CASE STUDY 1: MONITORING AND FAILURE PREDICTION OF HEALTH MONITORING DEVICE.

Richard Han
rickyhan24@gmail.com

ABSTRACT

In this case study, I perform exploratory data analysis on a dataset consisting of daily aggregated device telemetry readings to identify the most significant attributes of the data. Secondly, I apply several classification models and identify the best-performing models in regard to accuracy, precision, and recall–with especial emphasis on recall due to the importance of identifying faulty medical devices.

1 Problem Statement

The task is to build a model to predict device failure given the daily aggregated device telemetry readings. Each device has a time-series for each of attributes 1 through 9. I need to identify which attributes are significant for prediction and to use those attributes to build a model that will classify a given device as faulty or not.

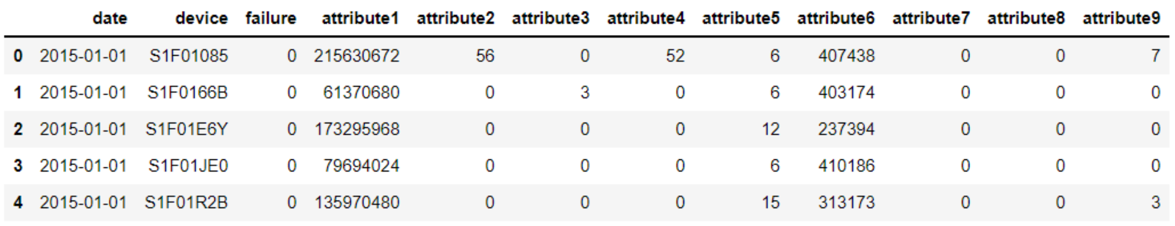
2 Data

The data consists of 124,494 rows of daily aggregated device telemetry readings with 12 columns. The first column gives the date, the second column gives the device id, and the third column indicates whether the device has failure or not. The remaining 9 columns correspond to attributes 1 through 9.

The rows correspond to dates from January 01, 2015 to November 02, 2015, and there are a total of 304 unique dates in this range.

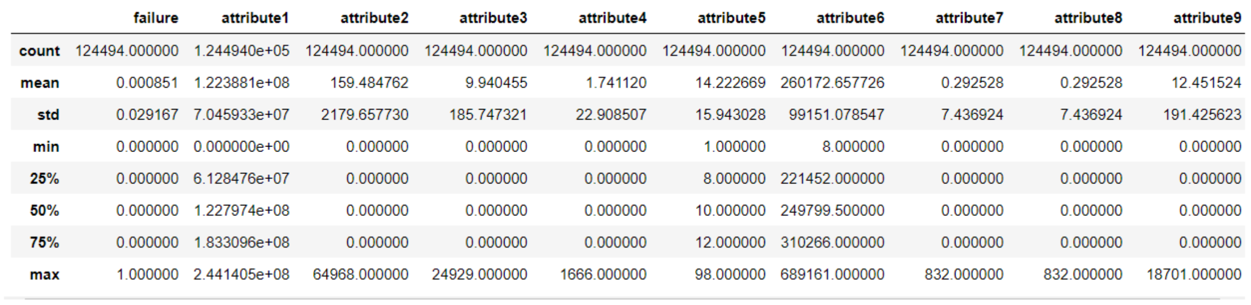
There are 1,169 unique device id's, and 106 eventually have a failure in the given time range.

Here is what the head of the table looks like:



	date	device	failure	attribute1	attribute2	attribute3	attribute4	attribute5	attribute6	attribute7	attribute8	attribute9
0	2015-01-01	S1F01085	0	215630672	56	0	52	6	407438	0	0	7
1	2015-01-01	S1F0166B	0	61370680	0	3	0	6	403174	0	0	0
2	2015-01-01	S1F01E6Y	0	173295968	0	0	0	12	237394	0	0	0
3	2015-01-01	S1F01JE0	0	79694024	0	0	0	6	410186	0	0	0
4	2015-01-01	S1F01R2B	0	135970480	0	0	0	15	313173	0	0	3

