Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
ΔΠΜΣ Επιστημη Δεδομενων και Μηχανικη Μαθηση

Υλοποίηση και Αξιολόγηση Συστημάτων Ανάπτυξης και Λειτουργίας Λογισμικού Μηχανικής Μάθησης (MLOps). Εφαρμογή στη Σημασιολογική Κατάτμηση Γεωχωρικών Δεδομένων.

Μεταπτυχιακη Διπλωματικη Εργασια

ΤΡΙΑΝΤΑΦΥΛΛΟΥ Ν. ΔΗΜΗΤΡΙΟΣ

Επιβλέπων: Κωνσταντίνος Καράντζαλος
Αναπλ. Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2022

National Technical University of Athens
School of Electrical and Computer Engineering
MSc in Data Science and Machine Learning

Implementation and Evaluation of Machine Learning Systems, focusing on Development and Operations (MLOps). Application in Semantic Segmentation of Geospatial Data.

Master Thesis

TRIANTAFYLLOU N. DIMITRIOS

Supervisor: Konstantinos Karantzalos
Associate Professor at NTUA

Athens, December 2022

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
ΔΠΜΣ Επιστημη Δεδομενων και Μηχανικη Μαθηση

Υλοποίηση και Αξιολόγηση Συστημάτων Ανάπτυξης και Λειτουργίας Λογισμικού Μηχανικής Μάθησης (MLOps). Εφαρμογή στη Σημασιολογική Κατάτμηση Γεωχωρικών Δεδομένων.

Μεταπτυχιακη Διπλωματικη Εργασια

ΤΡΙΑΝΤΑΦΥΛΛΟΥ Ν. ΔΗΜΗΤΡΙΟΣ

Επιβλέπων: Κωνσταντίνος Καράντζαλος
Αναπλ. Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 21 Δεκεμβρίου 2022.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Κωνσταντίνος Καράντζαλος	Μαρία Βακαλοπούλου	Γεώργιος Στάμου
Αναπλ. Καθηγητής Ε.Μ.Π.	Asst. Prof. at CentraleSupélec	Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2022

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
ΔΠΜΣ Επιστημη Δεδομενων και Μηχανικη Μαθηση

Copyright © – All rights reserved. Με την επιφύλαξη παντός δικαιώματος.
Δημήτριος Ν. Τριανταφύλλου, 2022.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα.

Το περιεχόμενο αυτής της εργασίας δεν απηχεί απαραίτητα τις απόψεις του Τμήματος, του Επιβλέποντα, ή της επιτροπής που την ενέκρινε.

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

Περίληψη

Τα τελευταία χρόνια η μηχανική μάθηση γνωρίζει μεγάλη άνθιση στον τομέα της πληροφορικής. Η εποχή ψηφιοποιείται και τα δεδομένα μεγαλώνουν. Καθώς η μηχανική μάθηση και η τεχνητή νοημοσύνη διαδίδονται σε προϊόντα και υπηρεσίες λογισμικού, κρίνεται αναγκαία η εφαρμογή βέλτιστων πρακτικών και εργαλείων για τη δοκιμή, την ανάπτυξη, τη διαχείριση και την παρακολούθηση μοντέλων μηχανικής μάθησης στην παραγωγή πραγματικού κόσμου. Αυτές οι πρακτικές συνθέτουν τις διαδικασίες μηχανικής μάθησης MLOps. Στόχος των MLOps είναι η μείωση το «τεχνικού χρέους» σε εφαρμογές μηχανικής μάθησης.

Η χρήση στρατηγικών εικονικοποίησης με κοντέινερ έχει επικρατήσει στις εταιρείες και το Kubernetes, ως ενορχηστρωτής κοντέινερ, κερδίζει όλο και μεγαλύτερο έδαφος, ειδικότερα σε περιπτώσεις μεγάλης κλίμακας υποδομών και δεδομένων. Παράλληλα, οι διαδικασίες που πραγματοποιούνται σε ένα σύστημα μηχανικής μάθησης στην παραγωγή, με την πάροδο του χρόνου, μεγεθύνουν τις απαιτήσεις για χρήση πόρων και διαχείριση. Ο συνδυασμός της μηχανικής μάθησης με το Kubernetes, ευνοείται από τα κύρια χαρακτηριστικά του για ενορχήστρωση, κλιμακωσιμότητα και φορητότητα.

