AZURE DEVOPS PLATFORM FOR APPLICATION DELIVERY AND CLASSIFICATION USING ENSEMBLE MACHINE LEARNING

Hemanth Swamy
Senior Software Engineer
Motorola Solutions
hemanth.swamy1@motorolasolutions.com

Abstract—**When it comes to software-intensive systems, system logs play a crucial role in Azures. Logs capture the system's status and major occurrences at crucial times in time. The utility of log entries for data analysis and machine learning is severely limited due to their ad hoc, unstructured, and uncoordinated creation. Particularly in a DevOps setting, activities in data automation pipelines, visualizations, and analytics are prone to frequent disruptions due to uncontrolled change in log data format. Using a comprehensive case study at a top telecommunications firm as a foundation, this article outlines the primary obstacles faced by current methods of creating, storing, and overseeing the development of system logs data for sophisticated, big, software-intensive systems. Secondly, we provide a solution that is designed for machine learning and avoids the aforementioned problems when it comes to creating and controlling the development of log information in a Microsoft Azure DevOps environment. Third, we validate the strategy by conducting expert interviews, which prove that it solves the difficulties and challenges that have been identified. As a result, ML models that are fine-tuned for a certain dataset may rapidly become insufficient. Due to modifications to the characteristics and input data, a model that was once quite accurate may become inaccurate over time. Consequently, it is common to need distributed learning that incorporates dynamic model selection. Such a selection process involves replacing underperforming models with new ones, even if the old ones were fine-tuned for the previous data. It is possible to enhance the general accuracy of a set of ML models by using the famous Ensemble ML (EML) technique. There are a number of drawbacks to EML that should be considered. These include the lengthy model-building process, significant risks of overfitting, extensive training dataset requirements, costly processing resources, and the need for continual training. This research presents a new cloud-based approach to autonomous ML model tweaking and selection that outperforms current approaches in terms of automation of both model construction and selection. Prior to the automatic construction of focused supervised learning models, we employ unsupervised learning to get a deeper understanding of the data space. Specifically, we use a novel autoscaling technique to generate and assess ML algorithm instantiations on the fly, and we build a Cloud DevOps framework for autotuning with selection using container orchestration and communications between containers. Datasets pertaining to cloud network security are used to illustrate the suggested technique and tool.**

Keywords—Intelligent DevOps, Machine learning, System logs, machine learning, data preprocessing, data generation, Azure Devops

I. Introduction

Much advancement in intelligent communication as well as the Internet sectors has occurred with the ongoing deepening and growth of IT businesses based on AI, machine learning, and blockchain technology. Established IT companies deal with massive volumes of data every day as a result of DevOps (Development & Operations) tasks. One of the most challenging aspects of that endeavor is the fact that these data come from a variety of sources and might be in a variety of forms. Hardware as well as software systems DevOps that is both efficient and cost-effective becomes a critical issue that must be addressed [1]. To shorten "development, security along with operation and maintenance"[1], the acronym DevSecOps is used. Every step of the software creation cycle, from design to integration to testing to deployment and delivery, is automatically enhanced with security. Application as well as infrastructure security may now be shared across development, security, and it operations teams thanks to DevSecOps, which can incorporate system security throughout the DevOps process [2]. An growing number of apps are being built and distributed to different users inside an organization using cloud-based platforms. The use of cloud-enabled platforms and resources allows for many advantages, such as the ability to pool a group of talented people via virtualized environments and inexpensive access to vast on-demand processing power, storage, and network capacities [3]. An open-source machine learning platform, Kubeflow is used all around the world. Machine learning along with deep learning process deployment, management, and scalability are its key functions. One of the most popular container orchestration platforms, Kubernetes, is the one this platform was designed to work with. Kubeflow is an all-inclusive MLOps platform that helps data analysts, ML scientists, and DevOps teams work together efficiently across the whole ML lifecycle [4]. Many industries have made use of digital twin technology, which creates a digital representation of a real product. The digital twin patient approach allows healthcare providers to digitally assess the effects of different therapies on a virtual patient, without the risk of harming an actual patient. When faced with the many decisions that must be made in the ICU, this may be a helpful tool [5]. To create enormously parallel asynchronous brain simulations, the current status of neuromorphic computing relies on complex hardware architecture. This reliance impedes the integration of biological information into technological systems. The time it takes to get from concept to completed hardware is usually in the years, and even then, it isn't usually accessible to the public [6]. Researchers and practitioners in the field of DevOps have increasingly turned to cloud computing as a paradigm for rapidly provisioning infrastructure. Due to many restrictions including performance, compliance regulations, geographical reach, as well as vendor lock-in, the old strategy of choosing just one cloud service has lately gone out of favor [7]. Network security involves implementing both software and physical safeguards to prevent unauthorized use, abuse, breakdown, tampering, or improper disclosure of the networking infrastructure. This

