PREDICTIVE MAINTENANCE FOR MACHINE TOOL FAILURES

1st TEJASWI REDDY ANAPALLI
Computer Science
University of Central Missouri
Lee Summit, US
txa05670@ucmo.edu

2nd BOPPANA VEERA VENKATA SATYANARAYANA
Computer Science
University of Central Missouri
Lee Summit, US
vxb08620@ucmo.edu

3rd GUTHIKONDA SAI PRANITHA
Computer Science
University of Central Missouri
Lee Summit, US
sxg08670@ucmo.edu

4th RAKESH PEDDAPALLI
Computer Science
University of Central Missouri
Lee Summit, US
rxp02940@ucmo.edu

Abstract*—Maintenance is mandatory for any corporate manufacturer administering intricate machinery and procedures on a daily basis to operate the business effectively, and responsibly, and to reach or surpass the bottom line. Manufacturing machinery is frequently used without a defined maintenance strategy. Due to unforeseen failures, such a technique frequently leads to unscheduled downtime. With data collecting and analysis, predictive maintenance foresees probable equipment breakdown. In order to forecast when service should be carried out, predictive maintenance procedures can be used to examine the situation of the equipment. Although service is only carried out when necessary, this method of management can ultimately lead to savings in costs over normal preventative maintenance. Addressing the necessity of predictive maintenance, this project focuses on applying different machine learning techniques to forecast the type of breakdown that occurs and at what time it's going to happen. Keeping the format and the properness of the dataset to build any machine learning model in mind, this project performs Exploratory Data Analysis(EDA) before building any model. The features in the dataset are Air temperature [kelvin], Type, Process temperature [kelvin], Rotational speed [rpm], Torque [Nm], and Tool wear [min] which defines the failure type and when a failure occurs. The data cleaning step is taken to analyze any noise or inconsistencies in the dataset. This project used Logistic Regression, Support Vector Classifier, K- Nearest Neighbor Classifier, and Random Forest to forecast the machine breakdown and compared the results of all four models. While the Logistic Regression is the same, the Support Vector Classifier and the other two models are built using GridSearchCV. The most accurate model out of all the training classification algorithms is chosen. The confusion matrix will be shown for those concepts, showing the precision and recall for each model as well as the real positive and real negative rate. On Comparing the results the Random Forest model with GridSearchCV gave the better performance out of the other three models.

***Index Terms*—Predictive Maintenance, Equipment Breakdown, Machine Learning, Exploratory Data Analysis, Data Cleaning.**

I. Introduction

In order to forecast when maintenance has to be done, predictive maintenance techniques are made to help assess the state of in-service machinery. Data gathering, administration, and wise utilization form the foundation of condition monitoring. This approach focuses on "equipment care," providing maintenance and repairs when a resource doesn't achieve predetermined performance goals. Recognizing what causes assets to fail as well as recognizing the alarm indications of prospective issues or failure are necessary for modifying a Predictive Maintenance approach. Where sensory receptors or ears may fail to be able to do so, computerized technologies offer many industries a practical answer for locating and gathering confidential material from hardware that is primarily motorized [1]. Utilizing previous and authentic data from multiple areas of your operation, predictive maintenance foresees issues before they arise. Additionally, efficiency, quality of products, and the general efficacy of industrial settings can all be improved with predictive maintenance. Predictive maintenance programs, as opposed to traditional preventative maintenance, are based on data gathered from detectors and planning is the process [2], [3]. Predictive maintenance reduces costs while making the most use of its resources. The abnormalities and failing trends will be found via predictive maintenance, which will also give early warnings. Predictive maintenance has a number of benefits, such as minimization or almost complete elimination of unplanned equipment damage due to platform or maintenance issues, increased use of labor increased capacity for production decreased maintenance expenses and longer hardware longevity.

EDA, Exploratory data analysis, as the name suggests is a process to explore the data in several and all the ways possible. It is used to perform investigations on the data to obtain several trends within the data and draw conclusions using graphical representations. It is not a tool but is implemented basically in python programming language as the language provides several flexible libraries to deal with data frames, NumPy arrays, and also plotting libraries. This is a necessary and important move to smooth the provided data set(s) in a varying type of investigation to comprehend the

perspectives of the key properties of the various bodies of the data set, such as column(s), row(s), by applying Pandas, NumPy, Statistical Methods, and Data visualisation packages. EDA allows us to comprehend the given dataset, assist in its cleanup, gain a detailed view of the characteristics and the correlations between them, manage null values or operator mistakes, and identify anomalies. It also offers instructions for omitting/removing quasi-variables and keeping just necessary variables.

In the field of artificial intelligence, machine learning (ML) has become a potent tool for generating smart predictive algorithms across a variety of fields. In challenging and changing situations, ML techniques have the capacity to handle large features and multidimensional data and to uncover correlations within them [4]. Many businesses are having trouble adapting to the realities of AI technologies as Industry 4.0 proceeds to garner global attention. Condition monitoring has many tactical advantages, including aiding in asset periodic inspection and identifying when the repair is necessary. It goes without saying that the use of ML-based applications can result in significant cost savings, improved predictability, and greater system reliability. The semi-manual method ignores the machinery's more intricate adaptive behavior responses or the context-specific information pertaining to the overall production process. Machine learning algorithms, conversely, are fed data from the production line, including information gathered from sensors, Programmable logic controllers, scholars, SCADA, relevant information from sources like ERP, performance, and MES, as well as details about the production process, such as the coincidences of the equipment and the frequency of supply chain. When performing maintenance, it is always best to perform it precisely when and how it will be most effective. Condition monitoring doesn't require much more than a simple intuitive mathematical calculation when equipment circumstances are at a stage where correction or even substitution is required. In order to identify the characteristics of a system crash, condition monitoring uses machine learning and tries to learn from both real-time data and previous data. The ML-based forecast strategy eliminates wastage of resources and sub-optimal resource utilization for maintenance chores, in contrast to conventional maintenance processes that rely on the life cycle of mechanical components.

