Journal of Engineering Science and Technology
21st EURECA 2024 Special Issue August (2025) 148 - 159

REAL-TIME DETECTION OF OIL PIPELINE LEAKAGE USING MACHINE LEARNING

RAVI PRASATH A/L POONGKUNTRAN[1], HAFISOH AHMAD[1,*], NURFARHANIM ABU BAKAR[2]

[1]School of Engineering, Taylor's University, Taylor's Lakeside Campus, No. 1 Jalan Taylor's, 47500, Subang Jaya, Selangor DE, Malaysia.
[2]Department of Engineering and Sciences, American Degree Program, School of Liberal Arts and Sciences, Taylor's University, Taylor's Lakeside Campus, No. 1 Jalan Taylor, 47500, Subang Jaya, Selangor Darul Ehsan, Malaysia.
*Corresponding Author: Hafisoh.Ahmad@taylors.edu.my

Abstract

The challenge of undetected corrosion in oil pipelines requires the development of a machine learning based solution to predict progression and optimize maintenance to prevent leaks and contamination. This project aims to determine the most effective machine learning algorithm among k-nearest Neighbours (k-NN), Support Vector Machine (SVM) and Random Forest and implement it in a real-time pipeline model with an alarm system. The study uses an open-source dataset from GitHub, originally intended for regression analysis, and applies binarization of labels to classify corrosion defects as 'high' or 'low' The research highlights the importance of data quality, starting with data cleaning, to ensure reliable training data. The data is split in an 8:2 ratio, with 80% training and 20% testing. Metrics such as the F1score, ROC-AUC and confusion matrix are used to assess performance. The results show that the SVM model achieves the highest accuracy with 95.82%, while the k-NN model has the lowest accuracy with 85.14%. However, as this dataset comes from an external source, the validity of the results still needs to be proven. The main goal remains to implement the model in a real-time system with an alert function to achieve individualised performance results. In summary, while the SVM model is effective in this analysis, its true validation will depend on its implementation in a real-time pipeline monitoring system. This step is crucial to ensure its practical applicability and reliability in operational environments.

Keywords: K-nearest neighbour, Machine learning, Oil pipeline, Random forest, Real-Time, SVM.

1. Introduction

Oil pipelines are essential in the energy sector for the cost-effective transportation of oil and gas over long distances. They connect lines between producers, production suppliers, and end users. Corrosion is a common issue in the oil pipeline industry and is the primary cause of both internal and external leaks. Internal corrosion occurs due to chemical reactions between the pipeline material and transported fluids, which can include water, salts, acids, and bacteria.

Excessive water content, especially fluids reaching up to 50%, significantly increases the likelihood of localized corrosion along the pipeline's bottom [1]. On the other hand, external corrosion, on the other hand, results from interactions between the pipeline and its surrounding soil and moisture in the environment, exacerbated by factors such as poor soil conditions, degradation of coatings, and inadequate cathodic protection [2].

Corrosion is critical to consider in efforts to mitigate leaks not only because it creates pits and holes in pipeline walls but also because as wall thickness decreases over time with more pits and holes, the risk of pipeline rupture and subsequent environmental damage, particularly to marine life, increases [3, 4].

Therefore, maintaining the integrity of pipelines is critical, as a pipeline leak can have catastrophic economic and environmental consequences, such as damage to the marine ecology, high legal costs, and litigation. Traditional leak detection methods are often ineffective, especially for early detection, since they primarily depend on visual inspections due to human bias [5]. This can result in tiny leakage problems not being detected early, leading to significant losses.

Machine learning models can be employed to detect early signs of corrosion by analysing sensor data such as pressure, flow rate, and acoustic signals throughout the pipeline. Machine learning is a rapidly expanding technology in various fields due to its ability to work without human intervention. Machine learning recognises patterns and makes predictions without extensive complex programming. Integrating sensor data and leakage detection results in oil pipeline management can significantly benefit from developing machine learning models [6].

The Support vector machine (SVM) model begins by inputting the sensor data from the various sensors along the pipelines as points in a multidimensional space, with each dimension representing a specific feature extracted from the sensor readings. The algorithm seeks to identify an optimal hyperplane within this space that effectively distinguishes between the data points, allowing a comparison between normal operating conditions and the situations that indicate potential leaks.