ensures that computers, users, along with programs can safely execute their authorized critical functions. Any collection of interconnected computer systems in a relatively compact physical location, such as an office, school, lab, or residence hall, is known as a LAN (local area network) [8].
A variety of developmental processes, the maintenance of cellular lineage specificity, and the definition or stratification of cancer and other disorders are all supported by cytosine changes in DNA, such as 5-methylcytosine (5mC). There is a need for standardized materials, methodologies, as well as rigorous benchmarking to enhance genome-wide methylome sequenced applications in fundamental and clinical research, given there is a large range of ways available to investigate these alterations [9]. More and more people are interested in building Machine Learning apps with their data as big data becomes commonplace in various industries. When developing an ML application, it is common for ML specialists and domain experts to work closely together. But a major issue is still the lack of ML engineers. Low-coding Machine learning tools/platforms (also known as AutoML) automate many repetitive processes in the ML pipeline, with the goal of making ML development accessible to domain experts. These tasks include data collection, feature engineering, simulation design, ideal hyper-parameter setup, and model evaluation [10].

This study makes a triple contribution. Utilizing a case study from a top-tier telecommunications corporation, we first outline the primary obstacles faced by current methods when it comes to creating and archiving log data for intricate software-loaded systems. Our second contribution is a method for creating log data that is free of these issues and ready to be applied by algorithms for machine learning right away. Thirdly, we validate the strategy by conducting expert interviews, which prove that it solves the difficulties and challenges that have been identified.