This paper used logistic regression, Random Forest, Support Vector Classifier, and K-Nearest Neighbors models to predict when a failure will occur and the type of the failure. The models used Air temperature [k], Type, Process temperature [k], Rotational Speed [rpm], Torque [Nm], Tool ware [min] as feature variables and Target as the response variable to predict when the failure is likely to occur and the attributes Air temperature [k], Type, Process temperature [k], Rotational Speed [rpm], Torque [Nm], and Tool ware [min] as features with Failure Type as a response variable for predicting the type of failure that will occur.

II. Motivation

Any type of gadget is susceptible to deterioration and harm. The regularity of their upkeep determines whether or not they will live longer. All machines eventually fail, but the effects might be very different. Organizations cannot afford any kind of breakdown, no matter how big or small. It has an impact on the speed of company achievement and operational efficiencies. A single missing bolt results in significant wear and tear, shortened equipment life, and unexpected downtime. You are left with increased recovery costs, decreased production value, no concurrent device inspection, and no operational visibility as a result. Maintenance management is the major focus of the most economical maintenance techniques. However, a key component of an effective carry-out of the activities approach is neglected by many maintenance departments. Companies require an effective maintenance policy to minimize the harm caused by breakdowns and prevent costly outages. This inspired us to explore machine learning techniques for predicting equipment failure using real-time data from the machinery. By identifying flaws at an early stage or the signs that may later create more problems, planned maintenance helps you move toward the future. Real-time observation and selection are facilitated for industries. Additionally, it aids in controlling the machine's lifespan, physical state, and productivity. Screening hardware that is high-cost to replace, sophisticated, or susceptible to a highly catastrophic collapse can benefit greatly from condition-based maintenance. Condition monitoring can minimize the amount of time spent performing maintenance and increase the amount of time that equipment is used productively. Condition monitoring is now a potent tool for site supervisors and building engineers because of developments in cloud computing, AI, and various wired leveraging IoT devices.

III. Objectives

This paper is focused on evaluating different Machine Learning algorithms in predicting when the failure occurs and what type of failure occurs. This paper has two main objectives:

- Predicting when a failure will occur using independent variables.
- Predicting type of failure will occur using predicting variables.

For this purpose, the first objective uses the Target attribute in the dataset as the response variable, and to achieve the second objective uses the Failure Type attribute of the dataset as the response variable. This paper explores the following Algorithms and analyses their performance on the dataset mentioned in section IV.

- Logistic Regression.
- Support Vector Classifier using GridSearchCV.
- Random Forest using GridSearchCV
- K-Nearest Neighbours using GridSearchCV

IV. Related Work

H. M. Hashemian et al.[1] discussed condition-based system maintenance for machinery and manufacturing uses. They attempted to use the sensor's current AC output to find obstructions in the pressure detecting lines and its preexisting DC output to confirm detector accuracy. Additionally, they reviewed wireless, test transmission, and other condition monitoring methods that use test sensors. The author came to the conclusion that industrial facilities should adopt predictive and interactive techniques that operate under the presumption that any malfunction can happen at any moment, rather than continuing to presume that malfunctions only happen after a certain period of time in service.

Sze-jung Wu et al.[2] talked about an automated neural network-based decision-assisting system for rotating maintenance and repair. In order to reduce the mean replacement cost across the life span of a component, the authors have devised a deterministic substitution strategy that may be utilised for condition-based predictive management. The approach they suggested is focused on how to use actual situation assessment data to estimate the lifespan of the system and choose the best replacement strategy for its components. They employed two possible cost techniques: the cost of a scheduled substitute and the value of corrective maintenance.

Emanuele Frontoni et al.[3] described the project HDOMO. They put out the novel concept of an adaptable deeply embedded efficient system, where several low-cost devices and sensors can evaluate usage patterns and quickly respond to their demands while also gathering statistical information that can be used for further computation. By modeling a home using each of the SOs HDOMO, the data processing software has been put to the test. The goal was to see how the platform behaved in high-speed conditions, or when every SO is concurrently transmitting status updates. Problems with data handling and storage have been brought to light by the tests. The data collecting and processing system underwent a validity test.

Thorsten Wuest e al.[4] Before proposing a structure for the broad subject of machine learning and providing an introduction to its fundamental language, they explored the constraints of efficient production systems, the constraints of machine learning, and its benefits from a mechanical point of view. The structure is classifying the available methods and applications into three categories: standard supervised learning, RL, and unsupervised machine learning. The traditional supervised learning algorithm SVMs was successfully applied in manufacturing, as demonstrated by the authors in their example. They claimed that the capability of Machine learning algorithms to handle large multidimensional challenges and datasets is one of their advantages

Thyago P. Carvalho et al.[5] using ML approaches, investigated a comprehensive literature review that covered the key PdM papers. As a result, the researchers were able to notice that each suggested approach only tackles a certain piece of equipment, making it more challenging to contrast it to other approaches. They may say that PdM itself arises as a new instrument for handling with service events and claimed that with the advancement of industry 4.0, PdM becomes more and more viable and promising. They noticed that Machine learning algorithms are gradually being used to create PdM applications and discovered that in some cases, combining PdM and ML produces beneficial outcomes with cost savings. The authors found that combining PdM approaches with modern sensor technology prevents the needless rebuilding of equipment, cuts costs, and boosts operational efficiencies, reliability, and sustainability.

Marina Paolanti,et al.[6] provided details on a Random Forest-based Machine Learning framework for Conditional Monitoring. By establishing the data collecting and data system analysis, using the machine learning approach, and contrasting it with the simulation tool analysis, the authors tested the on a real-world industry case. The data that was made accessible to Tool For Data analysis on the Azure Cloud environment by a variety of sensors, machine PLCs and communication networks were utilized by the authors. Dynamic decision rules can be used for planned maintenance using the proposed PdM methodology, which is accomplished by developing a random forest method on azure machine learning Studio. The study's findings demonstrated that the method worked as intended to anticipate various machine stages with a 95% accuracy rate using a collection of 530731 data values on 15 different hardware features that were gathered on a real-time basis from the evaluated cutting machine.

