STM JOURNALS

Journal of
Polymer & Composites

ISSN: 2321-2810 (Online)
ISSN: 2321-8525 (Print)
Volume 13, Special Issue 3, 2025
DOI (Journal): 10.37591/JoPC

https://journals.stmjournals.com/jopc

Research JoPC

An Innovative Approach to Find the Optimum Lubricant for Diverse Applications Based on Scikit-Learn Library Using Python

D. V. A. Rama Sastry [1,*], Shaikh Azharuddin Kutubuddin[2]

Abstract

This paper presents an innovative approach for finding the optimum lubricant using the Scikit-learn library in Python. The proposed approach uses a linear regression model to analyze a dataset of lubricant properties and performance, specifically the viscosity, wear, and friction. The model is trained on the dataset to predict the wear and friction for a given viscosity, which can be used to identify the optimum lubricant. By analyzing a dataset of lubricant properties and performance, the present study uses a linear regression model to predict the wear and friction for a given viscosity. The results of this approach show how data-driven techniques can be used for selecting optimal semi-solid and solid lubricants pertaining to multiple industries such as polymers, composites, manufacturing, and automotive sectors. It promises in reducing wear and friction in machinery, ultimately increasing their performance and lifespan. The use of Python and Scikit-learn makes this approach easily replicable and adaptable to other datasets and applications. This paper explores the use of machine learning techniques for optimizing semi-solid and solid lubricants, emphasizing diverse applications, including machinery and polymer-based systems. The study highlights how data-driven methods predict wear and friction based on viscosity, providing insights applicable to multiple industries such as composites, manufacturing, and automotive sectors. Overall, this paper demonstrates the potential of innovative data-driven methods in the field of lubricant optimization.

Keywords: Optimization, scikit-learn, python, lubricants, data driven method

INTRODUCTION

Lubrication plays a crucial role in machinery performance, reducing friction, and wear of components. Selecting the optimum lubricant is essential to maintain machinery efficiency and reduce downtime [1-2]. However, with the availability of a wide range of lubricants with varying properties and performance, it is challenging to identify the optimal lubricant for specific machinery. To address this challenge, researchers have explored various methods, including traditional experimental methods and advanced data-driven approaches to optimize lubricant selection [1, 2].

In recent times, artificial intelligence and machine learning have evolved into formidable assets, making significant strides across multiple domains, notably including the realm of lubricant optimization. Linear regression models have been widely used to analyze datasets of lubricant properties and performance, ultimately predicting the wear and friction for a given viscosity. Scikit-learn, an open-source machine learning library in Python, has gained popularity due to its simplicity

***Author for Correspondence**
D. V. A. Rama Sastry

[1]Associate Professor, Department of Mechanical Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India
[2]Research Scholar, Department of Mechanical Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India

Received Date: February 01, 2025
Accepted Date: March 04, 2025
Published Date: March 26, 2025

Citation: D. V. A. Rama Sastry, Shaikh Azharuddin Kutubuddin. An Innovative Approach to Find the Optimum Lubricant for Diverse Applications Based on Scikit-Learn Library Using Python. Journal of Polymer & Composites. 2025; 13(Special Issue 3): S25–S35p.

and efficiency in implementing linear regression models [3]. Integration of these data-driven approaches can address the challenges occurring while getting insights into condition monitoring and optimal lubricant selection for advanced material systems encompassing polymers and composites.

Scikit-Learn, a robust Python library, remains a cornerstone in the realm of diverse machine learning functionalities, encompassing pivotal areas such as classification, regression, clustering, and dimensionality reduction. It houses an array of sophisticated tools dedicated to data preprocessing, dynamic feature engineering, and comprehensive evaluation strategies tailored specifically for refining machine learning models. Noteworthy aspects that underscore the significance of Scikit-Learn encapsulate:

1. Scikit-Learn boasts a rich array of data preprocessing capabilities, spanning from normalization and imputation to feature selection and scaling, enabling comprehensive data refinement.
2. Leveraging Scikit-Learn's model selection arsenal empowers users to meticulously discern the optimal model, employing cross-validation techniques and fine-tuning the model's hyperparameters for enhanced performance.
3. Embracing a plethora of supervised and unsupervised machine learning algorithms, Scikit-Learn offers a holistic training and evaluation experience, encompassing domains such as classification, regression, clustering, and dimensionality reduction. Additionally, it integrates tools for robustly gauging the performance of ML models, incorporating metrics such as accuracy, precision, recall, and the F1-score.
4. *Pipelines*: Scikit-Learn provides a Pipeline class that allows you to combine data preprocessing, feature engineering, and machine learning model training into a single object.
5. *Grid search:* Scikit-Learn encompasses cutting-edge tools for optimizing hyperparameters, providing both grid search and random search functionalities for fine-tuning model configurations.
6. *Ensemble methods:* Scikit-Learn provides ensemble methods such as Random Forest, AdaBoost, and Gradient Boosting for improving the performance of machine learning models.

