CMAPSS Jet Engine Failure Classification Based On Sensor Data

A. Overview

The United States space agency or popularly known as NASA some time ago shared a dataset containing jet engine simulation data. This data contains sensor data from a jet engine starting from the first time the engine is used until the engine dies. It is certainly interesting to discuss how we can recognize sensor data patterns and then perform classification to determine whether a jet engine is still functioning normally or has failed. Classification is one of the capabilities of machine learning that can predict an event based on historical data. The output of this classification is usually categorical data. One example is the classification of jet engine failures based on sensor data. This project will discuss how machine learning models learn to be used to predict engine health. This project is done by referring to the CRISP-DM concept, which is a workflow that organizes the data mining process. For more details, let's take a look together.

B. Table of Content

- Maximum Cycle Histogram
- Create labels for prediction targets
- View the proportion of classes in the dataset
- View the correlation between features with a heatmap
- Split the dataset into training and test data
- Sampling process
- Scaling process

4. Modeling & Evaluation
    - Building a random forest (RF) model
    - RF algorithm prediction and evaluation
    - Building an artificial neural network (ANN) model
    - ANN algorithm prediction and evaluation

Conclusion

1. Business Understanding

This stage will explain the background of the project, formulate the problems faced, and the ultimate goal to be achieved from the *jet engine predictive maintenance* project so that it can answer the problems that have been defined.

a. Why is machine failure prediction important?

Jet engines are one of the crucial components used in NASA's space industry. This engine is used as a source of power for a vehicle such as an airplane to be able to fly with the thrust generated from the engine. Seeing how crucial the role of the engine in a vehicle, an analysis is needed that is able to predict the health of the engine whether it is still functioning normally or has begun to require further maintenance. This aims to avoid sudden engine failure that could potentially endanger the vehicle. One way to measure engine performance is by using sensors. These sensors work to find out various things such as temperature, rotation, pressure, vibration in the engine, and others. Therefore, in this project, an analysis process will be carried out to predict engine health based on sensor data before the engine actually dies.

b. What is the problem?

Ignorance of machine health can potentially lead to sudden machine failure during use.

c. What is the goal to be achieved?

Classify machine health into normal or failure categories based on sensor data.

2. Data Understanding

This stage will explain the background of the project, formulate the problems faced, and the ultimate goal to be achieved from the *jet engine predictive maintenance* project so that it can answer the problems that have been defined.

a. Dataset information

The dataset that will be used in this project comes from: https://data.nasa.gov/Aerospace/CMAPSS-Jet-Engine-Simulated-Data/ff5v-kuh6/about_data

This dataset consists of several files which are broadly grouped into 3 namely train, test, and RUL data. However, in this project, we will only use train data, namely train_FD001.txt. This dataset has 26 columns and 20,631 data.

b. Feature explanation

Parameters	Symbol	Description	Unit
Unit	—	—	—
Time	—	—	t
Setting 1	—	Altitude	ft
Setting 2	—	Mach Number	M
Setting 3	—	Sea-level Temperature	°F
Sensor 1	T2	Total temperature at fan inlet	°R
Sensor 2	T24	Total temperature at LPC outlet	°R
Sensor 3	T30	Total temperature at HPC outlet	°R
Sensor 4	T50	Total temperature at LPT outlet	°R
Sensor 5	P2	Pressure at fan inlet	psia
Sensor 6	P15	Total pressure in bypass-duct	psia
Sensor 7	P30	Total pressure at HPC outlet	psia
Sensor 8	Nf	Physical fan speed	rpm
Sensor 9	Nc	Physical core speed	rpm
Sensor 10	epr	Engine pressure ratio	—
Sensor 11	Ps30	Static pressure at HPC outlet	psia
Sensor 12	phi	Ratio of fuel flow to Ps30	pps/psi
Sensor 13	NRf	Corrected fan speed	rpm
Sensor 14	NRc	Corrected core speed	rpm
Sensor 15	BPR	Bypass Ratio	—
Sensor 16	farB	Burner fuel-air ratio	—
Sensor 17	htBleed	Bleed Enthalpy	—
Sensor 18	Nf_dmd	Demanded fan speed	rpm
Sensor 19	PCNfR_dmd	Demanded corrected fan speed	rpm
Sensor 20	W31	HPT coolant bleed	lbm/s
Sensor 21	W32	LPT coolant bleed	lbm/s

LPC/HPC=Low/High Pressure Compressor - LPT/HPT= Low/High Pressure Turbine

c. Feature explanation

We can check the dimensions and view the raw data before further processing.

[Code to import library, call data, and view data]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	21.61	554.36	2388.06	9046.19	1.3	47.47	521.66	2388.02	8138.62	8.4195	0.03	392
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	21.61	553.75	2388.04	9044.07	1.3	47.49	522.28	2388.07	8131.49	8.4318	0.03	392
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	21.61	554.26	2388.08	9052.94	1.3	47.27	522.42	2388.03	8133.23	8.4178	0.03	390
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	21.61	554.45	2388.11	9049.48	1.3	47.13	522.86	2388.08	8133.83	8.3682	0.03	392
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	21.61	554.00	2388.06	9055.15	1.3	47.28	522.19	2388.04	8133.80	8.4294	0.03	393
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
20626	100	196	-0.0004	-0.0003	100.0	518.67	643.49	1597.98	1428.63	14.62	21.61	551.43	2388.19	9065.52	1.3	48.07	519.49	2388.26	8137.60	8.4956	0.03	397
20627	100	197	-0.0016	-0.0005	100.0	518.67	643.54	1604.50	1433.58	14.62	21.61	550.86	2388.23	9065.11	1.3	48.04	519.68	2388.22	8136.50	8.5139	0.03	395
20628	100	198	0.0004	0.0000	100.0	518.67	643.42	1602.46	1428.18	14.62	21.61	550.94	2388.24	9065.90	1.3	48.09	520.01	2388.24	8141.05	8.5646	0.03	398
20629	100	199	-0.0011	0.0003	100.0	518.67	643.23	1605.26	1426.53	14.62	21.61	550.68	2388.25	9073.72	1.3	48.39	519.67	2388.23	8139.29	8.5389	0.03	395
20630	100	200	-0.0032	-0.0005	100.0	518.67	643.85	1600.38	1432.14	14.62	21.61	550.79	2388.26	9061.48	1.3	48.20	519.30	2388.26	8137.33	8.5036	0.03	396

20631 rows × 28 columns

Notes:

- /content/train_FD001.txt is the location and name of the dataset. Specify the location of the dataset on your computer.
- Data.shape returns 2 values, namely the number of data and the number of columns with the format (number of data, number of columns).