Li Bin et al.[7] In order to predict the failure of drilling tools, they suggested a novel forecasting technique. The authors' suggested model first chooses a number of important variables that have a significant impact on drilling machine breakdowns as input characteristics of SVM, and then, to boost performance, optimization using particle swarms is used to optimise the nuclear parameters of SVM. The authors discovered that their model performed well and accurately when compared to actual engineering data, offering a fresh way to predict drilling tool failure. The PSO-SVM joint prediction model, which has been used in real-world projects, fully exploits the singular superiority of SVM in handling the categorized learning of small sets of extracts as well as the descriptor of global parallel search optimization through the use of particle swarm optimization.

Sohyung Cho et al.[8] created and put into use a milling process-based support vector regression tool breakage detection system. To create the model, the authors used several sensors. Their study demonstrates that the rate of tool breakage detection can be raised by using cooperative multiple sensors. They found that their approach (SVR) operates well and with a strict predefined threshold for tool breakage determination when compared to a model constructed using a conventional multiple variable regression approach. According to the authors, this finding suggests that the model utilising SVR can be advised because it was found to have somewhat higher accuracy in a large-scale flow shop environment with a conservative instrument replacement plan where even a small performance

increase results in significant savings. The authors noted that a very difficult procedure of modifying design parameters is required to train their suggested model.

Hongfei Li et al.[9] reported the study on large and multi-predictive modeling for railway efficient maintenance management along with numerous analytical methodologies like correlation test, inferential analysis, time series forecasting, and pattern recognition to extract knowledge rules and breakdown prediction model. This was done using huge quantities of historical sensor data, combined with breakdown data, service operation data, examination agenda data, train type information, and weather information. They stressed how the human capacity for interpretability aids the maintenance staff's decision-making process. According to the authors, many other companies that employ sensor networks for equipment health assessment, such as those that maintain aircraft, large earthmoving equipment, power plants, chemical plants, etc., can more broadly use their methods.

Xiang Li et al.[10] combines the actual forecasting work for equipment maintenance, which is based on engine lifespan prediction data, and methodically analyses the construction concepts, procedures, and Python implementation concepts utilizing the random forest model for machine learning.

Chang-Ching Lin et al.[11] introduced the CMAC neural network-based device performance estimation model, which they used to merge sensory inputs and to forecast machine reliabilities, and they combined the conventional reliability modeling approach and resonance machine situation observing methods to estimate machine reliability. Results from the CMAC-PEM have been validated using the Weibull proportional hazards model. They asserted that the CMAC-PEM is a trustworthy and dependable method for online machine reliability analysis and can realize the notion of condition-based predictive maintenance.

V. Data Preparation

Data Preparation is the process of collecting the data, cleaning, and preparing the data to train the model. It is crucial for any Machine Learning model to have clean data to give accurate predictions and hence this process also consists of data cleaning procedure.

A. Dataset Collection

To check and analyze the capability of various machine learning algorithms on predicting the time and type of failure that may occur, we have considered the dataset from [22]. The dataset consists of 10000 rows and 10 columns. The columns include UDI, the Product ID, and the Type which is a categorical variable of three categories Low(L), Medium(M), and High(H). The Attribute Failure Type contains No Failure, Heat Dissipation Failure, Over Strain Failure, Power Failure, Random Failure, and Tool Wear Failure. The Target attribute contains 0 and 1 where 0 denotes No Failure and 1 denotes any one of the Failures. This dataset after the data cleaning process is used to train and test the considered machine learning models.

B. Data Cleaning

Before giving the dataset to the model to get trained, it is important to clean the data for fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. For the dataset considered in this process, there is no cell that is null and hence the following data-cleaning steps have been considered:

1) Dropping the unwanted features
2) Visualizing the data
3) Encoding
4) Scaling

C. Dropping the unwanted features

This stage carefully removes any features or undesirable information from your dataset, such as redundant or pointless information. When you observe observations that aren't related to the particular issue you are attempting to study, those discoveries are deemed irrelevant. This can improve analytical effectiveness, reduce deviance from your main objective, and produce a dataset that is easier to handle and performs better. In the dataset considered, out of all the 10 attributes, the attributes UDI (Unique Device Identification) and Product ID have nothing to do with classification because the device may not malfunction based on its ID. So, these two attributes are removed from the dataset and the updated dataset is considered for further analysis. The columns are dropped using the drop() function with its attribute column being the list of columns to be removed, inplace been set to true, and axis to 1. The result is the dataset without the attributes UDI and Product ID.

D. Visualizing the data

The data visualization step gives a clear graphical analysis of different representations and attributes in the dataset. The pictorial display of information and data is known as data visualization. Data visualization tools offer an easy approach to observing and analyzing trends, outliers, and themes in the data by utilizing visual cues like charting, infographics, and mappings. By visualizing data we can easily share information, Interactively explore opportunities and also visualize patterns and relationships. For the dataset chosen, we have visualized all the categorical variables in the dataset like Type, Failure type and Target. The countplot() function is used to know the no. of machines or observations under each category of all categorical variables. Fig 1 represents the result of applying countplot() where the target attribute takes the value of 1 and counts the failure types.

Fig 2, and Fig 3 represent the count of occurrences of the Type and the Target attributes.

E. Encoding Categorical Variables

Encoding categorical variables is one of the important data preparation tasks. The real-life data may consist of attributes that are categorical string values. Most machine learning models perform several mathematical computations to reach