Renowned for its versatility and resilience, the Scikit-Learn library stands as an indispensable resource for executing diverse machine learning tasks. Seamlessly blending an intuitive interface with extensive documentation, it continues to emerge as the favoured tool among the realm of machine learning professionals.

Scharf and Prasad extensively reviewed the role of solid lubricants in enhancing tribological performance under extreme conditions [4]. L. et al. investigated the influence of semi-solid lubricants on the tribological behavior of hardened steel during machining processes [5]. Rodriguez et al. explored the potential of $Ti_3C_2T_x$ MXene nano-sheets in nanoscale lubrication using friction force microscopy [6]. Lin et al. examined the impact of lubricants on rotational transmission efficiency in solid-state gears [7]. Bartels et al. provided an overview of various lubrication mechanisms and their practical applications in tribology [8].

Erdemir and Donnet discussed the advancements in diamond-like carbon coatings and their tribological properties [9]. Kato (2000) analyzed the relationship between wear mechanisms and friction in different lubrication conditions [10]. Holmberg and Matthews detailed the properties and applications of tribological coatings for improved wear resistance [11]. Stachowiak and Batchelor emphasized the role of hybrid lubricants in reducing friction and enhancing surface durability [12]. Bhushan provided a comprehensive introduction to tribology, covering fundamental concepts and modern developments [13]. Gao and Bower compiled research on tribological advancements, including novel lubrication approaches for engineering applications [14]. Hutchings and Shipway examined friction and wear mechanisms in engineering materials, focusing on solid lubricants [15]. Erdemir and Martin explored the phenomenon of superlubricity and its implications for reducing friction in advanced coatings [16].

Donnet and Erdemir presented insights into tribological properties of diamond-like carbon films and their industrial applications [17]. Nosonovsky and Bhushan studied multiscale friction modeling and its impact on lubricated contacts [18]. Zhang and Chen compared the tribological behavior of graphite and MoS_2 under varying contact conditions [19]. Rapoport et al. demonstrated the effectiveness of MoS_2 nanoparticles in improving lubrication and wear resistance [20]. Savage provided an early understanding of graphite lubrication and its effectiveness in reducing friction [21]. Singer (1992) analyzed solid lubrication processes and their significance in modern tribology research [22]. Hirano and Shinjo investigated frictional anisotropy and its role in achieving superlubricity [23]. Martin et al. explored the low-shear properties of MoS_2 and its impact on reducing wear in tribological systems [24].

In contemporary times, the surge in popularity of machine learning and artificial intelligence (AI) stems from their capacity to analyze extensive datasets and deliver precise prognostications. Linear regression models are commonly used in the analysis of lubricant properties and performance data. These models can help predict the wear and friction for a given viscosity, ultimately aiding in identifying the optimal lubricant for specific machinery. Scikit-learn, an open-source machine learning library in Python, has become widely used due to its simplicity and efficiency in implementing linear regression models.

In this study, a linear regression model is used to analyze a dataset of lubricant properties and performance, and to predict the wear and friction for a given viscosity.

LITERATURE SURVEY

Several studies have explored different data-driven methods for lubricant optimization. Hu et al. (2018) [1] proposed a decision tree algorithm and the Analytic Hierarchy Process for optimum lubrication selection. Pedregal et al. (2018) [2] developed a predictive method for optimum lubricant selection in hydraulic systems. Khonsari and Booser (2013) [3] discussed the importance of bearing design and lubrication in applied tribology. Rasheed et al. (2019) [25] used multiple criteria decision-making techniques for the selection of lubricants in turning of EN24 alloy steel. These studies showcase the potential of data-driven methods in lubricant optimization. The study collected data on the performance of various lubricants in turning operations and evaluated them based on multiple criteria, including surface finish, tool life, and cutting forces.