From the dataset we can see that the column names are not representative (still in the form of consecutive numbers) and there are NaN (Not a Number) columns in the last 2 columns. This needs to be done further process to clean the data. The data cleaning process will be carried out at the next stage, namely data preparation.

3. Data Preparation

This stage is a process to clean the data so that the output of this stage is data that is clean and ready to be used for the *Machine Learning* modeling process. There is a term Garbage In, Garbage Out (GIGO) which means that if the data trained is garbage data, it will create a garbage model too. A model that is not good for the prediction process. To avoid this, a data preparation process is needed. Some of the processes carried out at this stage include:

a. Delete NaN-valued columns & Rename columns

NaN values should be removed from the dataset because NaN values have no influence on the data. In addition, it is also important to rename the columns to make them easier to read and more representative.

[Code to rename column]

	engine	cycle	setting1	setting2	setting3	sensor1	sensor2	sensor3	sensor4	sensor5	sensor6	sensor7	sensor8	sensor9	sensor10	sensor11	sensor12	sensor13	sensor14
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	21.61	554.36	2388.06	9046.19	1.3	47.47	521.66	2388.02	8138.62
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	21.61	553.75	2388.04	9044.07	1.3	47.49	522.28	2388.07	8131.49
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	21.61	554.26	2388.08	9052.94	1.3	47.27	522.42	2388.03	8133.23
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	21.61	554.45	2388.11	9049.48	1.3	47.13	522.86	2388.08	8133.83
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	21.61	554.00	2388.06	9055.15	1.3	47.28	522.19	2388.04	8133.80
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
20626	100	196	-0.0004	-0.0003	100.0	518.67	643.49	1597.98	1428.63	14.62	21.61	551.43	2388.19	9065.52	1.3	48.07	519.49	2388.26	8137.60
20627	100	197	-0.0016	-0.0005	100.0	518.67	643.54	1604.50	1433.58	14.62	21.61	550.86	2388.23	9065.11	1.3	48.04	519.68	2388.22	8136.50
20628	100	198	0.0004	0.0000	100.0	518.67	643.42	1602.46	1428.18	14.62	21.61	550.94	2388.24	9065.90	1.3	48.09	520.01	2388.24	8141.05
20629	100	199	-0.0011	0.0003	100.0	518.67	643.23	1605.26	1426.53	14.62	21.61	550.68	2388.25	9073.72	1.3	48.39	519.67	2388.23	8139.29
20630	100	200	-0.0032	-0.0005	100.0	518.67	643.85	1600.38	1432.14	14.62	21.61	550.79	2388.26	9061.48	1.3	48.20	519.30	2388.26	8137.33

20631 rows × 26 columns

After the dataset is named after the column descriptions, the dataset looks easier to understand the meaning of the predictors. So, there are now only 26 columns (predictors) in the dataset.

b. View dataset statistics

This process is done to find out some statistical things from the data such as the average value, standard deviation, minimum value, Q1, median, Q2, and maximum value of data from each column.

[Code to view data statistics]

	count	mean	std	min	25%	50%	75%	max
engine	20631.0	51.506568	2.922763e+01	1.0000	26.0000	52.0000	77.0000	100.0000
cycle	20631.0	108.807862	6.888099e+01	1.0000	52.0000	104.0000	156.0000	362.0000
setting1	20631.0	-0.000009	2.187313e-03	-0.0087	-0.0015	0.0000	0.0015	0.0087
setting2	20631.0	0.000002	2.930621e-04	-0.0006	-0.0002	0.0000	0.0003	0.0006
setting3	20631.0	100.000000	0.000000e+00	100.0000	100.0000	100.0000	100.0000	100.0000
sensor1	20631.0	518.670000	0.000000e+00	518.6700	518.6700	518.6700	518.6700	518.6700
sensor2	20631.0	642.680934	5.000533e-01	641.2100	642.3250	642.6400	643.0000	644.5300
sensor3	20631.0	1590.523119	6.131150e+00	1571.0400	1586.2600	1590.1000	1594.3800	1616.9100
sensor4	20631.0	1408.933782	9.000605e+00	1382.2500	1402.3600	1408.0400	1414.5550	1441.4900
sensor5	20631.0	14.620000	1.776400e-15	14.6200	14.6200	14.6200	14.6200	14.6200
sensor6	20631.0	21.609803	1.388985e-03	21.6000	21.6100	21.6100	21.6100	21.6100
sensor7	20631.0	553.367711	8.850923e-01	549.8500	552.8100	553.4400	554.0100	556.0600
sensor8	20631.0	2388.096652	7.098548e-02	2387.9000	2388.0500	2388.0900	2388.1400	2388.5600
sensor9	20631.0	9065.242941	2.208288e+01	9021.7300	9053.1000	9060.6600	9069.4200	9244.5900
sensor10	20631.0	1.300000	0.000000e+00	1.3000	1.3000	1.3000	1.3000	1.3000
sensor11	20631.0	47.541168	2.670874e-01	46.8500	47.3500	47.5100	47.7000	48.5300
sensor12	20631.0	521.413470	7.375534e-01	518.6900	520.9600	521.4800	521.9500	523.3800
sensor13	20631.0	2388.096152	7.191892e-02	2387.8800	2388.0400	2388.0900	2388.1400	2388.5600
sensor14	20631.0	8143.752722	1.907618e+01	8099.9400	8133.2450	8140.5400	8148.3100	8293.7200
sensor15	20631.0	8.442146	3.750504e-02	8.3249	8.4149	8.4389	8.4656	8.5848
sensor16	20631.0	0.030000	1.387812e-17	0.0300	0.0300	0.0300	0.0300	0.0300
sensor17	20631.0	393.210654	1.548763e+00	388.0000	392.0000	393.0000	394.0000	400.0000
sensor18	20631.0	2388.000000	0.000000e+00	2388.0000	2388.0000	2388.0000	2388.0000	2388.0000
sensor19	20631.0	100.000000	0.000000e+00	100.0000	100.0000	100.0000	100.0000	100.0000
sensor20	20631.0	38.816271	1.807464e-01	38.1400	38.7000	38.8300	38.9500	39.4300
sensor21	20631.0	23.289705	1.082509e-01	22.8942	23.2218	23.2979	23.3668	23.6184

From the data, it can be seen that there are several predictors that have the same min and max values. This indicates that the predictor has a constant value, which is the same value for all rows. This will not affect the target so it is necessary to remove these predictors to reduce the computational burden.

c. Removing constant-value columns

A constant value is characterized by equal min and max values. Here is the function to remove the constant value.

