Automate End-to-End Deployment and Improve SHAP Interpretation and Model Performance with Large Language Model

[1]Chung-Chian Hsu and *[2]Pin-Han Chen

[1]Department of Information Management
National Yunlin University of Science and Technology, Yunlin, Taiwan
[2]International Graduate School of Artificial Intelligence
National Yunlin University of Science and Technology, Yunlin, Taiwan
*E-mail: m11263004@yuntech.edu.tw

ABSTRACT

With the rapid development of artificial intelligence, many research challenges have emerged, including the black-box nature of models and the automation of deployment and maintenance processes. The black-box problem forces developers to spend considerable time and resources understanding model decisions when optimizing model performance. As a result, Explainable AI (XAI) techniques are becoming increasingly important, but current XAI explanation methods are still underdeveloped, requiring significant time investment from researchers to learn. Similarly, without automation, model deployment and maintenance also require a large amount of time and effort, making Machine Learning Operations (MLOps) a growing trend. To address these challenges, this research proposes an innovative approach that combines XAI techniques with Large Language Model (LLM), utilizing custom graphic prompts for XAI techniques and text explanations provided by LLM. This enables developers to quickly and easily understand model decisions while addressing automation issues related to model deployment and maintenance with Kubeflow. Experimental results show that feature selection using SHAP improves model performance. Integrating these steps into Kubeflow allows for the automation of model deployment, training, and explanation processes, giving model developers more time to focus on improving their models.

Keywords: XAI, MLOps, LLM, SHAP, Pipeline

1. INTRODUCTION

In recent years, machine learning (ML) has achieved higher accuracy, but most high-precision models are black boxes, meaning the models lack interpretability. This phenomenon has garnered widespread attention and discussion in both academia and industry, leading to the rise of XAI.

A black box means that even if we can accurately predict outcomes, we cannot explain the underlying prediction logic and reasoning process, forcing developers to spend more time improving the model. Black box models are applied in many critical societal areas such as healthcare, finance, industry, and military fields, where inaccurate predictions can have irreversible consequences. Therefore, the models used must be transparent and trustworthy, making XAI technology a key solution to address the interpretability problem of models [1]. Through model explanations, developers can understand model decisions, thereby building trust in predictions and ensuring their reliability and validity.

Apart from the black box problem, machine learning models also face significant challenges in automation and maintenance, including setting up training environments, deploying models, and the manpower required for model maintenance. The development and application of models are closely related to these challenges, and Machine Learning Operations (MLOps) has emerged as a solution to address these issues.

MLOps provides a unified framework for managing the lifecycle of models, covering development, deployment, and monitoring processes. It allows for highly automated model maintenance while ensuring environmental consistency. These advantages can be applied to model development, controlling model versions and achieving updates through automated deployment functions.

This research proposes an innovative approach by combining Kubeflow Pipeline (KFP) [2] workflow, XAI, and LLM. By utilizing LLM with prompt templates, XAI-generated graphical information is transformed into concise and understandable text explanations. This provides model developers with more comprehensive explanations, enabling them to improve model performance based on these explanations, and enhances the understanding and trustworthiness of model decisions. Additionally, MLOps offers high automation for model explanation, training, and deployment,

addressing issues related to model deployment and training.

2. BACKGROUND KNOWLEDGE

2.1. Explainable AI (XAI)

In recent years, the rapid development of artificial intelligence has led to more precise and complex model predictions, increasing human reliance on these models. However, this raises a problem: these models often lack the ability to explain their reasoning and actions to human users. Gunning etc. [4] proposed a novel framework that addresses the challenge of model interpretability by combining ML development principles with human-computer interaction. In [5], the concept of XAI was introduced to explain the behavior of AI systems in simulation game scenarios. It was mentioned that to achieve the highest accuracy with modern large datasets, the models used are often so complex that even experts find them difficult to interpret. Therefore, a unified method for explaining model predictions, SHapley Additive exPlanations (SHAP), was proposed [6]. Moreover, Ribeiro etc. [7] proposed LIME, a local post-hoc explanation method that transforms unreliable models into trustworthy ones.