II. Related Work

While taking into account the principles of a DevOps culture, the authors of [11] provide a roadmap for the deployment of applications using micro services architecture, giving practitioners a starting point for developing the platform required by this design. Their study was carried out in accordance with the DSRM, which stands for Design Science Research Methodological for Information Systems. By doing a Systematic Research Mapping as well as a Gray Literature Evaluation, they were able to pinpoint the issue and establish the goals for the remedy; this allowed us to develop the suggested guide. Here is a brief overview of this work: (I) Looking for ways and resources to assist put microservices into place. (II) The best practices, potential roadblocks, and recommendations for carrying out the implementation are detailed. (III) Planning the rollout of microservices using SPEM. (IV) A compilation of data from a manual on how to establish microservices by using DevOps concepts. In order to assess how well agile approaches (like Scrum) and DevOps handle cloud-induced challenges, this study will first collect theoretical research that identifies these issues, then develop core theories to explain the case study that will be utilized to test these hypotheses. All things considered, the thorough investigation that will be discussed in a subsequent communication will build upon the foundation laid forth in this document. In particular, the latter will make an effort to clarify the issues that crop up when businesses use DevOps or Scrum to build their cloud-based applications. The goal of the study [13] is to "discover" certain guidelines for building Big Data platforms in the cloud that can handle complicated and diverse settings. The importance, difficulties, and architectural effects of Big Data are all part of the design scope. In order to discover the design principles and rules together with pertinent aspects of Big Data components, the Reverse Engineered Design Science Study technique is used. This is done using artifacts from top vendors. They draw the conclusion that the results have practical and relevant applications for DevOps designers and practitioners managing diverse and complicated Big Data systems in the cloud. In order to determine research goals in service that expand the bounds of existing research, the authors of this article [14] drew on feedback from service academics, practitioners, literature reviews, and key policy papers. Their emphasis in the companion post was on four areas of service research that are particularly important for managing and providing service during times of uncertainty. In addition to the four objectives and three priorities laid forth in the accompanying article, the authors additionally included research questions that link the stated aims to an array of stakeholder-wants culled from the relevant literature. Platforms as well as marketplaces (SRP6), solutions for disadvantaged people and their neighborhoods (SRP7), and complex service ecosystems on a broad scale for effect transformation (SRP5) are the three main areas of focus in this article. Sustainable service ecosystems are an important goal, and they stress the importance of both theory and practice in this area. The National Science Foundation supported this effort in order to fill the gap in primary care consultations with chronic care services [15]. Design science research techniques were used to study medical primary care procedures and develop a digital workflow. The cutting-edge digital platform of this project is already fulfilling the communication needs of COVID-19 patients and covering dozens of chronically ill people at home. Doctors and nurses may talk to their chronic patients in a safe and thorough way, give the medications they need, and tell them with the COVID-19 precautions. Patients undergoing quarantine or isolation may also have their vitals monitored and evaluated using that platform. By enabling follow-up patients, thateHealth digital platform presents a chance to manage chronic care throughout epidemics, reducing the likelihood that patients would become uncontrolled and need emergency room treatment. In [16], the authors describe their experience in creating the Pediatric Cancer Data Commons, highlighting the significance of that effort in fighting pediatric cancer and improving outcomes and providing vital information to persons creating resources in other areas of health. The PCDC group from the University of Chicago's School of Medicine has been working with researchers from all across the globe to create a pediatric cancer data commons since 2015. According to their findings, a data commons that is both well-designed and executed needs six main features. Here are some of the things that need to be done: establish and implement governance, build and deploy the technical infrastructure that the data commons needs, create a platform that researchers can easily use, encourage the sharing of knowledge and expertise among researchers, and finally, plan for the data commons to be around for a long time. Data commons are crucial for research on large patient cohorts, which is necessary to improve cancer outcomes for children. It is beneficial to combine high-quality clinical as well as phenotypic data with data from other sources, such as genomics, proteomics, and imaging. Examine the Cloud SaaS security trends in [17]. All the way from system and information security to privacy, we're working on patterns that encompass it all. Building foundational Cloud SaaS apps is our top priority, therefore we're documenting security guidelines and acquiring security expertise to help SaaS developers accomplish just that. Along with that, we provide a case study that delves into security trends and remedies in AWS with Azure.Create and monitor the distribution and deployment of project artifacts, including code and build scripts, as part of the project tasks described in [18]. For administering your IoT Edge installations, Azure DevOps is a good option because of its support for containers and its interaction with Docker. On top of that, there are specialized processes for developing IoT Edges that streamline the release and build procedures. Cloud DevOps in Azure is a vast subject. Our emphasis in this chapter will be on the processes and workflows needed for developing and deploying Azure IoT Edge applications, once we have established some of the core principles. It covers a lot of ground in terms of Kubeflow's performance, from configuring it on different cloud platforms to installing their learnt model for internet serving. Using stroke as an example, the research group in [19] intends to develop an online previsit app and, using a participative approach, discover how potential end users view it. Using service architecture principles, a panel of experts provided feedback while users provided qualitative data for the digital tool's development and testing. A team that includes a patient companion processed all aspects in workshops. Questions on health issues and health records were included in the final product. Stroke patients from a Swedish outpatient clinic as well as patient groups participated in the study. Data gathering and development happened simultaneously. The Post-Stroke Checklist along with additional studies served as the foundation for the original prototype's conception. In order to learn about the needs and relevance, they conducted focus groups. To learn about the perceived usefulness, simplicity of use, and acceptability, authors employed a web survey as well as individual interviews. Through the use of mobile aided language learning, the authors of that research want to further the idea of digital literacy education, particularly as it pertains to the use of mobile devices for media consumption [20]. With the use of an example scenario, this article compares and contrasts several load balancing strategies for resource allocation. Applications that use these load balancing methods include prioritized medical data storage systems and real-time systems. Under the load balancing component, this study work is carried out in Windows Azure Platforms as a cloud operating system.

III. Proposed WOrk

A. System Model

Continuous delivery of added value to end users is enabled by the union of employees, procedures, and products, which is known as DevOps. Bringing DevOps' lifecycle management to ML is what "DevOps for ML" is all about. Streamlining processes and cooperation, DevOps uses Machine Learning to manage, monitor, as well as version models with ease. For DevOps to be successful in real-time systems, the machine learning (ML) lifecycle must be managed effectively.

A strong automated engine for script-based tools for automation, Azure by Microsoft DevOps is robust, cross-platform, and powerful. With Azure DevOPS, you can create, test, and release apps that are either Cloud Native or Non-Cloud Native. The accessibility of automation techniques like infrastructure as code as well as the easy integration of verifiable structures like Machine Learning Operation (MLOps) into the DevOps computerized pipelines to provision as well as configure the computing resources that applications need to run form the core principle along with chief advantage of Azure DevOps. Azure DevOps is the most essential framework that many organizations are quickly incorporating into their business processes to reduce product building costs and improve customer success. This is because application complexity is increasing and software development lifecycles are incorporating Machine Learning (ML) as well as Artificial Intelligence (AI) techniques. This paper aims to help small-scale farmers at the base of the economic pyramid by proposing a new DevOps framework for developing intelligent Small ML dairy farming sensors and by outlining the benefits of DevOps in developing high-quality products efficiently.

