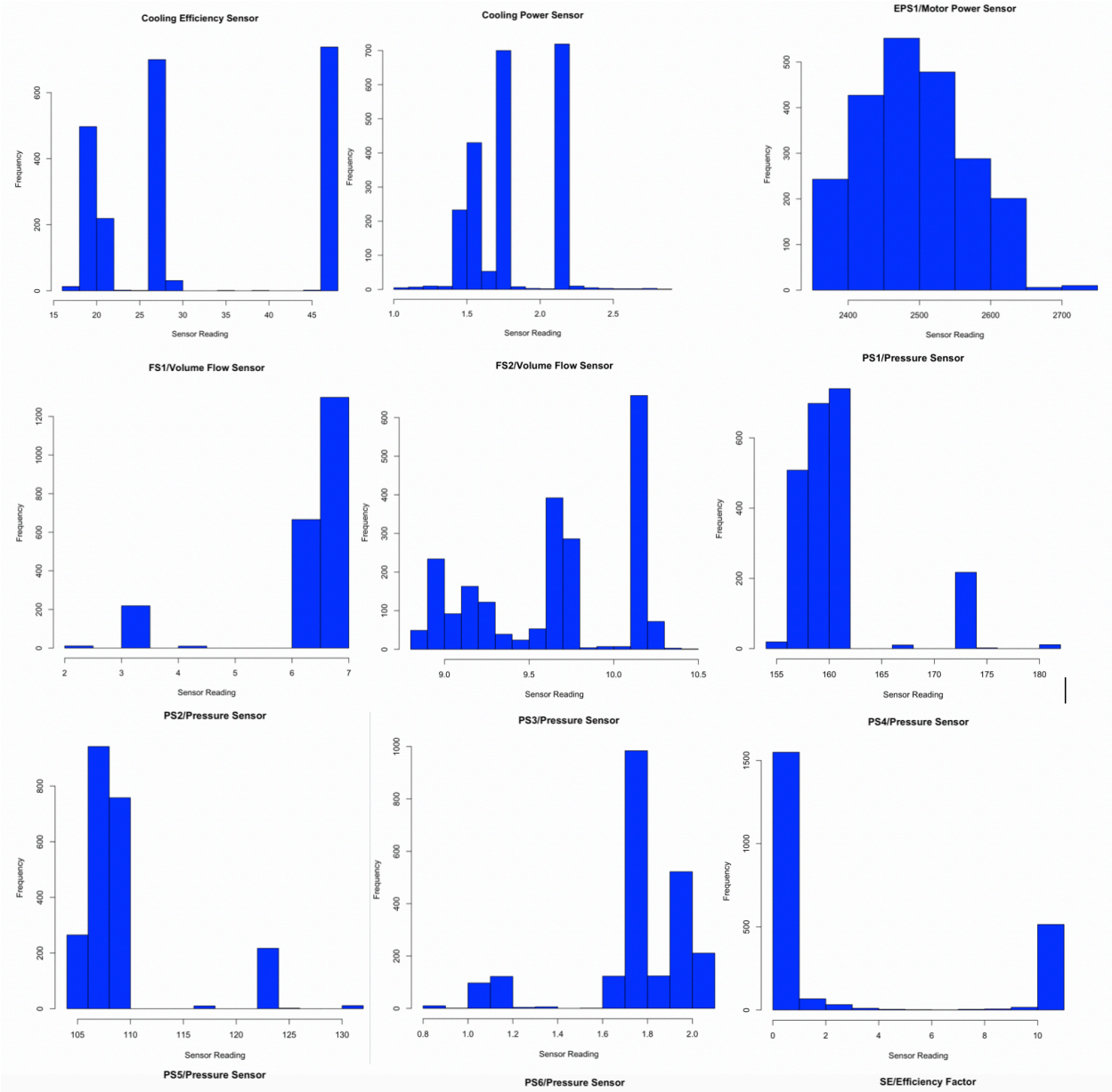
There was minimal data preparation necessary for this data set since there were no missing values and all values were numeric. Each individual dataset was uploaded and a mean value for each variable was created and combined into one data frame. Throughout the analysis, time readings were added to the data frame, the melt function was used for initial visualizations and the data frame was converted to tibble format to be able to perform anomaly detection.

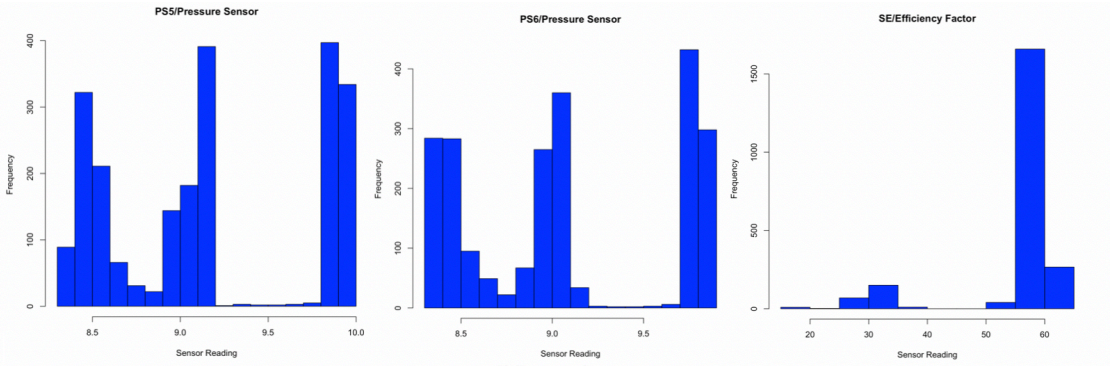
EDA (Exploratory Data Analysis)

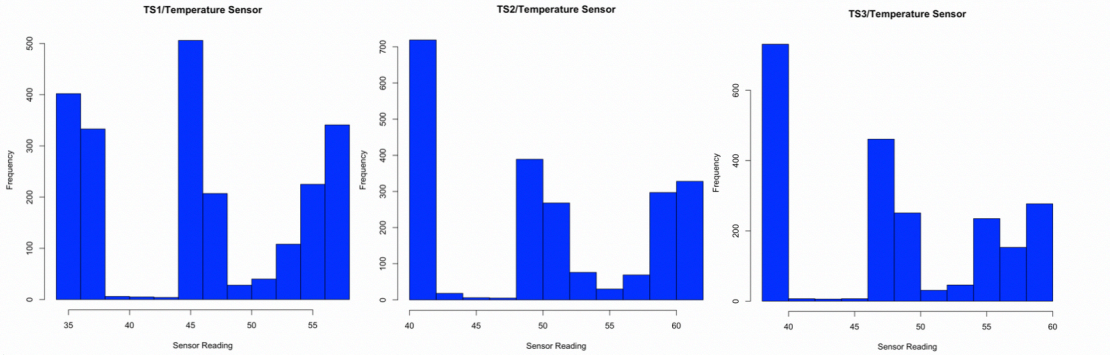
The summary function was used to provide an overall view of the dataset. This provides the minimum, 1st quartile, median, mean, 3rd quartile and maximum values for each of the sensor variables. This was done prior to normalizing the data and with the values having a wide range between the different sensors, it shows the need to normalize the data before progressing on with the project.

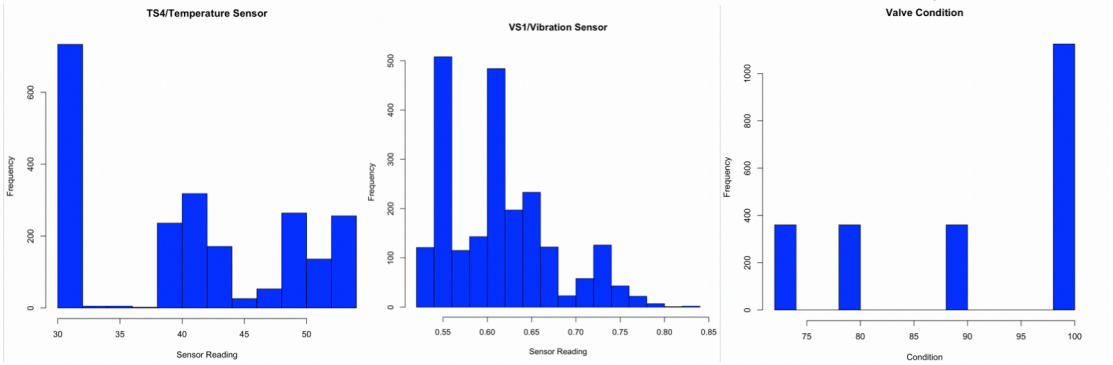
```
> summary(df)
       CE              CP             EPS1           FS1             FS2
 Min.   :17.56   Min.   :1.062   Min.   :2362   Min.   :2.019   Min.   : 8.858
 1st Qu.:20.08   1st Qu.:1.550   1st Qu.:2443   1st Qu.:6.392   1st Qu.: 9.203
 Median :27.39   Median :1.740   Median :2481   Median :6.577   Median : 9.692
 Mean   :31.30   Mean   :1.808   Mean   :2496   Mean   :6.199   Mean   : 9.649
 3rd Qu.:46.68   3rd Qu.:2.148   3rd Qu.:2548   3rd Qu.:6.658   3rd Qu.:10.155
 Max.   :47.90   Max.   :2.840   Max.   :2741   Max.   :6.723   Max.   :10.403
      PS1             PS2             PS3              PS4
 Min.   :155.4   Min.   :104.4   Min.   :0.8403   Min.   : 0.000
 1st Qu.:158.1   1st Qu.:107.0   1st Qu.:1.7297   1st Qu.: 0.000
 Median :159.0   Median :107.7   Median :1.7796   Median : 0.000
 Mean   :160.5   Mean   :109.4   Mean   :1.7532   Mean   : 2.600
 3rd Qu.:161.0   3rd Qu.:109.4   3rd Qu.:1.9320   3rd Qu.: 3.503
 Max.   :180.9   Max.   :131.6   Max.   :2.0234   Max.   :10.207
      PS5             PS6              SE             TS1             TS2
 Min.   :8.366   Min.   :8.322   Min.   :18.28   Min.   :35.31   Min.   :40.86
 1st Qu.:8.547   1st Qu.:8.487   1st Qu.:56.27   1st Qu.:36.24   1st Qu.:41.86
 Median :9.116   Median :9.032   Median :58.76   Median :44.84   Median :49.78
 Mean   :9.163   Mean   :9.079   Mean   :55.29   Mean   :45.42   Mean   :50.37
 3rd Qu.:9.844   3rd Qu.:9.729   3rd Qu.:59.66   3rd Qu.:54.10   3rd Qu.:58.58
 Max.   :9.979   Max.   :9.857   Max.   :60.76   Max.   :57.90   Max.   :61.96
      TS3             TS4             VS1
 Min.   :38.25   Min.   :30.39   Min.   :0.5244
 1st Qu.:39.12   1st Qu.:31.27   1st Qu.:0.5551
 Median :47.07   Median :40.43   Median :0.6102
 Mean   :47.66   Mean   :40.74   Mean   :0.6133
 3rd Qu.:55.69   3rd Qu.:49.41   3rd Qu.:0.6499
 Max.   :59.42   Max.   :53.06   Max.   :0.8391
```

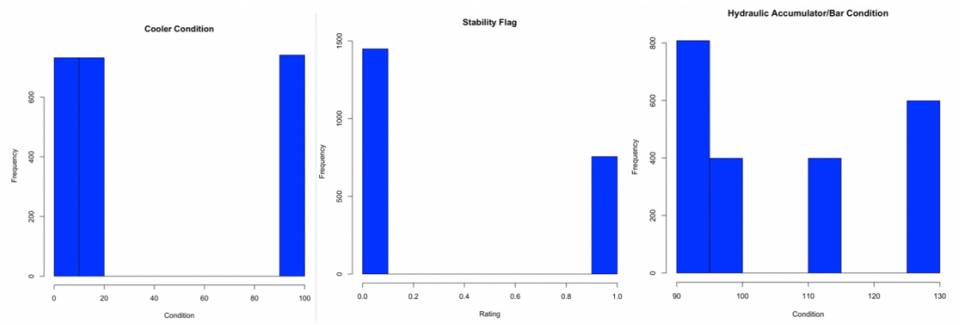
Histograms were created for the variables based on frequency and can be used to see how data is distributed.

