Predictive Maintenance: Capstone Project Report

HarvardX PH125.9x - Data Science: Capstone

Seyed M. Seyedtorabi

2024-Feb-14

I. Introduction

This document is the generated report for the "Chose Your Own Project" capstone of the "HarvardX PH125.9x: Data Science: Capstone" course provided by HarvardX. In the intoducion section of this document, we will go over the motivations and goals behind the project, the chosen data set, problem definition, and the adapted strategy to approach the problem. In later sections, we will go over the exploratory data analysis (EDA), different modelling strategies, model performance comparisons and final results. The document will end with concluding remarks and suggestions of possible improvements for a potential future work.

I.I Motivation

As the previous "Movielens" capstone project was essentially a regression problem, it was desired to have a classification problem for the final capstone project. Working on a **predictive maintenance** project presented a relevant and intellectually stimulating prospect for this capstone. Predictive maintenance is a growing field of interest in data-driven engineering. The goal of predictive maintenance is to monitor the health state of an asset (e.g. a machine, a component thereof, or even a part of the infrastructure) through analysis of the data collected by sensors which monitor the asset of interest. For instance, if vibration levels of a component is under continuous monitoring by a sensor which sends the data to a database, and the machine learning models detect a change point or a trend in these vibration levels, the asset owner can send inspection personnel to investigate the alert, and possibly carry a preemptive maintenance action to avoid total failure of the asset. In this way, the resources of the inspection personnel is used more efficiently, because the periodic and/or reactive maintenance is replaced by the more targeted approach of **predictive maintenance**. In this project, we will work on a data set to classify the state of machines and predict which failure mode they have based on the provided data.

I.II The Data set

The Data set chosen for this project is the **Machine Predictive Maintenance Classification** data set available on Kaggle website [1]. We can import the data set and have a first look after renaming the columns to convenient names:

```
# load data set
df_data <- read.csv("./data/predictive_maintenance.csv")

# rename columns to more convenient names
df_data <- df_data %>%
  rename("product_id" = Product.ID,
         "type" = Type,
         "air_t" = Air.temperature..K.,
         "process_t" = Process.temperature..K.,
         "rpm" = Rotational.speed..rpm.,
         "torque" = Torque..Nm.,
         "wear" = Tool.wear..min.,
         "failure_numeric" = Target,
         "failure_type" = Failure.Type)

# show the head
knitr::kable(head(df_data), row.names = FALSE)
```

UDI	product_id	type	air_t	process_t	rpm	torque	wear	failure_numeric	failure_type
1	M14860	M	298.1	308.6	1551	42.8	0	0	No Failure
2	L47181	L	298.2	308.7	1408	46.3	3	0	No Failure
3	L47182	L	298.1	308.5	1498	49.4	5	0	No Failure
4	L47183	L	298.2	308.6	1433	39.5	7	0	No Failure
5	L47184	L	298.2	308.7	1408	40.0	9	0	No Failure
6	M14865	M	298.1	308.6	1425	41.9	11	0	No Failure

A quick check shows that the first two columns have unique values for every row. Therefore, they are not relevant for any modelling considerations:

```
n_distinct(df_data$UDI)
```

```
## [1] 10000
```

```
n_distinct(df_data$product_id)
```

```
## [1] 10000
```

According to the data set description [1], the **type** column describes the type of the machine and contains possible 3 values of L, M, and H, which stand for low, medium, and high quality, respectively. the **air_t** and **process_t** columns contain respectively the air temperature, and the machine processing temperatures in Kelvin. The **rpm** column stores the angular speed of the machine in revolutions per minute (rpm), while the **torque** columns contains the angular force in Nm. The **wear** column indicates the tool wear in minutes. The last 2 columns are the labels. The column **failure_numeric** contains binary values, 1 for machine failure and 0 for no failure, followed by the **failure_type** columns which indicates the mode of failure. Here, we can see the failure modes:

```
unique(df_data$failure_type)
```

```
## [1] "No Failure"               "Power Failure"
## [3] "Tool Wear Failure"        "Overstrain Failure"
## [5] "Random Failures"          "Heat Dissipation Failure"
```

