**DOMAIN**

Three-dimensional rapid prototyping and manufacturing (3DRPM) involves using interconnected technologies to create physical objects directly from CAD data. Unlike traditional machining, which removes excess material, 3DRPM builds objects layer by layer horizontally. These systems are also called solid freeform fabrication and layered manufacturing, offering advantages over conventional methods like milling or turning (Pérès & Noyes, 2006).

1. Objects can be produced with intricate geometric complexities without requiring complex machine setups or final assembly.

2. Objects can be constructed using multiple materials, including composites, and materials can be varied in a controlled manner at any location within the object.

3. Solid freeform fabrication systems simplify the construction of complex objects, making it a manageable, straightforward, and relatively fast process.

4. The use of jigs and fixtures becomes unnecessary in this approach

Predictive Maintenance (PM) is a method used to monitor the status of machinery to prevent costly failures and perform maintenance only when necessary. It has a long history, evolving from the earliest form of visual inspection to automated techniques that utilize advanced signal processing. In traditional maintenance practices, there is a trade-off involved, where one must choose between maximizing the lifespan of a component and risking machine downtime (run-to-failure) or maximizing uptime by replacing parts early, even if they are still functional (time-based PM). However, it has been demonstrated that this time-based approach is ineffective for most equipment components, as they are flawed and unreliable in recent years (Mobley, 2022). PM aims to minimize maintenance by forecasting it ahead of time, allowing companies to maximize the useful life of assets. This is achieved by reducing maintenance frequency, avoiding unplanned breakdowns, and eliminating unnecessary preventive maintenance. As a result, substantial time and cost savings are realized, along with improved system reliability (Traini et al., 2019).

To implement a PM strategy, a Condition Monitoring (CM) system is required. CM involves monitoring one or more machine parameters to detect potential faults at an early stage. Specifically, in machining operations like milling and turning that utilize cutting tools, monitoring tool degradation is crucial. Worn-out tools can negatively impact workpiece quality and potentially cause damage to the machining system (Traini et al., 2021). Accurate tool condition assessment prevents the use of degraded tools, which can lead to lower work quality, increased costs, and longer production times due to excessive preventive replacements. Industry 4.0 emphasizes machine digitization and connectivity, enabling more effective condition monitoring. This is achieved through sensor data analysis, providing better insights into the tool's status (Żabiński et al., 2019). From the various descriptions, it can be understood that PM is highly necessary for milling machines as it serves to prevent more severe damage to the machine and avoid incurring larger costs.

**Business Understanding**

Explaining the background of the problem statement:

- Which features have the most influence on machine failure?

- How can we select/create the best model to predict machine failure?

Explaining the objectives of the problem statement:

1. Perform data pre-processing to identify the most correlated features with machine failure from the correlation table.

2. Build a machine learning model capable of predicting machine failure using algorithms such as KNeighbors Classifier, Random Forest Classifier, Support Vector Classification, GaussianNB, and Gradient Boosting Classifier.

3. To select the best model, the following steps can be taken:

- Calculate evaluation metrics, namely ROC AUC score and MCC, to determine the best model.

- The formula for ROC AUC score is as follows(Calders & Jaroszewicz, 2007):

- The formula for MCC is as follow (Matthews, 1975)

- All the mentioned metrics will be directly calculated using the sklearn library in Python.

Once the goals have been achieved, the next step is the implementation phase. During this stage, milling machine operators need to collaborate with the maintenance division to enforce policies related to machine usage, such as checking/monitoring the condition of tools, spindle rotation, machine temperature, and other factors that could potentially trigger machine failure. Once identified, they can discuss the safe and hazardous ranges for these variables to prevent failures that could damage the machine and impact performance quality.

**Data Understanding**

This synthetic dataset is modeled after an existing milling machine and consists of 10 000 data points from a stored as rows with 14 features in columns

UID: unique identifier ranging from 1 to 10000

product ID: consisting of a letter L, M, or H for low (50% of all products), medium (30%) and high (20%) as product quality variants and a variant-specific serial number

type: just the product type L, M or H from column 2

air temperature [K]: generated using a random walk process later normalized to a standard deviation of 2 K around 300 K

process temperature [K]: generated using a random walk process normalized to a standard deviation of 1 K, added to the air temperature plus 10 K.

rotational speed [rpm]: calculated from a power of 2860 W, overlaid with a normally distributed noise

torque [Nm]: torque values are normally distributed around 40 Nm with a SD = 10 Nm and no negative values.

tool wear [min]: The quality variants H/M/L add 5/3/2 minutes of tool wear to the used tool in the process.

a 'machine failure' label that indicates, whether the machine has failed in this particular datapoint for any of the following failure modes are true.