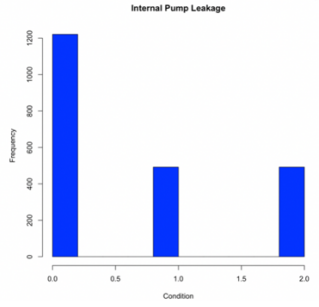



PS5/Pressure Sensor
Frequency
Sensor Reading
PS6/Pressure Sensor
Frequency
Sensor Reading
SE/Efficiency Factor
Frequency
Sensor Reading

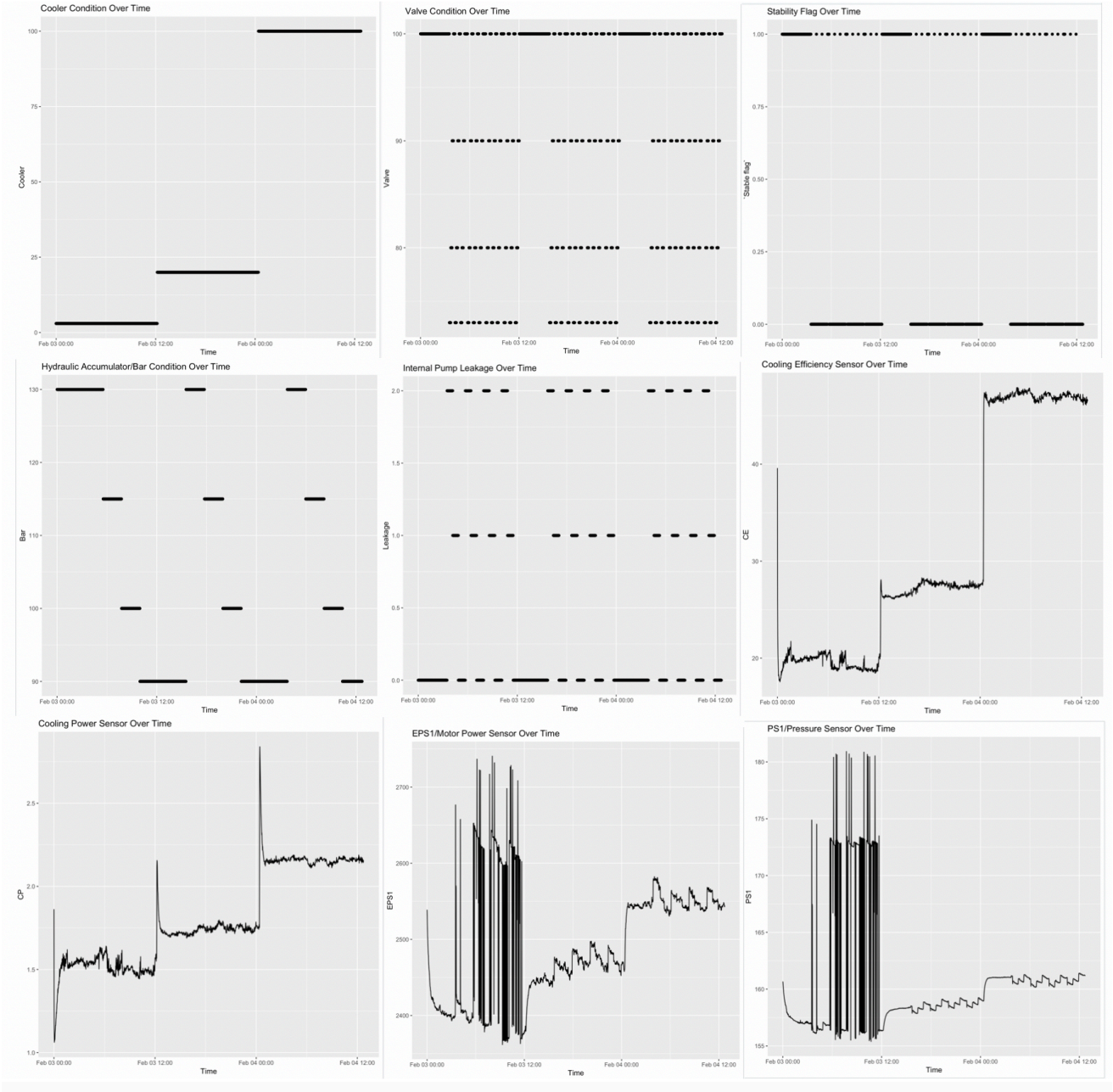

TS1/Temperature Sensor
Frequency
Sensor Reading
TS2/Temperature Sensor
Frequency
Sensor Reading
TS3/Temperature Sensor
Frequency
Sensor Reading

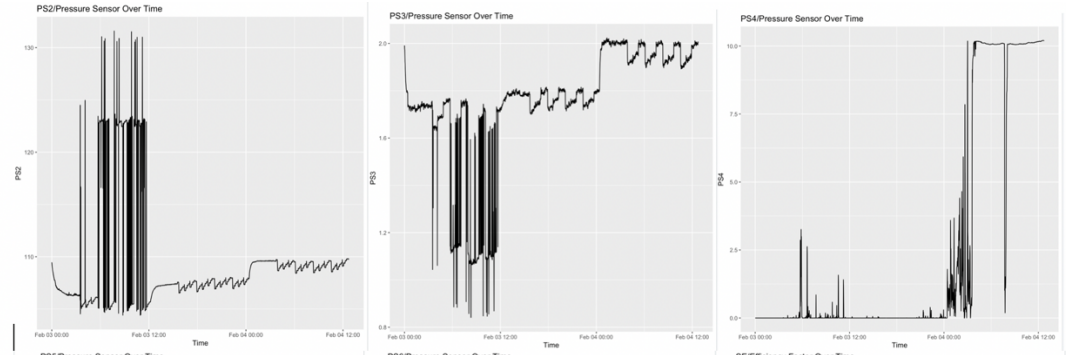

TS4/Temperature Sensor
Frequency
Sensor Reading
VS1/Vibration Sensor
Frequency
Sensor Reading
Valve Condition
Frequency
Condition


Cooler Condition
Frequency
Condition
Stability Flag
Frequency
Rating
Hydraulic Accumulator/Bar Condition
Frequency
Condition


Internal Pump Leakage
Frequency
Condition

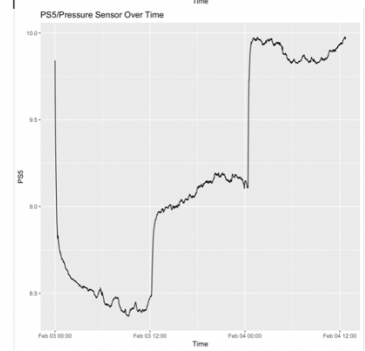
Scatter and line plots were created to see the readings during the time that the data was collected.

