Classification Methods for Anomaly Detection and Prediction

Yang Miaoqing A0263711L, Zhang Xiaoyan A0259868B,
Zhu Chuanlong A0263984M, Guo Shiping A0260014Y

April 14, 2023

Abstract

This project aimed to implement and evaluate failure classification or prediction methods for two given datasets. Four classification methods were studied, including Maximum Likelihood Gaussian Classifier, Decision tree, Adaboost, Mahalanobis and PCA+K-mean. In this report, we present the performance evaluation results for all the studied failure prediction methods according to six classification metrics and diagnostic data graphs and performance results were generated for each method. The results indicate that not all algorithms are suitable for this failure prediction. These findings can provide valuable insights for improving failure prediction in similar applications.

1 Introduction

In this project, we implemented and evaluated failure classification or prediction methods for two given datasets. We followed the instructions provided in the Google Colab skeleton file and each group member implemented one of the proposed classification methods. Specifically, we studied four classification methods: Maximum Likelihood Gaussian Classifier, Decision tree, Adaboost, Mahalanobis and PCA+K-mean.

One group member implemented the Maximum Likelihood Gaussian Classifier method as it was covered in Lecture 2. The rest of the group members implemented different classification methods. We generated diagnostic data graphs and performance results for each method.

In this report, we present the performance evaluation results for all the studied failure prediction methods. We tabulated six classification metrics, including Accuracy, GMean, Recall, Specificity, Precision, and F-score. The results will be discussed and compared in the following sections.

2 Classification Methods

2.1 Maximum Likelihood Gaussian Classifier

2.1.1 Concept and Operational Principles

At every time point, there is only one set of measurement data for classification, so according to the definition, this situation belongs to single measurement. The purpose of classifier in this situation is to determine whether a failure occurs based on the measurement data, so it is a binary classification problem, and it is assumed that the distribution of each class follows a Gaussian distribution. Denote the 2 possible target classes as $\omega_1, \omega_2 \in \Omega$, and the N-dim real-valued event feature vector as x, which belongs to class j^{th} with probability $P(\omega_j)$. Assume

that for class j, x has a Gaussian distribution with mean vector $\mu_j = E_j[x]$ and covariance matrix $\Sigma_j = E_j[(x-\mu_j)(x-\mu_j)^T]$, where $E_j[\bullet]$ denotes ensemble average over class j and superscript T denotes transpose. For class j, the probability density function of its Gaussian distribution is[1]:

$$p(x|\omega_j) = \frac{1}{(2\pi)^{N/2}|\Sigma_j|^{1/2}} \exp[-\frac{1}{2}(x-\mu_j)^T {\Sigma_j}^{-1}(x-\mu_j)]$$

Assuming we have a binary classification training dataset and the goal is to predict the class of new data. In other words, for a given new data x, we want to estimate $p(\omega_1|x)$ and $p(\omega_2|x)$ and assign it to the class with the highest probability. In this case, we need to use Bayes' theorem:

$$p(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{\sum_j p(x|\omega_j)P(\omega_j)}$$

Given that $p(x|\omega_j)$ follows a Gaussian distribution and $P(\omega_j)$ is the prior probability of class j, for the purpose of classification, we can ignore the denominator normalization constant and use the MAP decision rule:

$$C(x) = \arg\max_{j=1,2} p(x|\omega_j)P(\omega_j)$$

If the prior probabilities are the same, it can be simplified to the ML decision rule:

$$C(x) = \arg\max_{j=1,2} p(x|\omega_j)$$

According to the lecture note, the ML decision formula can be simplified as follow:

$$\begin{aligned} C(x) &= \arg\max_{j=1,2} p(x|\omega_j) = \arg\max_{j=1,2} \tfrac{1}{\pi^N|\Sigma_j|} \exp[-(x-\mu_j)^T {\Sigma_j}^{-1}(x-\mu_j)] \\ &= \arg\min_{j=1,2}[-\log p(x|\omega_j)] = \arg\min_{j=1,2} \log|\Sigma_j| + (x-\mu_j)^T {\Sigma_j}^{-1}(x-\mu_j) \end{aligned}$$