We can get a sense of the dataset by looking at summary statistics:



	failure	attribute1	attribute2	attribute3	attribute4	attribute5	attribute6	attribute7	attribute8	attribute9
count	124494.000000	1.244940e+05	124494.000000	124494.000000	124494.000000	124494.000000	124494.000000	124494.000000	124494.000000	124494.000000
mean	0.000851	1.223881e+08	159.484762	9.940455	1.741120	14.222669	260172.657726	0.292528	0.292528	12.451524
std	0.029167	7.045933e+07	2179.657730	185.747321	22.908507	15.943028	99151.078547	7.436924	7.436924	191.425623
min	0.000000	0.000000e+00	0.000000	0.000000	0.000000	1.000000	8.000000	0.000000	0.000000	0.000000
25%	0.000000	6.128476e+07	0.000000	0.000000	0.000000	8.000000	221452.000000	0.000000	0.000000	0.000000
50%	0.000000	1.227974e+08	0.000000	0.000000	0.000000	10.000000	249799.500000	0.000000	0.000000	0.000000
75%	0.000000	1.833096e+08	0.000000	0.000000	0.000000	12.000000	310266.000000	0.000000	0.000000	0.000000
max	1.000000	2.441405e+08	64968.000000	24929.000000	1666.000000	98.000000	689161.000000	832.000000	832.000000	18701.000000

The means for attributes 1 and 6 are relatively large, the minimum is 0 for most of the attributes, and the maximum is relatively high for attributes 1 and 6.

3 Exploratory Data Analysis

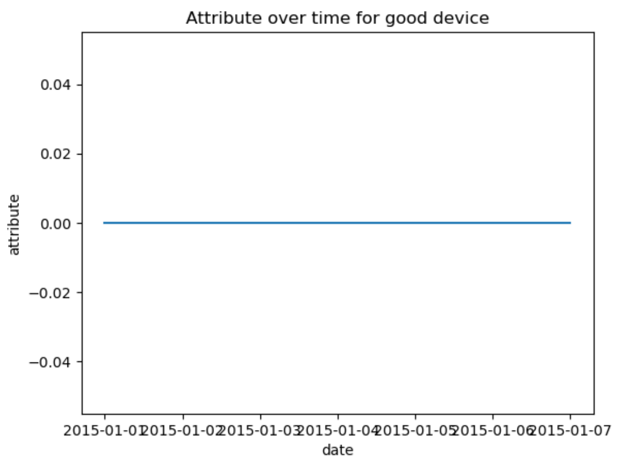
3.1 Methodology

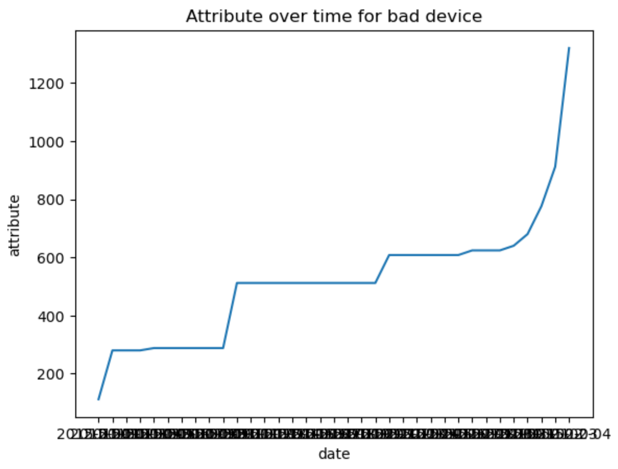
I want to see if there are any differences in the trajectories of the attributes between the good and bad devices. I can plot the trajectories of a given attribute for the good devices and compare these with the trajectories of the same given attribute for the bad devices. This is done by looking at the qualitative characteristics of the graphs of trajectories.

3.2 Results and Evaluation

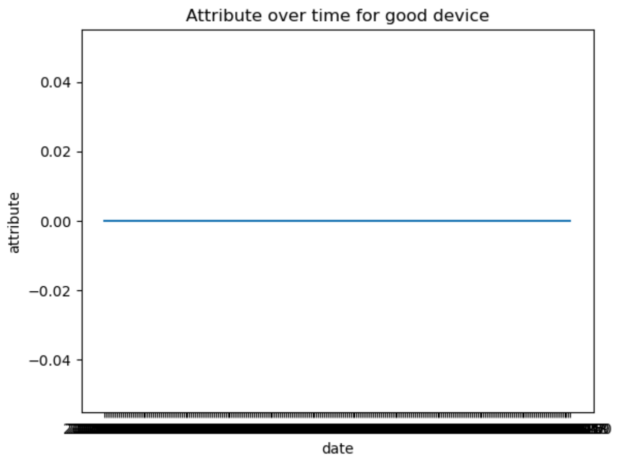
After comparing the graphs of the trajectories of attributes between the good and bad devices, I found that attributes 2, 4, 7, and 8 had noticeably different trajectories between the good and bad devices.