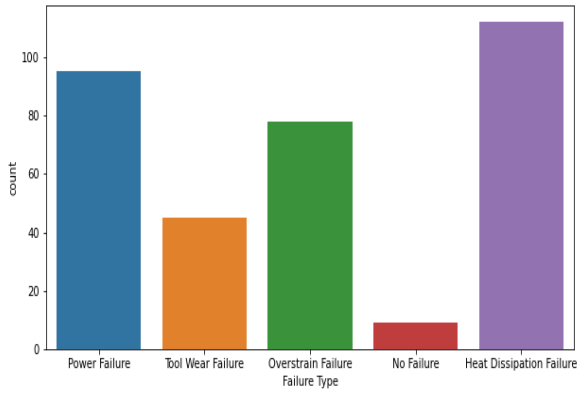

Fig. 1. Countplot on Failure Type

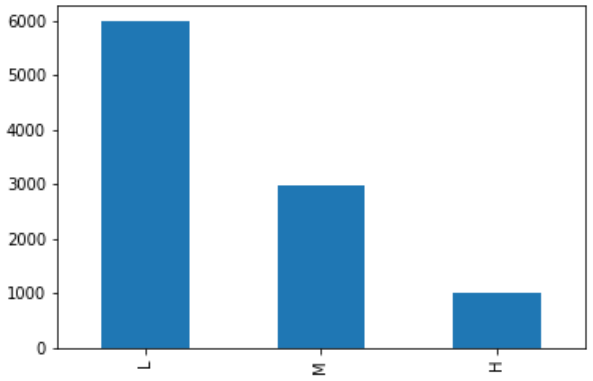

Fig. 2. Bar plot on the 'Type' attribute

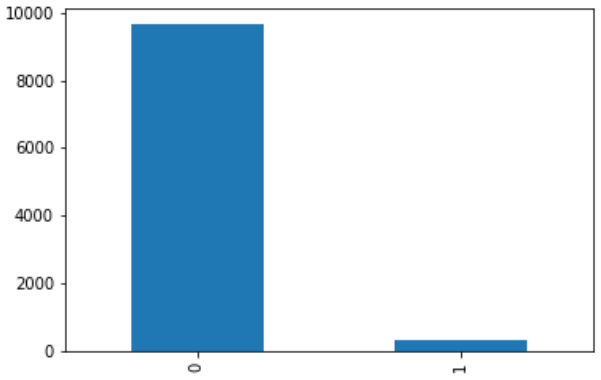

Fig. 3. Bar plot on the 'Target' attribute

their goal and hence they work only on numerical data and on other data types which are considered by that algorithm. Since the model will not work on the string values it is necessary to convert these string values into integer values. Encoding categorical variables is all about this process. Out of the many different types of encoding, we used the label encoding technique which converts each label into integer values and the encoded data represents the sequence of labels. In this dataset there are 3 attributes that are categorical. The type attribute is categorical consisting three categories Low(L), Medium(M) and High(H). The failure type attribute is also categorical consisting five categories which are Heat Dissipation Failure, Over Strain Failure, Power Failure, Random Failure, and Tool Wear Failure. The label encoding technique is applied to these variables only. After applying label encoding these attributes, the unique values of the Type attribute are [1,2,3] where 1 represents Low, 2 represents Medium and 3 represents High. The unique values of Failure type attribute are [0,1,2,3,4,5] where 0 represents Heat Dissipation Failure, 1 represents No Failure, 2 represents Over Strain Failure, 3 represents Power Failure, 4 represents Random Failure and 5 represents Tool Wear Failure.

F. Scaling

When continuous independent variables are measured at several scales, the idea of standardisation becomes apparent. This indicates that these factors do not contribute equally to the analysis. Scaling is the process of changing data to match a predetermined range, such as 0-100 or 0-1. When employing techniques dependent on how far away data points are, such as support vector machines (SVM) or k-nearest neighbours (KNN), you should scale the data. These algorithms give the same emphasis to changes of "1" in any numerical feature. By scaling your variables, you can make it easier to compare various variables side by side.

In the dataset here, The attribute 'air Temperature [k]' has values ranging from 295.3 to 304.5. The attribute 'Rotational speed [rpm]' ranges from 1168 to 2886 whereas the attribute 'Torque [Nm]' ranges from 3.8 to 76.6. The attribute 'Tool wear [min]' takes values ranging from 0 to 253. Therefore the attributes that take large values will be given more weightage by the model irrespective of whether or not those attributes contribute to such weightage towards the goal of the model. Therefore, it is required to transform the data to comparable scales. The idea is to rescale an original variable to have an equal range and/or variance.

The min-max scaling method is used in this paper to scale all the numerical variables in the dataset. The standardized value using this method lies between 0 and 1 and hence is also called 0-1 scaling. The mathematical representation of the min-max scaler is represented in "(1)".

$$x = min(x)/(max(x) - min(x)) \tag{1}$$

In this project, the attributes 'Air temperature [K]', 'Process temperature [K]', 'Rotational speed [rpm]', 'Torque [Nm]',

and 'Tool wear [min]' are all standardized using MinMaxScaler() function.

VI. Proposed Classification Models

A. Logistic Regression

By measuring each independent variable's distinct contribution, logistic regression is a quick and effective technique to examine the impact of a group of independent variables on a binary outcome. Logistic regression iteratively determines the strongest linear combination of variables with the highest likelihood of identifying the observed outcome using elements of linear regression indicated in the logit scale[12]. The "equation(2)" for logistic regression is as follows:

$$ProbabilityOfOutcome(\hat{Y}_i) = \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_i X_i}}{1+e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_i X_i}} \quad (2)$$

- Here $\hat{Y}$i represents the estimated probability of being in one binary outcome category (i) versus the other
- Here $e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_i X_i}$ represents the linear regression equation for independent variables expressed in the logit scale

The logit scale transforms the original equation of linear regression to obtain natural log of the odds of being in one outcome category ($\hat{Y}$) versus the other category (1 –$\hat{Y}$) is given by "(3)"

$$ln(\hat{Y}/1–\hat{Y}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_i X_i \quad (3)$$

Despite its name, logistic regression is more of a classification model than a regression model. For situations involving binary and linear classification, logistic regression is a straightforward and more effective approach. It's a classification model that's incredibly simple to implement and performs admirably with linearly separable classes.

B. GridSearchCV

GridSearchCV() is a function available in the Sklearn framework. GridSearchCV is used to accelerate the Hyperparameter Tuning. Hyperparameters are those parameters that cannot be learned directly. Prior to starting training, humans frequently choose them based on some intuition or trial and error. By enhancing the model's functionality, such as increasing the model's complexity or learning rate, these parameters demonstrate their significance. Models may have a large number of hyper-parameters, and determining the ideal set of parameters can be approached as a search issue. Hyperparameter tuning aids in identifying the best hyperparameter values for a specific model. The optimal values for hyperparameters cannot be determined at the outset. Hyperparameters would need to be manually adjusted, which would take a lot of time and resources. GridSearchCV tests all potential combinations of the dictionary's values using the Cross-Validation method then examines the model for each one. We can determine the accuracy for each set of hyperparameters and select the one that performs the best as a consequence.