Σκοπός της παρούσας διπλωματικής είναι η υλοποίηση και διερεύνηση διαδικασιών MLOps στην παραγωγή. Υλοποιείται μια υποδομή με Kubernetes, που προσωμοιώνει ένα παραγωγικό περιβάλλον, καθώς και επιλέγεται η πλατφόρμα Kubeflow της Google για την ανάπτυξη των MLOps. Υλοποιούνται οι διαδικασίες MLOps και εφαρμόζονται σε ένα ζήτημα σημασιολογικής ταξινόμησης των στοιχείων εικόνων υποθαλάσσιου εδάφους. Ταυτόχρονα, περιγράφεται η συνεργασία επιστήμονων δεδομένων και μηχανικών DevOps, σε ένα σύστημα μηχανικής μάθησης στην παραγωγή και πως ευννοούνται από την αυτοματοποίηση που παρέχουν οι διαδικασίες MLOps. Τελικό παραγόμενο είναι μια σωλήνωση Kubeflow Pipeline, η οποία αναλαμβάνει να αυτοματοποιήσει την τακτική επανεκπαίδευση, προώθηση και παρακολούθηση ενός μοντέλου στην παραγωγή.

Λέξεις Κλειδιά

Μηχανική Μάθηση, Τεχνητή Νοημοσύνη, Εικονικοποίηση, Ενορχηστρωτής Κοντέινερ, Τεχνικό χρέος, Διαδικασίες Μηχανικής Μάθησης, MLOps, DevOps, Kubernetes, Kubeflow, Pipeline

Abstract

Machine learning has made tremendous advances in the past decade. As machine learning and artificial intelligence proliferate in software products and services, there is a need to apply best practices and tools for testing, developing, managing, and monitoring machine learning models in real-world production. These practices are known as machine learning operations - MLOps. The goal of MLOps is to reduce the "technical debt" in machine learning applications.

The majority of IT enterprises have adopted a container strategy. Kubernetes, as a container orchestrator, is gaining more and more popularity, especially in large-scale infrastructure and big data cases. Meanwhile, production machine learning systems are getting more complex requiring more and more resources and management. The combination of machine learning with Kubernetes, is favored due to Kubernetes key features of orchestration, scalability, and portability.

The purpose of this thesis is the implementation and analysis of MLOps in production. An infrastructure with Kubernetes is implemented, simulating a production environment, and the Google's Kubeflow platform is chosen for the development of MLOps. MLOps are developed and applied to a problem of semantic classification of seabed images. Additionally, there is an approach that describes the collaboration of data scientists and DevOps engineers in a production machine learning system and the benefits of automation provided by MLOps. The final product is a Kubeflow Pipeline which automates the regular retraining, deploying and monitoring a model in production.

Keywords

Machine Learning, Artificial Intelligence, Containers, Container Orchestrator, MLOps, DevOps, Kubernetes, Kubeflow, Pipeline

Extended Summary

Nowadays, machine learning is used in a wide range of applications and has been the focus of interest of many companies. The interest lies both in developing products for direct use of machine learning for prediction and classification, as well as in the aspect of recognizing trends in customer and business behavior to support existing products or develop new ones. In this manner, some operations are needed to continuously develop the machine learning models, check their performance, enhance them and integrate them into new or existing products. These operations are known as MLOps [1] and are of vital importance in production environments.

To develop a machine learning system, data scientists need to apply and train a machine learning model with predictive performance on a dataset, taking into account relevant training data for their use case. However, the real challenge is not building a model, the challenge is building a complete machine learning system and running it continuously in production. Only a small fraction of a real machine learning system consists of code. The required components, coexisting with the machine learning code, are many and complex [2]. These components are responsible for operations such as data collection, data verification, resource management, automation, serving models, monitoring and many more.

Many teams have data scientists and researchers who can build state-of-the-art models, but the process of developing and deploying models is completely manual. This process is interactive and scenario-based. Every step is manual, including data analysis, data preparation, model training, and validation. It requires manual execution of each step and manual transition from one step to another. This process is typically driven by experimental code written and run by the data scientist interactively, until a working model is produced. This process depends on the data scientists who build the model and the engineers who make the model available as a predictive service. Data scientists deliver a trained model to the engineering team to incorporate into their infrastructure. This handoff may include placing the trained model in a storage location or code repository, or uploading it to a model registry. In this way, this process does not have frequent updates of the model, and the performance of the model is not actively monitored.

