VOL. 18, NO. 20, OCTOBER 2023 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

APPLICATION OF KUBEFLOW AS A UNIVERSAL APPROACH FOR THE DEVELOPMENT AND IMPLEMENTATION OF ARTIFICIAL INTELLIGENCE SYSTEMS

Mykhailo Kuzmich and Tetyana Gordiyenko
State University of Telecommunications, Ukraine
E-Mail: t_gord@hotmail.com

ABSTRACT

An overview of the concept of machine learning and processes (Machine Learning and Operation, MLOps), which is a set of techniques for implementation and automatic continuous integration, as well as delivery to the production environment and model training, is made. The concept of MLOps was considered in terms of Kubeflow tools - a cloud-native open-source system running on the Kubernetes platform. The possibility of using modern MLOps solutions to improve the development processes of machine learning information systems has been studied. The results of the operation of the model in the Kubeflow arsenal have been checked using such improvement factors as speed of development, implementation of changes, reduction of time to search for problems, recovery after global interruptions, and decrease of the number of errors in the model. For practical analysis, a publicly available model was deployed in a Kubeflow cluster using the Seldon Core Serving application manifest. The conducted research showed that Kubeflow consists of a set of various open-source components that have a high level of integration with each other through the Kubernetes platform. At the same time, Kubeflow uses the Kubernetes pattern of operators for machine-learning objects extremely effectively.

Keywords: Kubeflow, MLOps, data science, data analytics, machine learning, test model deployment, information technology.

Manuscript Received 18 September 2023; Revised 27 November 2023; Published 30 December 2023

1. INTRODUCTION

Data science, data analytics, and machine learning (ML) at the current stage of information technology development are becoming one of the main tools for solving complex applied problems in various fields of activity. ML is one of the methods of functioning of artificial intelligence, namely, the practical implementation of its capabilities by creating algorithms for identifying regularities during the analysis of big data, and their further use for self-learning [1-3]. In general, both fields are concerned with the use of computational and mathematical approaches to improve decision making. Rapid progress in various areas of ML research (such as computer vision, natural language understanding, and AI recommender systems). Therefore, many companies are investing in their data processing teams and ML capabilities to develop predictive models that can deliver business value to their users.

Various processes and technologies are used to develop, test, and maintain the infrastructure of a big data system.

Factors such as:

- access to large data sets;
- low price of computing resources;
- ready-made services based on cloud technologies;
- Rapid advances in artificial intelligence fields such as computer vision, natural language understanding, and recommender systems.

However, there are many additional complexities involved in ML development, which is why many ML systems fail in the early stages of development and do not reach the product environment.

To solve these problems, the concept of a joint approach to the processes of machine learning and operation (Machine Learning and Operation, MLOps) is used, which is a set of techniques for implementation and automatic continuous integration, as well as delivery to the product environment and model training. MLOps is a ML engineering culture and practice aimed at unifying ML system development (Dev) and their operation (Ops). An MLOps practice means you advocate for automation and monitoring at all stages of building an ML system, including integration, testing, release, deployment, and infrastructure management.

Data scientists can implement and train an ML model with predictive performance on an offline captured dataset with appropriate training data for their use case. However, the real challenge is not building an ML model, but building an integrated ML system and running it continuously in production. With a long history of production ML services, Google has discovered that there can be many pitfalls when operating ML-based systems in production. Some of these pitfalls are summarized in Machine Learning: The High Interest Credit Card of Technical Debt [4].

Therefore, software developers increasingly prefer the concept of containerization. Google is one of the few that needs to manage the deployment and development of a large number of service components on hundreds of thousands of servers. At the heart of the

concept of containerization and microservices was the development of an open source distributed container management system called Kubernetes. The latter can not only maintain complete independence of programs, but also improve the use of hardware resources, so it is widely used by most institutions.

Thus, it is advisable to consider the MLOps concept in terms of Kubeflow tools - a cloud native system with open source code running on the Kubernetes platform. The Kubeflow project is dedicated to simplifying the deployment of the workflow of ML systems in Kubernetes. The goal of the latter is not to duplicate other services but to provide an easy way to deploy the best open source systems for ML across different infrastructures, be it private production servers or public cloud solutions. That is, where Kubernetes runs, Kubeflow can run, making it flexible and portable.

2. LITERATURE REVIEW

Machine learning has proven its effectiveness in modern realities, and investments in the development of high-quality, predictable models have grown many times. The model development process in its essence corresponds to the assembly line in production, as a link of sequential or parallel stages [1-3]. The MLOps concept is a set of techniques for the implementation and automatic continuous integration, delivery, and training of models that covers the main patterns and anti-patterns on which to build a successful ML systems development automation pipeline. The effectiveness of using the MLOps approach was considered and proven in many works, where the main requirements for ML systems were formalized [4-5].