[Function code to delete constant value column]

	engine	cycle	setting1	setting2	sensor2	sensor3	sensor4	sensor6	sensor7	sensor8	sensor9	sensor11	sensor12	sensor13	sensor14	sensor15	sensor17	sensor20	sensor21
0	1	1	-0.0007	-0.0004	641.82	1589.70	1400.60	21.61	554.36	2388.06	9046.19	47.47	521.66	2388.02	8138.62	8.4195	392	39.06	23.4190
1	1	2	0.0019	-0.0003	642.15	1591.82	1403.14	21.61	553.75	2388.04	9044.07	47.49	522.28	2388.07	8131.49	8.4318	392	39.00	23.4236
2	1	3	-0.0043	0.0003	642.35	1587.99	1404.20	21.61	554.26	2388.08	9052.94	47.27	522.42	2388.03	8133.23	8.4178	390	38.95	23.3442
3	1	4	0.0007	0.0000	642.35	1582.79	1401.87	21.61	554.45	2388.11	9049.48	47.13	522.86	2388.08	8133.83	8.3682	392	38.88	23.3739
4	1	5	-0.0019	-0.0002	642.37	1582.85	1406.22	21.61	554.00	2388.06	9055.15	47.28	522.19	2388.04	8133.80	8.4294	393	38.90	23.4044
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
20626	100	196	-0.0004	-0.0003	643.49	1597.98	1428.63	21.61	551.43	2388.19	9065.52	48.07	519.49	2388.26	8137.60	8.4956	397	38.49	22.9735
20627	100	197	-0.0016	-0.0005	643.54	1604.50	1433.58	21.61	550.86	2388.23	9065.11	48.04	519.68	2388.22	8136.50	8.5139	395	38.30	23.1594
20628	100	198	0.0004	0.0000	643.42	1602.46	1428.18	21.61	550.94	2388.24	9065.90	48.09	520.01	2388.24	8141.05	8.5646	398	38.44	22.9333
20629	100	199	-0.0011	0.0003	643.23	1605.26	1426.53	21.61	550.68	2388.25	9073.72	48.39	519.67	2388.23	8139.29	8.5389	395	38.29	23.0640
20630	100	200	-0.0032	-0.0005	643.85	1600.38	1432.14	21.61	550.79	2388.26	9061.48	48.20	519.30	2388.26	8137.33	8.5036	396	38.37	23.0522

20631 rows × 19 columns

After the constant value removal process, the dataset left 19 predictors from the original 26 predictors. This shows that there are 7 predictors that have constant values.

d. Create labels for prediction targets

Since this is a classification job and the dataset does not have a target column, it is necessary to create a target column manually. The target to be created is whether the machine has a normal or failure status (binary classification case). In this project, normal status will be labeled 0 and failure will be labeled 1.

To determine whether a cycle is labeled as failure or normal, a threshold value of 20 is used. That is, for each engine if the cycle value has reached (maximum cycle - threshold), then the cycle will be labeled as failure. For example, engine 1 has a maximum cycle of 120. Then cycle 101 to 120 will be labeled as failure. This is done to prevent the engine from completely shutting down so that maintenance and preparation for engine replacement can be done earlier.

Here is the function to create a machine status label.

[Function code assign label]

	engine	cycle	setting1	setting2	sensor2	sensor3	sensor4	sensor6	sensor7	sensor8	sensor9	sensor11	sensor12	sensor13	sensor14	sensor15	sensor17	sensor20	sensor21	status
0	1	1	-0.0007	-0.0004	641.82	1589.70	1400.60	21.61	554.36	2388.06	9046.19	47.47	521.66	2388.02	8138.62	8.4195	392	39.06	23.4190	0.0
1	1	2	0.0019	-0.0003	642.15	1591.82	1403.14	21.61	553.75	2388.04	9044.07	47.49	522.28	2388.07	8131.49	8.4318	392	39.00	23.4236	0.0
2	1	3	-0.0043	0.0003	642.35	1587.99	1404.20	21.61	554.26	2388.08	9052.94	47.27	522.42	2388.03	8133.23	8.4178	390	38.95	23.3442	0.0
3	1	4	0.0007	0.0000	642.35	1582.79	1401.87	21.61	554.45	2388.11	9049.48	47.13	522.86	2388.08	8133.83	8.3682	392	38.88	23.3739	0.0
4	1	5	-0.0019	-0.0002	642.37	1582.85	1406.22	21.61	554.00	2388.06	9055.15	47.28	522.19	2388.04	8133.80	8.4294	393	38.90	23.4044	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
20626	100	196	-0.0004	-0.0003	643.49	1597.98	1428.63	21.61	551.43	2388.19	9065.52	48.07	519.49	2388.26	8137.60	8.4956	397	38.49	22.9735	1.0
20627	100	197	-0.0016	-0.0005	643.54	1604.50	1433.58	21.61	550.86	2388.23	9065.11	48.04	519.68	2388.22	8136.50	8.5139	395	38.30	23.1594	1.0
20628	100	198	0.0004	0.0000	643.42	1602.46	1428.18	21.61	550.94	2388.24	9065.90	48.09	520.01	2388.24	8141.05	8.5646	398	38.44	22.9333	1.0
20629	100	199	-0.0011	0.0003	643.23	1605.26	1426.53	21.61	550.68	2388.25	9073.72	48.39	519.67	2388.23	8139.29	8.5389	395	38.29	23.0640	1.0
20630	100	200	-0.0032	-0.0005	643.85	1600.38	1432.14	21.61	550.79	2388.26	9061.48	48.20	519.30	2388.26	8137.33	8.5036	396	38.37	23.0522	1.0