We can also check if the data set is balanced or not by defining and looking at the distribution in the **failure** column:

```
# crate the failure label column form 0 and 1 ==> no failure and failure
df_data <- df_data %>%
  mutate(failure = ifelse(failure_numeric==1, "Failure", "No Failure"))

table(df_data$failure)
```

```
##
##    Failure No Failure
##        339       9661
```

The data set contains 9661 observations without failure, and only 339 failure observations. This shows that the data set is quite unbalanced and a classification algorithm may be affected by this.

I.III The Objective

The objective in this project is to use different data analysis, data transformation, feature engineering, and machine learning methods to classify and predict the failure of the machines. Although there are 6 distinct categories in the **failure_type** column, one of them is essentially the negative case ("No Failure"), while the other categories are the positive cases. In other words, this classification problem is inherently a binary problem from this point of view. Additionally, it is possible to imagine a real-world scenario where the machine owner is more interested in the health status of the machine rather than the specific failure mode. It is important to mention that in our modelling efforts, we may use the multi-class classification approaches, but if the model predicts e.g. **Heat Dissipation Failure** while the real case is **Overstrain Failure**, we will still count it as a true positive. Since the data set is very unbalance, we will not use **accuracy** as the metric to evaluate the model performance. Instead, we will use **F1-score** for that purpose. As a reminder, the F1-score is calculated through the following formula:

$$F1 = 2 \times \frac{recall \times precision}{recall + precision}$$

Precision and recall are defined by the following formulas:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Additionally, we will use comparisons of receiver operating characteristic (ROC) curves to visualize the model performance. One more metric that will be calculated throughout some of the sections in this report is the true positive rate (TPR) at 10% false positive rate (FPR). This metric can helpful, because in a real-world scenario, the machine owner may require the data scientist to produce a model that does not have more than 10% false alarm rate, while maintaining a high true alarm rate. Please note that TPR is simply another name for recall, and FPR is defined as:

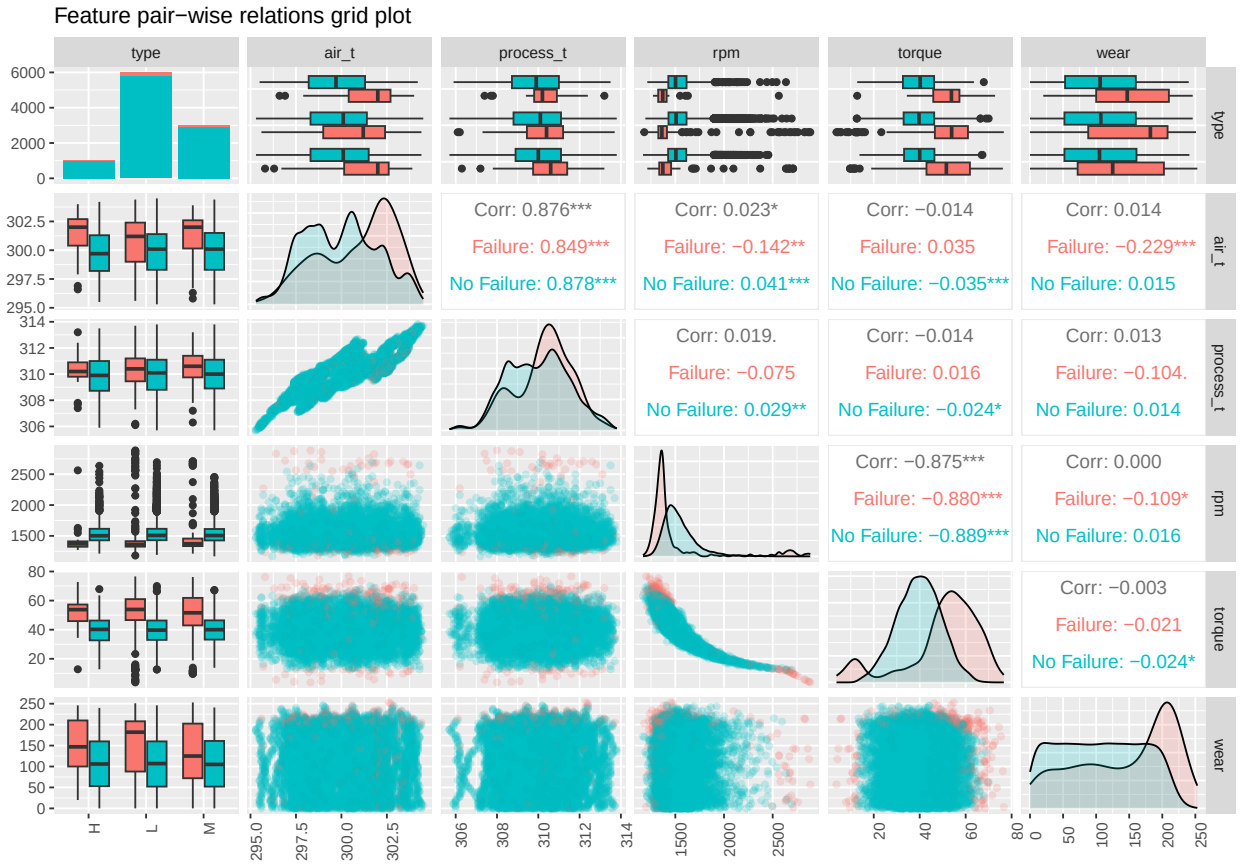
$$FPR = \frac{FP}{TN + FP}$$

As mentioned before, the objective in this project make the binary failure/no failure classification. We will try to achieve an F1-score of **0.95**. We will also use the ROC curves and the TPR at 10% FPR metric as helpful means that guide our modelling efforts.

II. Analysis

Prior to any modelling step, it is crucial to perform some level of EDA and visualization in order to be aware of the relationships and correlations that exist in our data. This will be the starting point in our journey to create a classification model. As the first step, let us create a detailed figure including multiple plots, so that we will see the big picture more clearly:

```
# select columns for pair plot
df_pair <- df_data %>%
  dplyr::select(-any_of(c("UDI", "failure", "failure_numeric",
                          "product_id", "failure_type")))

# create and show the pair plot
pair_plot <- ggpairs(df_pair,
                     mapping=aes(color=df_data$failure),
                     lower = list(continuous = wrap("points", alpha = 0.2),
                                  combo = "box_no_facet"),
                     diag = list(continuous = wrap("densityDiag", alpha = 0.2))) +
  labs(title ="Feature pair-wise relations grid plot") +
  theme(axis.text.x = element_text(angle = 90,))

pair_plot
```