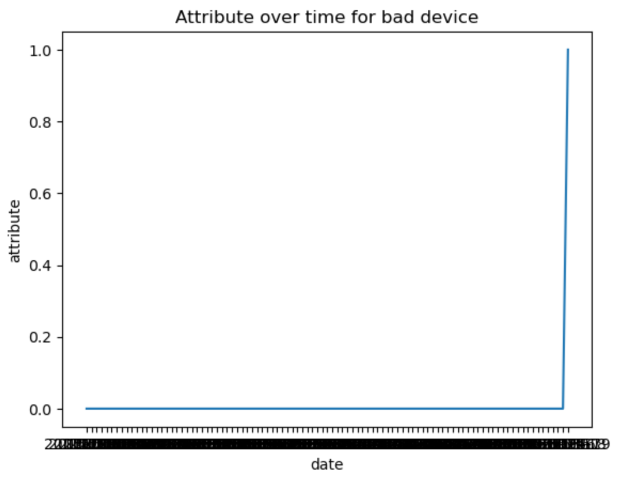
For attribute 2, the good devices had mostly constant graphs–horizontal lines–and the bad devices had either constant graphs or graphs that increased over time. For example:



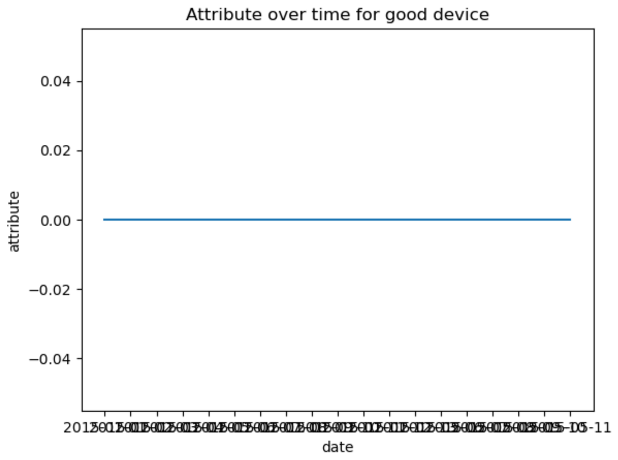


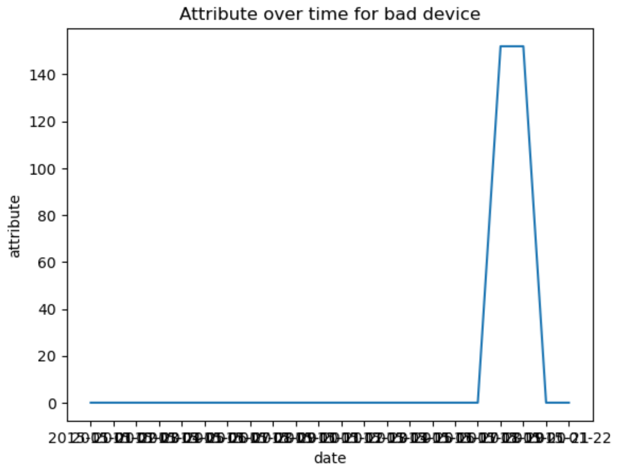
For attribute 4, the good devices had mostly constant graphs while the bad devices had either constant graphs or graphs that increased over time–many of them graphs with abrupt increases near the end. For example:





For attribute 7, the good devices had mostly graphs that stayed flat at 0 while the bad devices had graphs that were either flat at 0 or had abrupt sharp increases near the end of its trajectory–sometimes also suddenly crashing back down. For example:





For attribute 8, the result was similar to that for attribute 7. It turns out that attribute 7 is identical to attribute 8; so we will ignore attribute 8.

For attributes 1, 3, 5, 6, and 9, there was no noticeable difference in graphs between the good and bad devices. This suggests using only attributes 2, 4, and 7 in building our prediction model in the next section. More specifically, for a given device, I took the standard deviation of the device's time series data for attribute 2, the standard deviation of the device's time series data for attribute 4, etc. The reason for taking the standard deviation is to capture the difference between a constant graph and a graph that increases over time. The normal devices tend to have mostly constant graphs, and the faulty devices tend to have more increasing graphs than the normal devices do. Taking the standard deviations of each attribute for each device gives a table with devices as rows and 3 columns consisting of the standard deviations for attributes 2, 4, and 7. Adding a final column to indicate the device as 'failure' or not gives a table with 4 columns. This is the dataframe I used for the classification problem in the next section.

There are 106 faulty devices out of 1,168 devices considered in the dataframe, which is 9% of the devices. The training dataset has 848 records for the majority class and 86 records for the minority class.