It can be seen that the derivation above is slightly different from that in the lecture notes at the probability density function of the Gaussian distribution, although the simplification does not affect the ML decision result. In this assignment, because the probabilities of the two classes (failure and success) are different, the MAP classifier should be used. However, for comparison purposes, both decision rules have been implemented and compared in this report. (The parameters of Gaussian classifier are all fixed, so it is not discussed)

2.1.2 Performance

Firstly, we assume that 'failure' is positive and 'normal' is negative since we focus on failure detection. The results indicate that both methods have high accuracy. According to the figure 1 and 2, it's clear that the ML method detected much more 'failure', which can be shown in the table 3 and 4, so clearly the precision of ML is low while that of MAP is more satisfying according to table 1 and 2. However, due to the price of low precision, ML method has higher recall rate (sensitivity), because it is more sensitive. Since for most of the time, there is no failure, so it is reasonable that both methods have high specificity.

Now the definition of GMean score and F1 score are not given in the lecture notes, so here we state them first. GMean is the n^{th} root of the product of all True Positive Rates (TPRs) across all classes, where n is the number of classes. TPR represents the proportion of positive instances that are correctly identified by the classifier. The significance of GMean is that it can provide a more accurate evaluation in imbalanced datasets. The F1-score is a

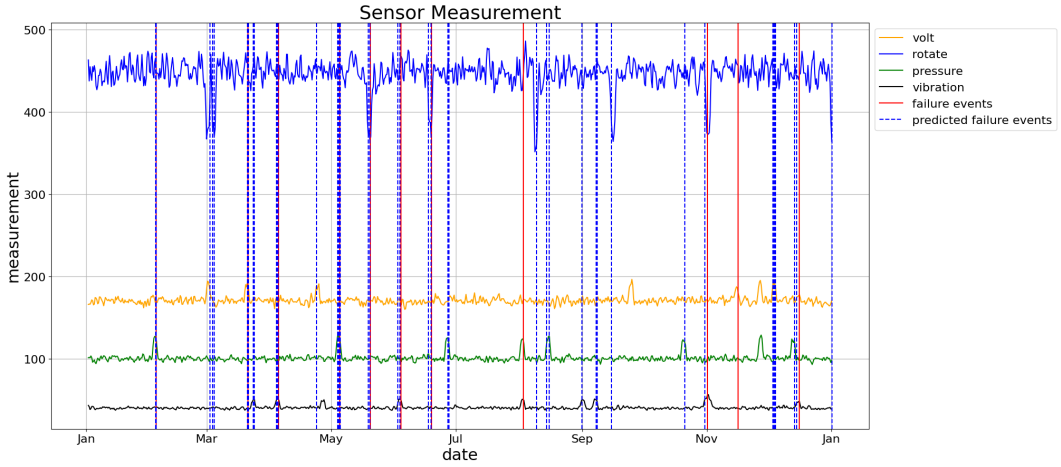

Figure 1: failure prediction result with ML Gaussian classifier

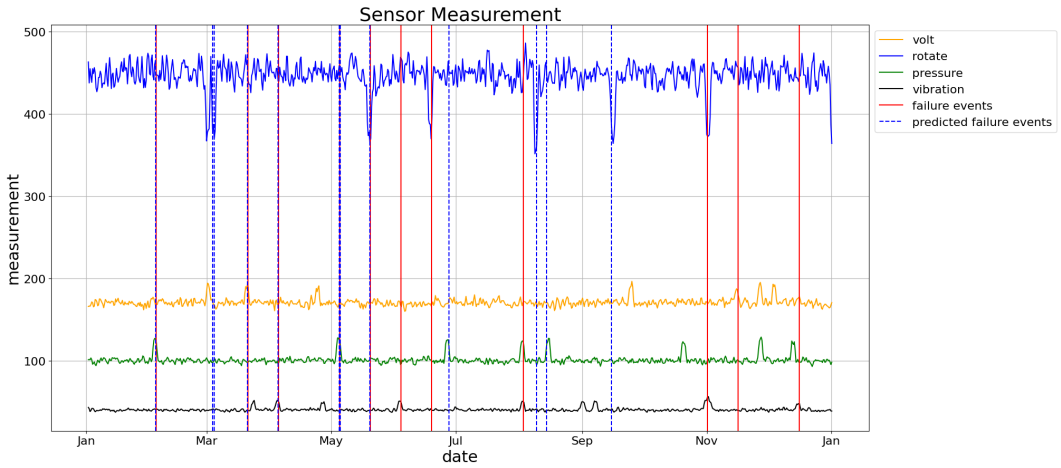

Figure 2: failure prediction result with MAP Gaussian classifier

Metrics	Values
Accuracy	0.954
Recall	0.545
Specificity	0.959
Precision	0.146
F1 Score	0.231
GMean score	0.723