C. Support Vector Classifier using GridSearchCV

Support vector machines are part of the category of supervised machine learning methods, which are typically applied to classification tasks. The Support Vector Machine's fundamental principle is to take data with relatively low dimensions and transform it into data with greater dimensions[15]. The data is then divided using a hyperplane to classify it into various groups. By tolerating misclassifications, a support vector classifier is utilised to determine the appropriate decision boundary. The choice of the kernel function and its parameters is critical when using SVM to tackle real-world problems. By choosing the right kernel function and parameters, one may create an SVM classifier with strong generalization capabilities. The most often used kernel function is the RBF, and it only has two parameters: C and γ.

Finding the best hyper-parameter for an SVM model is a highly challenging issue. These hyper-parameters include things like what C or gamma values to utilise. But it can be discovered by simply attempting all possible combinations and observing which inputs are most effective. Its main purpose is to build a grid of hyper-values and then simply test every possible combination of those parameters. So, here we employ the GridSearchCV approach to identify the most advantageous hyper-parameters and hence enhance the accuracy/prediction outcomes.

Here we are using GridSearchCV so we have to define a parameter grid which is a dictionary with parameter names as keys and lists of parameter values. To improve the model accuracy, there are several parameters need to be tuned. Two major parameters including Kernels and C. Kernels: A kernel's primary job is to turn a low-dimensional input space into a higher-dimensional one. It is most helpful in problems with non-linear separation. Here RBF and linear kernel are used. C (Regularization): The penalty parameter, C, stands for the term that indicates the misclassification or error. The SVM optimization is informed by the misclassification or error term regarding the acceptable level of error.

D. Random Forest using GridSearchCV

Popular machine learning algorithm Random Forest is a part of the supervised learning methodology. It can be applied to ML issues involving both classification and regression. It is built on the idea of ensemble learning, which is a method of integrating various classifiers to address difficult issues and enhance model performance. Random Forest, as the name implies, is a classifier that uses a number of decision trees on different subsets of the provided dataset and averages them to increase the dataset's predictive accuracy. Instead of depending on a single decision tree, the random forest uses estimates from each tree and predicts the result based on the votes of the majority of predictions[18]. Higher accuracy and overfitting are prevented by the larger number of trees in the forest.

Here we are using Randomforest using GridSearchCV so the parameter grid contains the max_ depth and criterion. The depth of each tree in the forest is represented by max_depth. The more splits a tree has, the more information it can collect

about the data. We plot the training and test errors after fitting each decision tree with a depth ranging from 1 to 32. The purpose of a criterion is to assess a split's quality. Here we considered both Gini and entropy as criterion to obtain the best outcome. The gini impurity counts the number of times a dataset piece will be incorrectly identified when it is randomly labelled. The Gini Index has a minimum value of 0. When a node is pure, which occurs when every element it contains belongs to a single, distinct class, this occurs. This node won't be split once more as a result. Therefore, the features with a lower Gini Index choose the best split. Additionally, it is at its highest value when the probabilities for the two classes are equal. Entropy is a unit of measurement for information that depicts the disorder of the target's features. The feature with the lowest entropy selects the optimal split, just like the Gini Index does. When the probability of the two classes is equal, it reaches its highest value, and a node is pure when the entropy is at its lowest point, which is zero.

E. KNN using GridSearchCV

One of the simplest machine learning algorithms, based on the supervised learning method, is K-Nearest Neighbour. The K-NN algorithm makes the assumption that the new case and the existing cases are comparable, and it places the new instance in the category that is most like the existing categories. A new data point is classified using the K-NN algorithm based on similarity after all the existing data has been stored. This means that utilising the K-NN method, fresh data can be quickly and accurately sorted into a suitable category[20]. Although the K-NN approach is most frequently employed for classification problems, it can also be utilised for regression. Since K-NN is a non-parametric technique, it makes no assumptions about the underlying data. It is also known as a lazy learner algorithm since it saves the training dataset rather than learning from it immediately. Instead, it uses the dataset to perform an action when classifying data.

The KNN method simply saves the information during the training phase, and when it receives new data, it organizes it into a category that is quite similar to the new data. Here GridSearchCV is used to find the best number of neighborhood points.

VII. Results & Comparision

A. Predicting when a failure will occur using independent variables.

Predictor Variables: Air temperature [k], Type, Process temperature [k], Rotational Speed [rpm], Torque [Nm], Tool ware [min].

Response Variables: Target

1) Logistic regression: A LogisticRegression() classifier is used here. The fit() function takes the parameters of x_train and y_train data. The predict() function takes test data as input and predicts the outcome. To calculate metrics here we have used the accuracy_score()function which is a ratio of correctly predicted observations to the total observations and the confusion_matrix() function is used to obtain a table that is used to define the performance of classification algorithms. The classification_report() function is used to measure the quality of predictions from a classification algorithm. The cohen_kappa_score() function balances the accuracy of a random classifier as assessed by the predicted accuracy and measures how well the instances identified by the machine learning classifier matched the data designated as ground truth. All these functions take the predicted values and the test data as parameters. For the given dataset when logistic regression is applied the accuracy score obtained is 96.9%. The kappa statistic obtained is 0.020. The metrics obtained are depicted in Fig 4.

```
Accuracy Score of logistic regression [Target] :  0.969
Confusion Matrix of logistic regression [Target] :
 [[2905    0]
 [  94    1]]
Classification Report of logistic regression[Target] :
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      2905
           1       1.00      0.01      0.02        95

    accuracy                           0.97      3000
   macro avg       0.98      0.51      0.50      3000
weighted avg       0.97      0.97      0.95      3000

Cohen Kappa: 0.020
```