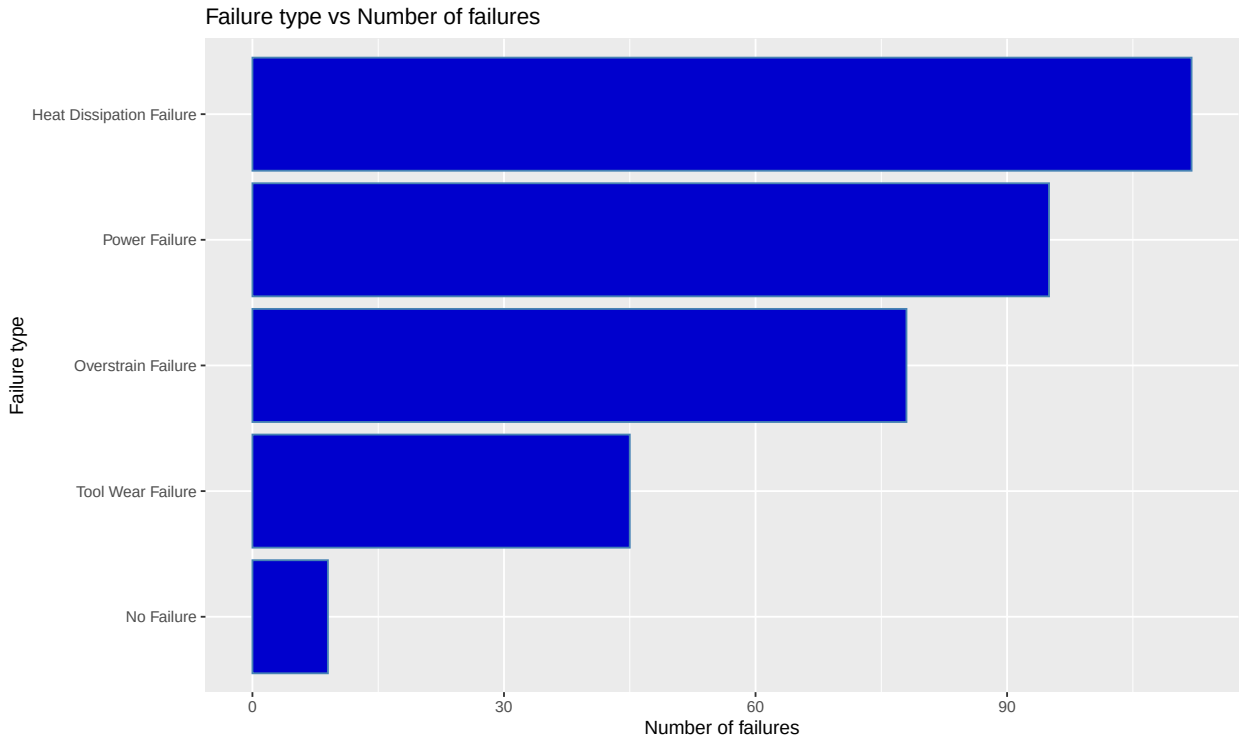


This is a very dense figure, but one can already draw very useful insights from it. The first thing to notice is that there are many more blue points compared to red points in the subplots. This means the data set is extremely unbalanced and has a lot more "No Failure" cases than "Failure" ones. From the bar plot at the

top left, one can see that L type machines have the most failures, followed by M and H type machines. The box plots on the left and top of the figure show us some rpm, torque, and wear have substantially different distributions for failed and not failed machines, regardless of the machine type. This is also visible in the density plots. The air and process temperatures also seem to be higher in case of failed machines. The figure also shows the correlation coefficients of each feature pair in both failure and no failure cases. In some pairs such as **air_t** vs **wear**, the difference in correlations for the 2 classes seem to be very high. However, one needs to take into consideration that the correlation values for failure cases are based on much smaller number of samples, and different failure types might actually have different correlation values. It is also clear from the figure that the relationship between torque and rpm is inverse. This is expected, because in machines higher torque always comes with a lower rotational speed assuming a constant power output.

Next, let us have a look at the distribution of the failure modes:

```
# see number of different types of failure
df_data %>%
  filter(failure_numeric==1) %>%
  group_by(failure_type) %>%
  summarize(n_failure = n()) %>%
  mutate(failure_type=reorder(failure_type,n_failure)) %>%
  ggplot(aes(x=failure_type, y=n_failure)) +
  geom_col(color = "steelblue", fill="blue3") +
  labs(x = "Failure type",
       y = "Number of failures",
       title ="Failure type vs Number of failures") +
  coord_flip()
```



There are 2 issues one can observe in this plot:
1) "No Failure" appears in the plot despite we filtered them out.
2) "Random Failures" are not in the plot despite the exist in the data set as we discussed in the introduction section.