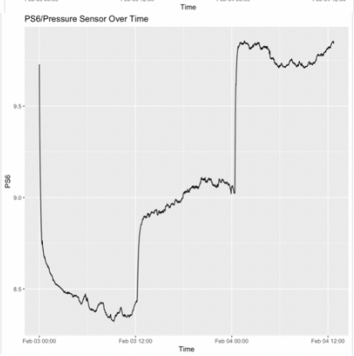


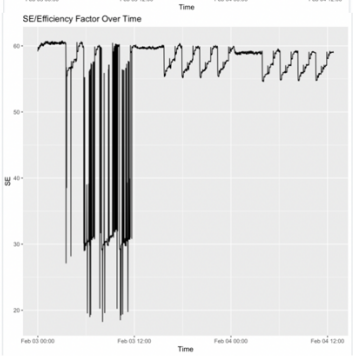

PS2/Pressure Sensor Over Time
PS2
Time

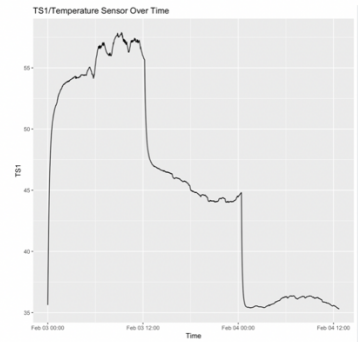
PS3/Pressure Sensor Over Time
PS3
Time

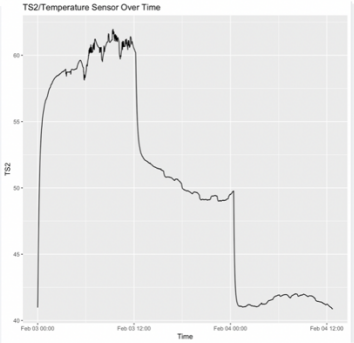
PS4/Pressure Sensor Over Time
PS4
Time

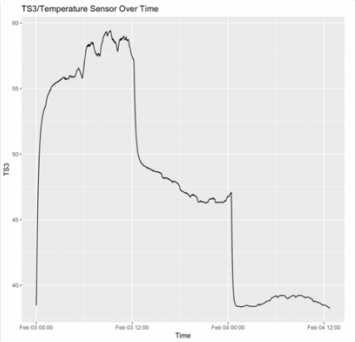

PS5/Pressure Sensor Over Time
PS5
Time

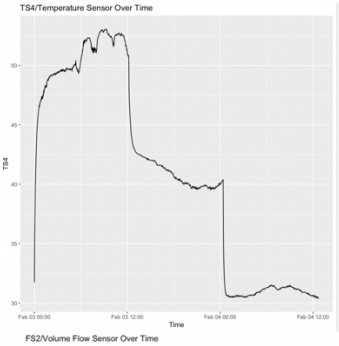

PS6/Pressure Sensor Over Time
PS6
Time

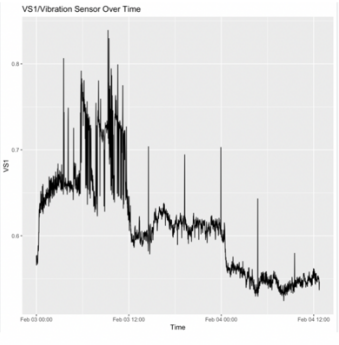

SE/Efficiency Factor Over Time
SE
Time

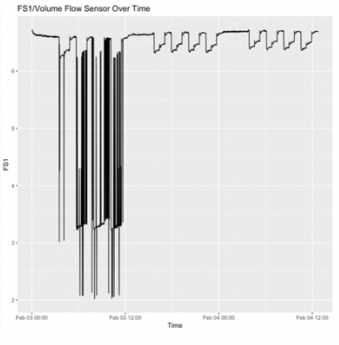

TS1/Temperature Sensor Over Time
TS1
Time

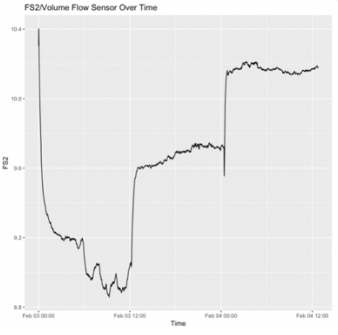

TS2/Temperature Sensor Over Time
TS2
Time


TS3/Temperature Sensor Over Time
TS3
Time

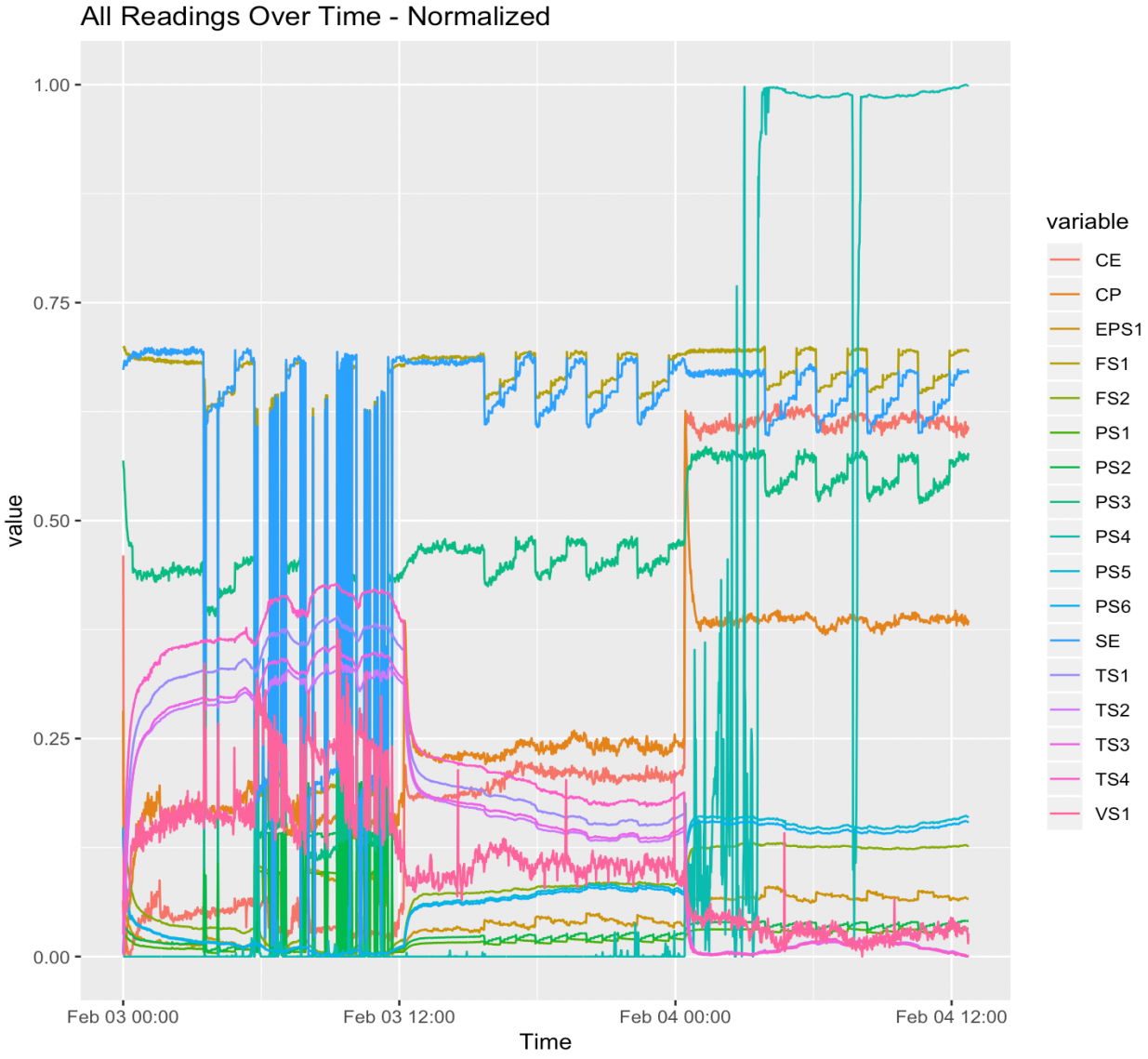

TS4/Temperature Sensor Over Time
TS4
Time

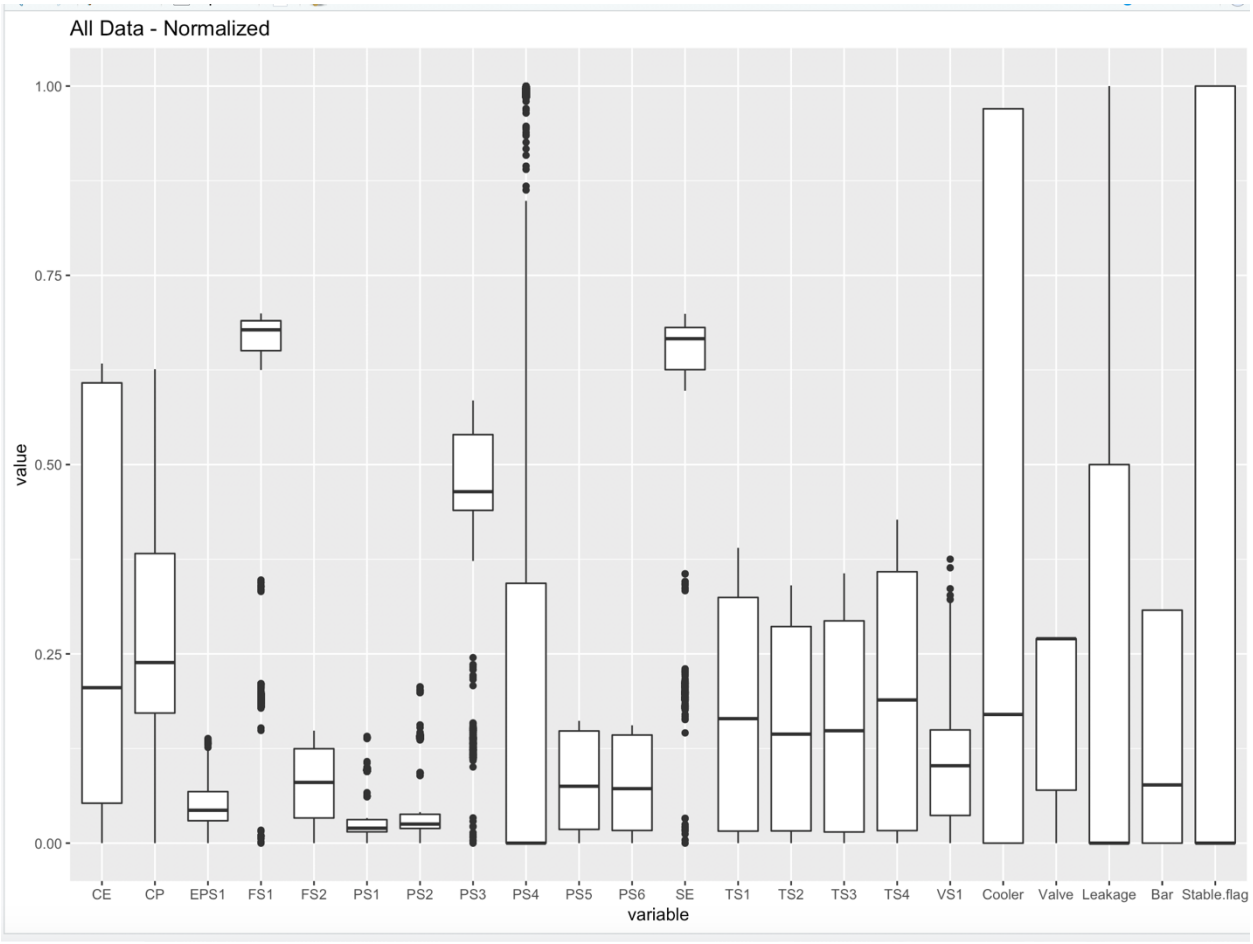

VS1/Vibration Sensor Over Time
VS1
Time


FS1/Volume Flow Sensor Over Time
FS1
Time

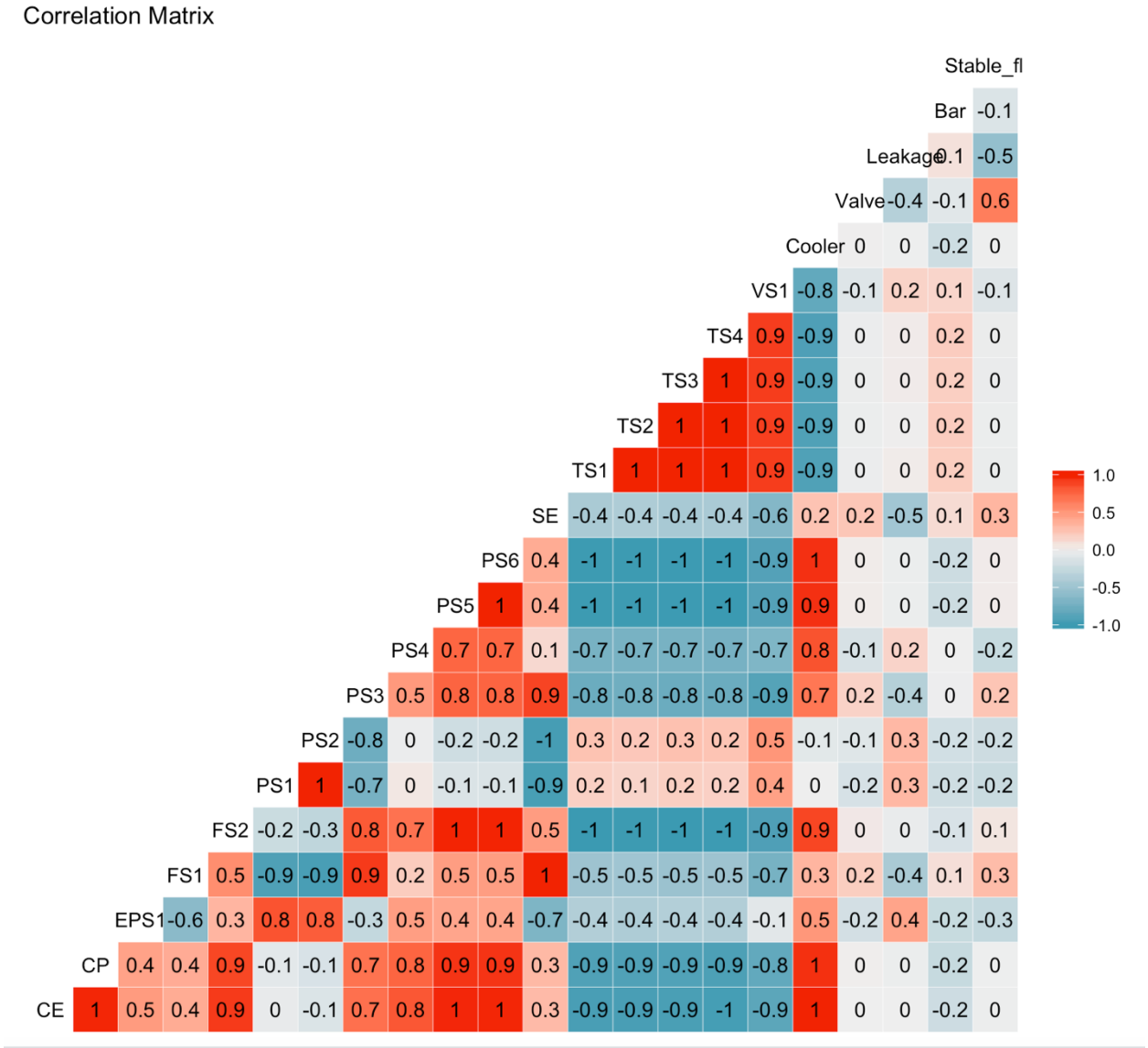

FS2/Volume Flow Sensor Over Time