The effectiveness of SVM, in this instance, stems from its ability to optimise the margin between different types of data points. This margin is known as the support vector representing the distance from the hyperplane to the nearest data points in each class [7]. By maximising this margin, SVM enhances its ability to generalise and accurately classify new data points. During training, SVM learns from labelled historical data, including normal pipeline activities and confirmed leak based. This supervised learning method allows SVM to modify and construct the ideal hyperplane that best distinguishes between these two states using the extracted sensor characteristics [8].

Once trained, the SVM model can classify real-time sensor data streams, quickly detecting deviations from normal operation that may reveal pipeline leaks. The effectiveness of SVM in this context lies in its capacity to maximize the margin between different types of data points; this margin represents the distance from the hyperplane to the nearest data points in each class, known as support vectors. By maximizing this margin, SVM enhances its ability to generalize and accurately classify new data points.

Another type of machine learning is the k-Nearest Neighbours (k-NN) algorithm uses supervised learning to classify pipeline conditions based on their proximity to historical data points. Each sensor data point is represented in a feature space, with dimensions corresponding to sensor readings or derived features such as mean or standard deviation.

Then, k-NN identifies the k-nearest neighbour to a new data point using distance metrics like Euclidean distance, where k specifies the number of neighbours considered [9]. The Euclidian distance formula is as shown in Eq. (1). The algorithm then classifies the new data point by majority voting if a classification task is performed or averages if a regression task is performed based on its nearest neighbours [10]. The assumption made in this model is that nearby data points in the feature space belong to the same class meaning that the class or boundary is determined by the closeness of the data cluster [10].

The efficacy of k-NN depends significantly on the choice of k whereby smaller k values create complex decision boundaries that may be sensitive to noise resulting in higher accuracy predictions while larger values smooth out boundaries but may overlook local patterns. The Euclidean distance formula is shown in Eq. (1).

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

where x and y are data points, and n is the number of features.

There are also Random Forest that operates by constructing multiple decision trees during the training process. It generates the final output by taking the mode of the classes for classification tasks or the mean of the predictions for regression tasks from the individual trees. Each decision tree is created independently and trained on a randomly sampled subset of the training data, known as bootstrap samples, with replacement. Rather than using all features at each node, Random Forest selects a random subset of features to make splits, introducing diversity and randomness among the trees in the ensemble.

A recursive binary splitting process develops each decision tree based on the chosen features. The splitting criteria differ between classification and regression tasks, where classification splits aim to maximize information gain, while regression splits focus on minimizing variance at each node. The majority vote (mode) of all the individual trees' predictions determines the Random Forest's final prediction in the classification tasks. In contrast, regression tasks are calculated as the average (mean) of the predictions from all the trees [11].

These models autonomously identify leakage patterns and predict potential leakage hot spots in the pipeline. Hence, this study aims to evaluate three machine learning models which are SVM, k-NN and Random Forest to evaluate the model with the highest accuracy in detecting leakage in oil pipelines. The motivation for this

study is to implement machine learning as part of the leakage detection system in pipelines that can learn patterns from the sensor data such as pressure sensor and flow rate sensor and make accurate predictions of identifying leakages and normal state.

2. Methods

Figure 1 shows the flowchart of the methodology for developing a machine-learning system for detecting oil pipeline leakages. The method used an external dataset with eight features from GitHub containing 10,293 data instances of leakage and normal states that will be used for model training and testing.

The first part of creating the model is data preprocessing, which involves binarizing the target attribute and standardising the feature value by applying a polynomial transformation to allow the model to classify the instances based on the threshold value.

The next step is creating the correlation matrix heatmap between the eight features in the dataset to determine the features with the highest correlation to the corrosion factor. Once the map is generated, the next part will split the data into 80:20 for training and testing purposes. Three machine learning models, SVM, k-NN, and Random Forest, were trained and evaluated using accuracy, precision, recall, specificity, F1-score, and ROC-AUC metrics. The results were analysed to identify the best-performing model.

These findings will be compared with those from the pipeline model developed using pressure and flow sensors, which were also trained on the same three models, to confirm that the model excels in both scenarios.