MLOps are a key aspect of machine learning engineering that focuses on simplifying and speeding up the process of delivering models to production, maintaining and monitoring them. They involve collaboration between different teams, including data scientists, DevOps engineers, IT specialists, and others. These operations aim at the unification of system development and the operation of a machine learning system. MLOps include the following practices : Continuous Integration (CI), Continuous Delivery (CD),

Continuous Training (CT) and Continuous Monitoring (CM) . CI focuses on testing and validating data, data schemas, and models. CD is about delivering of a training pipeline of a machine learning system that automatically makes another predictive model service available. CT is only found in MLOps, not DevOps, and it is concerned with automatically retraining and serving the models. CM refers to the monitoring of production data and model performance metrics. All these steps lead to an automated pipeline solution.

Cloud computing companies have invested a great amount of money in infrastructure and management. These companies invest in research and development of specialized hardware, software, and SaaS applications, but also MLOps software. Two of the leading commercial platforms are Amazon Sagemaker and Microsoft Azure MLOps with the individual tools AzureMachine Learning, Azure Pipelines, and Google Cloud MLOps with the tools Dataflow, AI Platform Notebook, TFX, and Kubeflow [3]. A tech company has to take into consideration several factors deciding if it should buy, build or go hybrid with its MLOps infrastructure. It can take over a year to build a functioning machine learning infrastructure. It may take even longer to build a data pipeline that can produce value. Moreover, cloud solution costs and human resources needed for building and maintaining in-house solutions are two factors needing leveraging. Tech companies that want to survive long-term usually have in-house teams and build custom solutions. If they have the skills, knowledge, and tools to tackle complex problems, then this approach is probably well suited.

For the purposes of this thesis, an infrastructure was implemented to host MLOps. More specifically, Kubeflow platform, originally created by Google, was chosen to run on top of a Kubernetes [4] cluster. Kubeflow is a many tool platform, hosting every component needed in machine learning workflow. It translates steps of a data science workflow into Kubernetes jobs, providing the cloud-native interface for ML libraries, frameworks, pipelines and notebooks. Kubeflow benefits of Kubernetes rich ecosystem. Kubernetes is commonly used in production environments, as an open-source system for automating deployment, scaling, and management of containerized applications.

In order to simulate a production environment and avoid single point failure, a high available Kubernetes cluster was created. Also, for the needs of a production ML system, the cluster was GPU enabled. Two alternatives were explored to create a cluster on a single physical machine. Firstly, a solution was implemented with Multipass tool of Canonical, creating three virtual machines. At each machine, that ran Ubuntu 20.04, Kubernetes was installed and afterwards they were all joined in cluster. For the use of GPU by the VMs, PCI passthrough was needed, but only one VM would be able to use it and the host machine would no longer be able to. So, this solution was not continued. Secondly, a solution with LXD manager was implemented. Three LXC containers were spawned to host Kubernetes nodes and finally create a cluster. With a simple LXC profile configuration the containers were able to use the GPU. In the matter of Kubernetes, three flavors were explored, two of Rancher, the RKE2 and the K3s, and one of Canonical, the MicroK8s. The latter was chosen due to the better open-source support of Kubeflow. Kubeflow was custom installed on the cluster with Kustomize tool, getting the latest features at the time, using the release candidate version v1.6.0-rc.0.

MLOps and infrastructure created were used to tackle a real machine learning problem of Seabed classification. In this case, offline data were available for three consecutive years of 2016, 2017 and 2018 for the Bedford Basin bay. Each year data contained a backscatter image, a bathynetry image and a ground truth image that was constructed as labeled data. Continuing the work of Mertikas P. and Karantzalos K. [5], an automated Kubeflow pipeline was created for hyperparameter tuning, training and model serving. Main tools of Kubeflow that were used are Jupyter Notebooks along with Kubeflow SDK [6], Katib [7] for hyperparameter tuning and KServe [8] for serving model, managing rollouts and monitoring the inference services. For the training of the model, a docker container was created with two ways, one solely with Kubeflow SDK and one with the normal process with Dockerfile. Kubeflow pipelines were solely created with the Kubeflow SDK coding in python. The created pipeline is depicted in the following picture.