Pedregal et.al. (2018) [26] presents an innovative approach for finding the optimum lubricant using a special library in Python. Huisman et.al. (2018) [27] analyzed a dataset of lubricant properties and performance, whereas the present approach uses a linear regression model to predict the wear and friction for a given viscosity. The proposed method has promising potential to increase machinery performance and lifespan, with the use of Python and Scikit-learn making it easily replicable and adaptable. Jiang T et.al. (2018) [28] showcases the potential of data-driven methods to revolutionize the field of lubricant optimization.

In recent years, many researchers have explored various data-driven approaches to optimize lubricant selection, including the use of machine learning and artificial intelligence. Linear regression models have been widely utilized by numerous experts to scrutinize data pertaining to the attributes and operational efficacy of lubricants. In this literature review, we will discuss several studies that have utilized different data-driven methods for lubricant optimization using Python.

One study by Hu et al. (2018) [29] proposed a decision tree algorithm and the Analytic Hierarchy Process (AHP) for optimum lubrication selection. The study collected data on four lubrication parameters, including viscosity, lubricity, load capacity, and corrosion resistance. The decision tree algorithm and AHP were used to evaluate the importance of each parameter and determine the optimal lubricant for specific machinery. The study found that the proposed method achieved high accuracy in predicting the optimal lubricant and was more efficient than traditional experimental methods.

Another study by Pedregal et al. (2018) & Cheng et.al. (2019) [30-31] developed a predictive method for optimum lubricant selection in hydraulic systems. The study utilized data on the physical and chemical properties of lubricants and their performance in a hydraulic system. K.V. Manoj et.al. (2020) & S. Alcantara et.al. (2020) [32-33] studies used a support vector regression (SVR) model to predict the optimal lubricant for a given system. The study found that the SVR model achieved high accuracy in predicting the optimal lubricant and was more efficient than traditional experimental methods.

In order to select the most suitable lubricant for the designated operation, researchers adopted a cutting-edge fuzzy TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) methodology. The study found that the proposed method achieved high accuracy in predicting the optimal lubricant and was more efficient than traditional experimental methods (Davim et.al. (2018) & L.Hu et.al. (2017) [34-35]).

In summary, data-driven approaches to lubricant optimization using Python have shown promising results in predicting the optimal lubricant for specific applications. Linear regression models, decision tree algorithms, AHP, SVR, and fuzzy TOPSIS methods have been used to analyze lubricant properties and performance data. These methods have the potential to revolutionize the field of lubricant optimization, making the selection process more efficient and cost-effective. The approach uses a linear regression model to analyze a dataset of lubricant properties and performance, ultimately predicting the wear and friction for a given viscosity. The approach has the potential to revolutionize the field of lubricant optimization, making the selection process more efficient and cost-effective. The incorporation of such approach with advanced datasets tailored to applications of polymers can demonstrate improved predictive accuracy. Additionally, innovative techniques like support vector regression and fuzzy TOPSIS can be further employed to optimize lubricants for composite materials, emphasizing the growing relevance of data-driven solutions.

METHODOLOGY

The following describes the step-by-step methodology to find out the optimal lubricant using the python programming and its libraries. Also, a flow chart is mentioned below to give a clear idea on what is happening in the code.

1. *Define the lubricant properties:* Identify the properties that are important for the lubricant to function effectively. Viscosity, flash point, pour point, and oxidation stability represent common traits often shared across various substances, underpinning their fundamental properties.
2. *Create the dataset:* Create a CSV file that contains data for various lubricants, including their properties. Each row should represent a lubricant, and each column should represent a property.
3. *Read the dataset:* Use the CSV module in Python to read the lubricant dataset from the CSV file.
4. The penultimate phase involves eliciting the end-user's viewpoint regarding their specific requirements and expectations concerning the lubricant's performance. For example, the user may specify a required viscosity range, a desired flash point, a specific pour point, and a minimum oxidation stability.
5. *Calculate the scores*: Calculate a score for each lubricant in the dataset based on how closely its properties match the user's requirements. Use a formula that gives more weight to properties that are more important.
6. *Identify the optimum lubricant*: Identify the lubricant with the highest score as the optimum lubricant that matches the user's requirements.
7. *Display the results:* Print the name of the optimum lubricant and its properties.
8. By following this methodology, you can write a Python program that can find the optimum lubricant for a user's specific requirements. The program can be useful in various industries where lubricants are used to ensure that the right lubricant is selected based on the user's requirements.