Fig. 4. Metrics obtained when Logistic Regression is applied

2) SupportVectorClassifier using GridSearchCV: Here the parameter grid contains C, degree, and kernel.GridSearchCV takes the following parameters: estimator which is a Support Vector Classifier, Parameter grid which is a dictionary, Refit which is an estimator using the best-found parameters on the whole dataset by default set to True and Verbose which controls the verbosity, here it is set to 3 where the score is also displayed. After fitting the model we can find the best parameters using the best_params_() function. The best parameters are obtained when "C=10,degree=1,kernel=rbf". We use the fit() function to train the model with the data and predict() function to test. To calculate metrics here we have used the accuracy_score() function which is a ratio of correctly predicted observations to the total observations. The confusion_matrix() function is used to obtain a table that is used to define the performance of classification algorithms. The classification_report() is used to measure the quality of predictions from a classification algorithm. The cohen_kappa_score() function balances the accuracy of a random classifier as assessed by the predicted accuracy and measures how well the instances identified by the machine learning classifier matched the data designated as ground truth. All these functions take the predicted values and the test data as parameters. For the given dataset when SVC using GridSearchCV is applied the accuracy score obtained is 97.5%. The kappa statistic obtained is 0.341. The metrics obtained are depicted in Fig 5.

```
Best params from Support Vector Classifier [Target] :  {'C': 10, 'degree': 1, 'kernel': 'rbf'}
Accuracy Score of Support Vector Classifier [Target] :  0.975
Confusion Matrix of Support Vector Classifier [Target] :
 [[2905    0]
 [  75   20]]
Classification Report of Support Vector Classifier[Target] :
              precision    recall  f1-score   support

           0       0.97      1.00      0.99      2905
           1       1.00      0.21      0.35        95

    accuracy                           0.97      3000
   macro avg       0.99      0.61      0.67      3000
weighted avg       0.98      0.97      0.97      3000

Cohen Kappa: 0.341
```

Fig. 5. Metrics obtained when SVC using GridSearchCV is applied

```
Best params from Random Forest Classifier [Target] :  {'criterion': 'entropy', 'max_depth': 8}
Accuracy Score of Random Forest Classifier [Target] :  0.98
Confusion Matrix of Random Forest Classifier [Target] :
 [[2901    4]
 [  56   39]]
Classification Report of Random Forest Classifier[Target] :
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      2905
           1       0.91      0.41      0.57        95

    accuracy                           0.98      3000
   macro avg       0.94      0.70      0.78      3000
weighted avg       0.98      0.98      0.98      3000

Cohen Kappa: 0.556
```

Fig. 6. Metrics obtained when RandomForest using GridSearchCV is applied

3) Random Forest using GridSearchCV: Here we are using Randomforest using GridSearchCV so the parameter grid contains the max_ depth and criterion.GridSearchCV takes the following parameters: estimator which is RandomForest-Classifier(), Parameter grid which is a dictionary as defined above, Refit which is an estimator using the best-found parameters on the whole dataset by default set to True and Verbose which controls the verbosity, here it is set to 3 where the score is also displayed. After fitting the model we can find the best parameters using the best_params_() function. The best parameters are obtained when "criterion= entropy, max_depth= 8". We use the fit() function to train the model with the data and predict function to test. To calculate metrics here we have used the accuracy_score() function which is a ratio of correctly predicted observations to the total observations. The confusion_matrix() function is used to obtain a table that is used to define the performance of classification algorithms. The classification_report() is used to measure the quality of predictions from a classification algorithm. The cohen_kappa_score() function balances the accuracy of a random classifier as assessed by the predicted accuracy and measures how well the instances identified by the machine learning classifier matched the data designated as ground truth. All these functions take the predicted values and the test data as parameters. For the given dataset when RandomForest using GridSearchCV is applied the accuracy score obtained is 98%. The kappa statistic obtained is 0.556. The metrics obtained are depicted in Fig 6.

4) KNN using GridSearchCV: Here the parameter_grid takes n_neighbours as the parameter which tells the number of neighborhood points. GridSearchCV takes the following parameters: estimator which is KNeighborsClassifier(), Parameter grid which is a dictionary as defined above, Refit which is an estimator using the best-found parameters on the whole dataset by default set to True and Verbose which controls the verbosity, here it is set to 3 where the score is also displayed. After fitting the model we can find the best parameters using the best_params_() function. The best parameters are obtained when "n_neighbours=3". We use the fit() function to train the model with the data and predict function to test. To calculate metrics here we have used the accuracy_score() function which is a ratio of correctly predicted observations to the total observations. The confusion_matrix() function is used to obtain a table that is used to define the performance of a classification algorithm. The classification_report() is used to measure the quality of predictions from a classification algorithm. The cohen_kappa_score() function balances the accuracy of a random classifier as assessed by the predicted accuracy and measures how well the instances identified by the machine learning classifier matched the data designated as ground truth. All these functions take the predicted values and the test data as parameters. For the given dataset when RandomForest using GridSearchCV is applied the accuracy score obtained is 97.6%. The kappa statistic obtained is 0.468. The metrics obtained are depicted in Fig 7.

```
Best params from K- Nearest Neighbor [Target] :  {'n_neighbors': 3}
Accuracy Score of K- Nearest Neighbor [Target] :  0.976
Confusion Matrix of K- Nearest Neighbor [Target] :
 [[2895   10]
 [  62   33]]
Classification Report of K- Nearest NeighborTarget] :
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      2905
           1       0.77      0.35      0.48        95

    accuracy                           0.98      3000
   macro avg       0.87      0.67      0.73      3000
weighted avg       0.97      0.98      0.97      3000

Cohen Kappa: 0.468
```

Fig. 7. Metrics obtained when KNN using GridSearchCV is applied

B. Predicting type of failure will occur using predicting variables

Predictor Variables: Air temperature [k], Type, Process temperature [k], Rotational Speed [rpm], Torque [Nm], Tool ware [min].