Alternatively, we may save the information as a CSV file or an Excel spreadsheet. Which at subsequent phases may be utilized for data analytics.

The logs include software application and services readings exported in CSV format from Power BI. We will forecast the machine's status using this data. The machine's state is classified in the following manner:

There are five levels of condition:

- excellent,
- good,
- rough,
- very rough,
- Dangerous, and severe.

Characteristics such as software type, application type, service type, time, and delivery type are known as Features, while the conditions corresponding to these numbers are known as Labels. In order to train this set of data to map the actual labels, we will use feature extraction on it. Next, the actual outcome is determined by comparing the Trained dataset with the test set.

C. Ensemble Machine Learning

In machine learning, an ensemble learns to merge several separate models into a single, more robust and accurate prediction model. Ensemble learning is a method for improving outcomes in machine learning as well as data analysis by combining the capabilities of many models to reduce the impact of mistakes, boost performance, and make predictions with more resilience. The use of ensemble learning in ML allows for the improvement of ML model performance. The idea behind it is straightforward. A more accurate model is obtained by combining several machine learning models.

To detect data from the model drift and automatically start the model development, many strategies are used.

- Data in real-time: receive the values of actual sensor data first.
- true_data = [0.02, 0.03, 0.05, 0.14, 0.16, 0.15, 0.12, 0.08, 0.1, 0.08, 0.07]

The sensors with signal data are being enhanced with machine learning, resulting in devices that are smarter than ever before. A major step forward for the Internet of Things (IoT) is the use of machine learning to sensor and signal data, which is making gadgets smarter than ever before. Utilizing a low-cost micro control system for processing, you can train a machine to identify and categorize events occurring at the edge in real-time, even with noisy, high-variation data. This can be done with a variety of sensor data types, including but not limited to: images, electrical signals, accelerometer readings, vibrations, and sound. The result is richer analytics.

Here are the steps involved in EML:

1. M algorithms are trained using the original training data.
2. A fresh training set is generated by using the result of each algorithm.
3. We build a meta-model algorithm using the updated training data.
4. The final forecast is made using the meta-model's findings. We average the weights of each outcome and then combine them.

In order to categorize network assaults, we have experimented with various arrangements of basic classes and used them in the voting approach. For our voting ensembles machine learning technique, we are using three basic classes: Decision Tree (DT), Random Forest (RF), and Logistic Regression (LR).

1) Logistic Regression: In statistics, logistic regression is a method for predicting a dependent variable using a collection of independent factors. It is among the most well-known algorithms used for binary problem solving within supervised machine learning. For our purposes, it linearly divides the data into two groups: those with values that are present (1) and those without values (0).

2) Decision Trees: Classification and regression problems are both amenable to Decision Trees. Databases may benefit from it for knowledge mining since it represents rules that people can understand. An upside-down tree represents the algorithm's structure, with each node representing an experiment on a single feature with each leaf node specifying the value of the target attribute. The greedy basic algorithm is used to build the tree. At the very base of the tree, we determine which qualities in the dataset are the best by calculating their Gini significance. Mean decrease impurity, also known as Gini important, is a metric of attribute selection that takes into account the overall reduction in node impurity across all trees in the ensemble. The splitting attribute is chosen based on which one minimizes the Gini index or maximizes the decrease in impurity. If you provide a tree depth as a hyperparameter, we'll keep splitting until we reach the leaf nodes, where the pure or majority class is located. When a tree has reached its maximum or minimum height, depending on the hyper-parameters, it is pruned, which entails deleting nodes from the tree. This technique aids in avoiding over-fitting and serves as a regularization variable. One benefit of using a choice tree is that the rules it generates are very human-friendly.

$$GiniIndex = 1 - PPq\ 2 \qquad (1)$$

where, Pq = frequency of class q in the dataset.

3) Random Forest: Additionally, there is a little difference between the Bagging method and the Random forest technique. The total amount of features that may be used for the split is restricted to the squared root of the total amount of input features, which is also picked at random, once m training sets have been formed. This little adjustment produces trees that are highly independent of one another, leading to more varied forecasts. Past research has shown that combined forecasting often outperforms either a single tree or the technique of bagging.

Voting

In order to arrive at a final forecast, the Voting Classifier ensembles method compiles output from several separate models, or base estimators. By considering the combined opinion of several models instead of relying on just one, it employs the "wisdom of the crowd" idea to get more precise forecasts. Both hard voting, in which every model predicts something, as well as soft voting, in which every model predicts how likely something is for every category or label, are used in the Voting Classifier. When all the models' predicted probabilities are added together, the final forecast is based on the class with the greatest average probability. You may manually apply weights to each model or let them learn from each other's performance; either way, they all have an impact on the final prediction thanks to weighted voting. Various models may have various effects on the final forecast due to this variety.