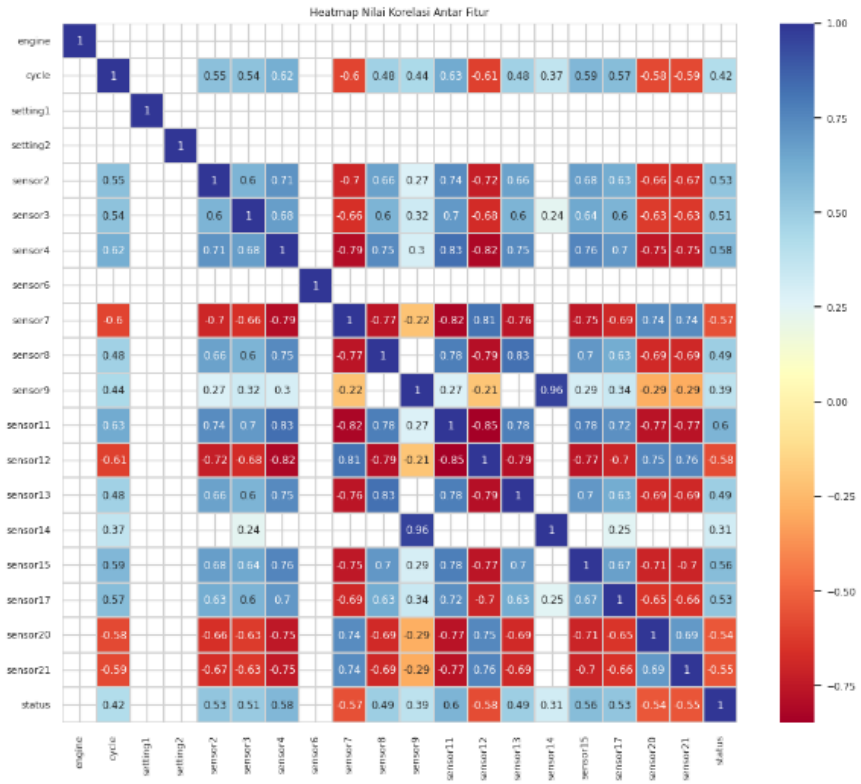
20631 rows × 20 columns

e. View feature correlation with heatmaps

The influence value or known as the correlation value in the dataset can be divided into 5 categories, namely:

Coefficient Interval	Correlation
0.00 – 0.199	Very Weak
0.20 – 0.399	Weak
0.40 – 0.599	Medium
0.60 – 0.799	Strong
0.80 – 1.000	Very Strong

To see the correlation value between the predictor and the target, a heatmap visualization is used. In this project, a threshold value of 0.20 will be used.



From the heatmap visualization above, only predictors that have an absolute value of correlation greater than or equal to the threshold will be displayed. A threshold value of 0.2 is used because a correlation value above 0.2 is a fairly

strong correlation. Whereas below 0.2 is categorized as very weak so it does not need to be used.

A negative value in the correlation indicates that the predictor has an opposite correlation with other predictors. For example, sensor 2 and sensor 7 have a correlation value of -0.7. This means that when the value of sensor 2 increases, the value of sensor 7 will decrease and vice versa. The higher the correlation value, the more they affect each other. The absolute value of the correlation value is between 0 and 1. A value of 0 means no correlation while 1 means a very strong correlation.

f. Feature selection

In some cases, not all predictors (columns) in the dataset have a strong enough influence on the target. For this reason, it is necessary to perform a feature selection process to remove features that have no influence. The goal is to reduce the time and computational burden used in the learning process. As in the previous stage, a threshold value of 0.2 will be used. So that predictors that have a correlation value <0.2 will be removed. Here is the function for feature selection.

[Code for feature selection]

	sensor2	sensor3	sensor4	sensor7	sensor8	sensor9	sensor11	sensor12	sensor13	sensor14	sensor15	sensor17	sensor20	sensor21	status
0	641.82	1589.70	1400.60	554.36	2388.06	9046.19	47.47	521.66	2388.02	8138.62	8.4195	392	39.06	23.4190	0.0
1	642.15	1591.82	1403.14	553.75	2388.04	9044.07	47.49	522.28	2388.07	8131.49	8.4318	392	39.00	23.4236	0.0
2	642.35	1587.99	1404.20	554.26	2388.08	9052.94	47.27	522.42	2388.03	8133.23	8.4178	390	38.95	23.3442	0.0
3	642.35	1582.79	1401.87	554.45	2388.11	9049.48	47.13	522.86	2388.08	8133.83	8.3682	392	38.88	23.3739	0.0
4	642.37	1582.85	1406.22	554.00	2388.06	9055.15	47.28	522.19	2388.04	8133.80	8.4294	393	38.90	23.4044	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
20626	643.49	1597.98	1428.63	551.43	2388.19	9065.52	48.07	519.49	2388.26	8137.60	8.4956	397	38.49	22.9735	1.0
20627	643.54	1604.50	1433.58	550.86	2388.23	9065.11	48.04	519.68	2388.22	8136.50	8.5139	395	38.30	23.1594	1.0
20628	643.42	1602.46	1428.18	550.94	2388.24	9065.90	48.09	520.01	2388.24	8141.05	8.5646	398	38.44	22.9333	1.0
20629	643.23	1605.26	1426.53	550.68	2388.25	9073.72	48.39	519.67	2388.23	8139.29	8.5389	395	38.29	23.0640	1.0
20630	643.85	1600.38	1432.14	550.79	2388.26	9061.48	48.20	519.30	2388.26	8137.33	8.5036	396	38.37	23.0522	1.0