Response Variables: Failure Type

1) Logistic Regression: A LogisticRegression() classifier is used here. The fit() function takes the parameters of x_train and y_train data. The predict() function takes test data as input and predicts the outcome. To calculate metrics here we have used the accuracy_score()function which is a ratio of correctly predicted observations to the total observations and the confusion_matrix() function is used to obtain a table that is used to define the performance of classification algorithms. The classification_report() function is used to measure the quality of predictions from a classification algorithm. The cohen_kappa_score() function balances the accuracy of a random classifier as assessed by the predicted accuracy and measures how well the instances identified by the machine learning classifier matched the data designated as ground truth. All these functions take the predicted values and the test data as parameters. For the given dataset when logistic regression is applied the accuracy score obtained is 98.7%. The kappa statistic obtained is 0.801. The metrics obtained are depicted in Fig 8.

```
Accuracy Score of Logistic Regression [Failure Type] :  0.9876666666666667
Confusion Matrix of Logistic Regression [Failure Type] :
 [[  23    0    0    0    0    0]
 [   2 2900    0    1    0    0]
 [   3    0   19    0    0    0]
 [   7    0    8   16    0    0]
 [   0    7    0    0    0    0]
 [   2    0    7    0    0    5]]
Classification Report of Logistic Regression [Failure Type] :
              precision    recall  f1-score   support

           0       0.62      1.00      0.77        23
           1       1.00      1.00      1.00      2903
           2       0.56      0.86      0.68        22
           3       0.94      0.52      0.67        31
           4       0.00      0.00      0.00         7
           5       1.00      0.36      0.53        14

    accuracy                           0.99      3000
   macro avg       0.69      0.62      0.61      3000
weighted avg       0.99      0.99      0.99      3000

Cohen Kappa: 0.801
```

Fig. 8. Metrics obtained when Logistic Regression is applied

2) SVC using GridSearchCV: GridSearchCV takes the following parameters: estimator which is a Support Vector Classifier, Parameter grid which is a dictionary, Refit which is an estimator using the best-found parameters on the whole dataset by default set to True and Verbose which controls the verbosity, here it is set to 3 where the score is also displayed. After fitting the model we can find the best parameters using the best_params_() function. The best parameters are obtained when "C=10,degree=1,kernel=linear". We use the fit() function to train the model with the data and predict() function to test. To calculate metrics here we have used the accuracy_score() function which is a ratio of correctly predicted observations to the total observations. The confusion_matrix() function is used to obtain a table that is used to define the performance of a classification algorithm. The classification_report() is used to measure the quality of predictions from a classification algorithm. The cohen_kappa_score() function balances the accuracy of a random classifier as assessed by the predicted accuracy and measures how well the instances identified by the machine learning classifier matched the data designated as ground truth. All these functions take the predicted values and the test data as parameters. For the given dataset when SVC using GridSearchCV is applied the accuracy score obtained is 99.3% The kappa statistic obtained is 0.892. The metrics obtained are depicted in Fig 9.

```
Best params from Support Vector Classifier [Failure Type] :  {'C': 10, 'degree': 1, 'kernel': 'linear'}
Accuracy Score of Support Vector Classifier [Failure Type] :  0.9933333333333333
Confusion Matrix of Support Vector Classifier [Failure Type] :
 [[  23    0    0    0    0    0]
 [   1 2901    0    1    0    0]
 [   1    0   21    0    0    0]
 [   1    0    5   25    0    0]
 [   0    7    0    0    0    0]
 [   1    0    3    0    0   10]]
Classification Report of SVC using GridSearchCV [Failure Type] :
              precision    recall  f1-score   support

           0       0.85      1.00      0.92        23
           1       1.00      1.00      1.00      2903
           2       0.72      0.95      0.82        22
           3       0.96      0.81      0.88        31
           4       0.00      0.00      0.00         7
           5       1.00      0.71      0.83        14

    accuracy                           0.99      3000
   macro avg       0.76      0.75      0.74      3000
weighted avg       0.99      0.99      0.99      3000

Cohen Kappa: 0.892
```

Fig. 9. Metrics obtained when SVC using GridSearchCV is applied

3) RandomForest using GridSearchCV: GridSearchCV takes the following parameters: estimator which is RandomForestClassifier(), Parameter grid which is a dictionary as defined above, Refit which is an estimator using the best-found parameters on the whole dataset by default set to True and Verbose which controls the verbosity, here it is set to 3 where the score is also displayed. After fitting the model we can find the best parameters using the best_params_() function. The best parameters are obtained when "criterion= entropy, max_depth= 7". We use the fit() function to train the model with the data and predict function to test. To calculate metrics here we have used the accuracy_score() function which is a ratio of correctly predicted observations to the total observations. The confusion_matrix() function is used to obtain a table that is used to define the performance of classification algorithms. The classification_report() is used to measure the quality of predictions from a classification algorithm. The cohen_kappa_score() function balances the accuracy of a random classifier as assessed by the predicted accuracy and measures how well the instances identified by the machine learning classifier matched the data designated as ground truth. All these functions take the predicted values and the test data as parameters. For the given dataset when RandomForest using GridSearchCV is applied the accuracy score obtained is 99.5%. The kappa statistic obtained is 0.929. The metrics obtained are depicted in Fig 10.

```
Best params from Random Forest Classifier [Failure Type] :  {'criterion': 'entropy', 'max_depth': 7}
Accuracy Score of Random Forest Classifier [Failure Type] :  0.9956666666666667
Confusion Matrix of Random Forest Classifier [Failure Type] :
 [[ 23    0    0    0    0    0]
 [   0 2903    0    0    0    0]
 [   1    0   20    0    0    1]
 [   0    0    3   28    0    0]
 [   0    7    0    0    0    0]
 [   0    0    1    0    0   13]]
Classification Report of Random Forest using GridSearchCV [Failure Type] :
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        23
           1       1.00      1.00      1.00      2903
           2       0.83      0.91      0.87        22
           3       1.00      0.90      0.95        31
           4       0.00      0.00      0.00         7
           5       0.93      0.93      0.93        14

    accuracy                           1.00      3000
   macro avg       0.79      0.79      0.79      3000
weighted avg       0.99      1.00      0.99      3000

Cohen Kappa: 0.929
```

Fig. 10. Metrics obtained when RandomForest using GridSearchCV is applied

```
Best params from KNN [Failure Type] :  {'n_neighbors': 4}
Accuracy Score of KNN [Failure Type] :  0.9893333333333333
Confusion Matrix of KNN [Failure Type] :
 [[  22    0    1    0    0    0]
 [   2 2900    0    0    0    1]
 [   1    0   19    1    0    1]
 [   8    0    7   15    0    1]
 [   0    7    0    0    0    0]
 [   1    0    0    1    0   12]]
Classification Report of KNN using GridSearchCV [Failure Type] :
              precision    recall  f1-score   support

           0       0.65      0.96      0.77        23
           1       1.00      1.00      1.00      2903
           2       0.70      0.86      0.78        22
           3       0.88      0.48      0.62        31
           4       0.00      0.00      0.00         7
           5       0.80      0.86      0.83        14

    accuracy                           0.99      3000
   macro avg       0.67      0.69      0.67      3000
weighted avg       0.99      0.99      0.99      3000

Cohen Kappa: 0.828
```