The Voting Classifier improves overall performance and resilience by integrating many models, which is particularly useful when different models have different features and provide separate predictions. By combining the decisions of many models, the Voting Classifier is able to overcome the limitations or biases of a single model, resulting in more reliable and accurate predictions. By combining the strengths of many models, Voting Classifier is able to provide more precise predictions in a wide range of machine learning tasks. There are several advantages to using the Voting Classifier, a flexible ensemble method in machine learning. It reduces prediction bias and variance while increasing model variety, accuracy, and resilience by the integration of many models with different strengths and limitations.

Due to its ensemble structure, the Voting Classifier reduces overfitting and increases model variation, both of which contribute to improved model stability. You may tailor it to the goals and features of individual models by choosing from a variety of voting techniques, such as hard casting votes, soft voting, as well as weighted voting. In addition, by analyzing the inputs of many designs, the Voting Classifier may enhance interpretability, which aids in comprehending the decision-making process and underlying patterns. As a whole, the Voting Classifier does a good job of improving the prediction capabilities of different machine learning jobs.

It is possible to categorize the network assault by voting as an ensemble. The first step of a voting algorithm is to develop a voting classifier; the second step is to use this classifier to generate a prediction. Below is the pseudocode for the voting prediction:

- Think about the test characteristics, apply the rules of every base classifier to make a prediction, then save the target prediction.
- You'll need to figure out how many votes each predicted target deserves.
- The highest number of votes will be used to determine the final outcome.
- Make an educated guess using the voting algorithm

Here are the steps we took to put the voting system into action:

- Step one: load the dataset.
- Step two: initialize the basic classifiers.
- Step three: use the base classifiers to apply a voting approach.
- Step four: predict the outcome based on the overwhelming vote.
- Calculate the detection rate using the confusion matrix value.

IV. Results & Discussion

A. Azure DevOpsComponents

Azure DevOps has six key components that make it a full-featured cloud Software Activity Monitoring (ALM) solution. Tables in Azure: Now is the time to start documenting the business requirements and user stories so the sprint can begin. A variety of version control systems are compatible with Azure Repos, including Azure Git, GitHub Enterprise, Public GitHub, External Git, as well as Microsoft's (TFVC). In a collaborative working environment, we may use each of these source control tools to make a pull request, send our work for review, and participate in the process. You may integrate continuous integration and ongoing delivery (CI/CD) for your source code using Azure Pipelines. Azure Test Plans allows you to create and execute automated, manual, and load test cases. Azure Artifacts allows you to publish your packages using Maven,NPM, NuGet, as well as Python.

B. Implementation Model

After obtaining the dataset, the initial phase in this study has been data cleaning using the computer language Python. It became clear in several use cases that we are receiving an overwhelming quantity of data. Some of the info was relevant, while some was unimportant. By compiling this information and constructing a machine learning model, useful sensor data may be retrieved. What follows is an instructional on:

- This tutorial will walk you through the steps of setting up an ESP32 board,
- Acquiring an IoT long-range wireless humidity and temperature sensor.
- Connecting the two, flashing the ESP 32, collecting data from the sensor.
- Analyzing and displaying the data in Power BI, exporting the data in CSV, and finally, creating a data set.

The results from the wireless humidity and temperature sensors are as follows:

Number of degrees Celsius

• Celsius (the degree of heat)

Factors Like:

• Humidity Level

• Charge Time

After that, Azure IoT hub is used to display and analyze this data. This guide will walk you through the steps of setting up Azure IoT Hub. To transmit the values to the Azure IoT hub, follow these steps.

Data publication and subscription in Azure IoT hub is facilitated using the MQTT protocol.

- The Azure site also offers the useful Azure functionalities. Azure functions allow us to create cloud-based code or functions. As part of this project, we are using an Azure function to extract the actual temperature and humidity readings from raw sensor data contained in JSON format. Follow this instruction to set up an Azure function.

- By parsing JSON raw data, we can obtain the actual humidity and temperature readings.

C. Analyzing Data in PowerBi

Power BI is being used to display the data. It offers tools to study data in an interactive fashion. Line graphs, bar graphs, pie charts, and other similar visual representations may be used to understand this data. Get going by registering for and logging into Power Bi. Using a stream analytics task, we had already built up Power Bi along with fed data to it in the previous article. In this article, we will show you how to transmit information from sensors to power Bi using an Azure function. Peruse this blog post for instructions on configuring Power Bi.