20631 rows × 15 columns

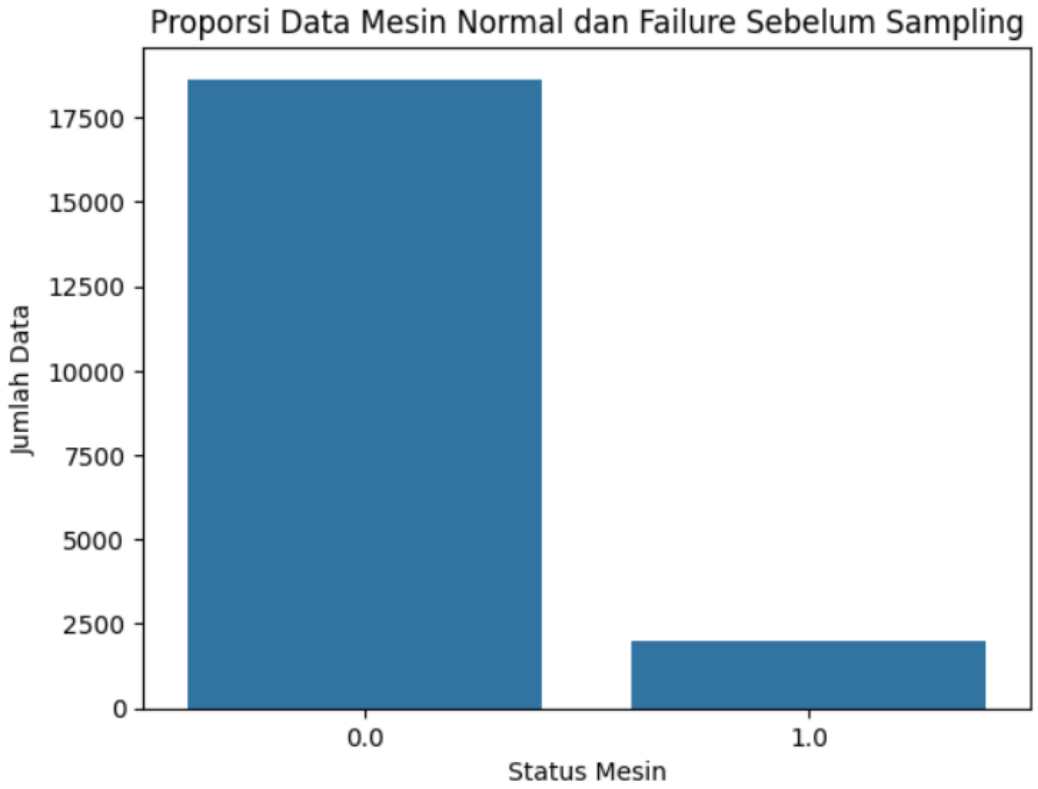
After the feature selection process, we are left with 15 columns consisting of 14 predictors and 1 target.

g. View the proportion of classes in the dataset

The next step is to look at the proportion of classes in the dataset. We will look at the proportion of normal (0) and failure (1) classes. This is done to determine the balance of the dataset.

[Code to view dataset balance]

```
0:  18631  data
1:  2000  data
```



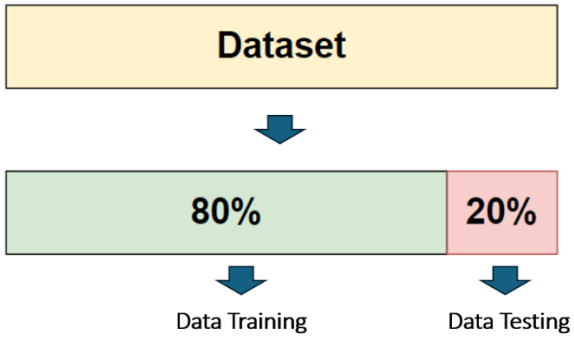
From the visualization above, it can be seen that the machine cycle data classified as normal is 18,631 cycles and failure is 2,000 cycles. From this amount it can be said that the proportion of minority values is 9.7% of the total data in the dataset. This proportion is included in the moderate category so it is necessary to do a sampling process to add data to the minority data. This phenomenon is referred to as an unbalanced dataset. The article about unbalanced datasets can be seen on the page: https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data

h. Split the dataset into training and test data

Before the data balancing process is carried out (sampling process), the data needs to be divided first into 2 parts, namely train data and test data. Train

data will be used in making machine learning models while test data is used to see the performance of the resulting machine learning model.

In this project, the dataset sharing process will use the 80:20 scheme which means 80% of the data will be used as train data and 20% of the data is used as test data. There is no specific rule for the scheme used. Some projects use 60:40, 70:30, 75:25, 80:20, and 90:10 schemes. But one thing for sure is that the amount of test data should not exceed the train data. In addition, it will also be divided between the predictor column (symbolized by the prefix X) and the target column (symbolized by the prefix y).



[Code to view split dataset]

After the dataset is divided, we look at the number of train data and test data by using the shape function.

[Code to view train and test data dimensions]

From a total of 20,631 data in the dataset, it can be seen that the amount of data for the training process is 16,504 data and the amount of data for the testing process is 4,127 data. The number 14 indicates that there are 14 predictors that will be searched for patterns during the learning process.

i. Sampling process

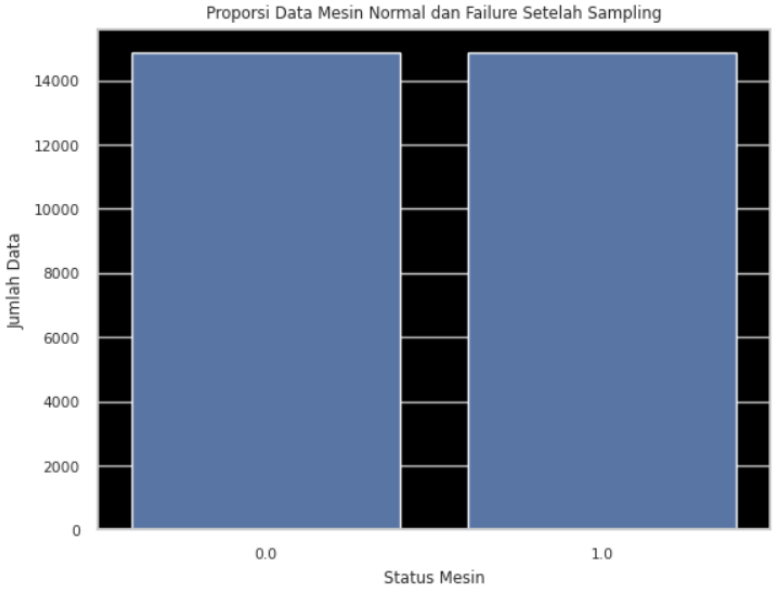
The sampling process is used to overcome the problem of unbalanced datasets. The purpose of this process is to balance the proportion of classes in the dataset so that the normal and failure classes will have the same amount of data. This will make the machine learning model sensitive to both classes of data (normal and failure) not just to one of them.

The sampling process should only be done on the train data to prevent data leakage from the test data. Therefore, in the previous stage, the process of dividing the train data and test data was carried out first.

