Gas Turbine Engine Fault Detection (Machine Learning 2020 Course)

Igor Usachev [1] **Andrey Zogov** [1] **Soodanbek Kasymaliev** [1]

Abstract

In this report, we present a hybrid approach, based on physics and data-guided software for a Gas Turbine diagnostic algorithm. The said algorithm was implemented under steady-state operating conditions to a two-shaft commercial Gas Turbine engine, where single, double, triple and quadruple part fault scenarios were considered.

1. Introduction

Like other machines, gas turbine engines show the signs of wear and tear over time. This contributes to degradation of engine reliability. The deterioration of gas turbine efficiency has a significant effect on its performance, availability, and lifespan. Therefore, a Gas Turbine part diagnostic system needs to be accurate and reliable to enable a stable, productive, clean, and cost-effective service. Gas Turbine components' health status is defined by health parameters (flow capability, isentropic performance, produced power) that may alter due to degradation of the engines. They may be validated based on a series of measurement deviations (such as pressure, temperature, fuel flow rate and shaft speed) named fault signatures.

The degrading trend of Gas Turbine engines will make their service uneconomical for airline/power plant companies as well as dangerous and untrustworthy for their customers. This brings us to the fact that at a given moment, we need sufficient information on the specific condition of the gas turbine. We can not assess its efficacy and satisfy the criteria for operating maintenance without the corresponding knowledge. Hence, in order to be able to evaluate the engine's current state, we need data. These data can be acquired via simulation. The two most popular modeling approaches are physics-based modeling and data-driven modeling.

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Igor Usachev <Igor.Usachev@skoltech.ru>.

Final Projects of the Machine Learning 2020 Course, Skoltech, Moscow, Russian Federation, 2020.

1.1. Physics-based approach

Physics-based modeling enables simulation of the actions of mechanical, thermodynamic, fluid mechanics, and other equation-resolution structures. Practically, the explanation for using physics-based modeling is that it adapts to a particular model without any exploitation of data. It plays a significant role in certain situations where the technical nature of the device or the working conditions (for instance, high temperature) makes it difficult to set up sensors and gather data.

1.2. Data-driven approach

Data-driven simulation is another method. The shortage of data is a common issue in machine learning. Machine health predictions are difficult due to the limited amount of data available that define malfunction and its growth. Health monitoring of the engine is realized during its exploitation and it only provides the monitoring of potential upcoming fault when the abnormality in the engine's operation is observed. Normally, the exploiting companies do not have a lot of labeled data to make a prescriptive model that can determine the exact reason for the breakage and can guide their actions.

1.3. Hybrid modeling approach

There is also a third method named hybrid modeling, which is essentially a combination of physics-based and data-driven modeling. In this method, the physical model produces data that are then tailored to the machine learning models. A synthesis of two methods makes it easier to address the question even better than simply using a computer simulation. It is useful to use this method when the engineer does not have any labels or has a limited amount of labeled data.

As there is not much data explaining the degraded engine actions and engine malfunctions, hybrid modeling is useful. The key concept is to replicate the faults of the engine using a physical model. We may achieve so by intentionally reducing the output of certain portions of the gas turbine system.

1.4. The Task and Workflow

In this work, we use a machine learning model to predict the health state of the Gas Turbine engine. The data used for training and testing are simulated using a physical model of the Pratt & Whitney JT8D Gas Turbine engine. The Gas Turbine is a complex system consisting of many parts. In this work, we consider the malfunctions of four of the most important elements: Low-Pressure Compressor (LPC), High-Pressure Compressor (HPC), High-Pressure Turbine (HPT), Low-Pressure Turbine (LPT). These parts have the most impact on the overall engine's performance.

The rest of this report is organized as follows. In section 2, we make a thorough literature review on the related topics. Section 3 presents the description of the dataset, preprocessing procedures, and the models that we picked for this work. Experimental results and discussion are given in section 4 and 5 respectively. Finally, conclusions are presented in section 6.

2. Related Work

Comparing with physics-based modeling, data-driven approach requires a high amount of exploitation data. Such algorithms as Support Vector Machines have shown good results in (Allen & Holcomb, 2017; Wong & et al., 2017). It has been used for gas turbine fault detection, where it has shown the accuracy 80% on the test data. This model studies the vibration level to predict the fault in real-time. For this task such algorithms as Random Forest or Gradient Boosting are used as well (Mulewicz & et al., 2017). The main problem of these algorithms is the lack of the data as it is described in the papers.

Another promising algorithm in industrial predictive maintenance is neural network. An interesting application of the neural network to gas turbines is an unsupervised feature learning (Yan & Yu, 2015). In this work, the problem of gas turbine combustion monitoring is observed. The performance of an unsupervised model in comparison with a model with handcrafted features was higher. The same promising result of faults prediction has Long Short-Term Memory neurons (Nataraj). In simple words in all these works researchers use the sets of experimental data to learn a wide variety of the gas turbine behavior.