However, the explanation method described above is too static, lacking interaction between the explainer and the recipient. The best explanation method should be human-centered. As mentioned in [8], explanations should involve interaction between the explainer and the recipient. The essence of explanations should be social, and communication with users should be conducted interactively. XAI technology should focus on users, providing explanations that help clarify and simplify model principles for better understanding. [3] suggests that LLM should be used to increase interaction between XAI and human users, and hopes that researchers will develop interactive explainable systems using LLM.

2.2. Kubeflow

Kubeflow is a model development platform built on Kubernetes [10]. Building machine learning systems typically involves system setup, requiring the installation of various packages, which makes the system increasingly complex and difficult to migrate when development environments change. Kubeflow addresses all environmental issues by providing the tools needed for model development. It integrates various open-source projects from different areas of machine learning, such as Optuna, Argo, and Kserve, to manage different stages of machine learning, including hyperparameter tuning, workflows, prediction, and service management.

Kubernetes, like Kubeflow, is an open-source system developed by Google. Kubernetes is an open-source container management system, where the outermost container in Kubernetes is called a Pod. A Pod is a collection of one or more containers, with the main program running through the Pod's child containers while other containers provide supporting functions. Through Pods, different teams can develop various functionalities, and Pods can enhance robustness, composability, and fine-grained resource adjustment capabilities [10].

3. METHOD

We used KFP to divide the machine learning process into three parts. The first part involves model development and training. The second part provides XAI techniques for the model trained by KFP The third part is the deployment of the trained model for use. Figure 1 illustrates the workflow of this research.

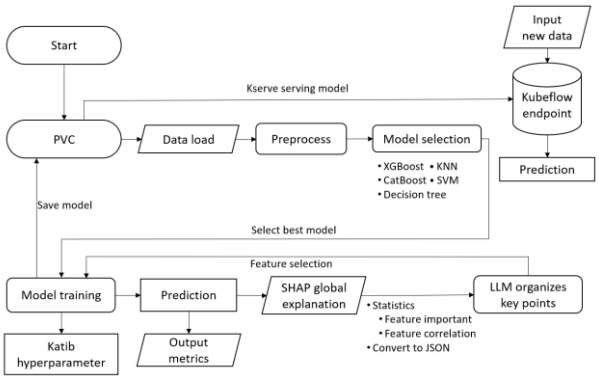

Fig. 1. The proposed framework: XAI and LLM are integrated in the KFP .

Kubeflow provides powerful workflow orchestration capabilities for model deployment and training, but it lacks XAI techniques. This limitation means that developers cannot improve black-box models effectively. Therefore, we propose an innovative idea: integrating XAI techniques into KFP for machine learning models, and using LLM to provide more in-depth explanations for developers. This approach allows developers to better understand feature importance and key factors within the model. Using this information enables developers to improve model performance.

Pipelines are primarily composed of components, each performing different functions. Data transfer between components is facilitated through Kubeflow APIs. Pipelines offer high portability because each component is an independent function. Furthermore, debugging generally becomes easier.

3.1. Model Development

3.1.1. Data Preprocess

In the preprocessing phase, we addressed missing values by imputing them, using categorical clustering, and applied this to a water resources dataset [1] related to environmental issues. This dataset had missing values, so

[1] https://www.kaggle.com/datasets/adityakadiwal/water-potability

we imputed missing values for the clustered features—PH, Sulfate, and Trihalomethanes—using the mean of the features. We then split the dataset into 70% training and 30% testing sets. Finally, we standardized the feature values using StandardScaler to ensure model stability.

3.1.1. Katib

Before training, we perform hyperparameter tuning to optimize model performance. Katib is a hyperparameter tuning tool within Kubeflow that orchestrates hyperparameter tuning workflows through Pipelines. Katib provides various algorithms for hyperparameter optimization and allows custom-developed algorithms to be packaged into Docker containers for use within Katib. In this study, we package the model into a Docker container and use Katib for hyperparameter tuning. The hyperparameter tuning was divided into five steps:

1. Algorithm Selection: Katib offers multiple algorithms for hyperparameter optimization. After comparison, we chose Tree of Parzen Estimators (TPE) as the primary optimization algorithm.
2. Defining Hyperparameter Ranges: We limit the range of hyperparameters to reduce the burden on the search algorithm, maximizing the effectiveness of hyperparameter tuning and further improving model performance.
3. Choosing Evaluation Metrics: We select evaluation metrics to assess model performance under different hyperparameters. Katib allows setting performance thresholds. This study uses F1-score as the primary evaluation metric, while also displaying Recall and Precision as additional metrics for reference.
4. Output: The optimized hyperparameters are outputted for model training.
5. Cleaning Up Experimental Environment: In Katib, using the same name can lead to errors. To avoid continuous occupation of system resources, we delete completed experimental environments.

3.2. Explain model

We used XAI techniques to interpret the model and combined them with LLM for further explanation. Developers can use SHAP feature importance and key points extracted by LLM to improve the model. Many XAI techniques are discussed in the literature, with SHAP being one of the most commonly used techniques.

3.2.1. SHAP

SHAP is an XAI technique based on game theory [6]. It helps in understanding model predictions by providing insights into how features contribute to the output. SHAP exhibits unique advantages among XAI techniques, offering both global explanations of the model's behavior and local explanations for individual predictions. The formula for local explanations in SHAP is represented as follows:

$$\phi_i(x) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!} [f(x_{S \cup \{i\}}) - f(x_S)] \quad (1)$$

where x represents a data point, i denotes a feature, and $\phi_i(x)$ indicates the contribution of the i^{th} feature for the data point. N represents the set of all features, and S is the subset obtained by excluding feature i $\sum_{S \subseteq N \setminus \{i\}}$ represents require computing and summing all possible subsets S, where $|S|$ and $|N|$ denote the number of features in the sets S and N. This formula calculates an attribution score $\phi_i(x)$ for each feature, quantifying its contribution to the model's prediction probability $f(x)$. The model prediction probability $f(x)$ is determined by summing the baseline prediction ϕ_0 and the contributions from all features $\phi_i(x)$. The baseline prediction ϕ_0 is calculated as the average prediction across all feature distributions. The formula is expressed as follows:

$$f(x) \approx \sum_i^N \phi_i(x) + \phi_0 \quad (2)$$

$$\boldsymbol{\phi_0} = \mathbb{E}_x[\boldsymbol{f(x)}] \quad (3)$$

Global explanations help developers understand the importance and positive or negative correlations of features within the model. The overall importance of a single feature in global explanations is represented by the sum of its contributions across all data points. This can be expressed with the following formula:

$$\Phi_i = \sum_x^X |\phi_i(x)| \quad (4)$$

The higher the value of Φ_i, the greater the importance of the feature. SHAP values and plots only show how feature values impact the model. This study further proposed to use Pearson correlation coefficient to quantify the positive or negative correlation between feature values and their impact (i.e., SHAP values) on model output. The Pearson correlation coefficient is calculated using the following formula:

$$\gamma = \frac{\sum(X_i - \hat{X})(Y_i - \hat{Y})}{\sqrt{\sum(X_i - \hat{X})^2 \sum(Y_i - \hat{Y})^2}} \quad (5)$$

where X_i represents the SHAP value, Y_i represents the feature value, $\hat{X}$ is the average SHAP value, and $\hat{Y}$ is the average feature value.

3.2.2. LLM Explain SHAP Feature Important

This study uses LLM to provide extended explanations for visualizations generated by SHAP (e.g., Figure 2) and displays both the graphics and text explanations on the Kubeflow UI using its visualization technology. However, LLM are not specifically trained for XAI purposes and thus lack inherent XAI knowledge. To address this, we propose to employ customized prompt templates to enhance the LLM understanding of SHAP data (importance, positive/negative correlations) and guide the LLM in generating specific and relevant responses. Experimental results show that prompt templates can effectively improve the LLM understanding of SHAP data and increase the stability of the generated text. Texts generated by the LLM provide developers with insights into feature importance and the

positive/negative correlations of features, which can be used to improve model performance.