FS2
Time

After normalizing the data and using the melt function to create a new data frame, a line plot and boxplot with all of the sensor readings were created. We can see in the line plot that there's a lot of activity at the beginning of the time series. This also coincides with the cooler condition reported at near total failure. The boxplot also shows the condition ratings and we can quickly see which variables have outliers, such as the EPS1, FS1, PS1, PS2, PS3, PS4, SE, and VS1 variables.

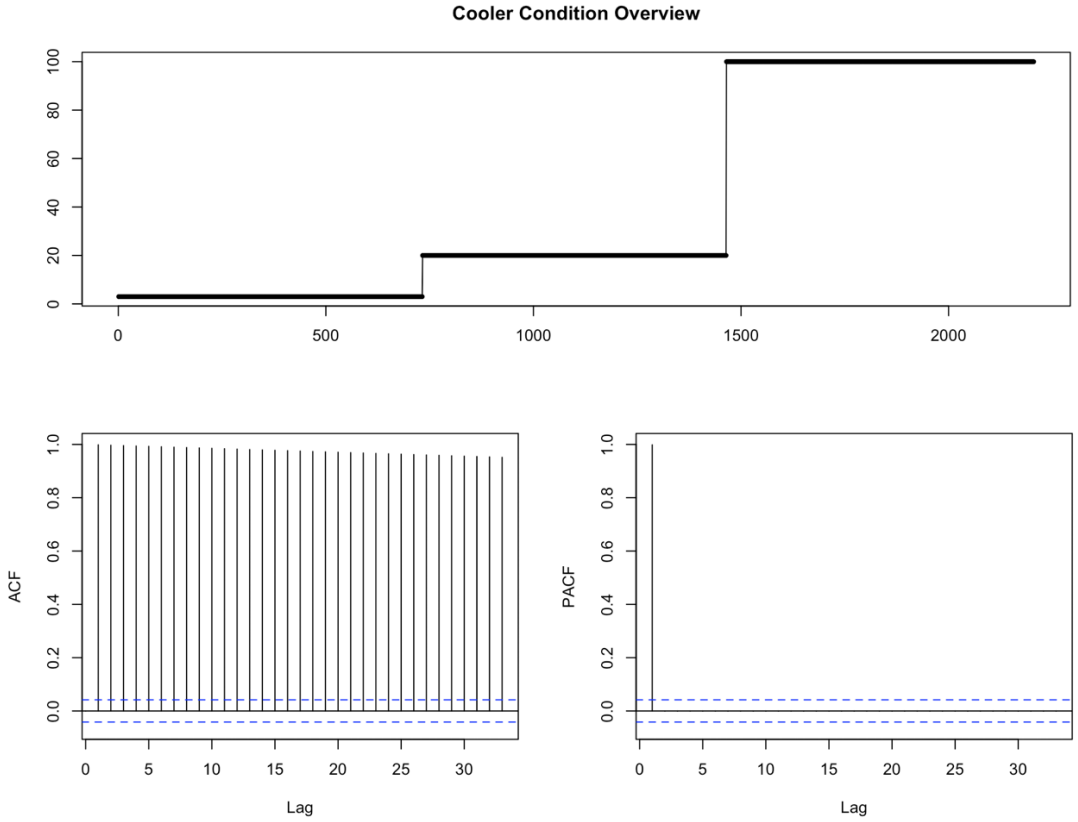


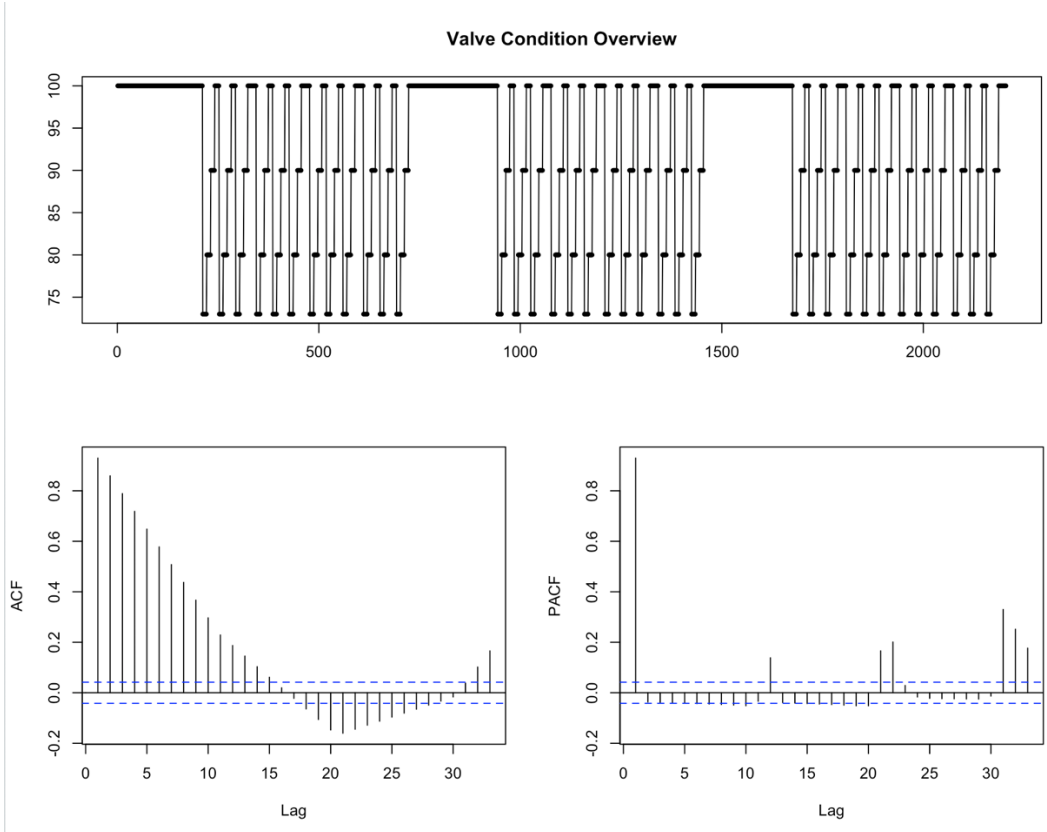


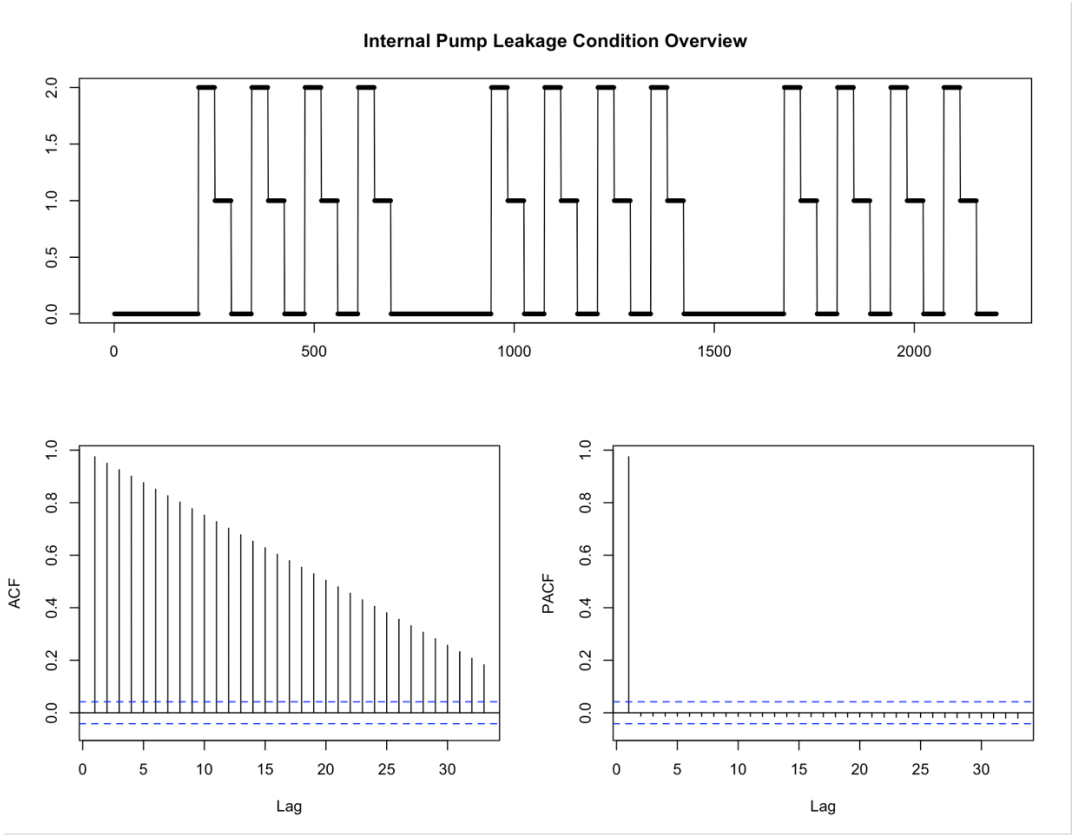
By creating a correlation heat matrix, we can determine the correlation between variables. The closer the correlation coefficient is to 1, the stronger the relationship is. Those in red show the highest correlation with a value of 1, so we can quickly see that the cooler condition is highly correlated with the PS6, CP and CE variables and to a slightly lesser extent, the PS6, PS4, and PS3 variables.