The sampling technique that will be used in this project is oversampling where the minority data (failure) will be made synthetic data so that the number will be the same as the majority class (normal). The algorithm used is Synthetic Minority Oversampling Technique (SMOTE). Read more about SMOTE at the following link:
https://medium.com/@corymaklin/synthetic-minority-over-sampling-technique-smote-7d419696b88c

[Code for sampling process]

```
0:  14861  data
1:  14861  data
```



It can be seen in the barplot above that after the oversampling process, the amount of data between normal and failure machines is balanced with each status having 14,861 data.

j. Scaling process

Just like the sampling process, the scaling process should also only be done on the train data to prevent data leakage from the test data. In addition, the

scaling process is also done after the sampling process. It cannot be reversed. Therefore, in the previous stage, the process of dividing the train data and test data is carried out first, then continued with the sampling process, and finally scaling.

The scaling process is used to equalize the range of values of all features. This aims to reduce the computational burden during the training process and improve the performance of the resulting model. The scaling process is carried out if there is a predictor that has a value far above the value of other predictors. For example, the value of sensor 11 is worth tens while sensor 21 is worth tens of thousands.

In this project, the Z-Score method will be used for the scaling process. More information about Z-Score normalization can be found at the following link: https://www.statology.org/z-score-normalization/

[Code for scaling process]

	sensor2	sensor3	sensor4	sensor7	sensor8	sensor9	sensor11	sensor12	sensor13	sensor14	sensor15	sensor17	sensor20	sensor21
0	-0.938390	-0.556666	-0.909093	0.212688	0.177875	-0.622271	-1.015749	0.792598	-0.397959	-0.579051	-0.892908	-0.094989	0.888048	0.960463
1	-1.392110	-0.144523	-1.030363	1.533963	-1.576763	0.055578	-1.046601	2.175096	-1.319749	0.095758	-2.040381	-2.481010	0.746002	0.934275
2	-1.095447	-2.171478	-1.281174	0.592911	-1.459787	-0.782529	-1.817908	1.107313	-0.397959	-0.414380	-1.480251	-0.691494	1.456235	0.495435
3	-0.467220	-1.711510	-1.025770	1.524457	-1.342811	-0.519402	-1.324272	1.545666	-1.665420	-0.247229	-1.121949	-1.288000	1.124793	1.821479
4	-0.606826	-1.057426	-1.002802	0.298238	0.294851	-0.806351	-0.676374	0.736399	0.178160	-0.679218	-1.072059	-0.691494	0.414559	0.469248
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29717	0.445690	1.209339	0.834500	-0.819891	1.347633	-1.180792	0.415787	-0.347741	1.498290	-0.881607	0.491178	1.098022	-0.813797	0.271253
29718	0.957616	1.364339	1.040986	-1.121383	1.268609	-0.059821	0.799902	-1.229398	0.719684	0.102551	1.195119	0.501517	-1.430785	-1.405503
29719	0.274123	1.338818	0.614799	-0.441951	-0.496831	1.982762	0.222743	-0.283947	-0.589316	2.069433	0.583104	0.501517	-0.790301	-0.595673
29720	1.199629	1.536722	0.874485	-0.822968	-0.032885	2.017594	1.380752	-0.790664	0.265570	1.737952	1.163457	1.098022	-1.161015	-0.405578
29721	0.933562	2.213714	0.340163	-1.071757	-0.153191	1.880993	0.931628	-0.742242	0.217289	1.845271	1.369915	1.098022	-1.905538	-0.436191

29722 rows × 14 columns

From the scaling results, it can be seen that all predictors have a range of data that is not much different. This will facilitate the process of building machine learning models and reduce the time and computational resources required.

4. Modeling & Evaluation

This stage is a process to create a machine learning model that will later be used for the prediction process. Some of the things done at this stage are:

- Selecting the machine learning algorithm to be used and hyperparameter tuning.
- The fitting process or learning model process.
- Model evaluation process to determine the performance of the model.

The output of this stage is a model that has been trained and is ready to be used for the prediction process.

a. Modeling the random forest (RF) algorithm

Random forest is a popular classification algorithm due to its excellent performance. This article does not discuss the details of random forest so you can read more about random forest in the following sources:
https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

After the data is cleaned in the pre-processing process, the next step is to build a machine learning model. To create an ML model from random forest, we will use the library provided by scikit-learn.
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

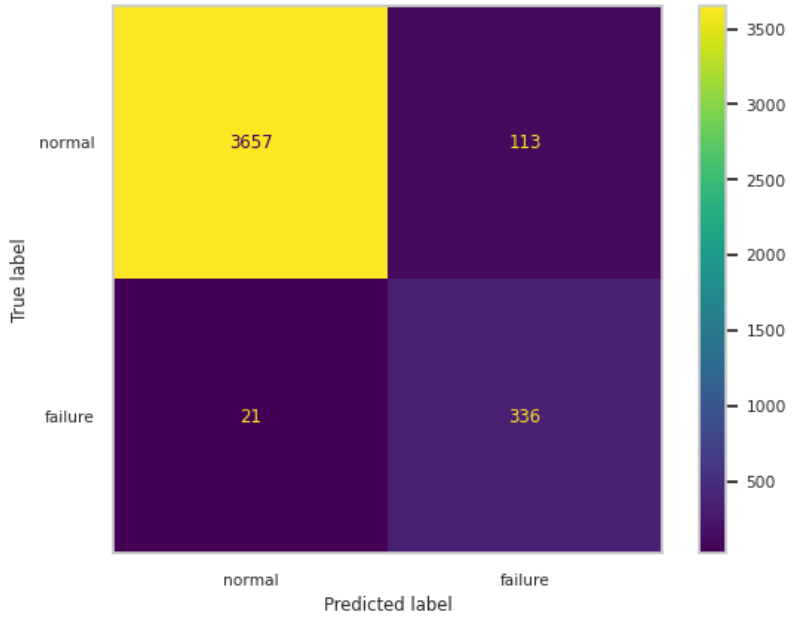
[Code for modeling with random forest]

Notes:

- The RandomForestClassifier() function is a function from the scikit-learn library that is used to create ML models with the random forest algorithm.
- The fit() function is used for the training and machine learning process to create the ML model. The fit() function requires 2 data namely X_train and y_train. X_train is data that contains predictor data while y_train contains target data.