But with opportunities come additional challenges. This problem, along with others, was revealed in a paper written by Google employees regarding the invisible technical debt in ML systems (Figure-1) [6]. It should be noted that although the term MLOps does not appear in the mentioned article, the main patterns and anti-patterns on the basis of which a successful pipeline is built are still relevant today. Modern ML systems consist of a large number of components. The ML code is only a small part of this system and is therefore depicted as a small black rectangle in the center (Figure-1). A complex and extensive infrastructure is required for the full operation of ML systems.

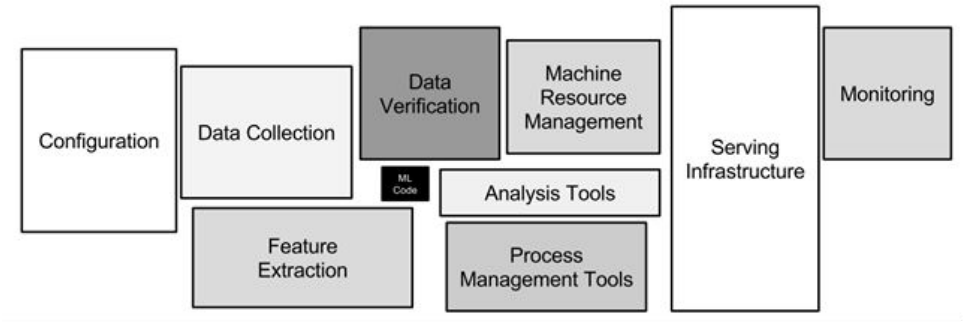

Figure-1. Modern ML system.

The description of ML concepts for industrial engineering and operations research using real programs in the form of courses for students is considered in [7]. Such studies increase students' interest in ML and appreciation of the real impact that analytics can have. It promotes practical skills and, when combined with ML and operations research, can be a valuable addition to undergraduate curricula. However, there is currently limited educational research literature on learning about ML that can be used for course design, especially for non-computer science majors. These problems are addressed in works [8, 9].

In [10], studies were conducted on how students conceptualize their understanding of the unsupervised ML method after interacting with interactive visualizations developed using a computational cognitive learning approach to integrate the knowledge of Data Science concepts. It is also investigated how the capabilities of interactive visualization support or hinder the integration of students' knowledge in ML unsupervised learning. However, this study does not make references to modern ML development systems, which does not allow us to evaluate it from a practical point of view. It would be worthwhile to evaluate one of the current ML systems (Kubeflow, MLFlow) in terms of ease of use for learners or students.

With the increasing availability of data and advances in ML and optimization methodology, data analytics is increasingly used to solve operations management problems. A common approach to solving real operational management problems using the "predict, then optimize" algorithm is considered in [11, 12]. In this approach, the key parameters of the optimization model are predicted by ML before or simultaneously with solving the optimization models. Works [11, 12] consider the application of data analytics for operations management in three main areas - supply chain management, revenue management, and health care operations. However, these papers do not address any specific recommendations for improving or modifying the model, for example in response to changing supply chains.

In [14], a model update and control pipeline structure for AIOps models in microservices systems is proposed. This is done for the purpose of training, encapsulation, and deployment of the model, and to simplify the process. A prototype system based on Kubernetes and Gitlab has been developed to provide preliminary framework implementation and testing. The packaging and deployment process is automated using continuous integration technology. This work is a useful resource for building an integrated and optimized AIOps model control system. However, the proposed technology has an insufficient level of integration between links and a low level of use of Kubeflow abstractions to simplify the model. For example, instead of writing a Dockerfile for a model, you can use the Seldon core component. Also, the concept of automatic retraining of the model was not disclosed in the work. After all, the Kubeflow pipeline can be triggered by the presence of new data for training. These changes would add more versatility and portability to the described ML system, as it automatically adapted to the specific environment of microservices and did not require manual training.

The pipeline component included in Kubeflow can create an entire ML workflow. However, this component still depends on the Google Cloud Platform, so it is not friendly enough for developers who do not have the conditions to rent it. In work [15], the problems of the Kubeflow pipeline were investigated; the feasibility of the Kubernetes independence solution from Google Cloud Service was verified using the example of supporting an ML program based on Pytorch. This solution enables data scientists to build Pytorch-based deep learning applications in Kubernetes-based distributed systems development. The proposed approach ensures stable and continuous operation of the program and reasonable planning on a cluster containing heterogeneous computing resources.

The structure of the ML life cycle (MLife) is proposed in [16] - a cyclical process of creating an effective ML system. The framework is based on the fact that the data flow in MLife is a closed loop driven by failures. They have the greatest impact on ML model performance but also provide the most value for further ML model development. This allows businesses to fast-track their ML capabilities. However, the proposed MLife framework is only suitable for ML systems in their early stages.

The conducted analysis showed the need to consider the concept of MLOps in terms of Kubeflow tools using a number of identified improvement factors, in particular, such as speed of development, reduction of the number of errors in the model, time to find problems, etc. The capabilities of modern MLOps solutions can contribute to the improvement of ML information systems development processes.