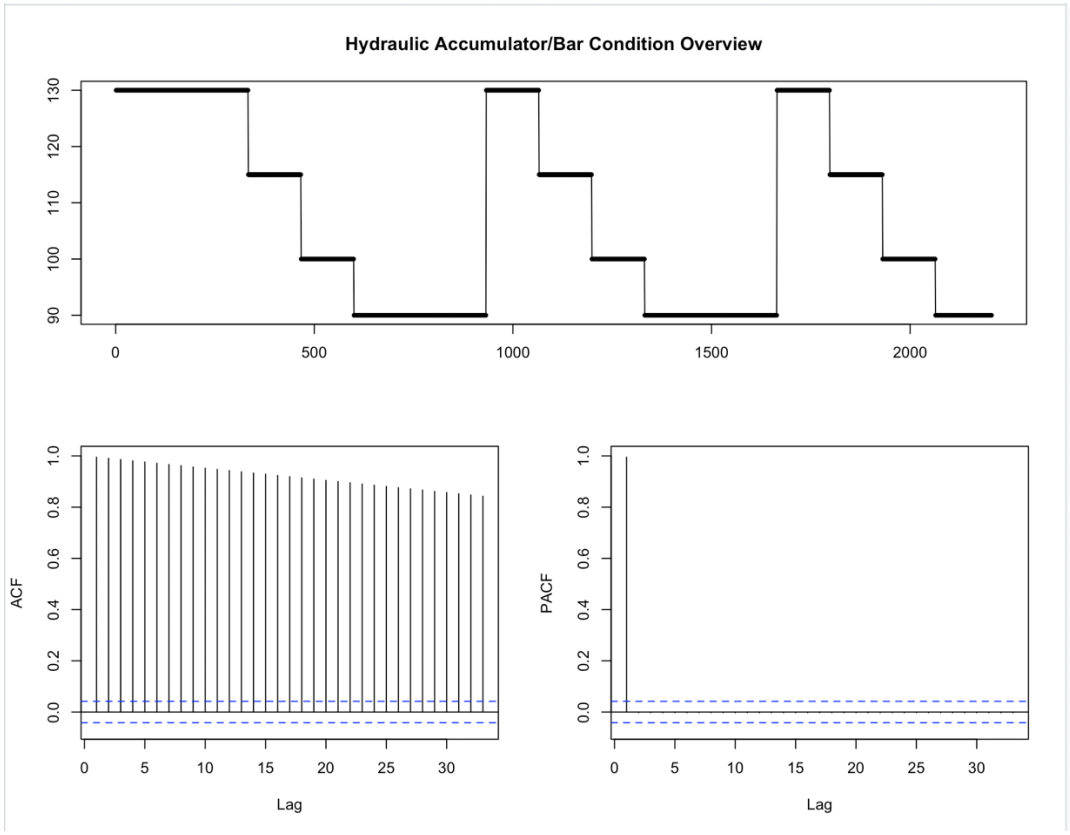


Using the tsdisplay function, we can visualize the auto-correlation function (ACF) with lag values and partial auto-correlation (PACF) with lag. ACF takes trend, seasonality, cyclic, and residuals into consideration where PACF finds correlation of the residuals. This was done for the condition variables used to determine the state of the equipment.


Cooler Condition Overview
0
20
40
60
80
100
0
500
1000
1500
2000
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
Lag
PACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
Lag


Valve Condition Overview
75
80
85
90
95
100
0
500
1000
1500
2000
ACF
-0.2
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
Lag
PACF
-0.2
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
Lag


Internal Pump Leakage Condition Overview
0.0
0.5
1.0
1.5
2.0
0
500
1000
1500
2000
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
Lag
PACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
Lag


Hydraulic Accumulator/Bar Condition Overview
90
100
110
120
130
0
500
1000
1500
2000
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
Lag
PACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
Lag

Anomaly Detection

In order to see anomalies within the data for the sensor variables, the data frame had been converted to tibble time series format and the anomalize package was applied. I prefer to use the anomazlize package for anomaly detection because it's easy to apply with visuals showing the anomalies in red.

The anomalize package was used to determine anomalies within the data and consists of three main functions

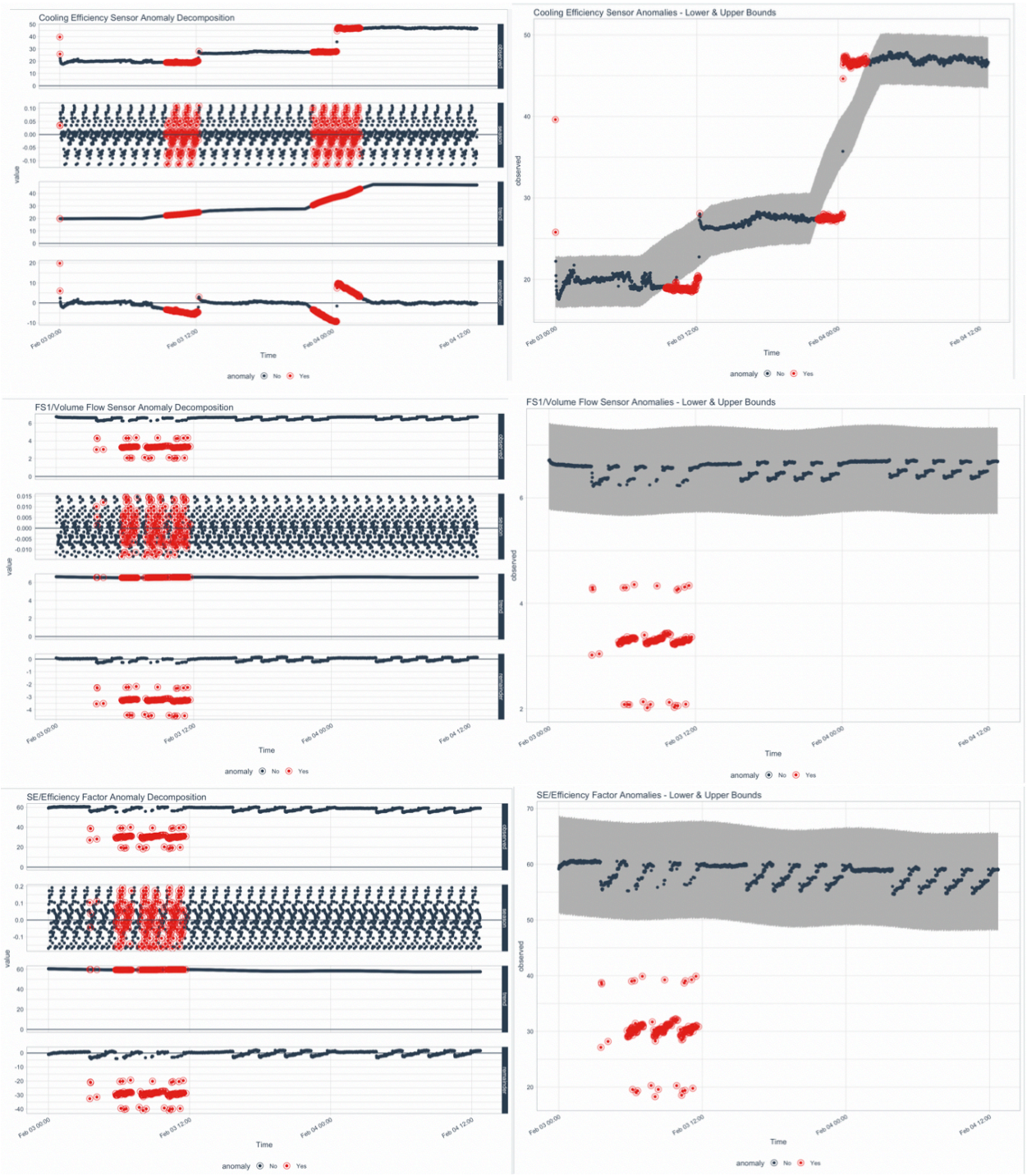
- The time_decompose function separates the data into seasonal, trend and remainder components.
- The anomalize function applies anomaly detection methods to the remainder component and produces three new columns: "remainder_l1" (lower limit), "remainder_l2" (upper limit), and "anomaly" (Yes/No flag).
- The time_recompose function is used to calculate limits that separate the "normal" data from anomalies.

Using the decompose function, we are able to see the breakout for observed, season, trend and remainder with each anomaly highlighted. Using the recompose, we are able to see the upper and upper bounds. The STL method, which uses seasonal decomposition with a Loess Smoother and the IQR method, which is a fast and relatively accurate method for detecting anomalies, was applied to variables within the data set. The default parameters of 0.05 alpha and 0.2 max anomalies were also used.

```
ts %>%
  time_decompose(CE, method = "stl",
                 frequency = "auto", trend = "auto") %>%
  anomalize(remainder, method = "iqr", alpha = 0.05,
            max_anoms = 0.2) %>%
  #visualization
  plot_anomaly_decomposition() +
  ggtitle("Cooling Efficiency Sensor Anomaly Decomposition")
```

```
ts %>%
  time_decompose(CE, method = "stl",
                 frequency = "auto", trend = "auto") %>%
  anomalize(remainder, method = "iqr", alpha = 0.05,
            max_anoms = 0.2) %>%
  time_recompose() %>%
  #visualize
  plot_anomalies(time_recomposed = TRUE) +
  ggtitle("Cooling Efficiency Sensor Anomalies - Lower & Upper Bounds")
```

Since the decompose function breaks out the observed, season, trend and remainder, it was easiest to apply this to each variable individually. I also chose to do the same for the lower and upper bounds. Here are some examples of the visuals as a result of applying the anomalize functions for anomaly detection.



We can see that there's a common timeframe that most of the anomalies occur, which is early on in the timeframe when it was reported that the equipment was close to total failure for cooler condition but also follows the change in state for the other condition variables.

Failure Prediction

To predict failure on the equipment, machine learning techniques were applied. After creating training and test data sets with a 75/25 ratio, I first started by applying the support vector method to the condition variables for the hydraulic system.