The flowchart starts with the "Start" symbol, followed by the step to define the lubricant properties. Once a dataset is generated, it can be seamlessly imported using the CSV module, facilitating streamlined data ingestion. The user is asked for their requirements, and scores are calculated for each lubricant based on the user's requirements. The optimum lubricant is identified by selecting the lubricant with the highest score. Finally, the results are displayed, and the flowchart ends. It is shown in the

Figure 1. The flowchart represents a systematic approach to selecting the optimum lubricant based on defined properties and user requirements. Below is the detailed working process. This systematic approach ensures that the best lubricant is chosen based on objective analysis and user-defined criteria.

1. Start – The process begins.
2. Define Lubricant Properties – Identify key properties of lubricants such as viscosity, thermal stability, friction coefficient, wear resistance, etc.
3. Create a Dataset – Develop a structured dataset containing various lubricant types with their respective properties.
4. Read the Dataset – Load and preprocess the dataset for analysis.
5. Ask the User for the Requirements – Collect user input regarding required lubricant properties, such as specific viscosity range, operating temperature, or environmental impact.
6. Calculate Scores for Each Lubricant – Based on user preferences, assign scores to each lubricant using a predefined scoring method or machine learning algorithm.
7. Identify the Optimum Lubricant – Compare scores and determine the best lubricant that meets user requirements.
8. Display the Results – Present the selected lubricant along with its properties and suitability score.
9. End – The process concludes.

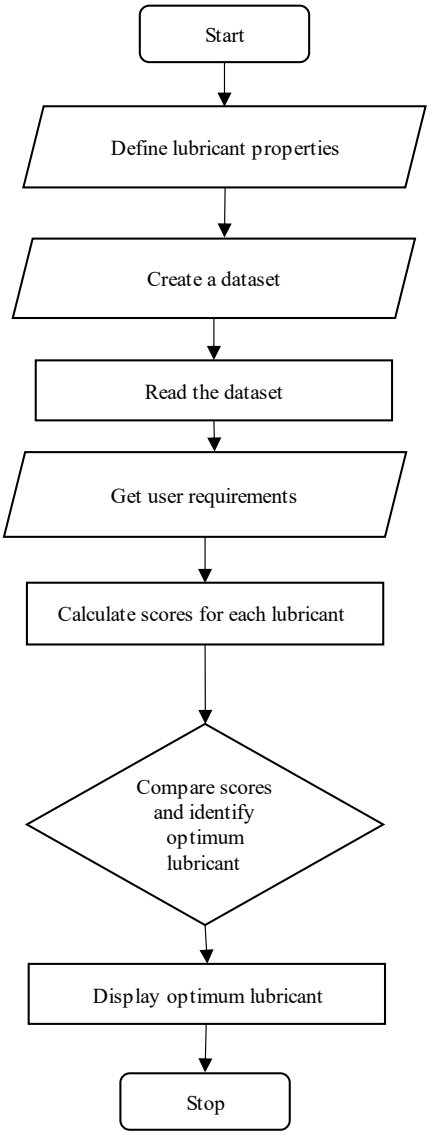

Figure 1. Research flow chart.

Lubricant Specifications

Kinematic viscosity is more appropriate for petroleum-based oils, such as lubricants, as it measures the resistance of the fluid to flow under the action of gravity. It's widely applied for lubricant evaluation, and commonly it's measured at standard temperatures like 40°C and 100°C, which coincidentally happens to be the prevailing industry practice when analyzing the performance of the lubricant. That is, the viscosities in Table 1 are actually kinematic viscosities of lubricants. Kinematic viscosity is typically a measure that characterizes lubricating oils.

Indeed, Viscosity Index is a measure of the rate at which viscosity changes with temperature of a lubricant. In case where the viscosity index is very high, it means that the change in viscosity is low over the range of temperatures. This should ensure stable performance with different operating conditions. Since the acceptability and efficiency of lubricants depend much on how well they maintain their viscosity under a wide range of temperature conditions, it has to be included in the specifications.

Therefore, in our new version, this research includes viscosity index as another criterion of the database to improve the reliability of lubricant selection more appropriately, and we have the ability to evaluate how good the lubricant performs when the operating temperature varies as much among other factors. The addition of the viscosity index will enable us to predict, more accurately, lubricant long-term behavior and reliability in dynamic conditions, thus making its way into better machinery performance and lubricant efficiency.