3. MATERIALS AND METHODS

When working with ML, steps are defined that can be performed both manually and automatically.

a) Data collection: finding and selecting relevant data from various sources to perform ML tasks.

b) Data analysis: conducting exploratory analysis - preliminary express analysis of data by transforming and/or presenting it in a convenient form: graphic, tabular, scheme, diagram, etc.

The result will be the following:

- understanding of the structure and characteristics of the data expected by the model;
- determination of the stages of data preparation and development of the main features required for the model.

c) Data processing: data must be ready for machine learning tasks to be performed on it. They are cleaned and divided into data for training, validation, and testing.

d) Model training: Engineers use various algorithms and processed data to train the model. In addition, algorithms are used to select hyperparameters to improve the accuracy of the model.

e) Model evaluation: model is evaluated based on available data. The result is a set of model quality metrics.

f) Validation of the model: checking the model for admissibility for deployment in the required environment.

g) Exploitation of the model: tested model is deployed in a finite environment. The expanded model can have the following form:

- a microservice with a REST API that processes forecasts in real-time;
- the built-in model in the final device;
- part of a complex forecasting system.

h) Model monitoring: accuracy and prediction speed of the model are measured for consideration in subsequent development iterations.

The level of automation of these processes shows the maturity of ML processes, which affects the speed of development of new models. There are 3 main levels of MLOps, starting with the entry-level, which includes 0

automation to level, with automation of ML tasks and a full pipeline of continuous integration and deployment.
The simplest MLOps pipeline usually consists of such stages as data preparation, model training, model validation and, accordingly, maintenance/launch of the final model.

Kubeflow is a cloud-native open-source platform for machine learning developed by Google [17]. A feature of this platform is a high level of integration with the Kubernetes orchestration system and a large set of tools for ML tasks.

Kubeflow developers are organized into working groups with corresponding repositories focusing on specific ML tasks:

- automatic machine learning;
- deployment;
- manifests - Kubernetes configuration files;
- Jupyter notebook;
- conveyors;
- training.

A schematic representation of the pipeline for the development of ML solutions in the Kubeflow platform is shown in Figure-2.

The study reviewed the main components of Kubeflow and their feasibility in creating a pipeline. Kubeflow version 1.12, which was the latest at the time of writing, was selected for review. The installation process is relatively simple, as a command line interface (CLI) application has been created for this purpose. The main functional task of the CLI application is the creation of manifests for Kubernetes (Figure-3). After installation, the components will be available in the namespace Kubeflow. The platform has a large number of integrated components. However, the work considers only those that are minimally necessary for the construction of a complete conveyor.

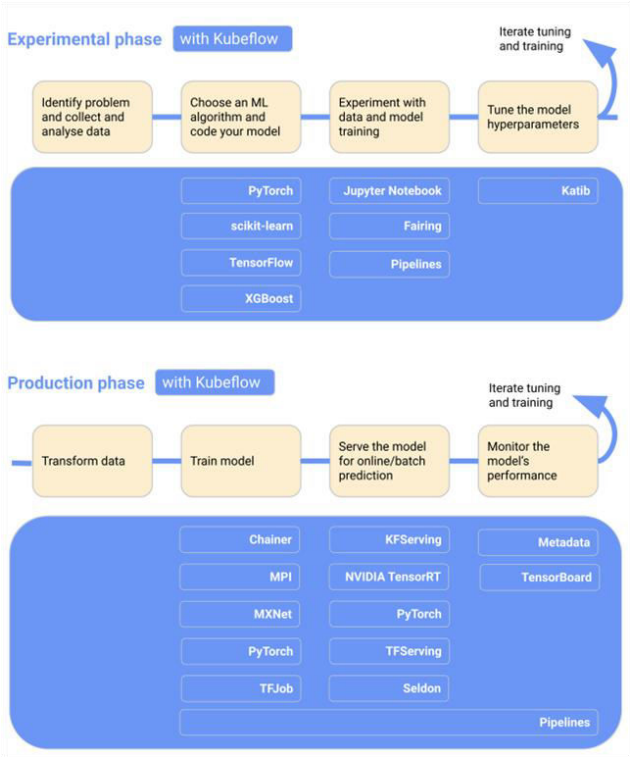

Figure-2. Schematic representation of the ML pipeline in the Kubeflow platform.

```yaml
apiVersion: kfdef.apps.kubeflow.org/v1
kind: KfDef
metadata:
  clusterName: kubeflow.eu-central-1.eksctl.io
  creationTimestamp: null
  name: kubeflow
  namespace: kubeflow
spec:
  applications:
  - kustomizeConfig:
      repoRef:
        name: manifests
        path: namespaces/base
    name: namespaces
  - kustomizeConfig:
      repoRef:
        name: manifests
        path: application/v3
    name: application
  - kustomizeConfig:
      repoRef:
        name: manifests
        path: stacks/aws/application/istio-1-3-1-stack
    name: istio-stack
```

Figure-3. An example of a manifest created by a CLI application.