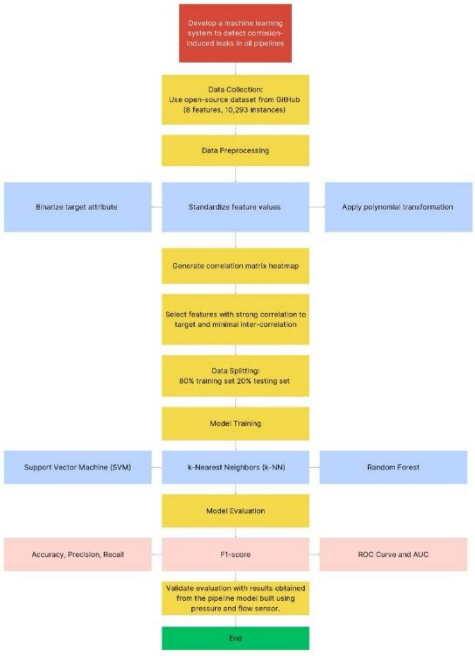

Fig. 1. Main flow chart of the methodology for developing a machine learning system for detecting oil pipeline leakage.

An open-source dataset on oil pipeline corrosion defects from GitHub was used, containing eight features and 10,293 instances, including attributes like temperature, wellhead pressure, gas flow, oil production, water production, CO_2 concentration, and gas gravity.

The target attribute, "corrosion defect," was adapted from a regression to a classification problem by binarizing it: values ≤0.211 were labelled 'low' (no leaks), and values >0.211 were labelled 'high' (leaks present).

Feature values were transformed into a NumPy array to establish the leakage threshold and standardized to achieve a mean of 0 and a standard deviation of 1, as illustrated in Fig. 2. This standardization guarantees that all features contribute equally to the model's predictions, thereby improving algorithm performance. A polynomial transformation was applied to capture non-linear relationships within the data. Finally, a pre-trained model was utilized to predict corrosion defects based on these processed features.

Out[10]:

	Wellhead Temp. (C)	Wellhead Press (psi)	MMCFD-gas	BOPD (barrel of oil produced per day)	BWPD (barrel of water produced per day)	BSW - basic solid and water (%)	CO2 mol. (%) @ 25 C & 1 Atm.	Gas Grav.
0	53.3549	1105.1310	12.8663	1378.9315	2812.6157	75.6442	3.3628	0.7205
1	72.2534	1026.3148	3.4239	1028.7463	919.9169	44.2063	3.8679	0.8940
2	65.0794	722.9642	6.2303	2017.9195	1212.4212	17.5518	2.3552	0.7661
3	60.7060	1557.2321	11.7114	558.2210	1716.0908	65.7869	1.7253	0.7738
4	46.1874	1304.4192	8.5750	1280.4693	1929.2197	37.4468	1.8327	0.7611

In [11]:
```
X2 = df2.iloc[:,1:7]
X2.head()
X2 = X2.values
X2 = StandardScaler().fit_transform(X2)
X2_poly = pre_process.fit_transform(X2)
```

In [12]:
```
df2['CR-corrosion defect']= model.predict(X2_poly)
```

Fig. 2. Derivation of corrosion defect value from the eight features.

A correlation matrix heatmap was produced to display the correlation coefficients between the features in the dataset and comprehend the linear relationships between numerical variables. The correlation coefficients between all pairs of variables were determined using the Pearson correlation method, which quantifies the linear correlation between two continuous variables. Figure 3 shows the generation of the correlation matrix heatmap.

```
# Correlation matrix heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix Heatmap')
plt.show()
```

Fig. 3. Generation of correlation matrix heatmap.

The heatmap illustrates linear relationships between variables, demonstrating positive correlations (increases in tandem) and negative correlations (varies inversely). To improve the interpretability and performance of the model, features that are strongly correlated with the target variable and have minimal inter-correlation among themselves are chosen. The dataset was then pre-processed by specifying features and binarizing the target variable based on a threshold of 0.211

(corresponding to 'high' for values above and 'low' for values at or below). The distribution of the target variable was visualized using a count plot, which was used to evaluate potential imbalances and class balance as shown in Fig. 4.

```
[ ]  # Define features and target variable
     X = data.drop('CR-corrosion defect', axis=1)
     y = np.where(data['CR-corrosion defect'] > 0.211, 'high', 'low')
```

```
[ ]  # Plot count plot for 'y'
     plt.figure(figsize=(6, 4))
     sns.countplot(x=y)
     plt.title('Distribution of Target Variable')
     plt.xlabel('CR-corrosion defect')
     plt.ylabel('Count')
     plt.show()
```

Fig. 4. Features and target variable splitting.