Table 1: ML classification metrics

Metrics	Values
Accuracy	0.985
Recall	0.455
Specificity	0.992
Precision	0.417
F1 Score	0.435
GMean score	0.671

Table 2: MAP classification metrics

	failure	normal	(predicted)
failure	6	5	
normal	35	829	
(true)			

Table 3: ML confusion matrix

	failure	normal	(predicted)
failure	5	6	
normal	7	857	
(true)			

Table 4: MAP confusion matrix

harmonic mean of precision and recall. It provides a balance between the two metrics and is often used when both precision and recall are important. According to table 1 and 2, ML method have higher GMean score because MAP considers the prior while ML does not and the prior deliberately want to treat 2 classes differently. The lack of 'failure' data is because of low happening probability, not because we collect less 'failure' on purpose. So I think we should treat the two classes differently. Finally, MAP method has higher F1 score, which shows better performance in some sense.

What is not shown in the table is that the ML method captures most of failure in the neighbour time point of true failure, however they are categorized as false positive. So if we want to detect most of the failure roughly, we should use ML method instead of MAP method.

2.2 Decision Tree

Decision tree is a basic classification and regression method. Decision tree learning usually consists of three steps: feature selection, tree generation and tree pruning[2].

2.2.1 Concept and Operational Principles

A complete decision tree consists of three parts: the root node (the node at the top of the tree); Leaf node (the node at the bottom of the tree, which is the decision result); Internal nodes (nodes other than top and bottom) Each internal node in the tree represents a test on an feature, each branch represents a test output, and each leaf node represents a category.

The process of generating a decision tree is the recursive process of splitting data over and over again, each time splitting, trying to keep the same kind of data on one side of the tree, in order to improve purity.

The measure of information clutter is entropy. Entropy stands for uncertainty, and the greater the uncertainty, the greater the entropy. The maximum level of entropy or disorder is given by 1 and minimum entropy is given by a value 0. The formula of entropy is:

$$E = \sum_{i=0}^{n} p_i \log_2 p_i$$

where the pi is the probability of randomly selecting an example in class i.

Now essentially what a Decision Tree does to determine the root node is to calculate the entropy for each variable and its potential splits. For this we have to calculate a potential split from each variable, calculate the average entropy across both or all the nodes and then the change in entropy vis a vis the parent node. This change in entropy is termed Information Gain and represents how much information a feature provides for the target variable.During the

Metrics	Values
Accuracy	0.981
Recall	0.545
Specificity	0.986
Precision	0.333
F1 Score	0.414
GMean score	0.733

Table 5: Decision tree classification metrics

construction of the decision tree, the attribute with the maximum information gain is selected as the splitting condition, so that the maximum category gain can be obtained during the test on each non-leaf node, and the entropy of the data set after classification is minimized. Such a processing method makes the average depth of the tree smaller, thus effectively improving the classification efficiency.

$$Informationgain = Entropy_(parent) - Entropy_(children)$$

The other way of splitting a decision tree is via the Gini Index. The Entropy and Information Gain method focuses on purity and impurity in a node. The Gini Index or Impurity measures the probability for a random instance being misclassified when chosen randomly. The lower the Gini Index, the better the lower the likelihood of misclassification.

$$Gini = \sum_{i=1}^{j} p_i^2$$

where j represents the no. of classes in the target variable and pi represents the ratio of Pass/Total no. of observations in node.The Gini index has a maximum impurity is 0.5 and maximum purity is 0.

2.2.2 Parameter Settings

In this project, sklearn.tree.DecisionTreeClassifier() is used. The main parameters include: criterion, max_depth, and splitter.