Using a ready-made CLI application greatly simplifies the installation process, as it allows you to avoid manual editing of a massive configuration file, reducing the entry threshold for users. The list of installed Kubeflow components is shown in Figure-4.

```
→  ~ kubectl get pods -n kubeflow
NAME                                                  READY   STATUS    RESTARTS   AGE
admission-webhook-bootstrap-stateful-set-0            1/1     Running   0          8m24s
admission-webhook-deployment-5d9ccb5696-b4k2n         1/1     Running   0          8m21s
alb-ingress-controller-7dcf69ff67-w4wxl               1/1     Running   0          9m4s
application-controller-stateful-set-0                 1/1     Running   0          8m53s
argo-ui-684bcb587f-rq4j9                              1/1     Running   0          9m4s
cache-deployer-deployment-6667847478-p5l6t            2/2     Running   1          9m4s
cache-server-bd9c859db-svhq8                          2/2     Running   0          9m3s
centraldashboard-895c4c768-r4ssh                      1/1     Running   0          9m3s
jupyter-web-app-deployment-54758f7bc4-g2svl           1/1     Running   0          9m3s
katib-controller-75c8d47f8c-9nr9c                     1/1     Running   0          9m3s
katib-db-manager-6c88c68d79-fbnzt                     1/1     Running   0          9m3s
katib-mysql-858f68f588-zz7dn                          1/1     Running   0          9m3s
katib-ui-68f59498d4-prl5s                             1/1     Running   0          9m3s
kfserving-controller-manager-0                        2/2     Running   0          8m44s
kubeflow-pipelines-profile-controller-69c94df75b-zfwff 1/1    Running   0          9m2s
metacontroller-0                                      1/1     Running   0          8m34s
metadata-db-757dc9c7b5-6xvjw                          1/1     Running   0          9m2s
metadata-envoy-deployment-6ff58757f6-dkj64            1/1     Running   0          9m2s
metadata-grpc-deployment-76d69f69c8-bwp8b             1/1     Running   2          9m2s
metadata-writer-6d94ffb7df-7d4lg                      2/2     Running   0          9m2s
minio-66c9cd74c9-n6659                                1/1     Running   0          9m2s
ml-pipeline-54989c9946-m6gq8                          2/2     Running   0          9m1s
ml-pipeline-persistenceagent-7f6bf7646-x29rq          2/2     Running   0          9m1s
ml-pipeline-scheduledworkflow-66db7bcf5d-5sb6f        2/2     Running   0          9m1s
ml-pipeline-ui-756b58fb-tcvtr                         2/2     Running   0          9m1s
ml-pipeline-viewer-crd-58f59f87db-g29f9               2/2     Running   1          9m
ml-pipeline-visualizationserver-6f9ff4974-jgcwr       2/2     Running   0          9m
mpi-operator-77bb5d8f4b-68z2f                         1/1     Running   0          9m
mxnet-operator-68b688bb69-xrr24                       1/1     Running   0          9m
mysql-7694c6b8b7-ghbhv                                2/2     Running   0          9m
notebook-controller-deployment-58447d4b4c-ndqkt       1/1     Running   0          9m
nvidia-device-plugin-daemonset-dt792                  2/2     Running   0          8m29s
nvidia-device-plugin-daemonset-jbw9z                  2/2     Running   0          8m38s
profiles-deployment-78d4549cbc-n45cx                  2/2     Running   0          8m59s
pytorch-operator-b79799447-2k6t2                      1/1     Running   0          8m59s
seldon-controller-manager-5fc5dfc86c-lht2f            1/1     Running   0          8m59s
spark-operatorsparkoperator-67c6bc65fb-7rn7f          1/1     Running   0          8m59s
tf-job-operator-5c97f4bf7-8zr5z                       1/1     Running   0          8m59s
workflow-controller-5c7cc7976d-5t8m2                  1/1     Running   0          8m59s
→  ~
```

Figure-4. List of installed kubeflow components.

A simple development cycle, where all actions are performed manually, can be divided into 4 stages:

a) Data processing.

b) Model training.

c) Evaluation and validation of the model.

d) Starting the model.

Let's consider the process of automating this pipeline using the Kubeflow platform component by component since a high-quality pipeline requires a certain level of isolation between individual stages and a clear understanding of what comes at the input and what will be at the output.

Jupyter Notebook is an open-source web application that allows you to create and share documents containing working code, equations, visualizations, and text. Notebook is used for data cleaning and transformation, numerical modeling, statistical modeling, data visualization, machine learning, and much more. The range of use of Jupyter notebook is very wide and will be used at all stages of the pipeline.

Storing metadata about training results is an important stage in the development of ML systems, which allows you to evaluate different versions of the model and compare them with each other. The Metadata project, which is an analogue of MLFlow, was created in order to solve the problems of users with understanding their machine learning processes by tracking and processing the metadata that is created by this process. In this context, metadata means information about models, datasets, and other artifacts. Artifacts are files that form the input and output between different stages of the pipeline. In this case, Metadata allows you to answer the following questions:

What dataset was the model trained on?
What hyperparameters were used to train the model?
Which training iteration does the model belong to?
What version of TensorFlow was the model built on?
When was the non-working model uploaded?