The dataset was partitioned into training and testing sets to facilitate the model's development and evaluation. As shown in Fig. 5, 80% of the data was designated for training, while the remaining 20% was designated for assessment. To guarantee reproducible results across multiple trials, a random state of 42 was established. Setting a random state ensures that the random processes involved in data splitting yield the same result each time the code is run.

```
[ ]  # Split data into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Fig. 5. Train and test data splitting.

Next, feature scaling-a data preprocessing step-was applied to the training and validation sets to ensure that all features contributed equally to the model's performance. This was achieved using the StandardScaler function from sci-kit-learn, which standardises the features by removing the mean and scaling them to unit variance. As a result, each feature is adjusted to have a mean of zero and a standard deviation of one. The code for the feature scaling step is shown in Fig. 6.

```
[ ]  # Feature scaling
     scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)
```

Fig. 6. Feature scaling.

The dataset was utilised to train three machine learning models-SVM, k-NN, and Random Forest-to detect anomalies in parameters such as temperature, pressure, and

flow rate, allowing the models to learn the underlying patterns and relationships. After completing the training phase, the models underwent testing to assess their performance using various metrics, including accuracy, precision, recall, specificity, F1-score, and ROC-AUC. These metrics were used to calculate the confusion matrix, which determines the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), thus helping to evaluate the accuracy of the predictions. Additionally, a Receiver Operating Characteristic (ROC) Curve was produced to analyse the classification capabilities of the models by considering the Area Under the Curve (AUC) value. The AUC value reflects the classifier's effectiveness in distinguishing between the 'low' and 'high' classes, with a higher AUC indicating a greater accuracy in the model's predictions.

3. Results and Discussion

3.1. Correlation matrix heatmap

Figure 7 shows the correlation matrix heatmap generated to study the relationships between the 8 key attributes used in the dataset which are Wellhead Temperature (°C), Wellhead Pressure (psi), Gas Flow Rate (MMCFD), Oil Production Rate (BOPD), Water Production Rate (BWPD), Basic Solid and Water Content (BSW, %), CO_2 Mole Fraction (%) and Gas Gravity to the Corrosion Defect Rate (CR).

The analysis of each attribute based on the matrix is as follows:

1. Wellhead Temperature (°C): Weak correlations with all parameters where the highest value obtained is 0.02 with Gas Gravity indicating that it has minimal influence on the other factors.
2. Wellhead Pressure (psi): Has a notable negative correlation with Corrosion Defect Rate with a value of -0.37, suggesting that when the pressure is higher, the rate of corrosion defects is lower.
3. Gas Flow Rate (MMCFD): Moderate positive correlation with Corrosion Defect Rate with a value of 0.22 implying that the rate of corrosion defects is higher when the flow rate is higher.
4. Oil Production Rate (BOPD): Negligible correlations with all parameters.
5. Water Production Rate (BWPD): Weak correlations with all parameters where the highest value obtained is -0.12 with Corrosion Defect Rate indicating that it has minimal influence on the other factors.
6. Basic Solid and Water Content (BSW, %): Weak correlations with all parameters where the highest value obtained is 0.12 with Corrosion Defect Rate indicating that it has slightly above average influence towards corrosion defect compared to the other attributes.
7. CO_2 Mole Fraction (%): Weak correlations with all parameters.
8. Gas Gravity: Negligible correlations with all parameters.
9. Corrosion Defect Rate (CR): Has significant negative correlations with Wellhead Pressure producing a value of -0.37 and positive correlations with the Gas Flow Rate producing a value of 0.22.

Hence, despite many parameters showing weak interdependencies with each other, the highlight of this matrix is the notable correlations between wellhead pressure and gas flow rate to the corrosion defect rate, which underscores that the

pressure and flow rate parameters play a pivotal role in detecting the corrosion defect in an oil pipeline. Therefore, for the next phase of the project, which involves constructing the pipeline model, the pressure and flow sensors should be considered and prioritized since they have a strong correlation to the corrosion defects rate.

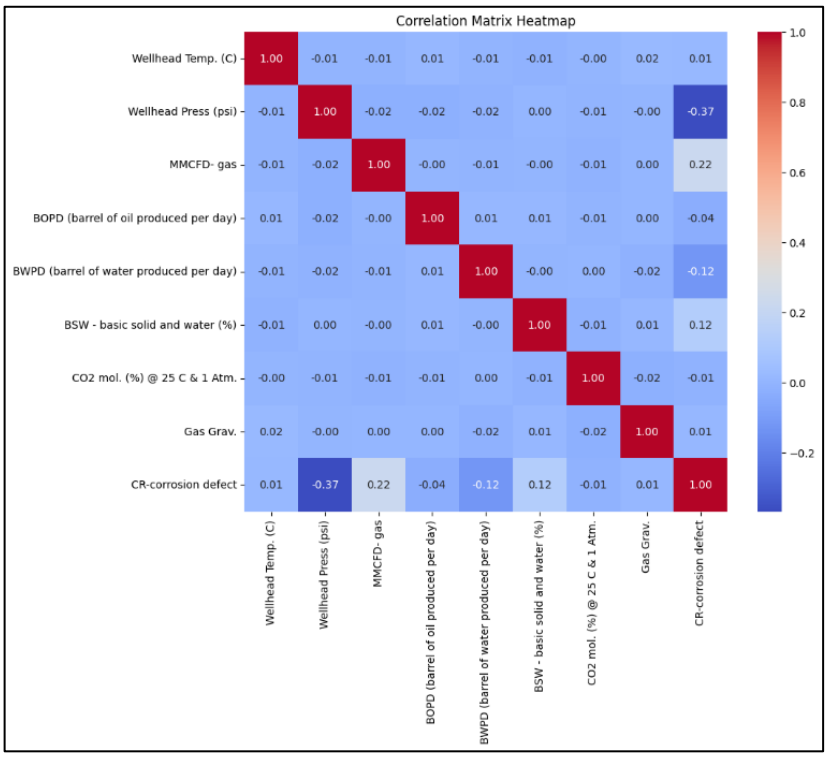

Fig. 7. Correlation matrix heatmap of the features.

3.2. Machine learning models

Figure 8 illustrates the calculations used to derive evaluation metrics, including precision, recall, F1-score, support, accuracy, macro average, and weighted average for the Random Forest model. In contrast, Fig. 9(a) presents the confusion matrix for the same model. The ROC curve is a graphical representation of the model's performance, with the x-axis depicting the False Positive Rate (FPR) and the y-axis representing the True Positive Rate (TPR).

The AUC value of 0.97, as shown in Fig. 9(b), indicates a high level of accuracy, suggesting that the Random Forest model effectively distinguishes between the two classes: leakage and no leakage. An AUC of 1.0 indicates a perfect model, while an AUC of 0.5 represents a model with no ability to discriminate between classes. The confusion matrix confirms the model's robust performance, showing a significant number of true positives (858) and true negatives (996). However, it also reveals some false positives (98) and false negatives (107), indicating areas where improvement is needed. The model's overall accuracy is 0.90, meaning it accurately predicted the class 90% of the time. These findings suggest that while the Random

Forest model performs well for this task, there is still potential for improvement, particularly in reducing false positives and false negatives.