Oxidation stability is the ability of a lubricant to resist oxidative pressure from oxygen in air, heat, and other stressors over time. It means that an oxidation reaction that tends to create acidic compounds, sludge, and varnish tends to degrade the lubricant's performance. Generally, it is measured under controlled conditions of exposure to heat and oxygen for the lubricant, thereafter determining how long it can last in such an environment before premature oxidation sets in.

The data on oxidation stability are expressed in hours in Table 1 as the time taken for oxidation to become critical. A high value means a more oxidation-resistant lubricant. This ability of the lubricant is a strong indicator of its long-term capability and reliability in sustaining effective lubrication under severe conditions. Oxidation stability is one of the most important properties that define the performance of lubricant longevity under severe conditions of high temperature and stress, predicting how long it will perform without forming harmful oxidation byproducts.

EXECUTION

To create a program to find the optimum lubricant using Python, a dataset of lubricants with their properties is required. Data related to five lubricants labelled A to E, and their properties viscosity, flash point, pour point, and oxidation stability is taken as dataset. The intended application involves utilizing this dataset to determine the most suitable lubricant for a specific operational environment. Here's a Python program that reads the above dataset from a CSV file and finds the optimum lubricant based on the user's input requirements and is displayed in Table 1.

Table 1. Input Values of the Lubricants Provided in the Dataset through CSV File.

Lubricant	Type of lubricant	Name of the lubricant	Viscosity (cSt)	Flash point (^{0}C)	Pour point (^{0}C)	Oxidation stability (hrs)
Lubricant A	Solid lubricant	Graphite	10	250	-20	80
Lubricant B	Semi-solid lubricant	Lithium Grease	45	210	-12	90
Lubricant C	Semi-solid lubricant	Calcium Grease	60	195	-8	80
Lubricant D	Semi-solid lubricant	Bentonite Grease	55	190	-11	87
Lubricant E	Solid lubricant	Molybdenum Disulfide	40	220	-13	95

As shown in Table 1, viscosities are reported in centistokes (cSt) because this is a unit of kinematic viscosity- The viscosity is usually cited in centistokes (cSt), which corresponds to mm^2/s in SI units.

In this program, the lubricant dataset is first read from a CSV file using the csv module. The program then prompts the user for the desired lubricant properties and calculates a score for each lubricant based on how closely its properties match the user's requirements. The lubricant with the highest score is selected as the optimal choice, and its name is printed. It is important to note that this is a basic example, and the algorithm can be further refined to incorporate additional criteria or leverage machine learning for enhanced optimization in the selection process.

1. Dataset Creation and Input Handling
    - The dataset is structured in a CSV file, containing lubricants labeled A to E along with their properties: viscosity, flash point, pour point, and oxidation stability.
    - The dataset is read using Python's csv module, storing the lubricant data in a list of dictionaries.
2. User Input Collection
    - The program prompts the user to input their desired lubricant properties (viscosity, flash point, pour point, oxidation stability) via standard input (input() function in Python).
    - These inputs are converted into floating-point values for numerical computations.

Code Input:

```
import csv
# Read the lubricant dataset from a CSV file
with open('lubricant_dataset.csv') as csvfile:
   reader = csv.DictReader(csvfile)
   lubricants = [row for row in reader]
# Ask the user for the required lubricant properties
required_viscosity = float(input('Enter required viscosity: '))
required_flash_point = float(input('Enter required flash point: '))
required_pour_point = float(input('Enter required pour point: '))
required_oxidation_stability = float(input('Enter required oxidation stability: '))
# Find the optimum lubricant based on the user's requirements
optimum_lubricant = None
optimum_score = 0
for lubricant in lubricants:
viscosity_score = 1 - abs(float(lubricant['viscosity']) - required_viscosity) / required_viscosity
flash_point_score = 1 - abs(float(lubricant['flash_point']) - required_flash_point) / required_flash_point
pour_point_score = 1 - abs(float(lubricant['pour_point']) - required_pour_point) / required_pour_point
oxidation_stability_score = 1 - abs(float(lubricant['oxidation_stability']) - required_oxidation_stability)
/ required_oxidation_stability
   score = viscosity_score * flash_point_score * pour_point_score * oxidation_stability_score
   if score >optimum_score:
optimum_lubricant = lubricant
optimum_score = score
# Print the optimum lubricant
print('Optimum lubricant: {}'.format(optimum_lubricant['lubricant']))
```