Four different ways exist for transferring data to Power Bi:

- Immediate data transmission to power Bi via Internet of Things hub.
- Sending information to Power BI using API.
- Making use of web-hook features Utilizing PubNub.

In this example, we're using the Power BI API to interact with Power BI via an Azure function to provide an HTTP response. The Visualization panel offers many graphs, graph types, line charts, pie charts, etc. Using the Visualization panel, we can choose a graph to use as a basis for the chart.

Our data set has been split into two halves:

- **Training Set**- Includes 800 features in the training set.
- **Test Set**- Contains 800 characteristics for use in training.

After training the data using linear regression, we go on to linear discriminant analysis, or LDA. After LDA feature extraction is complete.

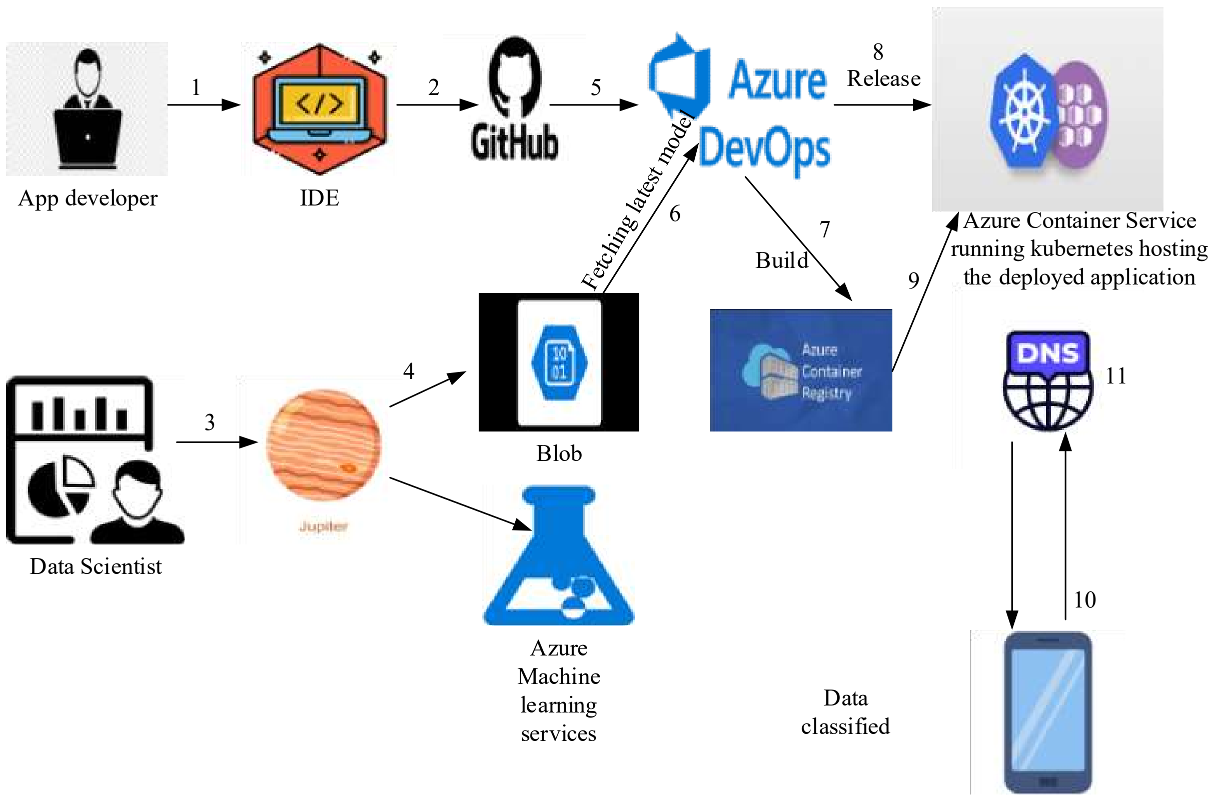

Fig.1. Azure Devops for Machine Learning in Risks Classification **C. Results**

Although these methods improve learning models, there is still a need for solutions that automate both tuning and selection inside a Cloud DevOps setting, as well as methods for dynamically selecting among adjusted models. This paper aims to offer an evolving method and context for choosing a model as well as tuning in circumstances where signal integrity and information content change frequently. It also discusses how tuned algorithms need to be revised in a DevOps framework. Previous work has successfully found optimal machine learning algorithms.