```
Accuracy: 0.9004371053909664

Classification Report:
              precision    recall  f1-score   support

        high       0.90      0.91      0.91      1094
         low       0.90      0.89      0.89       965

    accuracy                           0.90      2059
   macro avg       0.90      0.90      0.90      2059
weighted avg       0.90      0.90      0.90      2059
```

Fig. 8. The classification report for the Random Forest model includes precision, recall, F1-score, support, accuracy, macro average, and weighted average.

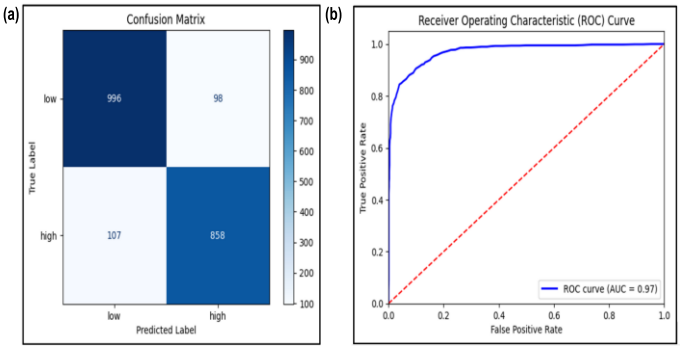

Fig. 8. Confusion matrix of the Random Forest model (b) ROC curve of the Random Forest model.

Figure 10 displays the calculations for the evaluation metrics, including precision, recall, F1-score, support, accuracy, macro average, and weighted average for the k-NN model. In contrast, Fig. 11(a) presents the confusion matrix associated with the k-NN model. The AUC value of 0.93 indicates a high level of accuracy, as shown in Fig. 11(b). The overall accuracy of the k-NN model is 0.85, indicating that it correctly predicts the class 85.13% of the time. The confusion matrix reveals a substantial number of true positives (804) and true negatives (949), reflecting strong performance. While these results demonstrate that the k-NN model is effective for this task, there is still potential for improvement, particularly in minimizing false positives (145) and false negatives (161).