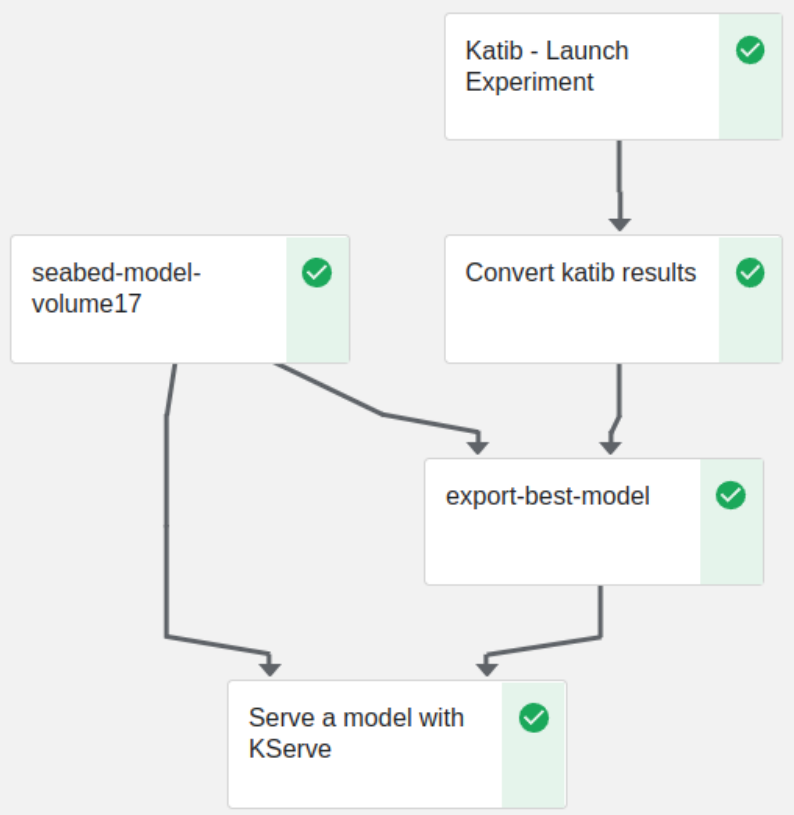


Figure 1: *End to end pipeline for seabed classification*

The pipeline starts with the hyperparameter tuning, using the Katib tool. Several runs are made, trying different hyperparameter sets with Bayesian optimization. The output of this stage is the best set of hyperparameters based on one chosen metric. In this case, Katib tries to maximize accuracy of the model, while also logging Cohen's kappa, precision and recall of classes. Hyperparameters available for tuning are the classifier used, the percentage of training set against the validation, the embedding dimensions and embedding sigma.