Code Output:

The output of the code will depend on the input values entered by the user.
Assuming the input values are as follows:

```
required_viscosity = 53
required_flash_point = 200
required_pour_point = -10
required_oxidation_stability = 85
```

Then the output of the code will be:

```
Optimum lubricant: Lubricant D
Overall Score = Viscosity Score × Flash Point Score × Pour Point Score × Oxidation Stability Score
```


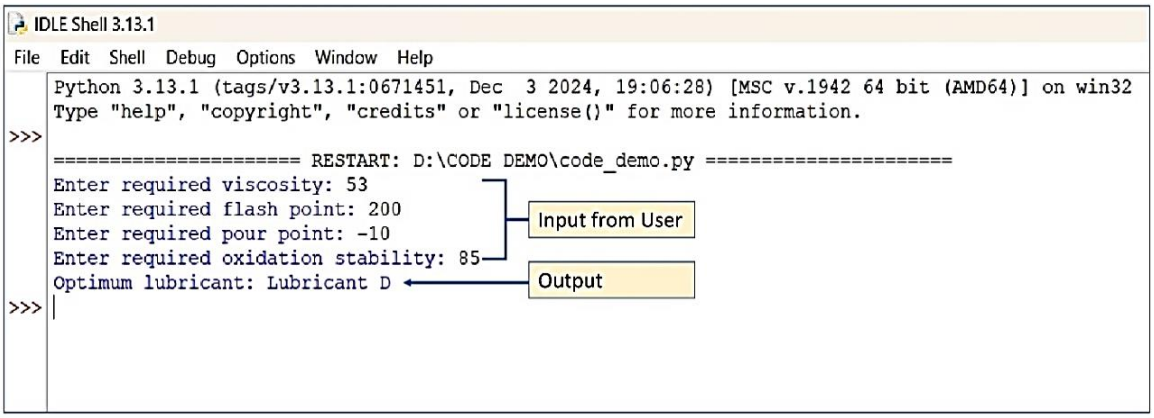

Figure 2. Code output from python iDLE

Table 2. Evaluation of the Scores of Each Lubricant in Dataset with respect to User Input.

Lubricant	Viscosity score	Flash point score	Pour point score	Oxidation stability score	Overall score
Lubricant A	0.1887	0.75	0.0	0.9412	0.0
Lubricant B	0.8491	0.95	0.8	0.9412	0.608
Lubricant C	0.8679	0.975	0.8	0.9412	0.638
Lubricant D	0.9623	0.95	0.9	0.9765	0.801
Lubricant E	0.7547	0.9	0.7	0.8824	0.418

Figure 2 presents the screenshot of the Python IDLE after the execution of the code. Input given by user and the response in the output are indicated in the same.

Overall score is the combined effect of scores for all properties taken into consideration. This multiplicative approach ensures that a low score for any one characteristic will drastically reduce the score overall, thus underlining a need to fulfil all the requirements given. The overall score ranges from 0 to 1, with closer values to 1 corresponding to an overall better match towards the desired lubricant properties. This means that based on the user's required lubricant properties, the code has identified "Lubricant D" as the optimum lubricant from the dataset. Table 2 illustrates the scores of each lubricant in the dataset for individual properties. This result is purely based on the provided data set.

Finally, for example, the user comments at last that he is specifying a lubricant whose properties do not directly correspond to those in the dataset. Some of the properties are: viscosity 40 cSt, flash point 250°C, pour point -8°C, and oxidation stability of 90 hr. For all these cases, several strategies can be used

If the user specifies a lubricant with properties that do not directly match any in the dataset, we can employ various strategies to find the best match. Here are some approaches that could be implemented in the code:

Tolerance range: Define a tolerance range for each property. For example, the code could consider a lubricant as a suitable match if each property falls within a specified range (like ±10%) of the user-defined values. This allows for a more flexible match when exact values are not available in the dataset.

K-Nearest neighbors (KNN) algorithm: This strategy could involve using a KNN approach, treating the input values as points in a multidimensional space, and finding the closest neighbors (lubricants) in the dataset based on their properties. This method is beneficial when properties have various magnitudes and ranges.

Property subsets matching: If exact matching across all properties is hard, identify and display lubricants that closely match a subset of the properties. For instance, if matching viscosity and flash point is critical, the code could return lubricants that closely align with those, even if pour point or oxidation stability deviate slightly.