```
Accuracy: 0.8513841670713939

Classification Report:
              precision    recall  f1-score   support

        high       0.85      0.87      0.86      1094
         low       0.85      0.83      0.84       965

    accuracy                           0.85      2059
   macro avg       0.85      0.85      0.85      2059
weighted avg       0.85      0.85      0.85      2059
```

Fig. 10. The classification report for the k-NN model encompasses precision, recall, f1-score, support, accuracy, macro average, and weighted average.

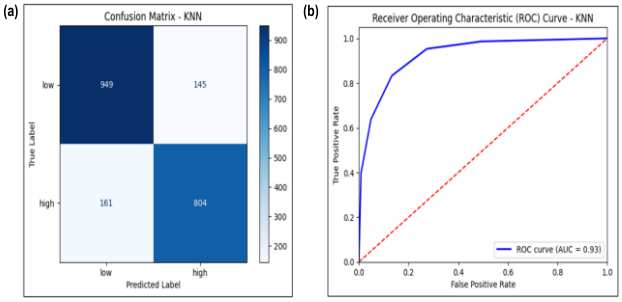

Fig. 11. (a) Confusion matrix of the k-NN model, (b) ROC curve of the k-NN model.

Figure 12 presents the calculations used to derive the evaluation metrics for the SVM model. Figure 13(a) showcases the confusion matrix for the SVM model, which achieves an impressive AUC value of 0.99, as shown in Fig. 13(b). This high AUC indicates the model's outstanding ability to distinguish between the two classes, confirming its effectiveness in identifying leakage versus no leakage conditions. The confusion matrix highlights a substantial number of true positives (917) and true negatives (1056), emphasizing the model's strong performance. Furthermore, the low counts of false positives (38) and false negatives (48) indicate that the model is both reliable and precise in distinguishing between normal and leakage states in a pipeline. As such, the SVM model stands out as the preferred option for implementation in a pipeline model that incorporates real-time data.

```
Accuracy: 0.9582321515298688

Classification Report:
              precision    recall  f1-score   support

        high       0.96      0.97      0.96      1094
         low       0.96      0.95      0.96       965