The machine failure consists of five independent failure modes

tool wear failure (TWF): the tool will be replaced of fail at a randomly selected tool wear time between 200 - 240 mins (120 times in our dataset). At this point in time, the tool is replaced 69 times, and fails 51 times (randomly assigned).

heat dissipation failure (HDF): heat dissipation causes a process failure, if the difference between air- and process temperature is below 8.6 K and the tools rotational speed is below 1380 rpm. This is the case for 115 data points.

power failure (PWF): the product of torque and rotational speed (in rad/s) equals the power required for the process. If this power is below 3500 W or above 9000 W, the process fails, which is the case 95 times in our dataset.

overstrain failure (OSF): if the product of tool wear and torque exceeds 11,000 minNm for the L product variant (12,000 M, 13,000 H), the process fails due to overstrain. This is true for 98 datapoints.

random failures (RNF): each process has a chance of 0,1 % to fail regardless of its process parameters. This is the case for only 5 datapoints, less than could be expected for 10,000 datapoints in our dataset.

If at least one of the above failure modes is true, the process fails and the 'machine failure' label is set to 1. It is therefore not transparent to the machine learning method, which of the failure modes has caused the process to fail.

**Data Preparation**

The techniques used on the dataset are as follows:

- Since the categorical feature "machine\_failure" has been properly formatted as 1 or 0, there is no need to perform category encoding.

- Variable reduction using PCA.

- Splitting the dataset into training and testing sets.

- Normalization.

Large datasets are becoming more common and are often challenging to understand. Principal component analysis (PCA) is a method used to simplify such datasets by reducing their dimensions, which enhances interpretability while minimizing loss of information. This is achieved by generating new variables that are uncorrelated and maximize variance. The process of finding these new variables, called principal components, involves solving an eigenvalue/eigenvector problem. Unlike predetermined variables, PCA adapts to the specific dataset being analyzed. Additionally, different versions of PCA have been developed to suit various data types and structures (Jolliffe & Cadima, 2016).

Kode di atas memanggil class PCA() dari library scikit-learn. Paremeter yang kita masukkan ke dalam class adalah n\_components dan random\_state. Parameter n\_components merupakan jumlah komponen atau dimensi, dalam kasus kita jumlahnya ada 2 yaitu air\_temperature\_k dan process\_temperature\_k. Sedangkan, parameter random\_state berfungsi untuk mengontrol random number generator yang digunakan. Parameter ini berupa bilangan integer dan nilainya bebas. Menentukan parameter random\_state bertujuan untuk dapat memastikan bahwa hasil pembagian dataset konsisten dan memberikan data yang sama setiap kali model dijalankan. Jika tidak ditentukan, maka tiap kali melakukan split, kita akan mendapatkan data train dan tes berbeda. Hal ini berpengaruh terhadap akurasi model ML yang menjadi berbeda tiap kali di-run. setelah menerapkan class PCA, kita bisa mengetahui proporsi informasi dari ketiga komponen tadi.

1. pca.explained\_variance\_ratio\_.round(3)

array([0.944, 0.056])

arti dari output diatas adalah bahwa 94,4% informasi pada kedua fitur air\_temperature\_k dan process\_temperature\_k terdapat pada PC pertama, sedangkan sisanya sebesar 5,6% pada PC kedua. Karena adanya pembulatan menyebabkan jumlah total informasi lebih dari 100%. Berdasarkan hasil tersebut, maka PC pertama akan menjadi fitur degree menggantikan air\_temperature\_k dan process\_temperature\_k yang diberi nama degree. Pembuatan fitur baru bernama degree adalah sebagai berikut.