The parameter "criterion" can be set as "gini" or "entropy'. Entropy is better when there are fewer features, entropy works better when there are more features and gini works better, but this is not always the case. Gini criterion is chosen. "max_depth" prevents overfitting by limiting the maximum depth of the tree. It is set as "None ", that is, the maximum depth is not limited. "splitter" can be set to "best" or "random". The first is to find the best splitter feature of all the features and the second is to find the best among some of the features. The default "best" is suitable for when the sample size is small, and if the sample data size is very large, then the decision tree will construct the recommendation "random". It is set as "random" here.

2.2.3 Performance

The decision tree prediction result using is shown in the figure and table. The results indicate that the model has a high accuracy score of 0.981. However, the recall score of 0.545 suggests that the model is not very good at identifying the true failure instances in the data. On the other hand, the specificity score of 0.986 suggests that the model is very good at identifying the true normal instances in the data. The precision score of 0.333 indicates that when the model predicts that an instance is a failure, it is only correct 0.333 of the time. The F1 score of 0.414

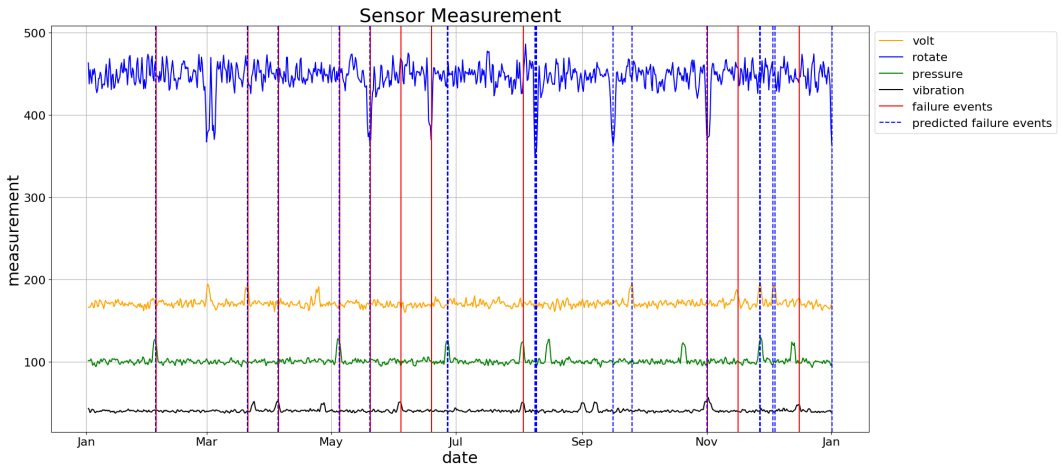

Figure 3: Actual and predicted failures of decision tree

is a harmonic mean of precision and recall, it shows that the model has poor performance on detecting failures. The GMean score of 0.733 suggests that the model's performance is somewhere in between its recall and specificity scores, and indicates that the model is fairly good at balancing the detection of normal and failure instances. Overall, the model's high accuracy is promising, but its low recall and precision scores suggest that it could benefit from further refinement and tuning.

A possible reason why a decision tree model may not be able to predict the actual failure instances accurately is that the decision tree model may not be able to capture the complex relationships between the features and the target variable. Decision trees are simple models that make predictions based on a set of if-else statements, and may struggle to capture more subtle or nuanced relationships between the features and the target variable. In this project, the result is related to the trend of the feature over time series, but the decision tree is only based on the independent value of the feature.

2.3 Adaboost

2.3.1 Concept and Operational Principles

To introduce AdaBoost, the basic boosting algorithm should be introduced first. Boosting algorithm is the process of upgrading a "weak learning algorithm" to a "strong learning algorithm". The main idea is that two heads are better than one. The boosting algorithm involves two parts, the additive model and the forward stepwise algorithm. The additive model means that the strong classifier is made by linearly adding a series of weak classifiers. Forward stepping algorithm means that during the training process, the classifier produced in the next iteration is trained based on the previous one. Boosting algorithms are classified using different loss functions, and AdaBoost is a Boosting algorithm with an exponential loss loss function.