The problem with abnormal behavior detection, or fault detection, is that it does not determine the reason for such a behavior. The new paradigm in machinery maintenance is prescriptive maintenance. The future model should determine not only the abnormal behavior of the gas turbine or any other equipment but also the reason for such behavior.

The new approach that starting to take place is using physical model simulation results as training data for machine learning algorithms. Such approach is called hybrid modelling approach. The main obvious reason such approach is being utilized is the lack of real data. Indeed, there are lots of data from normal operating gas turbine engines, but since such equipment is expensive, it's profitable in terms of cost for companies to deliberately malfunction the real engine in order to get faulty engine data. Hybrid modelling approach is used, for example, in (Nicolai et al., 2018) for mechanical fault detection. In (Fentaye & et al., 2018) it is used for Gas Turbine fault detection based on Support Vector Machines and Multi-Layer Perceptron.

3. Algorithms and Models

3.1. Dataset Description

The dataset was generated using physical model of Gas Turbine Engine JT8D. The physical model is based on LMS Amesim software. The Gas Turbine simulator takes as input 3 external parameters: fuel flow rate, external temperature, external pressure and as a result of simulation produces 28 features. These features are basically physical parameters of the engine's parts including temperature and pressure in combustion chamber, turbines and compressors revolutions per minute, pressure ratios, temperatures, mass flow rate, exhaust temperature etc. The physical model is quite sensitive to extra-large or extra-low input values so it may stop simulation and give an error. This is why we tried to avoid input parameters surges and to make output as smooth as possible. For all engine's models we have used the same fuel flow profile and to for each model we randomly assigned external operating temperature in range of 290 to 310 Kelvins.

As it was mentioned, we consider engines with malfunctions of four components: HPC, LPC, HPT, and LPT. Malfunction can be modelled by decreasing the efficiency of the corresponding component. For compressor components we considered the decrease of efficiency from 1.0 to 0.95 of initial efficiency with step 0.01. For turbine components we considered the decrease of efficiency from 1.0 to 0.99 of initial efficiency with step 0.005. Since high pressure components affect overall engine's efficiency more, any decrease of efficiency for HPC and HPT is considered as malfunction for these components. Low pressure components overall efficiency slightly, this is why LPT and LPC are considered as defect if their efficiency drop is more than 1%.

Thus, each of the components is either labeled as 0 or 1, where 0 state stands for normal condition, while state 1 states for fault condition. Since we have four components each having state 0 or 1, we have overall 16 possible states of the engine's condition: [LPC_state, HPC_state, HPT_state, LPT_state] (Table 1.). State 0, for example, stands for all four components being in normal condition, while state 15 means that all four components are deteriorated.

Table 1. Target States of the Gas Turbine Engine.

STATE	LPC	HPC	HPT	LPT
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

3.2. Pre-processing

After the generation of dataset from our digital model with 64000 samples, we dropped all the irrelevant features such as features with all unique data and non-sensor data, irrelevant for predictions. This way, we saved only 23 features with sensor data and one target for normal engine performance and 15 faults. Also, we divided our dataset on the stratified train (70%) and test (30%) parts. Standard Scaler was implemented to normalize the data for Logistic Regression.

3.3. Chosen Models

We made sure that our dataset is balanced (Figure 1) and decided to use three well-known machine learning classifiers, namely, Logistic Regression, Random Forest and XGBoost. All these algorithms support multi-class classification.

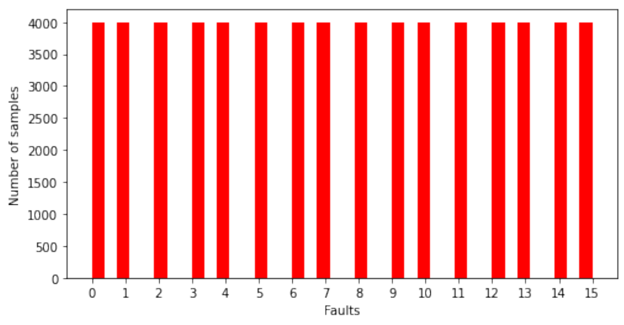

Figure 1. Our Balanced Dataset

We decided to tune the following hyperparameters: *'solver'* and *'max_iter'* for Logistic Regression, *'n_estimators'* for Random Forest and XGBoost Classifiers. The *'solver'* is the algorithm to use in the optimization problem. We chose the default *'lbfgs'* and *'saga'*, which is faster for larger datasets. The *'max_iter'* is maximum number of iterations taken for the solver to converge. *'n_estimators'* is the number of trees in the forest. Equivalent to number of boosting rounds for XGBoost.