For each variable I changed the target to a factor, trained the SVM model, checked the result of the model, created predictions using the predict function based on the model, used the confusionMatrix function to determine accuracy and plotted the predictions.

```
#train svm model
trctrl<-trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(3233)
svm_linear<-train(Valve ~., data = svm_train, method = "svmLinear",
                  trControl=trctrl,
                  preProcess=c("center", "scale"),
                  tuneLength = 10)
#predict classes in test set
svm_test_pred<-predict(svm_linear, newdata = svm_test)
```

When the SVM model was applied to the Valve variable, the confusionMatrix shows that the accuracy of this model was 80%. For the Cooler variable, the accuracy was 100%; for the internal pump leakage variable, the accuracy was 99.6%; and for the hydraulic accumulator/bar variable, the accuracy was 95.3%.

```
Confusion Matrix and Statistics

          Reference
Prediction  73  80  90 100
       73  105   2   1   1
       80    4  79  18   0
       90    0  11  35  38
       100   0   0  36 222

Overall Statistics

               Accuracy : 0.7989
                 95% CI : (0.763, 0.8316)
    No Information Rate : 0.4728
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7063

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 73 Class: 80 Class: 90 Class: 100
Sensitivity             0.9633    0.8587   0.38889     0.8506
Specificity             0.9910    0.9522   0.89394     0.8763
Pos Pred Value          0.9633    0.7822   0.41667     0.8605
Neg Pred Value          0.9910    0.9712   0.88248     0.8673
Prevalence              0.1975    0.1667   0.16304     0.4728
Detection Rate          0.1902    0.1431   0.06341     0.4022
Detection Prevalence    0.1975    0.1830   0.15217     0.4674
Balanced Accuracy       0.9771    0.9054   0.64141     0.8634
```

```
> #check model result
> svm_linear
Support Vector Machines with Linear Kernel

1653 samples
  22 predictor
   4 classes: '73', '80', '90', '100'

Pre-processing: centered (22), scaled (22)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 1488, 1487, 1489, 1488, 1488, 1487, ...
Resampling results:

  Accuracy   Kappa
  0.8082542  0.7019075

Tuning parameter 'C' was held constant at a value of 1
```

```
> #view predictions
> svm_test_pred
  [1] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
 [27] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 73  73  73  73  80
 [53] 80  80  80  80  90  90  90  100 73  73  80  80  80  73  73  73  73  73  80  80  80  100 100 100 100 100
 [79] 100 100 100 73  73  73  73  80  90  90  100 100 100 80  73  73  80  80  80  80  90  100 90  100 73  73
[105] 73  73  73  73  80  80  100 100 100 100 100 100 90  73  100 100 100 73  80  80  90  90  90  73  73  73
[131] 73  80  80  90  90  100 73  73  80  80  100 80  100 80  100 100 100 100 100 73  73  90  80  80  100 100
[157] 100 100 100 73  73  73  80  73  80  73  100 100 100 100 73  73  100 100 100 100 100 100 100 100 100 100
[183] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
[209] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 80  80  73  80  90  90  100 73  73  80  80
[235] 80  90  90  90  90  90  90  90  73  73  90  80  90  90  90  90  90  100 90  100 73  80  80  80  90  90
[261] 90  90  90  90  90  90  73  73  73  73  80  80  80  90  90  100 73  73  73  73  80  80  80  80  90  90
[287] 90  90  90  100 100 73  73  73  73  80  80  80  90  90  90  90  90  90  100 73  73  73  73  80  80  80
[313] 80  80  80  80  80  90  90  90  90  73  90  90  90  90  90  90  90  100 100 100 73  73  73  73  80  90
[339] 90  90  73  73  73  73  80  80  80  80  80  90  90  100 100 100 73  73  80  80  80  90  90  90  90  100
[365] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
[391] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
[417] 73  80  80  90  90  90  90  90  100 73  73  73  80  100 100 100 73  73  73  73  73  80  90  100 100 100
[443] 100 100 73  73  73  73  90  90  80  80  100 100 100 100 100 73  73  73  80  80  80  80  80  80  100 80
[469] 100 100 73  73  73  90  90  90  90  90  100 100 100 100 100 100 73  73  80  80  100 100 100 100 73  73
[495] 73  73  80  80  80  80  80  100 100 100 100 73  73  73  73  80  80  80  100 100 100 100 100 100 100 73
[521] 73  73  73  80  80  80  80  100 100 100 100 100 73  73  80  80  100 100 100 80  73  80  80  80  100 100
[547] 100 100 100 100 100 100 100
Levels: 73 80 90 100
```

```
Confusion Matrix and Statistics

          Reference
Prediction   3  20 100
       3   174   0   0
       20    0 189   0
       100   0   0 189

Overall Statistics

               Accuracy : 1
                 95% CI : (0.9933, 1)
    No Information Rate : 0.3424
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 3 Class: 20 Class: 100
Sensitivity            1.0000    1.0000     1.0000
Specificity            1.0000    1.0000     1.0000
Pos Pred Value         1.0000    1.0000     1.0000
Neg Pred Value         1.0000    1.0000     1.0000
Prevalence             0.3152    0.3424     0.3424
Detection Rate         0.3152    0.3424     0.3424
Detection Prevalence   0.3152    0.3424     0.3424
Balanced Accuracy      1.0000    1.0000     1.0000
```

```
> #check model result
> svm_linear_c
Support Vector Machines with Linear Kernel

1653 samples
  22 predictor
   3 classes: '3', '20', '100'

Pre-processing: centered (24), scaled (24)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 1489, 1487, 1488, 1488, 1488, 1488, ...
Resampling results:

  Accuracy   Kappa
  0.9985846  0.9978769

Tuning parameter 'C' was held constant at a value of 1
```

```
> #view predictions
> svm_test_pred_c
  [1] 3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
 [27] 3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
 [53] 3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
 [79] 3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
[105] 3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
[131] 3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
[157] 3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   20  20  20  20  20  20  20
[183] 20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20
[209] 20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20
[235] 20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20
[261] 20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20
[287] 20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20
[313] 20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20
[339] 20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  100
[365] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
[391] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
[417] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
[443] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
[469] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
[495] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
[521] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
[547] 100 100 100 100 100 100
Levels: 3 20 100
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1   2
         0 302   0   0
         1   0 126   0
         2   0   2 122

Overall Statistics

               Accuracy : 0.9964
                 95% CI : (0.987, 0.9996)
    No Information Rate : 0.5471
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9939

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2
Sensitivity            1.0000   0.9844   1.0000
Specificity            1.0000   1.0000   0.9953
Pos Pred Value         1.0000   1.0000   0.9839
Neg Pred Value         1.0000   0.9953   1.0000
Prevalence             0.5471   0.2319   0.2210
Detection Rate         0.5471   0.2283   0.2210
Detection Prevalence   0.5471   0.2283   0.2246
Balanced Accuracy      1.0000   0.9922   0.9977
```

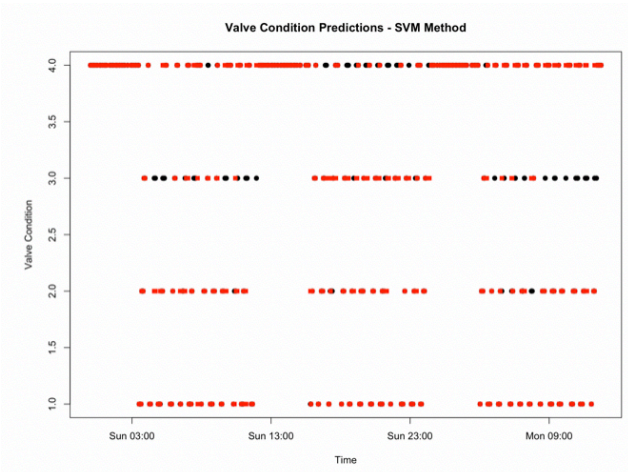
```
> #check model result
> svm_linear_l
Support Vector Machines with Linear Kernel

1653 samples
  22 predictor
   3 classes: '0', '1', '2'

Pre-processing: centered (25), scaled (25)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 1488, 1488, 1487, 1489, 1487, 1488, ...
Resampling results:

  Accuracy   Kappa
  0.9979859  0.9966022

Tuning parameter 'C' was held constant at a value of 1
```

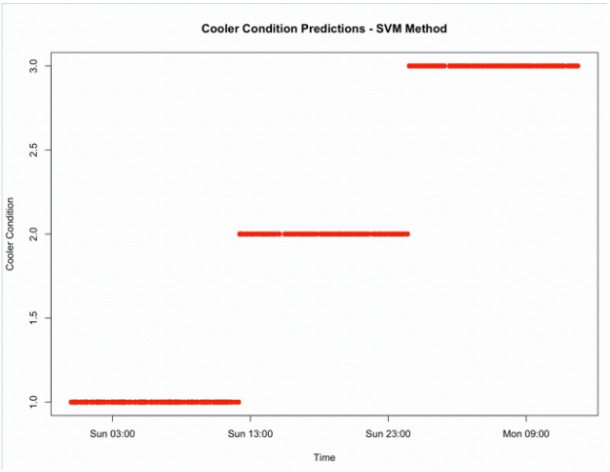
```
> #view predictions
> svm_test_pred_l
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2
 [54] 2 2 2 2 2 2 2 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 0 0 0 0 0 0
[107] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2
[160] 1 1 1 1 1 1 2 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[213] 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2
[266] 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
[319] 1 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[372] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2
[425] 1 1 1 2 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
[478] 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2
[531] 2 2 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
Levels: 0 1 2
```

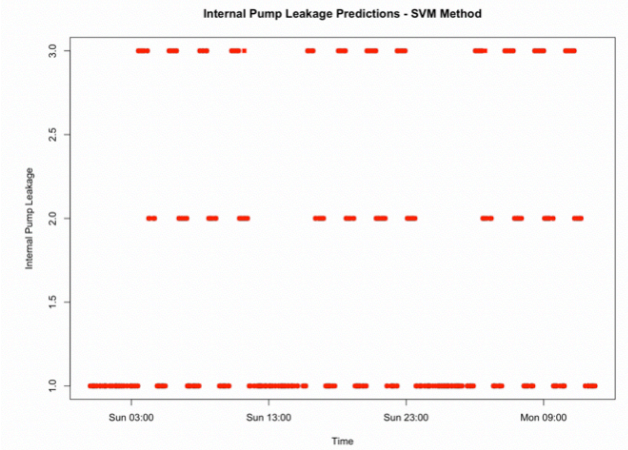
```
Confusion Matrix and Statistics

          Reference
Prediction  90 100 115 130
       90  194   8   8   7
       100   1  93   0   0
       115   0   0 103   1
       130   0   0   1 136

Overall Statistics

               Accuracy : 0.9529
                 95% CI : (0.9317, 0.969)
    No Information Rate : 0.3533
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9352

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 90 Class: 100 Class: 115 Class: 130
Sensitivity             0.9949     0.9208     0.9196     0.9444
Specificity             0.9356     0.9978     0.9977     0.9975
Pos Pred Value          0.8940     0.9894     0.9904     0.9927
Neg Pred Value          0.9970     0.9825     0.9799     0.9807
Prevalence              0.3533     0.1830     0.2029     0.2609
Detection Rate          0.3514     0.1685     0.1866     0.2464
Detection Prevalence    0.3931     0.1703     0.1884     0.2482
Balanced Accuracy       0.9652     0.9593     0.9587     0.9710
```