Model training is the process of optimizing the algorithm depending on the input data. Kubeflow includes operators that automate training for such machine learning frameworks as:

- MXNet;
- PyTorch;
- TensorFlow.

The concept of Kubernetes operators allows you to simplify the learning process and represent the process as a separate Kubernetes object. The complexity of the process is hidden behind an additional abstraction and, as a result, a relatively complex task of distributed machine learning (adding a picture) can be represented by a single Kubernetes object and described by a YAML manifest file.

Hyperparameter tuning (AutoML) is a machine learning task of selecting a set of optimal hyperparameters for a machine learning algorithm. Katib is a project for the development of ML systems that supports hyperparameter optimization, and neural architecture search (NAS). Katib allows you to set the learning rate, the number of layers in the neural network, and the number of nodes in each layer (Figure-5).

The Kubeflow pipeline is a key element of ML system development, combining all machine learning components into a complete pipeline (Figure-6). The advantage of the Kubeflow pipeline compared to other CI systems is the availability of a set of software libraries (SDK) in the Python programming language, which covers the main Kubeflow abstractions. Describing the development pipeline in the same programming language as the model code has a positive effect on the process because it breaks down the abstract wall between development and operation. One engineer can handle code development and process automation.

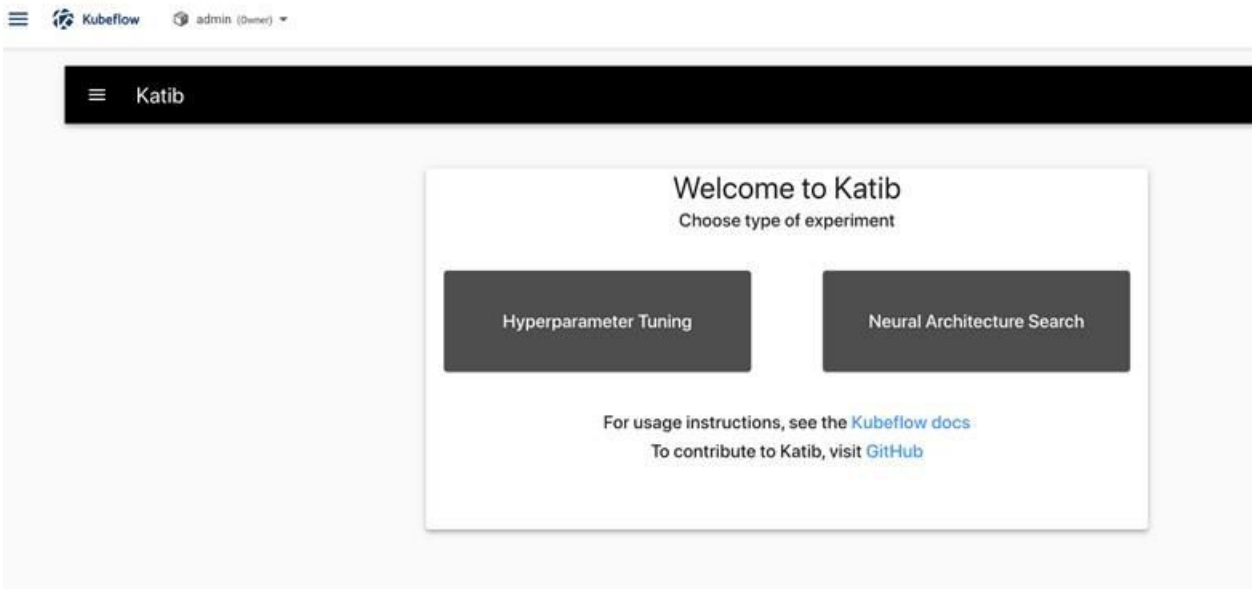

Figure-5. Katib web interface in Kubeflow.