- The dump() function of joblib is used to store the random forest model. We need to export the training model to embed it in an application to predict new data.
- The predict() function is used to predict new data. This function requires one data, X_test, which is the predictor data for the testing data. The result of this function is the target prediction of X_test which is then stored in the y_predict variable.

After successfully predicting the data using the predict() function, then we will evaluate the prediction results to find out whether the resulting model is good or not. To evaluate, there are several evaluation measures that will be used, namely accuracy, precision, recall, and f1 score. Before calculating the evaluation value, the confusion matrix will first be used to determine the value of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). More information about confusion matrix can be seen in the following article:
https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62
[Code for confusion matrix].



From the confusion matrix table above, the following can be seen:

- True Positive (TP) -> Cycle failure that is correctly predicted failure. There are 336 data.
- True Negative (TN) -> Cycle normal that is correctly predicted to be normal. There are 3,657 data.
- False Positive (FP) -> Cycle normal predicted failure. There are 113 data.
- False Negative (FN) -> Cycle failure that is predicted to be normal. There are 21 data.

[Code for calculating evaluation score]

```
[ ]  print("Accuracy  : ", metrics.accuracy_score(y_test, y_predict))
     print("Precision : ", metrics.precision_score(y_test, y_predict))
     print("Recall    : ", metrics.recall_score(y_test, y_predict))
     print("F1 Score  : ", metrics.f1_score(y_test, y_predict))
```

```
Accuracy  :  0.9675308941119457
Precision :  0.7483296213808464
Recall    :  0.9411764705882353
F1 Score  :  0.8337468982630273
```

From the evaluation scores above, we can conclude as follows:

- The accuracy value shows that the model is able to predict 96% of the data correctly. In other words, out of 4,127 test data the model can correctly predict 3,989 data.
- The precision value shows that of all the cycles predicted to fail by the model, only 74% are correct. In other words, of the 449 cycles predicted to fail, only 336 cycles were actually in failure status. The rest are normal.
- The recall value shows that of all the cycles that actually did have failure status, only 94% were successfully predicted as failure by the model. In other words, out of 357 cycles that were indeed failures, the model was able to correctly predict 337 cycles. Only 20 cycles with failure status were predicted normally by the model.
- The F1 value shows that the model is able to recognize normal and failure cycle conditions well. Not leaning towards one condition only.

b. Modeling an artificial neural network (ANN) algorithm

ANN is one of the machine learning algorithms that is the forerunner of deep learning algorithms. It is called neural because it mimics the way neurons work in the human brain to transfer signals to other neurons. Further discussion about ANN can be seen in the article: https://www.analyticsvidhya.com/blog/2021/09/introduction-to-artificial-neural-networks/

In this project, the Tensorflow library will be used to build the ANN model. Here is the code to build the ANN architecture.

[Code for ANN]

```
Epoch 1/200
186/186 [==============================] - 3s 9ms/step - loss: 4.5718 - accuracy: 0.9424 - val_loss: 3.4823 - val_accuracy: 0.9943
Epoch 2/200
186/186 [==============================] - 1s 8ms/step - loss: 2.7408 - accuracy: 0.9596 - val_loss: 2.0740 - val_accuracy: 0.9899
Epoch 3/200
186/186 [==============================] - 2s 13ms/step - loss: 1.6270 - accuracy: 0.9638 - val_loss: 1.2306 - val_accuracy: 0.9845
Epoch 4/200
186/186 [==============================] - 2s 10ms/step - loss: 0.9665 - accuracy: 0.9652 - val_loss: 0.7404 - val_accuracy: 0.9800
Epoch 5/200
186/186 [==============================] - 1s 8ms/step - loss: 0.5878 - accuracy: 0.9657 - val_loss: 0.4686 - val_accuracy: 0.9749
Epoch 6/200
186/186 [==============================] - 1s 8ms/step - loss: 0.3769 - accuracy: 0.9653 - val_loss: 0.3172 - val_accuracy: 0.9739
Epoch 7/200
186/186 [==============================] - 2s 8ms/step - loss: 0.2621 - accuracy: 0.9660 - val_loss: 0.2417 - val_accuracy: 0.9677
Epoch 8/200
186/186 [==============================] - 2s 8ms/step - loss: 0.2004 - accuracy: 0.9656 - val_loss: 0.1839 - val_accuracy: 0.9738
Epoch 9/200
186/186 [==============================] - 2s 8ms/step - loss: 0.1679 - accuracy: 0.9655 - val_loss: 0.1804 - val_accuracy: 0.9630
Epoch 10/200
186/186 [==============================] - 1s 8ms/step - loss: 0.1499 - accuracy: 0.9651 - val_loss: 0.1548 - val_accuracy: 0.9679
Epoch 11/200
186/186 [==============================] - 2s 12ms/step - loss: 0.1405 - accuracy: 0.9656 - val_loss: 0.1537 - val_accuracy: 0.9648
Epoch 12/200
186/186 [==============================] - 2s 11ms/step - loss: 0.1349 - accuracy: 0.9653 - val_loss: 0.1329 - val_accuracy: 0.9724
Epoch 13/200
186/186 [==============================] - 1s 8ms/step - loss: 0.1317 - accuracy: 0.9655 - val_loss: 0.1240 - val_accuracy: 0.9753
Epoch 14/200
186/186 [==============================] - 2s 8ms/step - loss: 0.1291 - accuracy: 0.9658 - val_loss: 0.1201 - val_accuracy: 0.9763
Epoch 15/200
186/186 [==============================] - 1s 7ms/step - loss: 0.1271 - accuracy: 0.9656 - val_loss: 0.1079 - val_accuracy: 0.9803
Epoch 16/200
186/186 [==============================] - 1s 8ms/step - loss: 0.1263 - accuracy: 0.9647 - val_loss: 0.1162 - val_accuracy: 0.9775
Epoch 17/200
186/186 [==============================] - 1s 8ms/step - loss: 0.1251 - accuracy: 0.9651 - val_loss: 0.1239 - val_accuracy: 0.9719
Epoch 18/200
186/186 [==============================] - 1s 8ms/step - loss: 0.1240 - accuracy: 0.9649 - val_loss: 0.1342 - val_accuracy: 0.9670
Epoch 19/200
186/186 [==============================] - 2s 10ms/step - loss: 0.1227 - accuracy: 0.9656 - val_loss: 0.1349 - val_accuracy: 0.9645
Epoch 20/200
186/186 [==============================] - 2s 13ms/step - loss: 0.1223 - accuracy: 0.9646 - val_loss: 0.1547 - val_accuracy: 0.9561
Epoch 20: early stopping
['model_nn.bin']
```