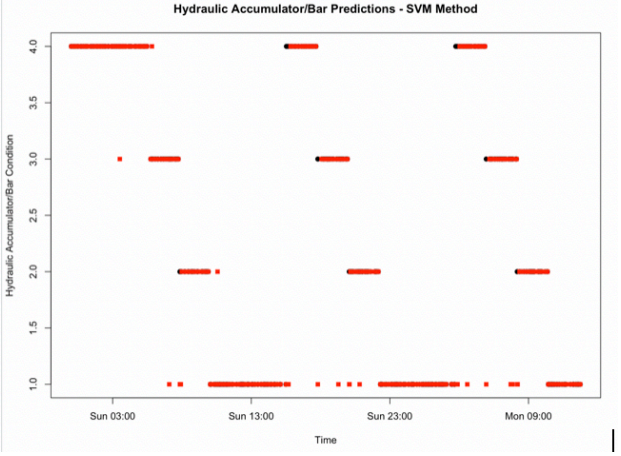
```
> #check model result
> svm_linear_b
Support Vector Machines with Linear Kernel

1653 samples
  22 predictor
   4 classes: '90', '100', '115', '130'

Pre-processing: centered (26), scaled (26)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 1488, 1488, 1487, 1487, 1487, 1488, ...
Resampling results:

  Accuracy   Kappa
  0.9457254  0.9240883

Tuning parameter 'C' was held constant at a value of 1
```

```
> #view predictions
> svm_test_pred_b
  [1] 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130
 [27] 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 115 130 130 130 130 130
 [53] 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130
 [79] 130 130 130 115 115 130 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 90  115 115
[105] 115 115 115 115 115 115 115 115 115 115 115 115 115 115 90  90  90  100 100 100 100 100 100 100 100 100
[131] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 90  90  90  90  90  90  90  100 90  90
[157] 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
[183] 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
[209] 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  130 130 130 130 130 130 130 130 130 130
[235] 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 90  115 115 115 115 115 115 115 115
[261] 115 115 115 115 115 115 115 115 115 115 115 115 115 115 90  115 115 115 115 115 115 115 115 115 115 115
[287] 115 115 115 90  90  100 100 100 100 100 100 100 100 100 100 100 100 100 90  100 100 100 100 100 100 100
[313] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 90  90  90  90  90  90  90  90
[339] 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
[365] 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
[391] 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
[417] 130 130 130 130 130 130 130 130 130 90  130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130
[443] 130 90  115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115 115
[469] 115 115 115 115 115 90  90  90  115 115 115 115 115 90  90  100 100 100 100 100 100 100 100 100 100
[495] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 90  90  90  90  90  90
[521] 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
[547] 90  90  90  90  90  90
Levels: 90 100 115 130
```









Forecasting

ETS and ARIMA methods were applied for forecasting the condition variables of the hydraulic equipment. ETS is a simple exponential smoothing model that uses a weighted average of past observations. ARIMA is an auto-regressive integrated moving average model. Both models use a three-character or digit code on the model for the error type, trend type and season type.

For both methods, the model was created, model results were viewed, residuals were checked, the forecast function was applied, and the results of the forecast viewed and plotted. Both methods had similar results for forecasting the condition variables, however the ARIMA model had better accuracy ratings.

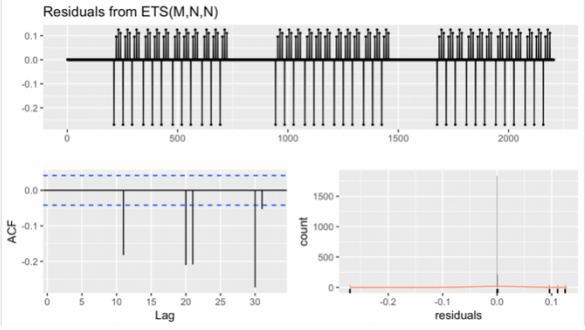
```
#use ets function to auto choose a model by default
cooler_fit_ets<-ets(ts$Cooler)
cooler_fit_ets #view results
#Check residuals
checkresiduals(cooler_fit_ets)
#accuracy function is used to see the predictive accuracy
accuracy(cooler_fit_ets)
#forecast for the next 500 cycles
cooler_pred_ets<-forecast(cooler_fit_ets, h=500)
#view forecast results
cooler_pred_ets
plot(cooler_pred_ets, main = "Cooler Condition Forecast = ETS Method",
     xlab = "Interval", ylab = "Cooler Condition")
```

```
#create model with auto.arima function
cooler_arima<-auto.arima(ts$Cooler)
#view results
cooler_arima
#check residuals
checkresiduals(cooler_arima)
#View accuracy
accuracy(cooler_arima)
#create forecast
cooler_arima_forecast<-forecast(cooler_arima, h=500)
#view forecast results
cooler_arima_forecast
#plot results
plot(cooler_arima_forecast,
     main = "Cooler Condition Forecast - ARIMA 0,1,0")
```

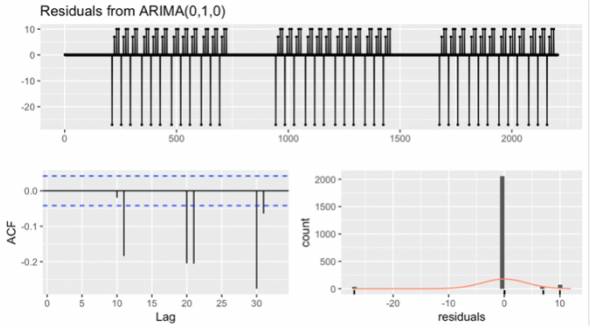
```
> valve_fit_ets #view results
ETS(M,N,N)

Call:
 ets(y = ts$Valve)

  Smoothing parameters:
    alpha = 0.9999

  Initial states:
    l = 99.8693

  sigma:  0.0424

     AIC     AICc      BIC
22887.62 22887.63 22904.71
```

```
> valve_arima
Series: ts$Valve
ARIMA(0,1,0)

sigma^2 estimated as 15.97:  log likelihood=-6180.98
AIC=12363.96   AICc=12363.97   BIC=12369.66
```

```
> checkresiduals(valve_fit_ets)

        Ljung-Box test

data:  Residuals from ETS(M,N,N)
Q* = 0.0074197, df = 8, p-value = 1

Model df: 2.   Total lags used: 10
```

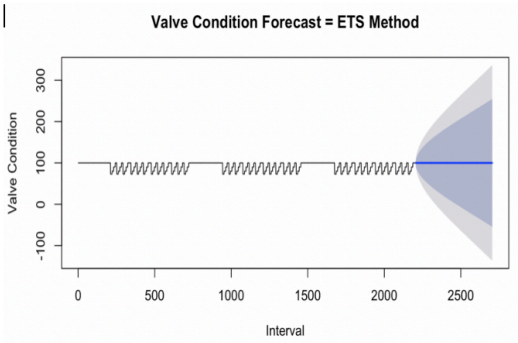


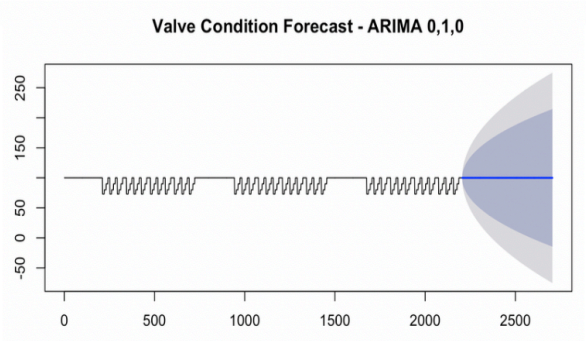

```
> checkresiduals(valve_arima)

        Ljung-Box test

data:  Residuals from ARIMA(0,1,0)
Q* = 0.83677, df = 10, p-value = 0.9999

Model df: 0.   Total lags used: 10
```




```
> accuracy(valve_fit_ets)
                      ME     RMSE      MAE       MPE     MAPE     MASE       ACF1
Training set 5.927871e-05 3.995917 0.8817945 -0.1162839 1.091573 0.99973 0.0001163172
```

```
> accuracy(valve_arima)
                      ME     RMSE      MAE       MPE     MAPE     MASE          ACF1
Training set 4.535145e-05 3.995917 0.881678 -0.1162843 1.091432 0.9995979 -5.841722e-14
```



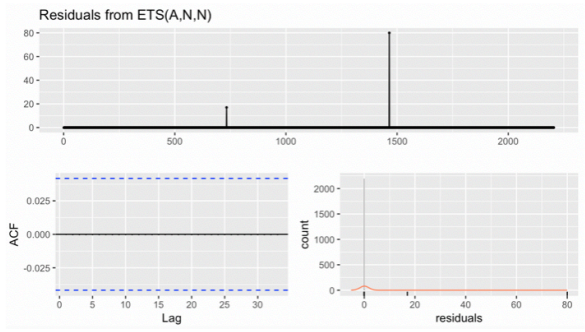


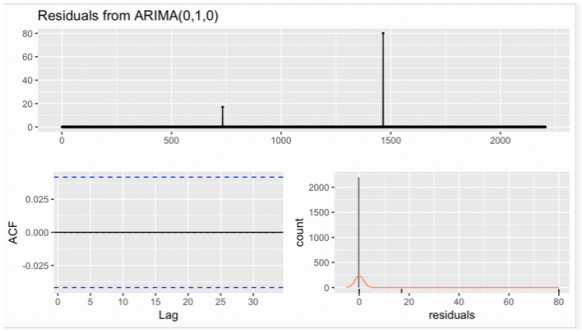

```
> cooler_fit_ets #view results
ETS(A,N,N)

Call:
 ets(y = ts$Cooler)

  Smoothing parameters:
    alpha = 0.9999

  Initial states:
    l = 3.113

  sigma:  1.7425

     AIC     AICc      BIC
19428.13 19428.14 19445.22
```

```
> cooler_arima
Series: ts$Cooler
ARIMA(0,1,0)

sigma^2 estimated as 3.035:  log likelihood=-4350.77
AIC=8703.54   AICc=8703.54   BIC=8709.24
```

```
> checkresiduals(cooler_fit_ets)

        Ljung-Box test

data:  Residuals from ETS(A,N,N)
Q* = 0.0087188, df = 8, p-value = 1

Model df: 2.   Total lags used: 10
```

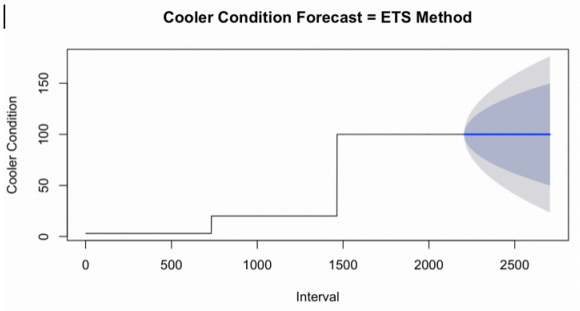
```
> checkresiduals(cooler_arima)

        Ljung-Box test

data:  Residuals from ARIMA(0,1,0)
Q* = 0.0090611, df = 10, p-value = 1

Model df: 0.   Total lags used: 10
```

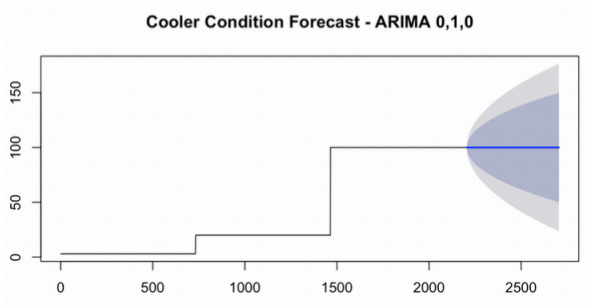





```
> accuracy(cooler_fit_ets)
                     ME     RMSE      MAE       MPE     MAPE     MASE       ACF1
Training set 0.04394531 1.741714 0.044047 0.07313079 0.07654826 1.000839 -0.0005097429
```

```
> accuracy(cooler_arima)
                     ME     RMSE      MAE       MPE     MAPE     MASE       ACF1
Training set 0.04399229 1.741712 0.043992 0.07487528 0.07487528 0.9995774 -0.0006386475
```