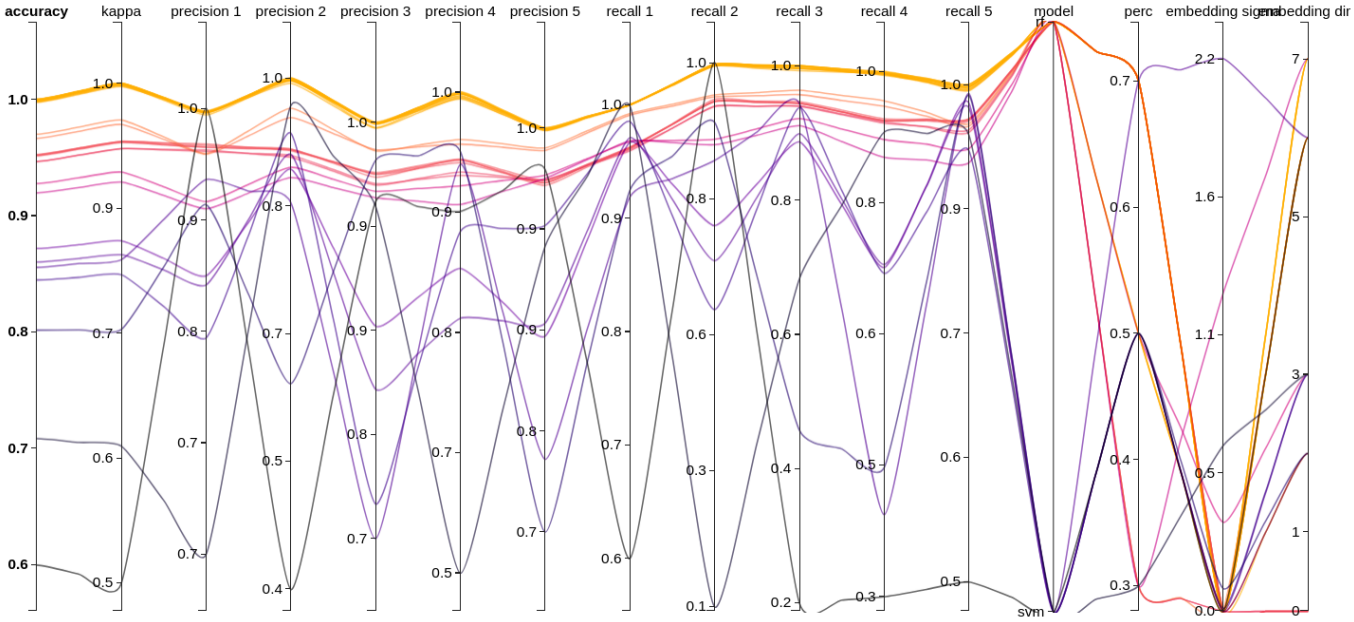


Figure 2: *Hyperparameter tuning with Katib*

Status	Accuracy	Kappa	Precision 1	Precision 2	Precision 3	Precision 4	Precision 5	Recall 1	Recall 2	Recall 3	Recall 4	Recall 5	Model	Perc	Embedding sigma	Embedding dir
Succeeded	0.99796	0.9974	0.99815	0.99702	0.99813	0.99691	0.9997	0.99969	0.99758	0.99816	0.99668	0.99579	rf	0.7	1.79734e-5	7
Succeeded	0.97306	0.96554	0.97668	0.96128	0.98521	0.95134	0.98447	0.99544	0.95933	0.97448	0.96482	0.95031	rf	0.3	7.00753e-3	6
Succeeded	0.86093	0.8207	0.91391	0.92941	0.73521	0.96297	0.88991	0.96811	0.64809	0.98613	0.60768	0.98028	svm	0.3	0.65418	3
Succeeded	0.88297	0.85003	0.90409	0.93559	0.85817	0.79646	0.93495	0.96691	0.63868	0.94697	0.91652	0.9041	svm	0.5	8.92547e-2	2
Succeeded	0.90313	0.87655	0.94798	0.82543	0.97341	0.84401	0.88609	0.94985	0.88602	0.83645	0.86477	0.97987	svm	0.5	6.9168e-6	2
Succeeded	0.80861	0.75375	0.97381	0.86384	0.69652	0.98707	0.69319	0.92711	0.81134	0.97126	0.10737	0.99937	svm	0.5	4.46636e-6	6
Succeeded	0.92857	0.9086	0.94716	0.87236	0.94521	0.9092	0.96161	0.97875	0.87766	0.91755	0.9226	0.91743	rf	0.5	0.3514	3
Succeeded	0.91812	0.89519	0.93772	0.85341	0.93873	0.89475	0.9598	0.97684	0.86214	0.9035	0.90973	0.90425	rf	0.3	1.25708	7
Succeeded	0.99757	0.99689	0.99773	0.99732	0.99845	0.99428	0.99949	0.99957	0.9968	0.9988	0.99636	0.99349	rf	0.7	7.28999e-6	6

Figure 3: *Table of trials of Katib*

The next step of pipeline is auxiliary for conforming the output of Katib to a proper format for the next step. Also auxiliary is the step of creating the PVC storage to store the model created. Kserve supports Azure blobs, Amazon S3, URI and PVC for model storage. In this case, the latter was chosen. Following up, in the next step, the model was trained with the best set of hyperparameters and stored at the PVC. In the final step, the model is served with KServe using a Canary rollout, routing the total traffic to the inference service created.

All Kubeflow procedures can be monitored by Central Dashboard of Kubeflow. The runs of a pipeline are organized in Kubeflow experiments. Every stage of a pipeline run produces logs, artifacts and visualations if implemented.