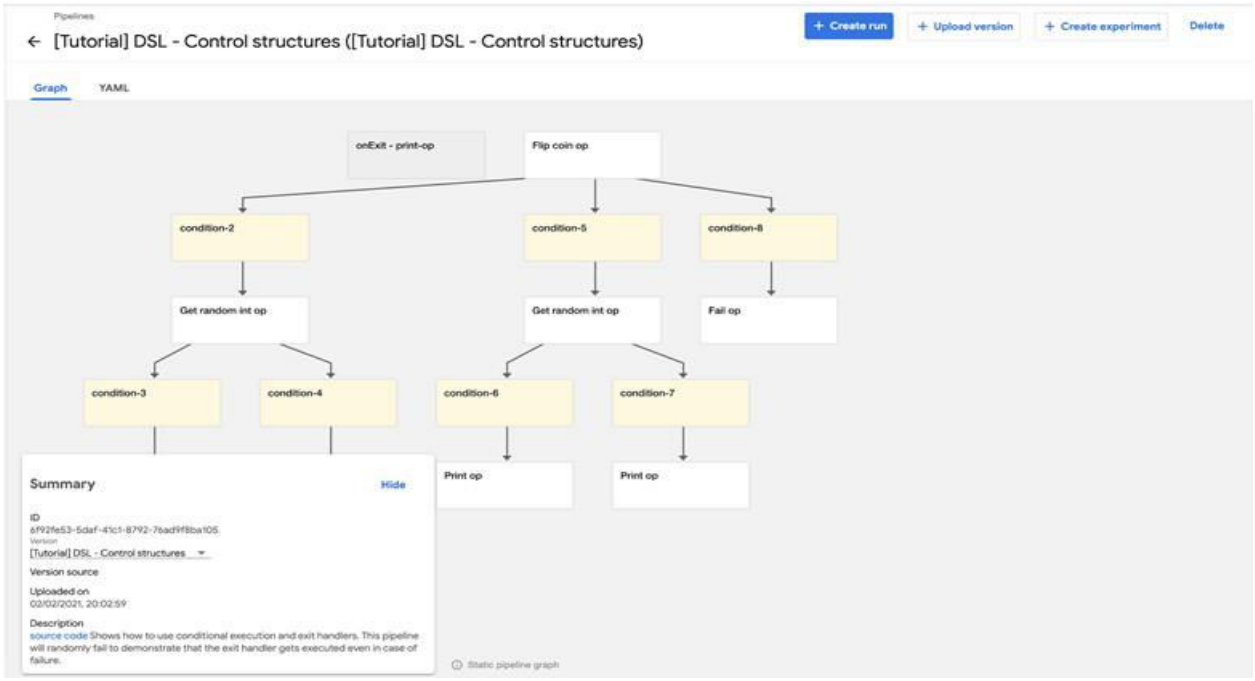

Figure-6. Screenshot of Kubeflow pipeline visualization.

The Kubeflow Pipelines SDK includes the following packages, the main ones being:

- kfp.compiler, includes classes and methods for compiling a pipeline written in Python DSL into YAML manifests used in Kubernetes;
- kfp.components, includes classes and methods for interacting with various components;
- kfp.dsl, includes object-oriented programming language (DSL) used to interact with the pipeline and its components;
- kfp.Client, includes Python client libraries for interacting with the Kubeflow Pipelines API.

4. RESULTS AND DISCUSSIONS

The modern software product development methodology is based on the principles of continuous implementation and does not stop when the model enters the product environment. The exploitation of the model in the classic form consists in writing code that would process requests via grpc/http and return the result via an open port. Also, taking into account the technological recommendations, the work should include the creation of a container, which is mostly equivalent to writing a Dockerfile, manifests for deployment in Kubernetes (deployment, service, ingress), expanding the functionality with the opening of an interface through which third-party services could receive metrics from the service's operation.

To solve the problem of operating the model, the Kubeflow arsenal includes a set of applications, such as Seldon Core Serving, NVIDIA Triton Inference Server, BentoML, TensorFlow Serving.

Seldon Core is an open source platform for exploiting machine learning models in Kubernetes. Seldon Core converts a machine learning model into a REST/GRPC microservice. Seldon supports scaling up to thousands of models instances and provides out-of-the-box advanced support for features such as monitoring metrics, query logging, model explainers, outlier detectors, A/B testing, and canary deployment. A hands-on review of Seldon Core Serving was made, although it should be noted that the above applications have similar functionality.

For practical analysis, a publicly available model was deployed in the Kubeflow cluster using a manifest (Figure-7). After deploying the manifest, the following set of objects was obtained at the output, which is shown in Figure-8.

prepacked.yaml expirement/seldon-core/prepacked.yaml

```
1   ---
2   apiVersion: machinelearning.seldon.io/v1
3   kind: SeldonDeployment
4   metadata:
5     name: iris-model
6     namespace: seldon
7   spec:
8     name: iris
9     predictors:
10    - graph:
11        implementation: SKLEARN_SERVER
12        modelUri: gs://seldon-models/sklearn/iris
13        name: classifier
14      name: default
15      replicas: 1
16
```

Figure-7. Seldon Deployment manifest.

```
→  ~ kubectl get all
NAME                                                READY   STATUS    RESTARTS   AGE
pod/iris-model-default-0-classifier-b546f575b-hsgfn   2/2     Running   0          35h

NAME                                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)             AGE
service/iris-model-default              ClusterIP   172.20.126.5    <none>        8000/TCP,5001/TCP   35h
service/iris-model-default-classifier   ClusterIP   172.20.95.216   <none>        9000/TCP            35h

NAME                                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/iris-model-default-0-classifier   1/1     1            1           35h

NAME                                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/iris-model-default-0-classifier-b546f575b   1         1         1       35h
→  ~ kubectl get virtualservices.networking.istio.io
NAME             GATEWAYS                      HOSTS   AGE
iris-model-grpc   [kubeflow/kubeflow-gateway]   [*]     35h
iris-model-http   [kubeflow/kubeflow-gateway]   [*]     35h
```

Figure-8. A set of objects obtained after deploying the manifest.