The Neural Network algorithm used has the following architecture:

- Number of layers => 5 consisting of 1 input layer, 3 hidden layers, and 1 output layer.
- The input layer has 14 neurons. This number is adjusted to the number of predictors in the train data.
- Hidden layers 1, 2, and 3 have 512, 256, and 128 neurons respectively.
- The output layer has 1 neuron with a sigmoid activation function. This allows it to produce an output in the form of a fractional value between 0 and 1. In this project using a threshold of 0.5. If the output value >= 0.5 then failure and if < 0.5 then normal.
- This architecture uses the ADAM optimizer function. This function is used to adjust the weight of each neuron in the learning process.
- The loss function used is binary_crossentropy. This function is used to calculate the error value in the output layer. The error value is obtained from the difference between actual data and predicted data.
- The evaluation metric measured during the machine learning process is the accuracy value.
- This learning process uses the EarlyStopping() function to stop the learning process if the model does not improve for a certain time.

Notes:

In general, the learning (training) process in neural networks consists of 2 main processes, namely forward and backward.

- Forward => The process to generate the output value. This process starts from input layer > hidden layer > output layer. In the output layer, the loss value calculation process is then carried out to determine the error value. How far is the predicted value from the actual value. If the error value is greater than the specified threshold, then the backward process is carried out.
- Backward => The process to update the weights. If the resulting error value is too large, then a backward process is performed to update the

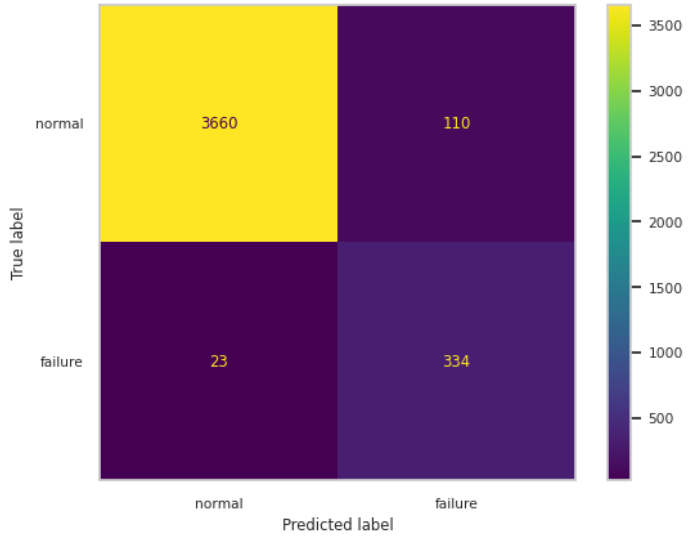
weights on each neuron in each layer. This process is done in order to reduce the resulting error value.

Explanation of some terms:

- Learning rate => How much the weights on the neurons change during the training process in order to minimize the error value. The LR value has a range between 0 and 1. The larger the LR value, the faster the training process. Vice versa, the smaller the LR value, the slower the training process. Hyperparameter tuning is required to find the right LR value.
- Epochs => Hyperparameters that determine how many times the deep learning algorithm works through the entire dataset both forward and backward. Forward serves to determine the prediction/output value. While backward serves to update the weights if the error value generated from the forward process is too large (exceeds the threshold).
- Batch size => It is the number of data samples that are distributed to the Neural Network. For example, there are 100 data and have a batch size of 10. Then the first 10 data will be distributed to the neural network for the learning process. Then the next 10 data. And so on until all 100 data are distributed. Every 1x data is spread all, then it is considered 1 epochs.
- Optimizer => A function used to update the weights in the backward process. This weight value needs to be updated in order to minimize the error value.
- Loss => A function used to calculate the error value. If the error value > threshold, then a backward process will be performed to update the weights. This forward and backward process is carried out continuously until 2 conditions are reached, namely the maximum epochs or the error value < the specified threshold.

After the training process is complete, just like in Random Forest, an evaluation process will be carried out to see the performance of the ANN model. The following is the confusion matrix code from ANN.

[Confusion matrix code for ANN]




```
[ ]  print("Accuracy  : ", metrics.accuracy_score(y_test, y_predict))
     print("Precision : ", metrics.precision_score(y_test, y_predict))
     print("Recall    : ", metrics.recall_score(y_test, y_predict))
     print("F1 Score  : ", metrics.f1_score(y_test, y_predict))
```

```
Accuracy  :  0.9677732008723043
Precision :  0.7522522522522522
Recall    :  0.9355742296918768
F1 Score  :  0.8339575530586767
```

From the evaluation scores above, we can conclude as follows:

- The accuracy value shows that the model is able to predict 96% of the data correctly. In other words, out of 4,127 test data the model can correctly predict 3,992 data.
- The precision value shows that of all the cycles predicted to fail by the model, only 75% are correct. In other words, of the 449 cycles predicted to fail, only 338 cycles were actually in failure status. The rest are normal.
- The recall value shows that of all the cycles that actually did have failure status, only 93% were successfully predicted as failure by the model. In other words, out of 357 cycles that were indeed failures, the model was able to correctly predict 335 cycles. Only 22 cycles with failure status were predicted normally by the model.

- The F1 value shows that the model is able to recognize normal and failure cycle conditions well. Not leaning towards one condition only.

Conclusion

Broadly speaking, this article discusses the classification of jet engines whether they are categorized as normal or failure based on sensor data. Some stages that need to be done in making ML models are from the business understanding stage to see the problem to be solved, data understanding to understand the data, data preparation to prepare and clean the data so that it becomes good data and is ready for the modeling process, modeling to create ML models, and evaluation to find out whether the resulting model is good or not. From the experiments that have been carried out, it can be concluded that the models produced by Random Forest and Artificial Neural Network are good because they have a high evaluation value so that they can be used for the prediction process on new data. You can do the last stage of CRISP-DM which is deployment which is not discussed here. You can use the exported model for the prediction process of new data embedded in an application. Good luck!