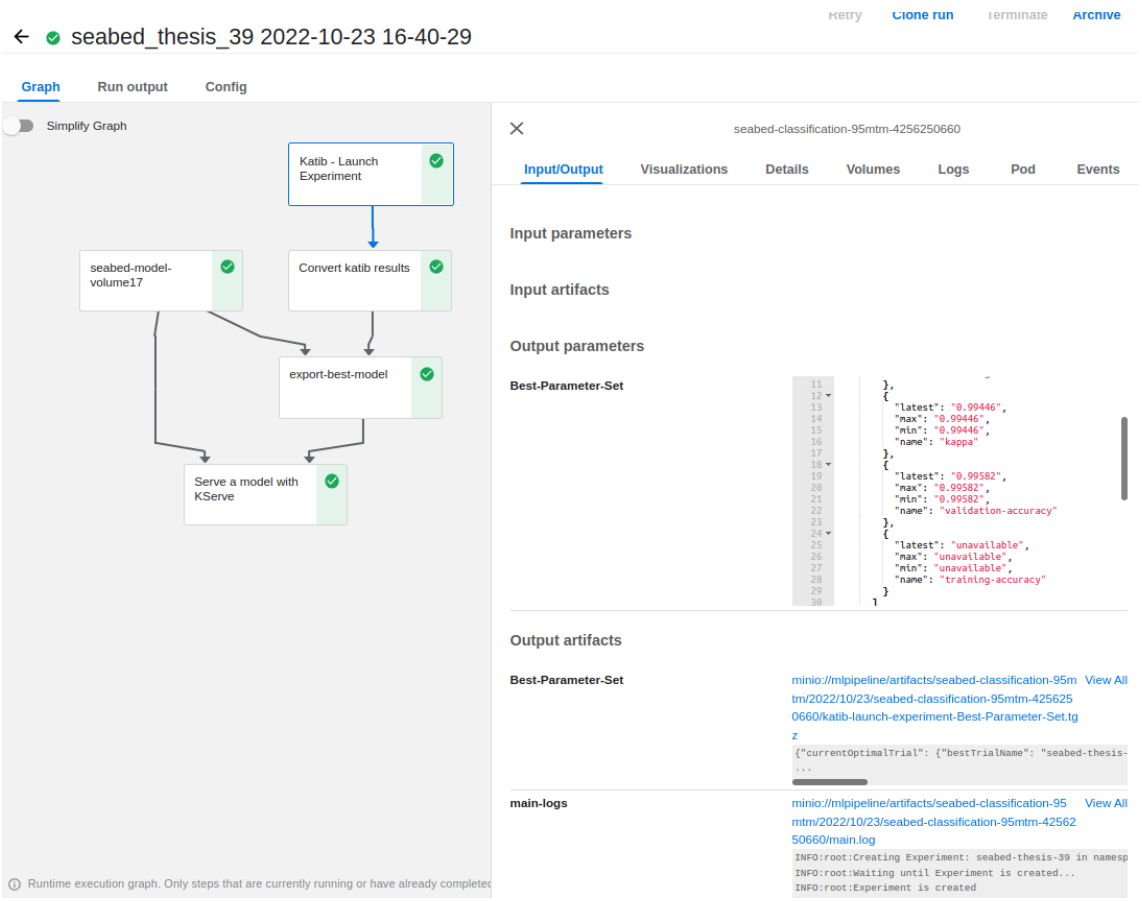


Figure 4: *Run page*

Figure 5: *Stage visualizations*

Furthermore, Kserve at the last step of the created pipeline, spaws an inference service known as Model Server. Central dashboard provides easy access to these procedures as shown bellow.