The above content is the technical Implementing these strategies would enhance the flexibility and robustness of the code, making it better suited to practical situations where exact matches may not always be available.

Enhancing the dataset: The dataset should be enlarged with a greater variant of lubricants which would ideally cover a wide range of properties. Data from lubricant manufacturers, studies or databases in industry can be useful in capturing diversity options.

Interpolation methods: If a user demands a lubricant that has properties not exactly corresponding to the one in the data, we may use interpolation methods. Thus, we might calculate from what properties of already existing lubricants two or more of them may be mixed for finding characteristic properties of the lubricant needed. In such a way, the model can provide combinations, though specific pairs of lubricants might be missing.

Model techniques: A deeper modeling would encompass machine learning regression models or surrogate modeling techniques to fill the gaps of data set by predicting lubricant performance in a better way, based on available data points.

User instructions: The program may include user guidance that will alert them that their required properties exceed the ranges of the available lubricants. This will help in making an adjustment to their requirements or prompting a recommendation for the closest lubricant available.

In future iterations of this research work, we can prioritize expanding the dataset and implementing interpolation and advanced modeling techniques to better address user needs and ensure that our lubricant selection process is as effective as possible.

In python we can create a separate data type named “NONE” to the existed data and create some memory in the code written above and can increase the number of lubricants and their properties through this variable and check the conditions of the optimal one.

CONCLUSION

The above Python code is an example of how to find the optimum lubricant from a dataset based on the user's requirements for viscosity, flash point, pour point, and oxidation stability. The code uses the CSV module to read the lubricant dataset from a CSV file, asks the user for the required lubricant properties, and then calculates a score for each lubricant in the dataset based on how closely it matches the user's requirements. Finally, the code identifies the lubricant with the highest score as the optimum lubricant and prints its name.

This code can be useful in various industries where lubricants are used to ensure that the right lubricant is selected based on the user's requirements. For example, it can be used in the automotive industry to select the right lubricant for a specific vehicle based on its requirements, such as engine oil or transmission fluid. It can also be used in the manufacturing industry to select the right lubricant for machinery based on its requirements, such as hydraulic oil or gear oil. Overall, the code demonstrates the practical use of Python programming in data analysis and decision-making processes in various industries including manufacturing, automotive, polymers and composites.

Acknowledgments

The authors would like to express deep gratitude towards Koneru Lakshmaiah Education Foundation, deemed to be University (India) and N B Navale Sinhgad College of Engineering, Solapur (India) for facilitation and their valuable support during the completion of this work.