The AdaBoost algorithm is implemented by changing the data distribution of the samples. AdaBoost determines whether each training sample is correctly classified, decreases its weight for correctly classified samples, and increases its weight for misclassified samples[3]. Based on the classification accuracy obtained in the last training, the weight of each sample in this training is determined. The new dataset with the modified weights is then passed to the next layer of the classifier for training. The advantage of this is that by dynamically weighting the training samples in each round, the training can be focused on the hard-to-

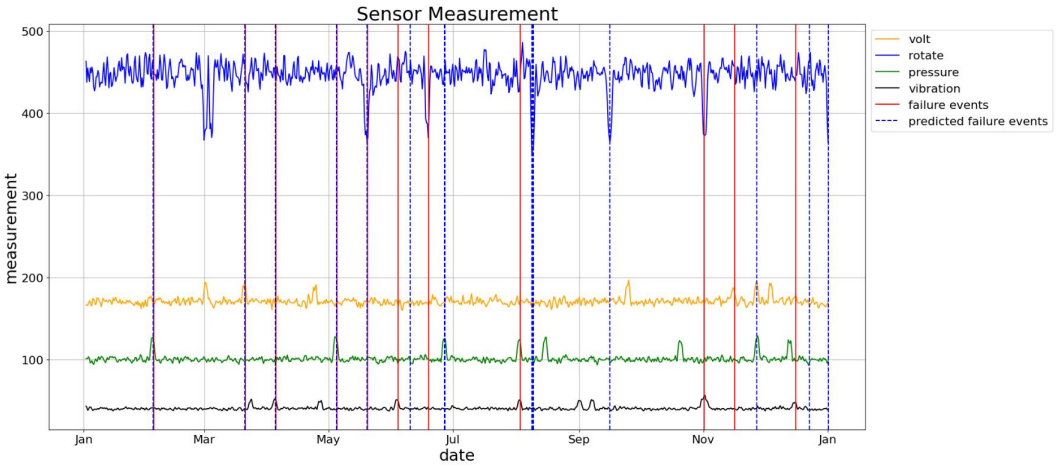

Figure 4: Actual and predicted failures of AdaBoost algorithm

classify samples, and the final combination of weak classifiers obtained is more likely to yield higher classification accuracy. We can use D_{k+1} to represent the set of weights of the samples in the (k+1)th round of training, which is

$$D_{k+1} = (w_{k+1,1}, w_{k+1,2}, ..., w_{k+1,N})$$

where $w_{k+1,1}$ represents the weight of the first sample in the (k+1)th round and $w_{k+1,N}$ represents the weight of the Nth sample in the (k+1)th round. The sample weight in the (k+1)th round is based on the weight of that sample in the kth round and the accuracy of the kth classifier, which is given by

$$w_{k+1,i} = \frac{w_{k,i} exp(-\alpha_k y_i G_k(x_i))}{Z_k}, i = 1, 2, ..., N$$

where Z_k is the normalization factor, which is $\sum (w_{k,i} exp(-\alpha_k y_i G_k(x_i)))$.

For parameter settings part, sklearn.ensemble.AdaBoostClassifier() is used in this method. The notable parameters are n_estimators, algorithm and criterion. Parameter "n_estimator" refers to the number of weak classifiers. If n_estimators is too small, it is easy to underfit, but if n_estimators is too large, it is easy to overfit. A moderate value of 50 is choosed in the experiment. Parameter "algorithm" refers to the classification algorithm, and SAMME and SAMME.R is optional. The difference between them is the measure of weak classifier weights. SAMME uses the classification effect of the classifier as the weak classifier weight, while SAMME. R uses the magnitude of the predicted probability of classifying the sample set as the weak classifier weight. Since SAMME.R uses continuous values of probability measures, iterations are generally faster than SAMME, so SAMME.R is used. Parameter "criterion" refers to the feature selection method, and gini (Gini coefficient) and entropy (information gain) are optional. Gini is the mean squared difference, and entropy is the sum of the absolute values of the differences from the mean. Gini is used in the experiment because it is more accurate and easy to calculate.