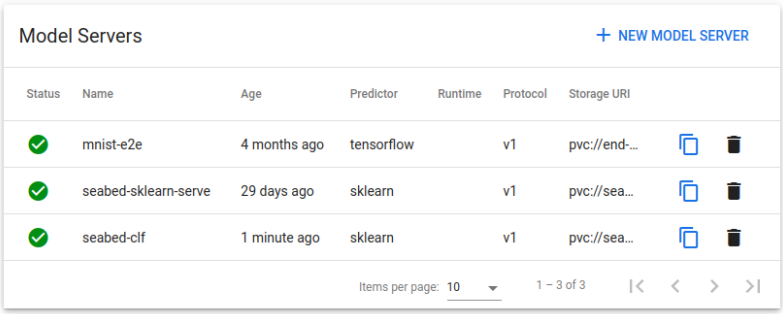


Figure 6: *Model servers*

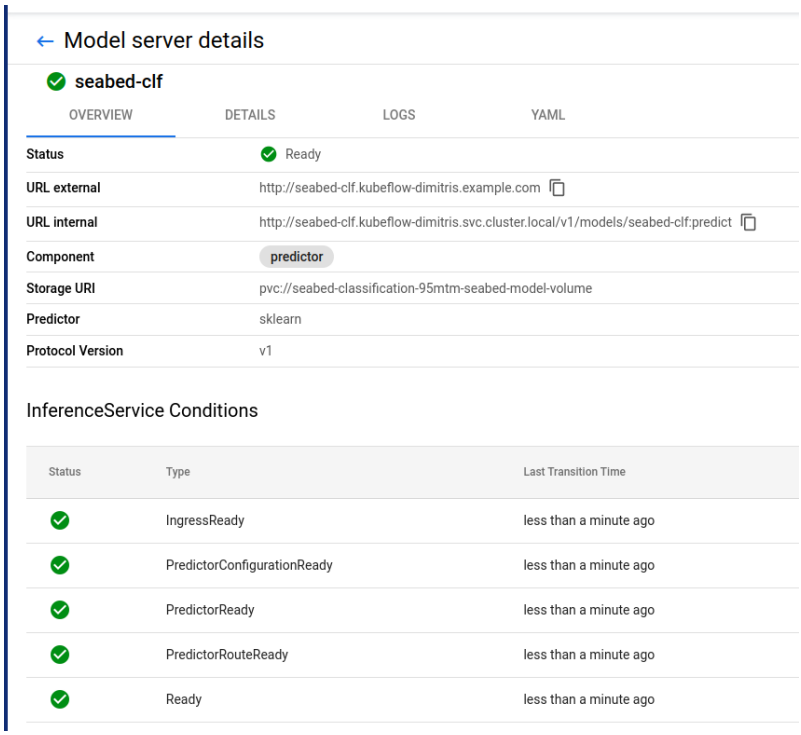
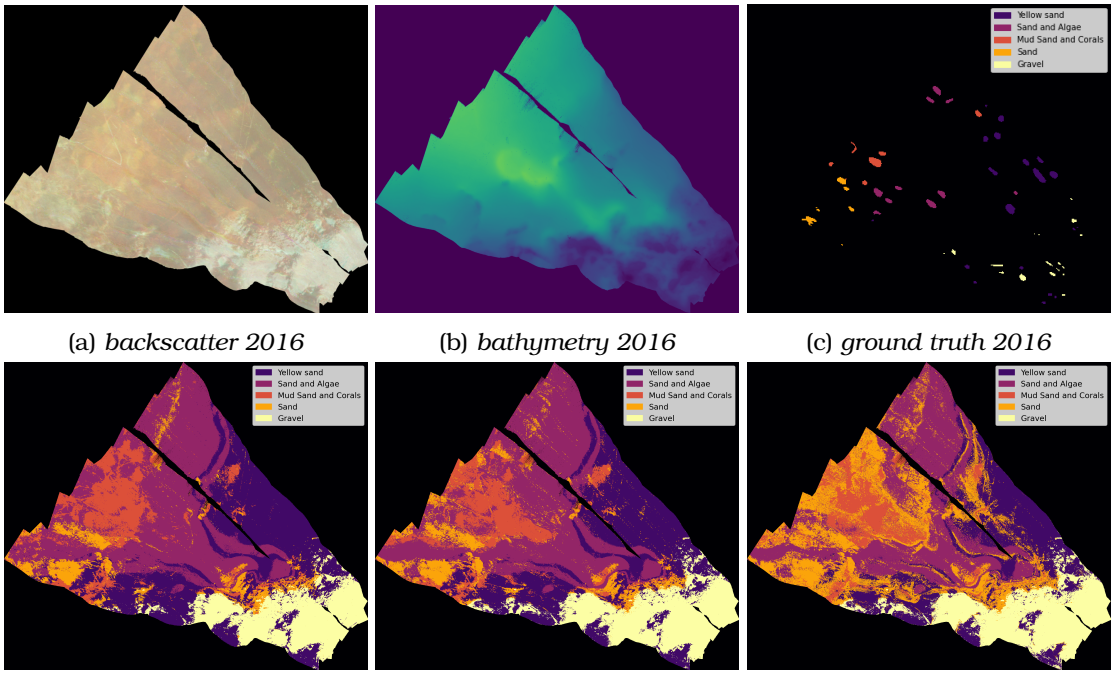


Figure 7: *Model server details*

In the matter of continuous training, the above pipeline is sufficient. In a more generic manner, a stable pipeline that is implemented and automated in production handling live data, in some stage collects updated data. The training code does not change, as well as the pipeline does not change. Only the data change. Following up, in this thesis retraining, or better rerunning the pipeline, will be simulated as if it happened at each year of our data 2016, 2017 and 2018. When training the model at year 2016 only data of 2016 are known. When training at year 2017, data of 2016 and 2017 are known and data of 2018 are unknown. When trying at year 2018, data of all years are known. Below is listed the performance of the models created by each pipeline run.