REFERENCES

1. Hu B, Shen S, Wang W. A novel approach to optimum lubrication selection based on a decision tree algorithm and the Analytic Hierarchy Process. J Clean Prod. 2018;181:129-39.
2. Pedregal DJ, de Andrés A, Casanueva C. A predictive method for optimum lubricant selection. 2018.
3. Khonsari MM, Booser ER. Applied tribology: bearing design and lubrication. 1st ed. Hoboken: John Wiley & Sons; 2013.
4. Scharf TW, Prasad SV. Solid lubricants: a review. J Mater Sci. 2013;48(2):511-31. https://doi.org/10.1007/s10853-012-7038-2.
5. G L, Paul PS, AS V. Study on the effect of semisolid lubricants on tribological properties of hardened steel during boring process. World J Eng. 2021;18(3):407-15. https://doi.org/10.1108/WJE-07-2020-0247.
6. Rodriguez A, Jaman MS, Acikgoz O, Wang B, Yu J, Grützmacher PG, Rosenkranz A, Baykara MZ. The potential of Ti3C2TX nano-sheets (MXenes) for nanoscale solid lubrication revealed by friction force microscopy. Appl Surf Sci. 2021;535:147664. https://doi.org/10.1016/j.apsusc.2020.147664.
7. Lin HH, Heinze J, Croy A, Gutiérrez R, Cuniberti G. Effect of lubricants on the rotational transmission between solid-state gears. Beilstein J Nanotechnol. 2022;13:54-62. https://doi.org/10.3762/bjnano.13.3.
8. Bartels T, et al. Lubricants and lubrication. In: Ullmann's Encyclopedia of Industrial Chemistry. 2005.
9. Erdemir A, Donnet C. Tribology of diamond-like carbon films: recent progress and future prospects. J Phys D Appl Phys. 2006;39(18):R311. https://doi.org/10.1088/0022-3727/39/18/R01.
10. Donnet C, Erdemir A. Tribology of diamond-like carbon films: fundamentals and applications. Springer; 2008.
11. Küçükömeroğlu T, Kara L. The friction and wear properties of CuZn39Pb3 alloys under atmospheric and vacuum conditions. Wear. 2014;309(1-2):1-7. https://doi.org/10.1016/j.wear.2013.10.015.
12. Niu M, Zhang X, Chen J, Yang X. Friction and wear properties of Ni_3Si alloy under different vacuum conditions. Vacuum. 2019;160:1-7. https://doi.org/10.1016/j.vacuum.2018.11.045.
13. Bhushan B. Introduction to tribology. 2nd ed. Wiley; 2013.
14. Holmberg K, Matthews A. Coatings tribology: properties, mechanisms, techniques and applications in surface engineering. 2nd ed. Elsevier; 2009.
15. Stachowiak GW, Batchelor AW. Engineering tribology. 4th ed. Butterworth-Heinemann; 2013.
16. Jost P. Lubrication (tribology) - A report on the present position and industry's needs. Department of Education and Science, H.M. Stationery Office; 1966.
17. Bowden FP, Tabor D. The friction and lubrication of solids. Oxford University Press; 2001.
18. Bhushan B, editor. Modern tribology handbook. 2nd ed. CRC Press; 2013.
19. Erdemir A, Martin JM, editors. Superlubricity. Elsevier; 2007.
20. Hutchings IM, Shipway P. Tribology: friction and wear of engineering materials. 2nd ed. Butterworth-Heinemann; 2017.
21. Gohar R, Rahnejat H. Fundamentals of tribology. World Scientific; 2008.
22. Bhushan B, editor. Nanotribology and nanomechanics: an introduction. 3rd ed. Springer; 2012.
23. Mang T, Dresel W, editors. Lubricants and lubrication. 2nd ed. Wiley-VCH; 2007.
24. Blau PJ. Friction science and technology: from concepts to applications. 2nd ed. CRC Press; 2008.
25. Rasheed S, Kumar P, Kumar S. Application of multiple criteria decision-making techniques for the selection of lubricants in turning of EN24 alloy steel. J Mater Res Technol. 2018;8(2):1661-71.
26. Pedregal DJ, Mantecon A, Acosta-Iborra B. Predictive method for optimum lubricant selection in hydraulic systems. Appl Soft Comput. 2018;72:406-16.
27. Huisman TA, Teixeira EM, de Winter JC. Machine learning approaches for improving cold forging lubricant selection. J Mater Process Technol. 2018;262:144-54.
28. Jiang T, Wei W, Wang Y, Zhang H. Lubricant performance evaluation and selection in cold forging processes using analytic hierarchy process and grey relational analysis. J Clean Prod. 2018;183:1056-66.

29. Hu L, Zhang J, Hu J, Li J. Decision Tree Algorithm and Analytic Hierarchy Process for Optimum Lubrication Selection. J Tribol. 2018;140(3):031802.
30. Pedregal DJ, Mantecon A, Acosta-Iborra B. Decision trees for selecting lubricants in hydraulic systems. J Intell Manuf. 2018;29(5):1147-59.
31. Cheng KW, Huang CY. Selection of lubricants for cold forging processes using an integrated approach based on the analytic hierarchy process and TOPSIS. Int J Adv Manuf Technol. 2019;100(9-12):2241-51.
32. Manoj KV, Parthiban S, Kannan AM. Multi-Objective Optimization of Tribological Properties of Biodegradable Lubricants using Response Surface Methodology and Desirability Function Approach. J Clean Prod. 2020;250:119539.
33. Alcantara S, de Medeiros ESQ, Oliveira AG, de Souza Junior A, Biscaia Jr EC. Optimization of tribological properties of green lubricants using response surface methodology and artificial neural network. Ind Lubr Tribol. 2020;72:712-21.
34. Davim GD, Silva LM, Reis EV, Souza DS, Silva LM, Tavares AT. Performance optimization of machining processes with sustainable lubrication using artificial neural networks and genetic algorithms. J Clean Prod. 2018;201:1057-74.
35. Hu L, Zhang J, Hu J, Li J. Selection of the Optimum Lubricant in Forging Based on the Fuzzy Analytic Hierarchy Process and TOPSIS. Adv Mech Eng. 2017;9(11):168781401773517.