There are 7 separate Kubernetes entities under the Selden Deployment umbrella. This level of system simplification for the end user is achieved using the operator pattern in Kubernetes. An operator is a software application that uses a custom resources object to service other applications and their components. In this case, the operator is pod-a seldon-controller-manager-5fc5dfc86c-lht2f, which is part of the Kubeflow cluster. It is this component, together with the basic Kuberentes controllers, that is responsible for creating the objects shown in Figure-9. In particular, these are pod, service, deployment, replica set, and virtual service. Also, when Selden Deployment is removed, all subordinate components will be automatically cleaned up, and the end user does not have to remember the above abstractions (service, deployment, pod, etc.).

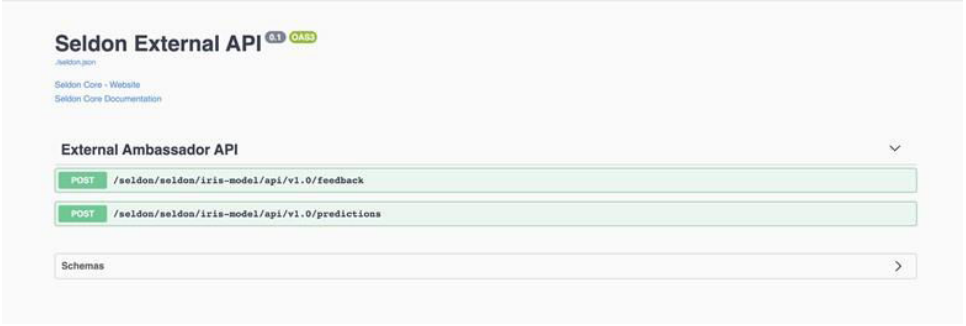

Figure-9. Swagger web model interface.

Seldon core has advanced monitoring with integrations with Prometheus and Grafana, which can be installed separately using the helm package manager. Metrics are automatically available to applications via the path 8080/metrics and Prometheus automatically reads them using a mechanism called service auto discovery (Figure-10).

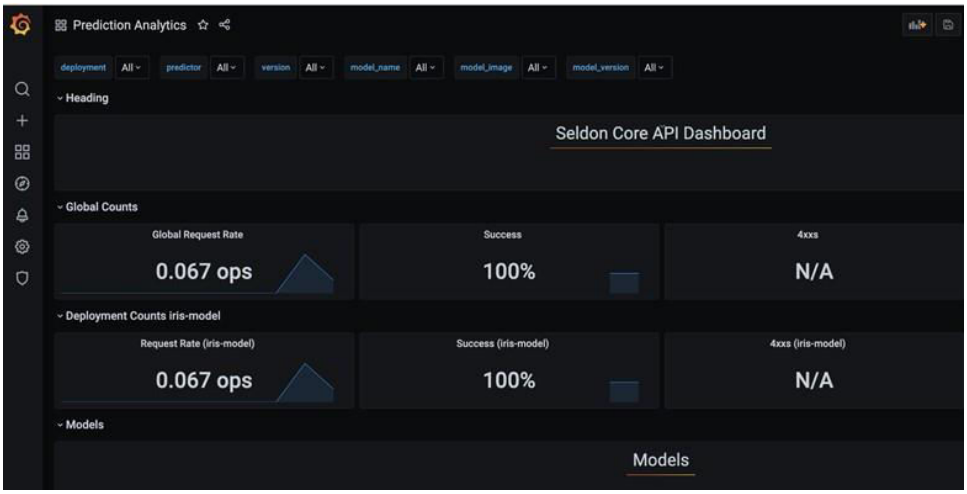

Figure-10. Screenshot of the Grafana monitoring panel based on available metrics in Seldon core

The conducted research showed that Kubeflow consists of a set of various open source components that have a high level of integration with each other through the Kubernetes platform. At the same time, Kubeflow uses the Kubernetes pattern of operators for machine learning objects extremely effectively. It has been demonstrated that writing model code is only a small part of the tasks of machine learning, which affects the need for automation - the presence of a full-fledged pipeline of continuous integration and delivery of the application to the end user.
The result of the research is the formation and substantiation of the conveyor using components of the Kubeflow platform, which is schematically shown in Figure-2. On the basis of this, it can be concluded that Kubeflow is a complete platform for the development and implementation of software products based on machine learning.

Representing abstractions in the form of separate platform resources allows you to reduce the entry threshold for the end user. This allows to inclusion of some components of the system in the course of the educational discipline "Decision support system", including for specialties not related to computer sciences.