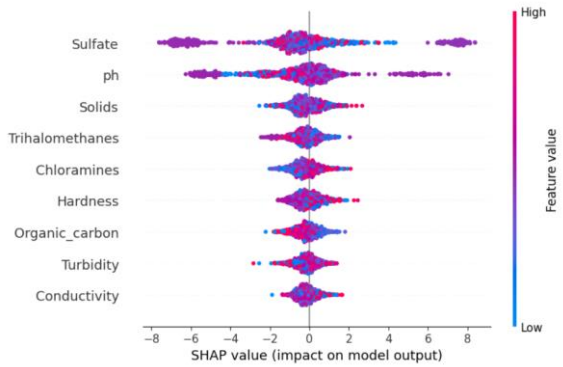

Fig. 2. SHAP-generated plot

In the proposed KFP, global information and statistical analysis from SHAP will be sent to another component for extended explanation by the LLM. Considering that the LLM used in this study is primarily for text analysis and does not support tabular data, those data will be converted to JSON format before being read by the LLM. Converting to JSON helps the LLM to understand the data more easily while preserving its structured information.

3.3. Serving Model

During the deployment phase, this study uses Kserve to deploy machine learning models for user predictions. Kserve is based on Kubernetes and utilizes Knative and Istio for model deployment and service management. It is highly compatible with Kubeflow, allowing Pipelines to use its services for model deployment.

In Kubeflow, we use PVC (Persistent Volume Claims) to store model data and create a YAML file to describe the configuration and parameters of the model service, including the location of the model data for deployment through Kserve. Once the deployment is complete, the deployed model can be viewed on the Kubeflow UI, and predictions can be made by invoking the model through this endpoint.

4. EXPERIMENT

In the field of environmental monitoring, this study used a water quality dataset to validate the proposed research framework. The dataset includes 10 features and 3,276 records, with the data split into 70% for training and 30% for testing. After comparing five prediction models, XGBoost demonstrated the highest overall performance, as shown in Table 1. Therefore, XGBoost was ultimately chosen as the prediction model. An XAI technique, specifically TreeSHAP, was used to explain the model. Following this, LLM was employed for further explanation of the SHAP plot, enabling model developers to more quickly understand the importance of features and their positive/negative correlations.

Table 1 Performance of all models shows that XGBoost outperformed the other models

Model	F1-score	Precision	Recall
XGBoost	**0.727**	0.808	**0.660**
CatBoost	0.722	**0.821**	0.645
Decision Tree	0.664	0.724	0.612
KNN	0.457	0.514	0.411
SVM	0.346	0.739	0.225

In the entire workflow, we used KFP as the workflow management tool to construct a pipeline, enhancing automation and efficiency. We also utilized the visualization capabilities of the Pipeline to display the results of LLM and SHAP, allowing model developers to easily interpret the outcomes. Additionally, the pipeline automates model deployment, as illustrated in Figure 3.

Fig. 3. a. The KFP automates data preprocessing, training, deployment, and integrates explanation functionalities. b. (1) The Kubeflow interface successfully displays SHAP plots and the explanations of SHAP plots generated by the LLM. (2) It can be observed that the red and blue data points in the SHAP plot are highly mixed, indicating that the dataset is difficult to predict. It generates explanations from the LLM regarding feature importance. (3) It explains the positive and negative correlations between feature values and model output values in the SHAP plots, making it easier for viewers to understand the SHAP result plots.

In this study, the ChatGPT API was used to build a prototype system for the LLM. As mentioned in Section 2.2.2, we have set up prompt templates to assist ChatGPT in understanding SHAP data and guiding it to generate

relevant responses. The important prompt template designed for this research includes two steps to explain feature importance in the model. The first part explains how feature importance impacts model predictions. The second part provides examples for the LLM on how to present feature names and importance scores, along with a template for explanations. Table 2 presents the customized prompt templates for the LLM in this study.

Table 2 Feature importance prompt templates.

Step1
Feature importance can evaluate the contribution of each feature to model prediction. The higher the value, the greater the impact on the model output result.
Step2
example: {"Feature": "A", "Importance": 90}, {"Feature": "B", "Importance": 82}, {"Feature": "C", "Importance": 66}, {"Feature": "D", "Importance": 62}, {"Feature": "E", "Importance": 20}, {"Feature": "F", "Importance": 19} Its important features are "A", "B", "C", and "D". These four features contribute the most to the target variable, and their important star scores are 90, 82, 66, and 62 respectively. In addition, features such as "E", "F" have slightly lower contributions to the target variable, and their importance scores are 20, and 19 respectively.