    accuracy                           0.96      2059
   macro avg       0.96      0.96      0.96      2059
weighted avg       0.96      0.96      0.96      2059
```

Fig. 12.9 The classification report for the SVM model encompasses precision, recall, f1-score, support, accuracy, macro average, and weighted average.

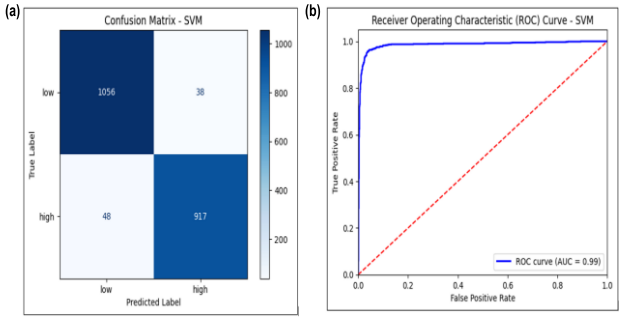

Fig. 13. (a) Confusion matrix of the SVM model, (b) . ROC Curve of the SVM model.

3.3. Model comparison

Based on the results, SVM shows the best model compared to the Random Forest and k-NN models, with an AUC score of 0.99 and an accuracy of 95.82%. This exceptional performance shows that SVM can distinguish between a leak and a non-leak state in an oil pipeline. The SVM model also has far fewer false positives and negatives than the other models, indicating that the probability of obtaining a false alarm is also relatively low compared to the other models.

The SVM outperformed Random Forest, and k-NN can be explained by its ability to work effectively in high dimensional space with many attributes such as in this study (wellhead temperature, pressure, flow rate, etc. [12]. Aside from that, SVM works well in recognizing patterns even with a small number of training samples, which is a great advantage in pipeline leakage detection, where the number of samples is often limited.

On the other hand, although the Random Forest and k-NN models performed well, both models still fell short of SVM. Despite its capability to handle noisy data, the Random Forest model tends to overfit training data, leading to slightly higher rates of false positives and false negatives. Meanwhile, the k-NN model relies heavily on local similarity measures and can struggle to distinguish boundaries for leak and non-leak states when the dataset is not uniformly distributed, causing the optimal k value not to be readily identifiable.

There are also limitations due to the usage of external datasets from GitHub. The quality and relevancy of the different oil pipeline conditions will be questionable. Moreover, the training models using external datasets may yield different results and exhibit sub-par performance when implemented using a pipeline model with real-time data in the subsequent phases. Challenges persist as variations in operational conditions and sensor readings can impact the model's capability to differentiate between normal fluctuations and genuine leaks.

4. Conclusion

In summary, this study illustrates the assessment of machine learning models for detecting oil pipeline leakage caused by corrosion using an external dataset from GitHub. ROC-AUC, F1-Score, and the Confusion Matrix are the evaluation metrics employed to assess the efficacy of the models. The results show that SVM obtained the highest accuracy at 95.82%, followed by Random Forest at 90.04% and k-NN at 85.14%. This may be attributed to the working principle of SVM, which is to maximize the margin between classes in high-dimensional spaces to establish a more precise decision boundary between the leak and no-leak classes. Nevertheless, it is important to acknowledge that the performance results derived from the external dataset are insufficient to validate the performance of SVM in real-time operational settings. Additional validation is required using individualized results to ensure that SVM still yields the highest accuracy of all three models, even in real-time operation.

References

1. Bai, Y.; and Bai, Q. (2014). *Corrosion and Corroded Pipelines*. In Bai, Y.; and Bai, Q. (Eds.), *Subsea Pipeline Integrity and Risk Management*. Elsevier, 3-25.
2. Pluvinage, G.; Capelle, J.; and Schmitt, C. (2016). *Methods for assessing defects leading to gas pipe failure*. In Pluvinage, G.; Capelle, J.; and Schmitt, C. (Eds.), *Handbook of Materials Failure Analysis with Case Studies from the Oil and Gas Industry*. Elsevier, 55-89.
3. Imran, M.M.H.; Jamaludin, S.; and Mohamad Ayob, A.F. (2024). A critical review of machine learning algorithms in maritime, offshore, and oil & gas corrosion research: A comprehensive analysis of ANN and RF models. *Ocean Engineering*, 295, 116796.
4. Xie, M.; and Tian, Z. (2018). A review on pipeline integrity management utilizing in-line inspection data. *Engineering Failure Analysis*, 92, 222-239.
5. Lu, H.; Xi, D.; and Qin, G. (2023). Environmental risk of oil pipeline accidents. *Science of The Total Environment*, 874, 162386.
6. Adegboye, M.A.; Fung, W.-K.; and Karnik, A. (2019). Recent advances in pipeline monitoring and oil leakage detection technologies: Principles and approaches. *Sensors*, 19(11), 2548.
7. Yakoubi, S.; Kobayashi, I.; Uemura, K.; Nakajima, M.; Hiroko, I.; and Neves, M.A. (2023). Recent advances in delivery systems optimization using machine learning approaches. *Chemical Engineering and Processing-Process Intensification*, 188, 109352.
8. Suthaharan, S. (2016). *Machine Learning Models and Algorithms for Big Data Classification*: *Thinking with Examples for Effective Learning*. Springer US.
9. Kuang, Q.; and Zhao, L. (2009). A practical GPU based KNN algorithm. *Proceedings of the Second Symposium International Computer Science and Computational Technology* (*ISCSCT '*09), 151-155.
10. Elyukhin, V.A. (2016). *Cluster variation method*. In Elyukhin, V.A. (Ed.), *Statistical Thermodynamics of Semiconductor Alloys*. Elsevier, 61-105.
11. Rezaei, N.; and Jabbari, P. (2022). *Random forests in R*. In Rezaei, N.; and Jabbari, P. (Eds.), *Immunoinformatics of Cancers*. Elsevier, 169-179.
12. Thanh Noi, P.; and Kappas, M. (2017). Comparison of random forest, k-nearest neighbor, and support vector machine classifiers for land cover classification using sentinel-2 imagery. *Sensors* (*Basel, Switzerland*), 18(1), 18.