Evaluation Metric: The detection rate is a key performance metric in an intrusion detection model. The proportion of log records that were successfully identified as classes divided by the overall number of log data.

$$TPR = TP/(TP + FN) \quad (2)$$

Where, TP = true positive , TN = true negative, FP = false positive and FN = false negative.

Several Machine Learning Algorithms are put into action using the data matrix that is created during pre-processing. To evaluate our progress, we have settled on the following metrics.

Correctly detected points as a percentage of total points or incidences constitute accuracy. We have a dataset where the classes of the target variable are very well distributed, thus this metric will work well for us.

It is important to limit the amount of false positives in order to construct a model that never forecasts data logs from numbers to log records, and sensitivity plays a key part in this process. Selecting a model having high sensitivity is essential for reducing false positives and increasing the number of real positives.

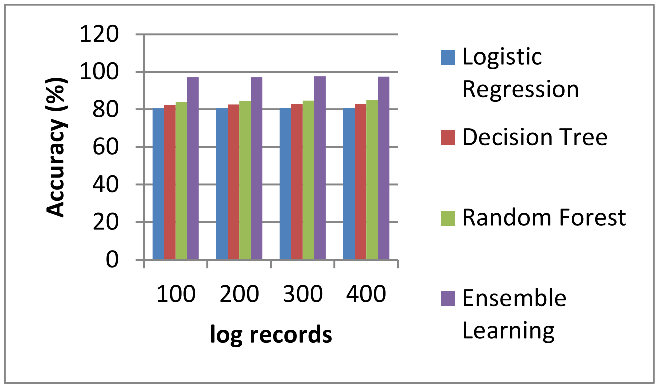

Fig.2 Accuracy Analysis

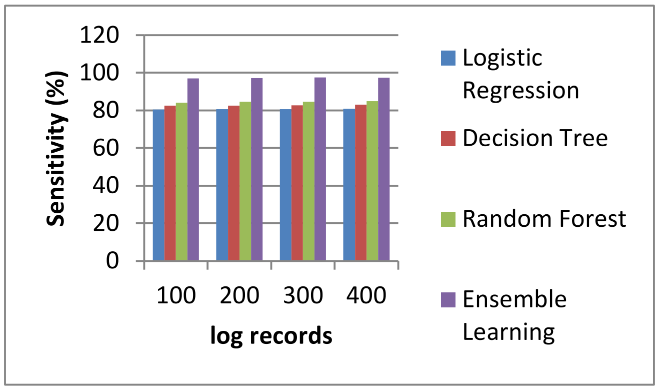

Fig.3 Sensitivity Analysis

Table.1 Comparison Analysis

Algorithm	Accuracy	Sensitivity
Logistic Regression	80.5	80.4
Decision Tree	82.5	82.45

Random Forest	83.6	83.45
Ensemble Voting	97.15	96.15

V. Conclusion

Model selection based on changing information evolution is shown using a DevMLOps approach. Older tuned algorithms are destroyed when they no longer achieve appropriate prediction accuracy, while fresh models undergo training offline then brought online as required. The prediction accuracy of the models that are brought online is further adjusted automatically to match the changing data set. Within the framework of distributed learning, the needs for dynamic feature selection are outlined. The future of our species is shaped by data, which also provides the greatest defense against catastrophic changes like climate change. In order to ensure continuous applicationsecurity, it is the organizer's responsibility to create a more involved atmosphere based on conclusions made from the facts. One important thing about tiny ML devices is that they make machine learning and data science accessible to small-scale farmers. Cost reductions for items that may be offered to struggling small-scale farmers who demand resources to benefit the world are made possible by systems with high performance, like Azure DevOps. We introduce DevOps approaches, which are procedures for system engineering. To show that dynamic model selection is better than statically trained ensemble learning, we compare it to boosting ensemble learning. We want to expand our current collection of machine learning (ML) models for autotuning and autoselection to include reinforcement learning models in the near future. We want to test the enhanced toolbox on several cloud computing datasets as well.

ReferencesArvola, M., Fuchs, I.E., Nyman, I., & Szczepanski, A. (2021). Mobile Augmented Reality and Outdoor Education. *Built Environment*.

[1] Mwotil, A., Bainomugisha, E., & Araka, S.G. (2022). mira: an Application Containerisation Pipeline for Small Software Development Teams in Low Resource Settings. *Proceedings of the Federated Africa and Middle East Conference on Software Engineering*.

[2] Fursin, G. (2020). The Collective Knowledge project: making ML models more portable and reproducible with open APIs, reusable best practices and MLOps. *ArXiv, abs/2006.07161*.