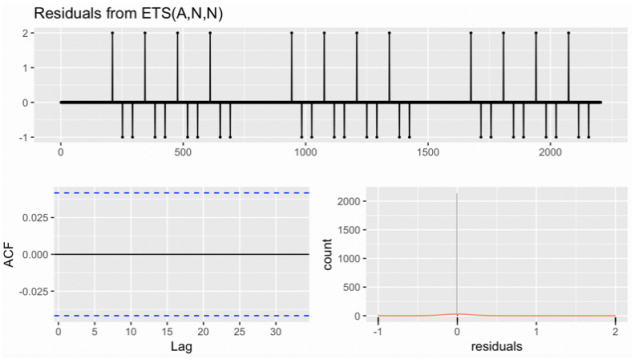


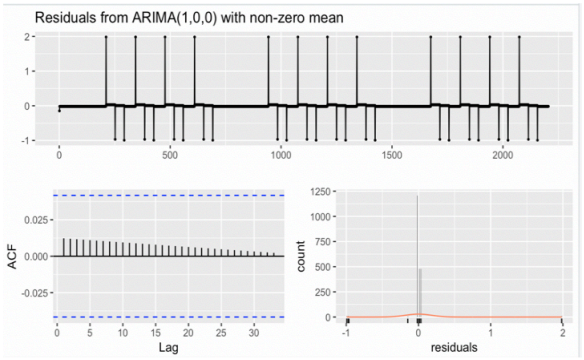
```
> leakage_fit_ets #view results
ETS(A,N,N)

Call:
 ets(y = ts$Leakage)

  Smoothing parameters:
    alpha = 0.9998

  Initial states:
    l = -9e-04

  sigma:  0.1808

     AIC     AICc      BIC
9436.049 9436.060 9453.144
```

```
> leakage_arima
Series: ts$Leakage
ARIMA(1,0,0) with non-zero mean

Coefficients:
         ar1    mean
      0.9754  0.6690
s.e.  0.0046  0.1525

sigma^2 estimated as 0.03229:  log likelihood=655.51
AIC=-1305.03   AICc=-1305.01   BIC=-1287.93
```

```
> checkresiduals(leakage_fit_ets)

        Ljung-Box test

data:  Residuals from ETS(A,N,N)
Q* = 7.8385e-05, df = 8, p-value = 1

Model df: 2.   Total lags used: 10
```

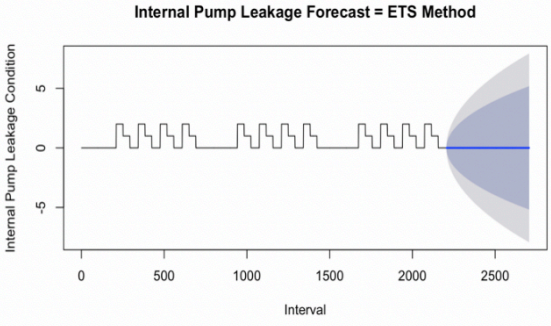
```
> checkresiduals(leakage_arima)

        Ljung-Box test

data:  Residuals from ARIMA(1,0,0) with non-zero mean
Q* = 2.6405, df = 8, p-value = 0.9549

Model df: 2.   Total lags used: 10
```

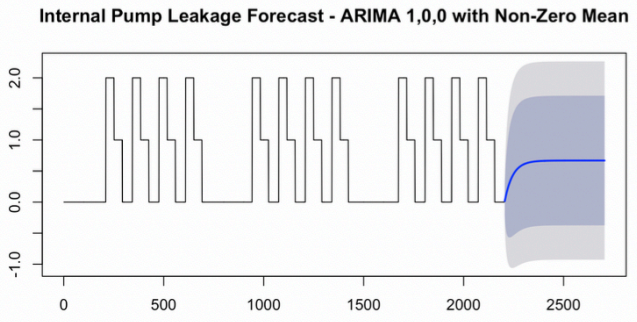





```
> accuracy(leakage_fit_ets)
                      ME      RMSE       MAE MPE MAPE      MASE         ACF1
Training set 4.005695e-07 0.1807016 0.02177321 NaN  Inf 0.9997532 0.0001884154
```

```
> accuracy(leakage_arima)
                       ME      RMSE       MAE  MPE MAPE     MASE      ACF1
Training set -4.977002e-05 0.1796206 0.03947423 -Inf  Inf 1.812525 0.0122895
```



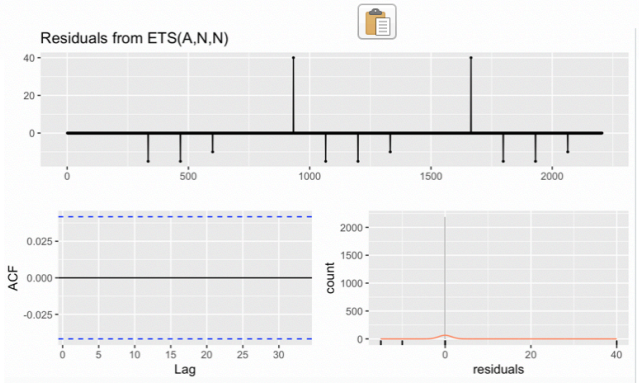


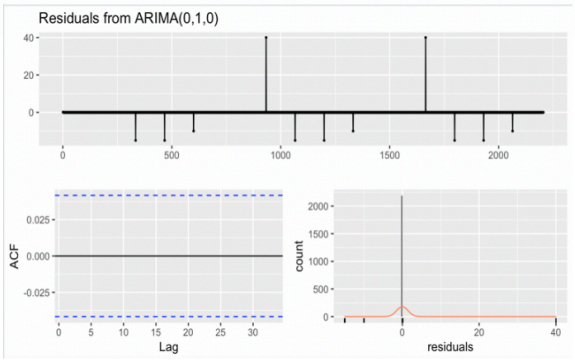
```
> bar_fit_ets #view results
ETS(A,N,N)

Call:
 ets(y = ts$Bar)

  Smoothing parameters:
    alpha = 0.9999

  Initial states:
    l = 130.0056

  sigma:  1.4838

     AIC     AICc      BIC
18719.25 18719.26 18736.34
```

```
> bar_arima
Series: ts$Bar
ARIMA(0,1,0)

sigma^2 estimated as 2.201:  log likelihood=-3996.49
AIC=7994.99   AICc=7994.99   BIC=8000.68
```

```
> checkresiduals(bar_fit_ets)

        Ljung-Box test

data:  Residuals from ETS(A,N,N)
Q* = 0.00044926, df = 8, p-value = 1

Model df: 2.   Total lags used: 10
```

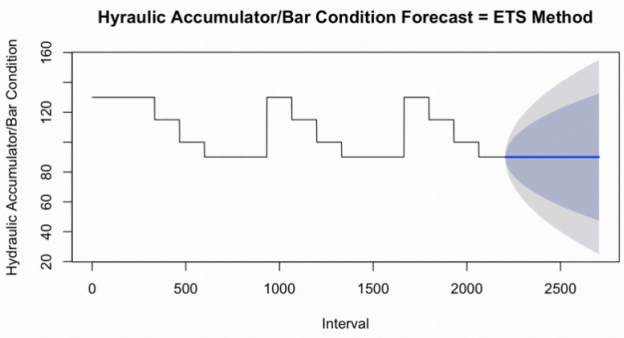
```
> checkresiduals(bar_arima)

        Ljung-Box test

data:  Residuals from ARIMA(0,1,0)
Q* = 0.00049463, df = 10, p-value = 1

Model df: 0.   Total lags used: 10
```

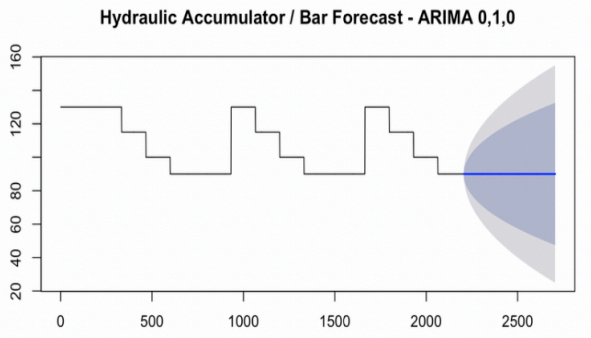





```
> accuracy(bar_fit_ets)
                     ME     RMSE      MAE         MPE      MAPE      MASE          ACF1
Training set -0.01814559 1.483087 0.0907177 -0.02536833 0.08119304 0.9997091 -1.52433e-05
```

```
> accuracy(bar_arima)
                     ME     RMSE      MAE        MPE      MAPE      MASE         ACF1
Training set -0.01808163 1.483089 0.0907619 -0.0253176 0.0812255 1.000196 -0.0001492159
```





Summary

In summary, this was a large data set that contained a lot of information that can be used to provide insight on the equipment and its status. Using anomaly detection, we can see points in which sensor readings are outside the normal bounds providing early detection that something is wrong with the equipment. Applying the SVM model to the equipment condition variables, we received a high accuracy rating indicating that this model could be used to predict failures with the equipment. Using the ARIMA or ETS method, we could forecast future readings on the equipment. Using this information, a company could begin transforming maintenance on their equipment from a break/fix or preventive maintenance model to a predictive maintenance model leading to more accurate planning and budgeting as well as cost savings.

References

Dancho, M. (2018). *Anomaly Detection Using Tidy and Anomalize*. Retrieved from https://www.business-science.io/code-tools/2018/04/08/introducing-anomalize.html

Helwig, N., Pignanelli, E., & SchÃtze, A. Condition Monitoring of a Complex Hydraulic System Using Multivariate Statistics™, in Proc. I2MTC-2015 - 2015 IEEE International Instrumentation and Measurement Technology Conference, paper PPS1-39, Pisa, Italy, May 11-14, 2015, doi: 10.1109/I2MTC.2015.7151267. Retrieved from https://archive.ics.uci.edu/ml/datasets/Condition+monitoring+of+hydraulic+systems#

Prabhakaran, S. (2017). *Time Series Analysis.* Retrieved from http://r-statistics.co/Time-Series-Analysis-With-R

Salvi, J. (2019). *Significance of ACF and PACF Plots in Time Series Analysis.* Retrieved from https://towardsdatascience.com/significance-of-acf-and-pacf-plots-in-time-series-analysis-2fa11a5d10a8

Saxena, R. (2017). *Support Vector Machine Classifier Implementation in R with Caret Package*. Retrieved from https://dataaspirant.com/2017/01/19/support-vector-machine-classifier-implementation-r-caret-package/

Stoklosa, J. and Blakey, R. (2016). *Forecasting with Time Series*. Retrieved from http://environmentalcomputing.net/forecasting-time-series/

Srivastava, T. (2015). *A Complete Tutorial on Time Series Modeling in R.* Retrieved from https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/