We used the Recall score as our key metric because, in the case of engine faults, we were more interested in the early identification of actual faults rather than preventing checks of well functioning engines. The F1 score metric was used to evaluate the correlation between the two scores: Precision and Recall.

4. Experiments and Results

4.1. Finding the best model

All the following computations were made using Google Colab.

As a starting point, we chose the best hyperparameters for our models using GridSearchCV. As a result, Random Forest Classifier (*'n_estimators'=200*, *random_state=0*) got the best Recall score on the test of all other models: 0.86. Logistic Regression Classifier (*'solver'='lbfgs'*, *'max_iter'*=1000, *random_state=0*) got the Recall: 0.51. XGBoost Classifier (*'n_estimators'*=200, *random_state*=0) got the Recall: 0.64.

To improve understanding of the data, we plotted feature importance for Random Forest (Figure 2) and XGBoost (Figure 3). They are different, because these classifiers use different feature importance types, for example, XGBoost has the default 'gain' type and Random Forest calculates 'gini importance'. But 3 of TOP 5 features are the same. They are 'high pressure compressor pressure ratio', 'high pressure compressor revolutions per minute' and 'pressure after turbine'.

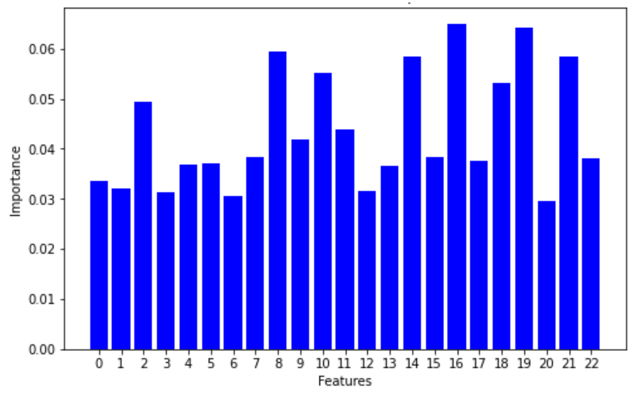

Figure 2. Feature Importance for Random Forest Classifier

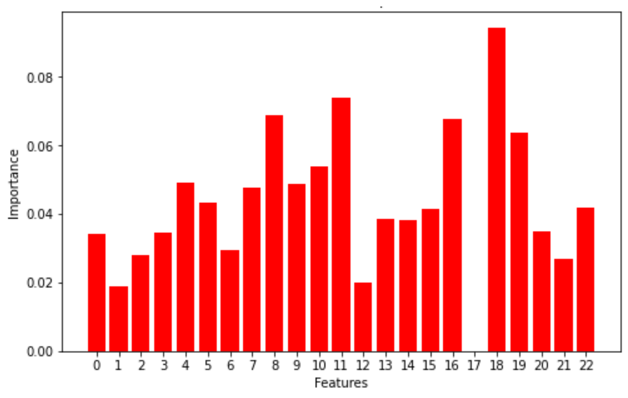

Figure 3. Feature Importance for XGBoost Classifier

4.2. Adding more data

According to the fact, that we were not satisfied with the result, we decided to generate extra data. Thus, we generated a new dataset with 144000 samples. This way, the Recall score on the test of Random Forest Classifier became 0.93.

4.3. Experiments with PyTorch

Next, we would like to try more complex ways to predict engine faults. First, we implemented Autoencoder and got the beautiful picture of our dataset in 2D latent space, as in Figure 4. Second, we added Multi Layer Perceptron with 3 hidden layers with and without Autoencoder, and got the maximum Recall score without Autoencoder at epoch 93/100: 0.96 (Figure 5).

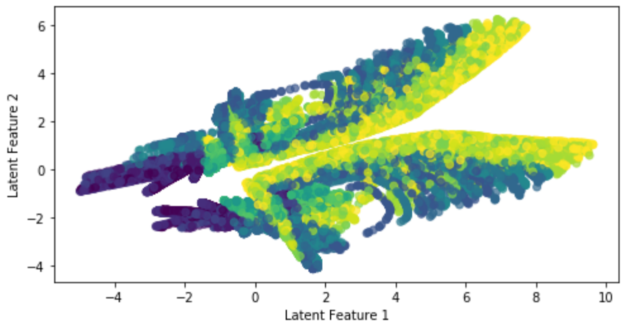

Figure 4. Our Dataset in Latent Space

4.4. Stacking

We also decided to implement Stacking to get even better results.