2.3.2 Performance

In the anomaly detection and prediction task, the failure prediction result can be seen in figure 4. Observe table 6 and table 7 and we can see that AdaBoost realizes a high accuracy value, because the training error decreases exponentially in each iteration. And AdaBoost

Metrics	Values
Accuracy	0.982
Recall	0.455
Specificity	0.988
Precision	0.333
F1 Score	0.385
GMean score	0.670

Table 6: AdaBoost classification metrics

	failure	normal	(predicted)
failure	5	6	
normal	10	854	
(true)			

Table 7: AdaBoost confusion matrix

has high recall and precision values, and therefore high F1-score and GMean values, because AdaBoost increases the weights of reliable weak classifiers in each iteration. AdaBoost also has the advantage of not having to do feature filtering and not worrying about overfitting. The disadvantage of AdaBoost is that it is sensitive to anomalous samples, and the weights of the anomalous samples may increase in each iteration, which may affect the prediction accuracy to some extent. In each iteration, a weak classifier is trained for each sample in the sample set, and each sample has many features, so the computational effort to train the optimal weak classifier from the large number of features increases, which causes AdaBoost to become time-consuming. AdaBoost is very flexible, and in this experiment the weak classifier is built by using the default decision tree model of AdaBoost, but other regression classification models can also be used to build it.

2.4 Mahalanobis+PCA+K-mean

2.4.1 Concept and Operational Principles

Mahalanobis distance: is a statistical measure that considers correlations between variables to determine the distance between a point and a distribution. This property makes it particularly useful for high-dimensional datasets where variables may be interdependent. In anomaly detection, Mahalanobis Distance can identify unusual data points that are distant from the distribution of normal data points by comparing the distance to a threshold value. To use Mahalanobis Distance in anomaly detection, the covariance matrix and mean vector of the normal data points must be computed to calculate the distance for each data point. The formula for Mahalanobis Distance is as follows: Given a probability distribution Q on R_N and positive covariance matrix S, the Mahalanobis distance of a point$\vec{x} = (x_1, x_2, x_3 x_N)_T$from Q is

$$d_M(\vec{x}, Q) = \sqrt{((\vec{x} - \vec{u})^T S^(- 1)(\vec{x} - \vec{u}))}$$

Once have computed the Mahalanobis Distance for each data point, can set a threshold value based on the distribution of distances. Data points with distances greater than the threshold are flagged as anomalies.

Principal Component Analysis: It is a statistical technique used for reducing the dimensionality of high-dimensional data by transforming it into a lower-dimensional space while retaining as much of the original information as possible. It works by identifying the directions in the data that explain the most variance, known as the principal components. These

principal components are linear combinations of the original variables, and they are ordered in terms of the amount of variance they explain in the data. The first principal component explains the most variance, followed by the second, third, and so on. In simulation, I tried components = 2 and 3, the following is the distribution under these two conditions: K-means:

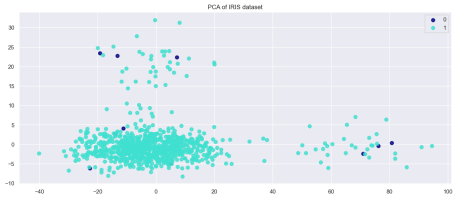

Figure 5: PCA components = 2

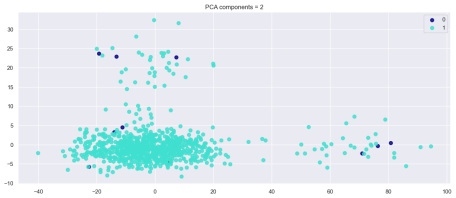

Figure 6: PCA components = 3

is a popular unsupervised machine learning algorithm for clustering analysis, which partitions a dataset into k clusters by assigning data points to their nearest centroid. It is efficient with large datasets and is widely used in various applications. In anomaly detection, K-means can identify clusters of normal data points and flag anomalies as data points that deviate significantly from these clusters.

2.4.2 Mahalanobis Distance application

Relationship volt rotate pressure vibration Figure 7 shows the relationship among the four factors. We can find that the some anomaly samples are messed with normal samples together, which is very hard to classify. By viewing the anomaly plot, We can found the four variables have a certain correlation. Mahalanobis distance is a good method to identify this correlation. The result is showed in figure 8. Compare with Figrue 7, Mahalanobis distance is very helpful. Next step, calculated the threshold and do classification. The result is showed in Figure 9. In