By using prompt templates to explain feature importance, the results are shown in Table 3. The input SHAP data is represented as X, which includes two main pieces of information: feature names and importance scores.

Table 3 Explanation generated by ChatGPT.

User Prompt
X
Response by ChatGPT
Feature importance can evaluate the contribution of each feature to model prediction. The higher the value, the greater the impact on the model output result. Its important features are 'Sulfate', 'ph', 'Solids', and 'Trihalomethanes'. These four features contribute the most to the target variable, and their importance scores are 2441, 1631, 580, and 497 respectively. In addition, features such as 'Chloramines', 'Hardness', 'Organic_carbon', 'Turbidity', and 'Conductivity' have slightly lower contributions to the target variable, and their importance scores are 489, 482, 433, 405, and 383. respectively.

From the text generated by ChatGPT, developers can learn the feature names and their corresponding importance scores for the dataset. The text also highlights the four most important features for reference. To explain feature correlations, we use a similarly structured prompt template. The first part introduces the relationship between correlations and SHAP plot, while the second part provides examples and a threshold for determining whether the red and blue regions in the SHAP plot are distinctly differentiated. Table 4 shows the prompt templates.

Table 4 SHAP value & feature value Pearson correlation prompt templates

Step1
Explain binary classification in the SHAP global diagram. Positive correlation means that the larger the value, the closer it is to category 1, and the smaller the value, the closer it is to category 0; negative correlation means The smaller the value, the closer it is to category 1, and the larger the value, the closer it is to category 0. Taking the X-axis SHAP value of 0 as the midline: a positive correlation means red is on the right and blue is on the left; a negative correlation means blue is on the right and red is on the left. The higher the correlation, the more obvious the distribution of red and blue.
Step2
example: {"feature": "A", "feature value& shap value Pearson Correlation": 0.75}, {"feature": "B", "Pearson Correlation": 0.26}, {"feature": "C", "Pearson Correlation": -0.51}, {"feature": "D", "Pearson Correlation": -0.81}, {"feature": "E", "Pearson Correlation": 0.70}, {"feature": "F", "Pearson Correlation": -0.53} Red on the right and blue on the left represent positive correlation, which means that the corresponding features are "A", " B ", "E", respectively, and their values are 0.75, 0.26 and 0.70; while blue is on the right and Red on the left represents negative correlation, and the corresponding features are "C", "D", "F", respectively, and their values are -0.51, -0.81, -0.53. If most positive or negative correlations are greater than 0.5 or less than -0.5: The higher the correlation, the more obvious the distribution of red and blue. For example, the resolution of "A", "E", "C", "D", "F is relatively high. If most positive or negative correlations are less than 0.5 or greater than -0.5: The higher the correlation, the more obvious the distribution of red and blue. Most of the relationships are weak, making it difficult to distinguish between the red and blue areas.

Table 5 presents the explanations of positive and negative correlations generated by the LLM. The input data is represented as Y, which includes feature names and the Pearson correlation coefficients between feature values and SHAP values.

Table 5 Explanation generated by ChatGPT.

User Prompt
Y
Response by ChatGPT
Red on the right and blue on the left represent positive correlation, which means that the corresponding features are 'ph', 'Solids', 'Chloramines', 'Hardness', and 'Conductivity' respectively, and their values are 0.08, 0.21, 0.29, 0.05, and 0.01; while blue is on the right and red on the left represents negative correlation, and the corresponding features are 'Sulfate', 'Trihalomethanes', 'Organic_carbon', and 'Turbidity' respectively, and their values are -0.04, -0.19, -0.46, and -0.02.Most of the positive or negative correlations are less than 0.5 or greater than -0.5: The higher the correlation, the more obvious the distribution of red and blue. Most of the relationships are weak, making it difficult to distinguish between the red and blue areas.