We combined Extra Trees Classifier (*n_estimators*=200, *random_state*=0), Extra Trees Classifier (*n_estimators*=100, *random_state*=0), Random Forest Classifier (*n_estimators*=200, *random_state*=0), Random Forest Classifier (*n_estimators*=100, *random_state*=0), XGBoost Classifier (*n_estimators*=200, *random_state*=0), XGBoost Classifier (*n_estimators*=100, *random_state*=0), and added XGBoost Classifier (*n_estimators*=100, *random_state*=0) as a final classifier.

Finally, we got the Recall on the test: 0.93.

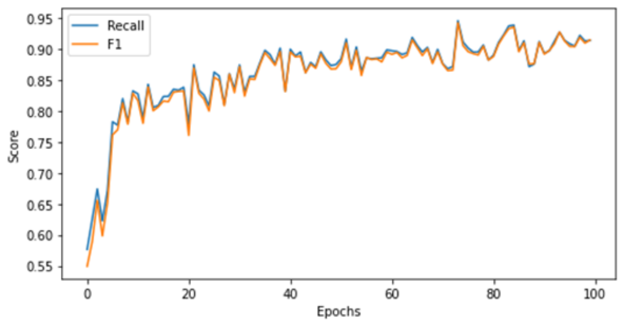

Figure 5. Recall Score on MLP

4.5. Github repo

On the following link you can find our code:
https://github.com/Muroger/ML_project

5. Results and Discussion

The first problem we encountered was the dataset generation. As it was mentioned, the physical model is quite sensitive to extra-low or extra-high input parameters which may lead to model giving an error and interrupting the simulation. Thus, almost two days were spent only to find appropriate input parameters for simulation, taking into account different types of the engine faults. To obtain an appropriate dataset it took about three days of continuous simulation using several computers and Google Colab. Initially, generated datasets were tend to give 100% accuracy and roc_auc until we figured that our data needed to be more variative. Since the first dataset was generated on constant fuel rate, which is does not represent real life scenario, we added more variance by setting complicated fuel flow rate, Only then were we able to move to the next step.

The best result was obtained by MLP, as it was said earlier. Though, it may look quite straightforward, the whole path was not as smooth as it may appear at first glance. It was decided to first try classical machine learning algorithms, then proceed to MLP, and compare the results. The aforementioned algorithms showed modest results, though, it is worth noting that the stacking strategy performed well. It was just behind the MLP in terms of metrics. However, it was twice as slow compared to the said MLP.

6. Conclusion

An architecture for gas turbine deterioration prediction was developed presented as a machine learning model for a two-shaft industrial Gas Turbine engine diagnosis. The test results showed that the proposed algorithms, in general, are capable of diagnosing major component faults with high accuracies.

The issue of preventive maintenance of Gas Turbine Engines is crucial. Early fault detection can save lots of human lives and money.

The proposed MLP model is able to assist experts to determine probable faults before a real accident happening. First, the model can predict all 15 main engine faults. Second, it has very good Recall and F1 scores on the test, 0.96 and 0.95 accordingly.

Also, we found top features responsible for faults: 'HPC pressure ratio', 'HPC revolutions per minute' and 'pressure after turbine'.

References

Allen, C. and Holcomb, C. Gas turbine machinery diagnostics: A brief review and a sample application. *The american society of mechanical engineers*, 2017.

Fentaye, A. D. and et al. Performance-based fault diagnosis of a gas turbine engine using an integrated svm and ann method. *Journal of Power and Energy*, 2018.

Mulewicz, B. and et al. Failures prediction based on performance monitoring of a gas turbine: a binary classifiation approach. *Schedae Informaticae*, 2017.

Nataraj, V. Dynamic modeling of mini sr-30 gas turbine engine.

Nicolai, M., Sobie, C., and Freitas, C. Simulation-driven machine learning: Bearing fault classification. *Mechanical Systems and Signal Processing*, 2018.

Wong, P. K. and et al. Real-time fault diagnosis for gas turbine generator systems using extreme learning machine. *Neurocomputing*, 2017.

Yan, W. and Yu, L. On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach. *Annual conference of the prognostics and health management society*, 2015.

A. Appendix A: Contributions of the Team Members

Igor Usachev

- Coding the main algorithm with Sklearn models
- Preparing the Sections 3, 4, 5 of this report
- Preparing the Video

Andrey Zogov

- Generating the dataset
- Reviewing literature on the topic (5 papers)
- Preparing the GitHub Repo
- Preparing the Sections 1, 2 of this report

Soodanbek Kasymaliev

- Coding the main algorithm with PyTorch neural network
- Implementing Stacking
- Preparing the Presentation

B. Appendix B: Reproducibility Checklist

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: We used the code from Homework 2 to create Autoencoder, MLP and Stacking

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

4. A complete description of the data collection process, including sample size, is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

9. The exact number of evaluation runs is included.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

10. A description of how experiments have been conducted is included.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

12. Clearly defined error bars are included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

13. A description of the computing infrastructure used is included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None