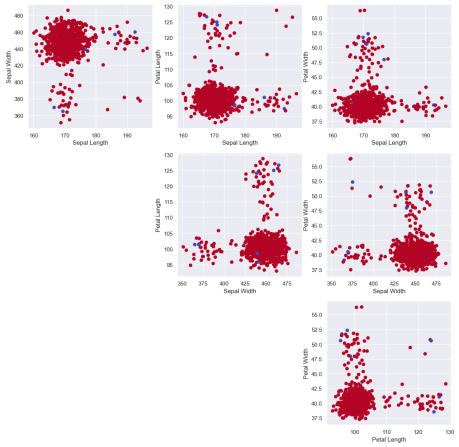

Figure 7: Four factors relationship

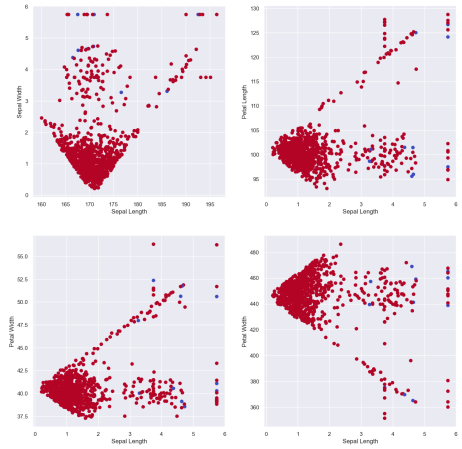

Figure 8: Four factors with mahal distance

the result, I found there are some continuous anomaly samples. I combine these continuous anomaly samples together as one anomaly sample which is more reasonable and calculate the result.

2.4.3 Mahalanobis Distance+PCA+Kmean

PCA is a very powerful tool to reduce the dimension of variables so I want to try reduce the dimension to see whether it is helpful to classification. After reducing dimension. The

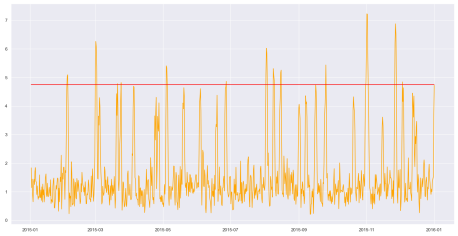

Figure 9: Mahalanobis and threshold

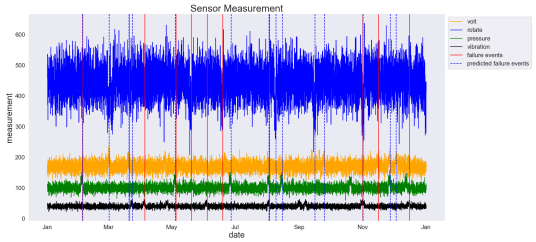

Figure 10: Mahala Distance predication result

Figure 11 shows the anomaly samples, it is hard to classify. The methods mentioned above are supervised learning. Here I tried to use unsupervised K-Mean to illustrate my suppose [4]. The predication result is shown in Figure 12. The result is worse than Mahalanobis distance and suppose is approved.

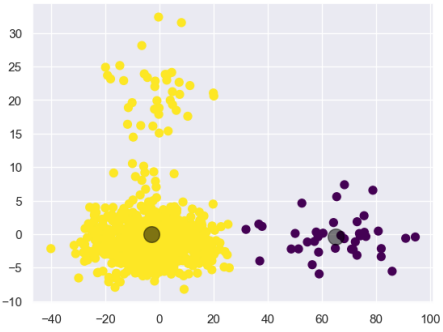

Figure 11: PCA+Kmean result

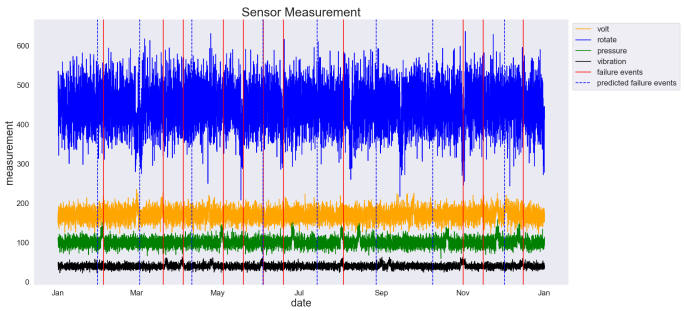

Figure 12: Mahala+PCA+Kmean result

3 Window Size and Step size

When transferring the original dataset by using a sliding window and calculating the rolling mean, the choice of window size and step size is crucial to ensure that the transformed data captures the desired patterns and characteristics.