This study uses prompt templates to enable ChatGPT to generate richer and more domain-specific responses, helping developers better understand the data, such as the significance of the red and blue regions and their distribution in SHAP plot, thus improving the speed of model development understanding.

We verify whether feature selection based on SHAP data can help to improve model performance by focusing on feature importance and positive and negative correlations. First, we input the top 5 most important features from Table 3. We then remove features with higher negative correlations compared to other negatively correlated features. If any of the top 5 features are removed, they are replaced with the next most important feature, and so on. Table 6 presents the results of this experiment.

Table 6 Performance of selected features is higher than that of all features

Model	F1-score	Precision	Recall
All Features	0.727	**0.808**	0.660
Selected Features	**0.733**	0.762	**0.706**

The results in Table 6 indicate that the performance of the model with feature selection is improved compared to the model without feature selection, confirming the feasibility and research potential of our approach.

5. CONCLUSION

This study proposes an innovative approach to address challenges in machine learning model research by using LLM with prompt templates to provide textual explanations of SHAP plot, and integrating this with Kubeflow to achieve efficient automation of model explanations, training, and deployment. The textual explanations provided by the LLM effectively enhance the interpretability of SHAP charts and improve model performance based on the explanations generated by SHAP and the LLM.

Experimental results show that incorporating prompt templates significantly improves the consistency and predictability of the LLM-generated textual explanations. Using these explanations to select features can enhance model performance. This innovative method is integrated into the KFP to achieve end-to-end automated deployment and management of machine learning models. The Kubeflow visualization features display SHAP plot and LLM textual explanations on the Kubeflow UI, facilitating easy reading by relevant personnel and demonstrating the practical potential of the proposed method.

Future research will continue to explore other XAI techniques, enabling developers to refer to various XAI technologies for model improvement. Additionally, these technologies will be applied to user interfaces, allowing users to better understand model decisions and XAI-generated graphics through Kserve predictions, thus expanding the application of AI technologies to critical fields such as healthcare and finance.

ACKNOWLEDGEMENT

This research is supported by the National Science and Technology Council, Taiwan under the project Design of a Command and Control Center for Sustainable Smart Cities (NSTC 112-2634-F-194-001).

REFERENCES

[1] Antwarg, L., Miller, R. M., Shapira, B., & Rokach, L., “Explaining anomalies detected by autoencoders using Shapley Additive Explanations”, Expert Systems with Applications, vol. 186, pp. 115736, 2021.

[2] E. Bisong, Kubeflow and kubeflow pipelines, Anonymous Berkeley, CA:Apress, pp. 671-685, 2019.

[3] H. Lakkaraju, D. Slack, Y. Chen, C. Tan, and S. Singh, “Rethinking explainability as a dialogue: A practitioner’s perspective”, *arXiv preprint arXiv:2202.01875*, 2022.

[4] Gunning, D., & Aha, D., “DARPA’s explainable artificial intelligence (XAI) program”, AI Magazine, vol. 40, no. 2, pp. 44-58, 2019.

[5] Van Lent, M., Fisher, W., & Mancuso, M., “An explainable artificial intelligence system for small-unit tactical behavior”, Proceedings of the National Conference on Artificial Intelligence, 2004.

[6] S. Lundberg, “A unified approach to interpreting model predictions,” arXiv preprint arXiv:1705.07874, 2017

[7] M. T. Ribeiro, S. Singh, and C. Guestrin, “” why should i trust you?” explaining the predictions of any classifier,” in Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135–1144.

[8] Miller, T., “Explanation in artificial intelligence: Insights from the social sciences”, Artificial Intelligence, vol. 267, pp. 1-38, 2019.

[9] Wong, W. K., & Lee, C. S. (2020, 7-15 Feb. 2020). An Implementation of Face Recognition with Deep Learning based on a Container-Orchestration Platform. 2020 Indo – Taiwan 2nd International Conference on Computing, Analytics and Networks (Indo-Taiwan ICAN).

[10] B. Burns, B. Grant, D. Oppenheimer, E. Brewer and J. Wilkes, "Borg Omega and Kubernetes", Commun. ACM, vol. 59, no. 5, pp. 50-57, Apr. 2016.