Therefore, we need to perform some data cleaning. We change the **failure_numeric** column to 0 and **failure** column to "No Failure" for all instances with "No Failure" in **failure_type** columns. We also change the failure to to "No Failure" for all instances with 0 in the **failure_numeric** column:

```
# reassign correct failure and failure_numeric to the problematic rows
df_data <- df_data %>%
  mutate(failure_numeric = ifelse(failure_type=="No Failure", 0, 1),
         failure = ifelse(failure_type=="No Failure", "No Failure", failure))

# reassign correct failure_type to the random failure rows
df_data <- df_data %>%
  mutate(failure_type = ifelse(failure=="No Failure",
                               "No Failure",
                               failure_type))
```

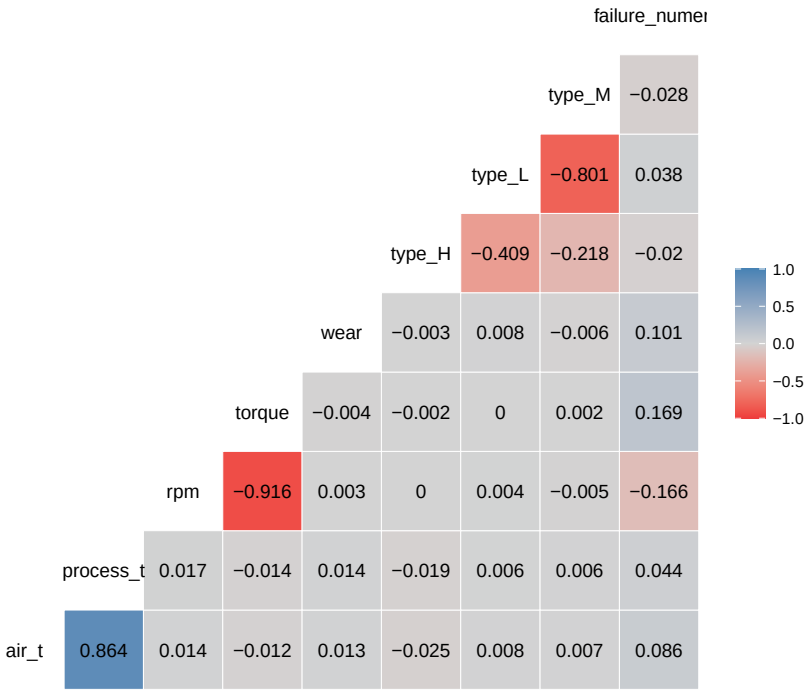
Next, let us visualize the correlations in a better way. This time we also want to include the machine type. However, the machine type in out data set is categorical. To overcome this, we do **one-hot encoding** of the type data, i.e. we encode the type data in a numeric way. After this encoding, we will have 3 new columns in our data set. Here, we see how these columns are related to the **type** column:

```
# one-hot encoding the categorical type column using the self-defined function
df_data <- one_hot_encode(df_data = df_data, categorical_col = "type")

# show encoding
tail(df_data %>% dplyr::select(type, type_L, type_M, type_H))
```

```
##       type type_L type_M type_H
## 9995     L      1      0      0
## 9996     M      0      1      0
## 9997     H      0      0      1
## 9998     M      0      1      0
## 9999     H      0      0      1
## 10000    M      0      1      0
```

Now that we have numerically encoded the machine types, we can make create a comprehensive correlation plot. Please note that the correlation coefficients are calculated using the Spearman method.

```
# select columns for correlations plot
df_corr <- df_data %>%
  dplyr::select(-any_of(c("UDI", "type", "failure",
                    "product_id", "failure_type")))

# plot the correlations using spearman method
ggcorr(df_corr, label = TRUE, label_round=3,
       method=c("pairwise", "spearman"),
       low = "brown2",
       mid = "lightgrey",
       high = "steelblue")
```



In this plot, we can see that higher torque and wear are correlated with failure. Also, rpm is inversely correlated. There is some variation in correlation coefficients between failure and different machine types, however, the coefficients are too small to conclude anything.