The window size determines the number of data points that are used to calculate the rolling mean. A larger window size will result in a smoother rolling mean but may miss short-term fluctuations, while a smaller window size will capture short-term fluctuations but may result in a noisy rolling mean. The choice of window size depends on the frequency and duration of the patterns you want to capture. The time span of the whole data is one year, and the fluctuation of the data is hourly, so we choose 20 hours, that is, nearly one day's data, as the window size.

The step size determines the amount of overlap between consecutive windows. A smaller step size will result in more overlap and smoother transitions between windows but may increase computational complexity and memory requirements. A larger step size will reduce computational complexity and memory requirements but may miss short-term fluctuations between windows. The choice of step size depends on the frequency and duration of the patterns you want to capture, as well as the computational and memory resources available. We set the step size as 10.

The experimental results also show that We get good estimates based on the window size and step size settings.

4 Comparation and analysis

Method	Accuracy	Gmean	Recall	Specificity	Precision	F-score
Decision tree	0.981	0.733	0.545	0.986	0.333	0.414
Gaussian classifier(ML)	0.954	0.723	0.545	0.959	0.146	0.231
Gaussian classifier(MAP)	0.985	0.671	0.455	0.992	0.417	0.435
Adaboost	0.982	0.670	0.455	0.988	0.333	0.385
Mahalanobis	0.978	0.519	0.273	0.987	0.214	0.240
PCA+K-mean	0.981	0.300	0.091	0.992	0.125	0.105

Table 8: Classification methods comparison

All classification algorithms have high accuracy and specificity.
Decision tree: high accuracy and specificity, but low precision and F-score

Gaussian classifier(ML): low precision, F-score, but high GMean and recall

Gaussian classifier(MAP): highest precision, F-score, and medium GMean, recall

Adaboost: high accuracy and specificity, but low precision, recall, F-score, and G-mean

Mahalanobis distance: lowest performance across all metrics except specificity

PCA+K-means: low performance across all metrics except specificity

Based on these results, Gaussian classifier with MAP decision has the highest precision, F-score, and G-mean score is relatively high among most of methods, indicating it performs well in identifying anomalies with high confidence, although it has medium recall and GMean compared to other methods. In comparison, Gaussian classifier with ML decision also has good performance, which is more sensitive to anomalies at the cost of low precision and F-score.

Decision tree and Adaboost have high accuracy and specificity, but lower precision, recall, and F-score.Because AdaBoost algorithm uses decision tree as the model for the weak classifier, it has a slightly better performance in terms of accuracy.

Mahalanobis distance is affected by the instability of the covariance matrix, PCA+K-means tends to be locally optimal when the data set is large as well as requires pre-set K values and is sensitive to the selection of the first K centroids, so their performance on Gmean and F1-score is inferior to that of other methods.They may not be the best methods for this problem.

It's also worth noting that Decision tree, Gaussian classifier and Adaboost are supervised learning methods, while Mahalanobis distance and PCA+K-means are unsupervised learning methods. Therefore, depending on the availability of labeled data, the choice of method also is influenced by the need for supervised or unsupervised learning.

References

[1] L. Zhu, "Application of naive bayesian classifier based on gaussian distribution in plant recognition," in *2021 IEEE 12th International Conference on Software Engineering and Service Science (ICSESS)*, 2021, pp. 135–139.

[2] J. Pärkkä, L. Cluitmans, and M. Ermes, "Personalization algorithm for real-time activity recognition using pda, wireless motion bands, and binary decision tree," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 5, pp. 1211–1215, 2010.

[3] F. Wang, Z. Li, F. He, R. Wang, W. Yu, and F. Nie, "Feature learning viewpoint of adaboost and a new algorithm," *IEEE Access*, vol. 7, pp. 149 890–149 899, 2019.

[4] S. A. Hosseini and H. Ghassemian, "A new hyperspectral image classification approach using fractal dimension of spectral response curve," in *2013 21st Iranian Conference on Electrical Engineering (ICEE)*, 2013, pp. 1–6.