Fig. 11. Metrics obtained when KNN using GridSearchCV is applied

4) KNN using GridSearchCV: GridSearchCV takes the following parameters: estimator which is KNeighborsClassifier(), Parameter grid which is a dictionary as defined above, Refit which is an estimator using the best-found parameters on the whole dataset by default set to True and Verbose which controls the verbosity, here it is set to 3 where the score is also displayed. After fitting the model we can find the best parameters using the best_params_() function. The best parameters are obtained when "n_neighbours=4". We use the fit() function to train the model with the data and predict function to test. To calculate metrics here we have used the accuracy_score() function which is a ratio of correctly predicted observations to the total observations. The confusion_matrix() function is used to obtain a table that is used to define the performance of classification algorithms. The classification_report() is used to measure the quality of predictions from a classification algorithm. The cohen_kappa_score() function balances the accuracy of a random classifier as assessed by the predicted accuracy and measures how well the instances identified by the machine learning classifier matched the data designated as ground truth. All these functions take the predicted values and the test data as parameters. For the given dataset when RandomForest using GridSearchCV is applied the accuracy score obtained is 98.93%. The kappa statistic obtained is 0.828. The metrics obtained are depicted in Fig 11.

CONCLUSION

This paper evaluated Logistic Regression, Support Vector Classifier, Random Forest Classifier, and K-Nearest Neighbors algorithms to predict when a failure may occur and what type of failure occurs at that time. It has been observed that Random Forest Classifier gave a better performance with 98% accuracy compared to the other three models.

REFERENCES

[1] H. M. Hashemian and W. C. Bean, "State-of-the-art predictive maintenance techniques," IEEE Transactions on Instrumentation and measurement, vol. 60, no. 10, pp. 3480–3492, 2011.

[2] S.-j. Wu, N. Gebraeel, M. A. Lawley, and Y. Yih, "A neural network integrated decision support system for condition-based optimal predictive maintenance policy," IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, vol. 37, no. 2, pp. 226–236, 2007.

[3] E. Frontoni, R. Pollini, P. Russo, P. Zingaretti, and G. Cerri, "Hdomo:Smart sensor integration for an active and independent longevity of the elderly," Sensors, vol. 17, no. 11, p. 2610, 2017.

[4] Wuest, T., Weimer, D., Irgens, C., Thoben, K. D.(2016). Machine learning in manufacturing: advantages, challenges, and applications. Production Manufacturing Research, 4(1), 23-45.

[5] Carvalho, T. P., Soares, F. A., Vita, R., Francisco, R. D. P., Basto, J. P., Alcalá, S. G. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. Computers Industrial Engineering, 137, 106024.

[6] Paolanti, M., Romeo, L., Felicetti, A., Mancini, A., Frontoni, E., Loncarski, J. (2018, July). Machine learning approach for predictive maintenance in industry 4.0. In 2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA) (pp. 1-6). IEEE.

[7] Bin, L., Min, Y. (2012, August). Analysis model of drilling tool failure based on PSO-SVM and its application. In 2012 Fourth International Conference on Computational and Information Sciences (pp. 1307-1310). IEEE.

[8] Cho, S., Asfour, S., Onar, A., Kaundinya, N. (2005). Tool breakage detection using support vector machine learning in a milling process. International Journal of Machine Tools and Manufacture, 45(3), 241-249.

[9] Li, H., Parikh, D., He, Q., Qian, B., Li, Z., Fang, D., Hampapur, A. (2014). Improving rail network velocity: A machine learning approach to predictive maintenance. Transportation Research Part C: Emerging Technologies, 45, 17-26.

[10] Li, X., Wei, L., He, J. (2018, December). Design and Implementation of Equipment Maintenance Predictive Model Based on Machine Learning. In IOP Conference Series: Materials Science and Engineering (Vol. 466, No. 1, p. 012001). IOP Publishing.

[11] Lin, C. C., Tseng, H. Y. (2005). A neural network application for reliability modelling and condition-based predictive maintenance. The International Journal of Advanced Manufacturing Technology, 25(1), 174-179.

[12] DeMaris, A., Selman, S. H. (2013). Logistic regression. In Converting Data into Evidence (pp. 115-136). Springer, New York, NY.

[13] Stoltzfus, J. C. (2011). Logistic regression: a brief primer. Academic emergency medicine, 18(10), 1099-1104.

[14] Kleinbaum, D. G., Dietz, K., Gail, M., Klein, M., Klein, M. (2002). Logistic regression (p. 536). New York: Springer-Verlag.

[15] Xiao, T., Ren, D., Lei, S., Zhang, J., Liu, X. (2014, June). Based on grid-search and PSO parameter optimization for Support Vector Machine. In Proceeding of the 11th World Congress on Intelligent Control and Automation (pp. 1529-1533). IEEE.

[16] Fayed, H. A., Atiya, A. F. (2019). Speed up grid-search for parameter selection of support vector machines. Applied Soft Computing, 80, 202-210.
[17] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.
[18] Probst, P., Wright, M. N., Boulesteix, A. L. (2019). Hyperparameters and tuning strategies for random forest. Wiley Interdisciplinary Reviews: data mining and knowledge discovery, 9(3), e1301.
[19] Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K. (2003, November). KNN model-based approach in classification. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 986-996). Springer, Berlin, Heidelberg.
[20] Zhang, S., Li, X., Zong, M., Zhu, X., Wang, R. (2017). Efficient kNN classification with different numbers of nearest neighbors. IEEE transactions on neural networks and learning systems, 29(5), 1774-1785.
[21] Deng, Z., Zhu, X., Cheng, D., Zong, M., Zhang, S. (2016). Efficient kNN classification algorithm for big data. Neurocomputing, 195, 143-148.
[22] https://drive.google.com/file/d/14pDiK0D-kzfM3eWX70a-di7_2hwekqC_/view - last accessed on 01/12/2022