The availability of extensive functionality out of the box makes Kubeflow an effective tool for developing and implementing ML models in enterprises, including those with a small engineering staff. On the other hand, a large number of abstractions in the system require the end user to have deep knowledge of the subject area when it is necessary to go one or more levels down for customization.

The presented studies are the first attempt to present a complex machine learning system as a complete object that can form additional value for the product. Further studies of the MLOps system can be developed for certain categories of enterprises that do not have a large number of employees, but need a high-quality model to solve current problems.

5. CONCLUSIONS

a) The possibility of using modern MLOps solutions to improve the development processes of machine learning information systems was studied. Proven feasibility of using Kubeflow to create pipeline MLOps. A schematic integrated pipeline for the development and operation of the artificial learning system has been created. It was important to raise the question of the integrity of the system because when one of the stages is not illuminated enough or not at all, the entire system begins to degrade.

b) Conducted practical use of the Kubeflow Seldon Core Serving component for test model deployment. This made it possible to achieve the best industry practices of model exploitation using simplified abstractions. The functional advantages include A/B testing, in-depth monitoring of internal metrics compatible with the Prometheus system, and a better level of protection against vulnerabilities available out of the box, that is, no additional code needs to be written.

c) MLOps can be a key missing component for the continual learning model, which is the ability to provide a foundation for AI systems to develop themselves adaptively to handle real-world dynamic changes. The basic concept of the continual learning model's pipeline based on Kubeflow components was discovered in this research and will be expanded in the future.

REFERENCES

[1] Muratahan Aykol, Patrick Herring, and Abraham Anapolsky. 2020. Machine learning for continuous innovation in battery technologies. Nat. Rev. Mater. 5, 10: 725-727.

[2] Mahendra Kumar Gourisaria, Rakshit Agrawal, G. M. Harshvardhan, Manjusha Pandey and Siddharth Swarup Rautaray. 2021. Application of Machine Learning in Industry 4.0. In Machine Learning: Theoretical Foundations and Practical Applications. Springer: 57-87.

[3] Ana De Las Heras, Amalia Luque-Sendra, and Francisco Zamora-Polo. 2020. Machine learning technologies for sustainability in smart cities in the post-covid era. Sustainability. 12, 22: 9320.

[4] Google Cloud. MLOps: Continuous Delivery and Automation Pipelines in Machine Learning. [Online]. Available: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

[5] Machine Learning Operations (MLOps): Overview, Definition and Architecture Dominik Kreuzberger KIT Germany dominik.kreuzberger@alumni.kit.edu Niklas Kühl KIT Germany kuehl@kit.edu Sebastian Hirschl IBM Germany sebastian.hirschl@de.ibm.com

[6] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, Dan Dennison. 2015. Hidden Technical Debt in Machine Learning Systems. Part of Advances in Neural Information Processing Systems 28 (NIPS 2015).

[7] Justin J. Boutilier, Timothy C. Y. Chan. 2021. Introducing and Integrating Machine Learning in an Operations Research Curriculum: An Application-Driven Course. INFORMS Transactions on Education. 23(2).

[8] Fiebrink R. 2019. Machine Learning Education for Artists, Musicians, and Other Creative Practitioners. ACM Transactions on Computing Education. 19(4, Art. No. 31): 1-32.

[9] Sulmont E., Patitsas E., Cooperstock J. R. 2019. What Is Hard about Teaching Machine Learning to Non-Majors? Insights from Classifying Instructors' Learning Goals. ACM Transactions on Computing Education. 19(4, Art. No. 33): 1-16.

[10] Sánchez-Peña M., Vieira C. and Magana A. 2022. Data science knowledge integration: Affordances of a computational cognitive apprenticeship on student conceptual understanding. Computer Applications in Engineering Education. 10.1002/cae.22580. 31(2): 239-259.

[11] Elmachtoub A. N., Grigas P. 2017. Smart predict, then optimize. Preprint arXivLabs: Optimization and Control.

[12] Mišić V. V., Perakis G. 2020. Data analytics in operations management: A review. Manufacturing Service Oper. Management. 22(1): 158-169.

[13] Hazzan O. and Mike K. 2023. Core Concepts of Machine Learning. Guide to Teaching Data Science: 209-22.

[14] Ruibo Chen, Yanjun Pu, Bowen Shi, Wenjun Wu. 2023. An automatic model management system and its implementation for AIOps on microservice platforms. The Journal of Supercomputing. 79: 11410-11426.

[15] Haoyu Cai, Chao Wang, Xuehai Zhou. 2021. Deployment and verification of machine learning tool-chain based on kubernetes distributed clusters. CCF Transactions on High Performance Computing, 3: 157-170.

[16] Cong Yang, Wenfeng Wang, Yunhui Zhang, Zhikai Zhang, Lina Shen. 2021. MLife: a lite framework for machine learning lifecycle initialization. Machine Learning. 110: 2993-3013.

[17] Kubeflow. Kubeflow Architecture. [Online]. Available: https://www.kubeflow.org/docs/started/architecture/.