[3] Larsen, L.B., Karnøe Stagsted, R., Strohmer, B., & Christensen, A.L. (2021). CloudBrain: Online neural computation in the cloud. *bioRxiv*.

[4] Achar, S. (2021). Enterprise SaaS Workloads on New-Generation Infrastructure-as-Code (IaC) on Multi-Cloud Platforms. *Global Disclosure of Economics and Business*.

[5] Alamin, M.A., & Uddin, G. (2022). Challenges and Barriers of Using Low Code Software for Machine Learning. *ArXiv, abs/2211.04661*.

[6] Schigel, D., Andersson, A.F., Bissett, A., Finstad, A.G., Fossøy, F., Grosjean, M., Hope, M., Kõljalg, U., Lundin, D., Nilsson, R.H., Prager, M., Jeppesen, T.S., & Svenningsen, C.S. (2020). Mapping and Publishing Sequence-Derived Data through Biodiversity Data Platforms.

[7] Islam, M.S., Pourmajidi, W., Zhang, L., Steinbacher, J., Erwin, T., & Miranskyy, A.V. (2020). Anomaly Detection in a Large-Scale Cloud Platform. *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 150-159.

[8] Iot, M. (2020). Microservices Iot And Azure Leveraging Devops And Microservice Architecture To Deliver Saas Solutions.

[9] Hansen, S., Hilberg, O., Ulrik, C.S., Bødtger, U., M. Rasmussen, L., D. Assing, K., Wimmer-Aune, A., B. Rasmussen, K., Bjerring, N., Christiansen, A., Schmid, J.M., Krogh, N.S., & Porsbjerg, C.M. (2020). The Danish severe asthma register: an electronic platform for severe asthma management and research. *European Clinical Respiratory Journal, 8*.

[10] Niño-Martínez, V.M., Ocharán-Hernández, J.O., Limón, X., & Arriaga, J.C. (2022). A Microservice Deployment Guide. *Programming and Computer Software, 48*, 632-645.

[11] Tsilionis, K., Sassenus, S., & Wautelet, Y. (2021). Determining the Benefits and Drawbacks of Agile (Scrum) and DevOps in Addressing the Development Challenges of Cloud Applications. *Research & Innovation Forum*.

[12] Sharma, R.S., Mannava, P.N., & Wingreen, S.C. (2022). Reverse-Engineering the Design Rules for Cloud-Based Big Data Platforms. *Cloud Computing and Data Science*.

[13] Field, J.M., Fotheringham, D., Subramony, M., Gustafsson, A., Ostrom, A.L., Lemon, K.N., Huang, M., & Mccoll-Kennedy, J.R. (2021). Service Research Priorities: Designing Sustainable Service Ecosystems. *Journal of Service Research, 24*, 462 - 479.

[14] TWycherley, 6., Victoria, Australia, Wellbeing, 2., and, Chronic, Disease, M., & School (2020). Dealing with COVID-19 Barriers to Care: Digital Platform to support and monitor chronic patients. *The European Journal of Public Health, 30*.

[15] Plana, A., Furner, B., Palese, M., Dussault, N., Birz, S., Graglia, L., Kush, M.A., Nicholson, J., Hecker-Nolting, S., Gaspar, N., Rasche, M., Bisogno, G., Reinhardt, D., Zwaan, C.M., Koscielniak, E., Frazier, A.L., Janeway, K.A., S Hawkins, D., Kolb, E.A., Cohn, S.L., Pearson, A.D., & Volchenboum, S.L. (2021). Pediatric Cancer Data Commons: Federating and Democratizing Data for Childhood Cancer Research. *JCO clinical cancer informatics, 5*, 1034-1043 .

[16] Rath, A.T., Spasic, B., Boucart, N., & Thiran, P. (2019). Security Pattern for Cloud SaaS: From System and Data Security to Privacy Case Study in AWS and Azure. *Comput., 8*, 34.

[17] Jensen, D. (2019). Azure DevOps for IoT Edge Solutions. *Beginning Azure IoT Edge Computing*.

Vol. 19, No. 1, (2022)
ISSN: 1005-0930

[18] Kjörk, E.K., Sunnerhagen, K.S., Lundgren-Nilsson, Å., Andersson, A.K., & Carlsson, G. (2022). Development of a Digital Tool for People With a Long-Term Condition Using Stroke as a Case Example: Participatory Design Approach. *JMIR Human Factors, 9*.

[19] (2019). Performance Analysis of Load Balancing Algorithms using Microsoft Windows Azure Cloud Platform. *VOLUME-8 ISSUE-10, AUGUST 2019, REGULAR ISSUE*.