4 Classification

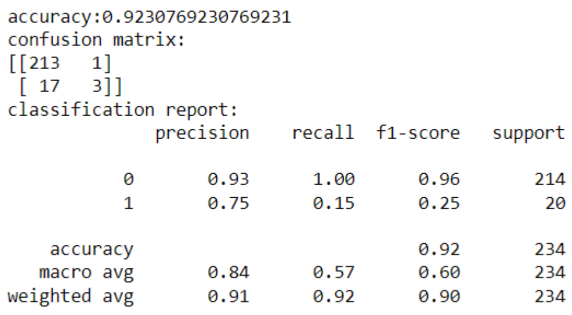
4.1 Methodology

Here, I would like to use classification models to distinguish between the good devices and the bad devices. I will apply logistic regression, decision tree classifier, random forest, XGBoost classifier, AdaBoost classifier, Gradient Boosting classifier, neural network, k-nearest neighbors, linear SVM, kernel SVM, and naive bayes.

It's not difficult to get a high test accuracy over 91%. However, the metric we care about more is recall because we don't want to overlook any faulty devices. Wrongly predicting faultiness for a good device may not be as costly as overlooking a faulty device. Therefore, we want to prioritize recall over precision; nevertheless, we want to maximize both if possible.

4.2 Results and Evaluation

Performing logistic regression gives the following test accuracy, confusion matrix, and classification report:



```
accuracy:0.9230769230769231
confusion matrix:
[[213   1]
 [ 17   3]]
classification report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.96       214
           1       0.75      0.15      0.25        20

    accuracy                           0.92       234
   macro avg       0.84      0.57      0.60       234
weighted avg       0.91      0.92      0.90       234
```

The precision, recall, and f1-score are high for class 0 but low for class 1. This is not good. We want high scores for both, especially recall. The recall for the minority class is 0.15, meaning only 15% of the faulty devices is captured by the model. Recall for the minority class is how many of the faulty devices the model correctly identifies out of all the faulty devices, and this is what we want to maximize. Precision for minority, on the other hand, is how many of the model's predictions of faultiness are correct.

Here is a summary table of the performance of all the models I considered. For each model, it gives the test accuracy, precision for the failure class (denoted PrecisionF), recall for the failure class (denoted RecallF), and f1-score for the failure class (denoted f1 scoreF):

Model	Test accuracy	PrecisionF	RecallF	f1 scoreF
KNN	91.9%	55%	30%	39%
Logistic regression	92.31%	75%	15%	25%
Linear SVM	92.31%	75%	15%	25%
Kernel SVM	91.03%	44%	20%	28%
Naive Bayes	92.31%	75%	15%	25%
Decision Tree	90.6%	43%	30%	35%
Random Forest	91.03%	46%	30%	36%
XGBoost	91.03%	45%	25%	32%
AdaBoost	91.45%	50%	30%	37%
Gradient Boosting	91.45%	50%	45%	47%
Neural network1	91.45%	50%	20%	29%
Neural network2	91.9%	67%	10%	17%

As you can see, all of the graphs have a high test accuracy of around 91%. Logistic regression, linear SVM, and naive bayes have the highest precision for the failure class, and gradient boosting, adaboost, KNN, decision tree, and random forest have the highest recall for the failure class. The models with the highest f1 score for the failure class are gradient boosting, KNN, adaboost, decision tree, and random forest.

Recall that the faulty devices make up only 9% of the devices so that the dataset is imbalanced. One way to handle this is to give more weight to the minority class when training our model. However, there is another way to handle imbalanced datasets–oversampling or undersampling. Oversampling balances the dataset by adding copies of the minority class until we reach a balance–in our case, 848 training data points of each class. Undersampling balanced the dataset by throwing out members of the majority class until we reach a balance–in our case, 86 training data points of each class.

Here are the results of doing oversampling for the winners of the previous competition–KNN, gradient boosting, adaboost, decision tree, and random forest:

Model	Test accuracy	PrecisionF	RecallF	f1 scoreF
KNN	90.17%	36%	20%	26%
Decision Tree	90.6%	42%	25%	31%
Random Forest	90.6%	43%	30%	35%
AdaBoost	91.03%	47%	40%	43%
Gradient Boosting	91.9%	54%	35%	42%