- gunakan n\_component = 1, karena kali ini, jumlah komponen kita hanya satu

- fit model dengan data masukan

- tambahkan fitur baru ke dataset dengan nama ‘degree’ dan transformasi

- drop kolom air\_temperature\_k dan process\_temperature\_k

\* table

Splitting a dataset into training and testing data is essential before building a model. It allows us to evaluate the model's generalization to new data by reserving a portion for testing. Transformations applied to the data are part of the model, so they should be performed on the training data. This division ensures that the test data remains untainted by information from the training data, making it crucial to perform the split before any transformations (Fuentes, 2018). This time, we will use an 80:20 ratio for the train and test data,with this code.

\*code

Result

\*gambar

**Standardization**

Standardization, also known as standardisation, is the act of establishing and creating technical standards through the agreement and consensus of various stakeholders. These stakeholders can include companies, users, interest groups, standards organizations, and government bodies (Xie et al., 2016). Standardization plays a crucial role in achieving optimal compatibility, interoperability, safety, repeatability, and quality. It also enables the transformation of previously customized processes into normalized ones. In the field of social sciences, including economics, standardization is of great significance(Blind, 2004). Normalizing data is essential in machine learning algorithms due to the wide variability in the range of values within raw data. Failure to normalize can lead to improper functioning of objective functions in certain algorithms. For instance, classifiers rely on Euclidean distance to measure the proximity between two points. When a feature has a broad range of values, it exerts a disproportionate influence on the distance calculation. To ensure balanced contributions from each feature, it is crucial to normalize the range of all features. Feature scaling is also applied because it significantly accelerates the convergence of gradient descent compared to when it is not used (Ioffe & Szegedy, 2015).

**Modeling**

**KNeighborsClassifier**

KNeighborsClassifier is the K-NN algorithm is a classification method that is conceptually one of the easiest to grasp. It is also known as a "Lazy Learner" in contrast to an "Eager Learner." Most classification algorithms are eager learners, where a set of training data with known classifications is used to construct a classification model. This model is then evaluated using test data with known classifications. If the results are satisfactory, the final model is used to predict classes for data with unknown classifications. In contrast, a lazy learner does not build a model beforehand but rather waits for unclassified data to make classification predictions through the algorithm. Lazy learners are more time-consuming as they require the model building process for each prediction (Ioffe & Szegedy, 2015).

In the k-nearest neighbor (K-NN) algorithm, the data examples are initially plotted in an n-dimensional space, where n represents the number of data attributes. Each point in this space is assigned a class label. To classify an unclassified data point, it is plotted in the same n-dimensional space, and the class labels of the nearest k data points are recorded. Typically, k is chosen as an odd number. The class that appears most frequently among the k nearest data points is assigned as the class for the new data point. In other words, the decision is made through a voting process based on the k neighboring points. One significant advantage of this K-NN algorithm is its compatibility with parallel operations, allowing for efficient computations (Ioffe & Szegedy, 2015).

Parameters:

n\_neighbors = 5 (default)

metric= minkowski(default)

p=2(default)

**RandomForestClassifier**

Random Forests classifier (RFC) is a widely acclaimed ensemble learning technique in pattern recognition and machine learning. It has gained immense popularity due to its effectiveness and power, particularly in addressing high-dimensional classification and skewed problems. Tree classifiers suffer from high variance, meaning even slight changes in the training data can result in significantly different trees. This is because errors at higher nodes propagate down to the leaves in the hierarchical structure of tree classifiers. To address this, the "random forest" methodology, initially proposed by Ho, Amit, Geman, and later integrated by Breiman, introduces decision forests as an ensemble of decision trees. They function as a single classifier with multiple classification methods and parameters. Each tree in the forest classifies new input data, and the final classification is determined by majority voting. Random forests build binary sub-trees using bootstrap samples from the training data and randomly select a subset of features at each node. This approach combines Breiman's "bagging" technique and Ho's "random selection features." Bagging involves creating a set of classifiers by aggregating approximately two-thirds of the dataset through bootstrapping, while the remaining instances form an out-of-bag set for evaluation. Random feature selection, usually based on the square root of the total number of features, is applied at each node. The sub-trees in random forests are maximal trees without pruning (Azar et al., 2014).

Parameters:

**We will use all parameter with default values.**

**Evaluation**

**ROC AUC score**

**MCC**

**Reference**