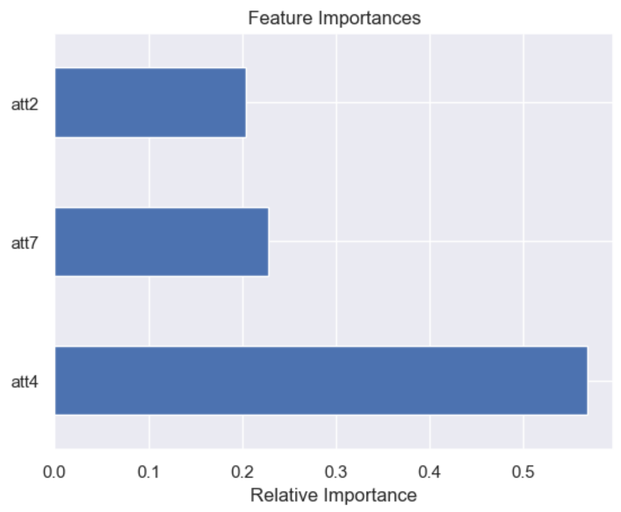
Gradient boosting has the highest precisionF while AdaBoost has the highest recallF.

Here are the results of doing undersampling for the same set of models–KNN, gradient boosting, adaboost, decision tree, and random forest:

Model	Test accuracy	PrecisionF	RecallF	f1 scoreF
KNN	91.45%	50%	60%	55%
Decision Tree	91.03%	48%	60%	53%
Random Forest	91.03%	48%	60%	53%
AdaBoost	90.6%	46%	60%	52%
Gradient Boosting	90.6%	46%	55%	50%

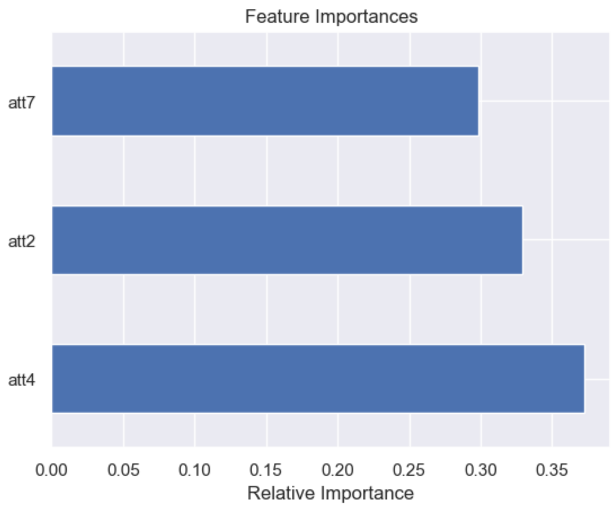
Comparing this table with the previous table shows that undersampling is superior to oversampling in our situation. The f1 scoreF's are all higher than those in the previous table, and the recallF scores are all higher too. Four models have a highest recallF score of 60%, and the three models with highest precisionF as well are KNN, decision tree, and random forest.

Here are the feature importances from the decision tree:



As you can see, the most important feature is att4–the standard deviation of attribute 4.

Here are the feature importances from the random forest:



The most important features are att4, att2, att7–in that order.

In order to improve the recallF score even more, I tried using the sign of the standard deviation for each of attributes 2, 4, and 7 rather than just the standard deviations themselves; a sign of 0 indicates a constant graph, and a sign of 1 indicates an increasing graph. Doing this, I was able to achieve a recallF score of 65%.

All of the following models give an accuracy of 90.6%, a precisionF of 46%, a recallF of 65%, and an f1 scoreF of 54%:

naive bayes, logistic regression with balanced weights, logistic regression with oversampling, adaboost with oversampling, gradient boosting with oversampling, random forest with oversampling, decision tree with oversampling, logistic regression with undersampling, adaboost with undersampling, gradient boosting with undersampling, random forest with undersampling, decision tree with undersampling.

5 Conclusion

Of all the models I've applied using standard deviation of attributes 2, 4, and 7, the KNN with undersampling, the decision tree with undersampling, and the random forest with undersampling were the best models when taking recall for the failure class as priority; the KNN with undersampling had a precision score of 50% for the failure class and a recall score of 60% for the failure class, while the decision tree and random forest both had a precision score of 48% for the failure class and a recall score of 60% for the failure class.

The most important feature was att4. There were differences in the time series graphs of attributes 2, 4, and 7 between the faulty devices and non-faulty devices such that the non-faulty devices tended to have constant graphs while the faulty devices tended to have many increasing graphs as well as constant graphs. These significant attributes were singled out to train machine learning models to predict the failure of devices.

Finally, I was able to achieve a recall score of 65% for the failure class while maintaining a precision score of 46% for the failure class by using the sign of the standard deviations of attributes 2, 4, and 7, respectively. Many of the models converged on an accuracy of 90.6%, a precisionF of 46%, a recallF of 65%, and an f1 scoreF of 54%.