	accuracy	kappa	prec_1	prec_2	prec_3	prec_4	prec_5	recall_1	recall_2	recall_3	recall_4	recall_5
model '16 at 2016	0.99983	0.99978	0.99988	0.99962	0.99984	0.99994	1.0	1.0	1.0	1.0	0.99893	0.9999
model '16 at 2017	0.60898	0.49144	0.65907	0.24476	0.55603	0.87916	0.99995	0.99949	0.255690	0.53317	0.09311	0.96927
model '16 at 2018	0.72483	0.58081	0.69983	0.71013	0.04479	0.93878	0.99195	0.94077	0.86175	0.00108	0.00106	0.75573
model '17 at 2016	0.99978	0.99971	0.99994	0.99939	0.99988	0.99978	0.99999	0.99998	0.9999	0.99994	0.9985	0.99995
model '17 at 2017	0.99968	0.9996	0.99907	0.99987	0.99987	0.99997	1.0	0.99996	0.99866	0.99993	0.99976	0.9999
model '17 at 2018	0.6699	0.48917	0.61697	0.71738	0.04586	0.99714	0.98888	0.99937	0.62684	0.00148	0.02406	0.69297
model '18 at 2016	0.99969	0.99959	0.9999	0.99935	0.9999	0.99922	0.99997	0.99982	0.99979	0.99994	0.99845	0.99999
model '18 at 2017	0.99967	0.99959	0.99911	0.99986	0.9998	0.99997	1.0	0.99993	0.9987	0.99997	0.99965	0.99989
model '18 at 2018	0.99686	0.99559	0.99634	0.99729	0.99662	0.99425	0.99972	1.0	0.99983	0.99783	0.97006	0.9947

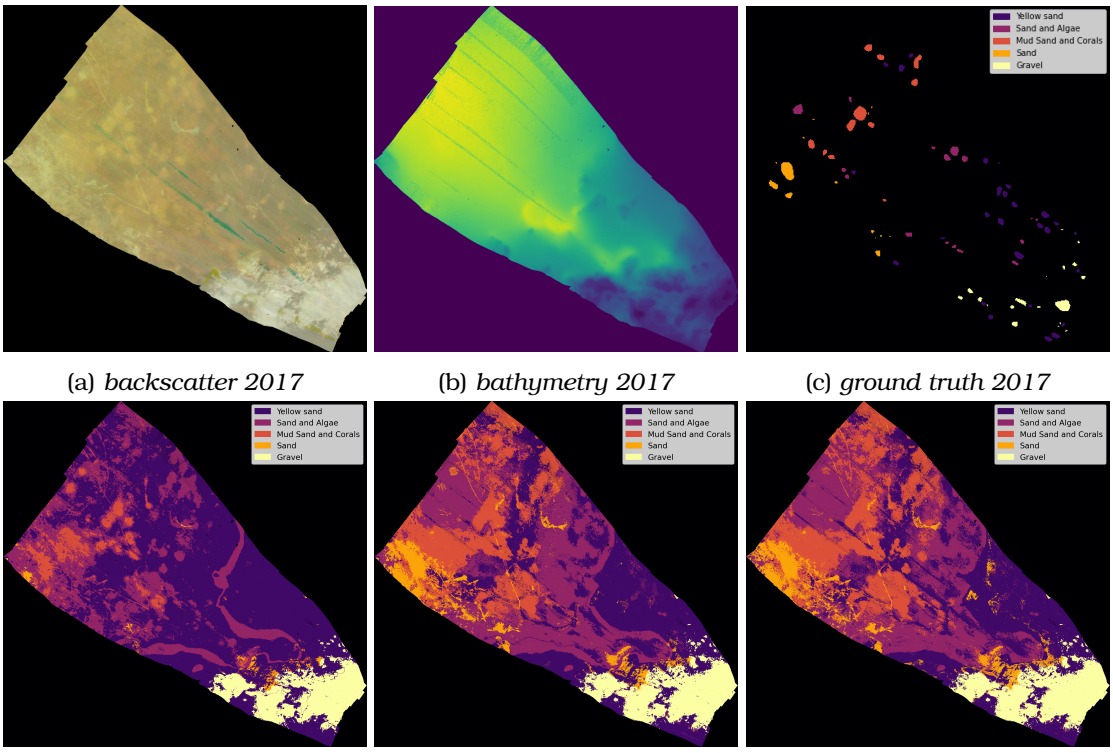
Table 1: *Performance of 2016, 2017, 2018 models*



(a) *backscatter 2016* (b) *bathymetry 2016* (c) *ground truth 2016*

(d) *model '16* (e) *model '17* (f) *model '18*

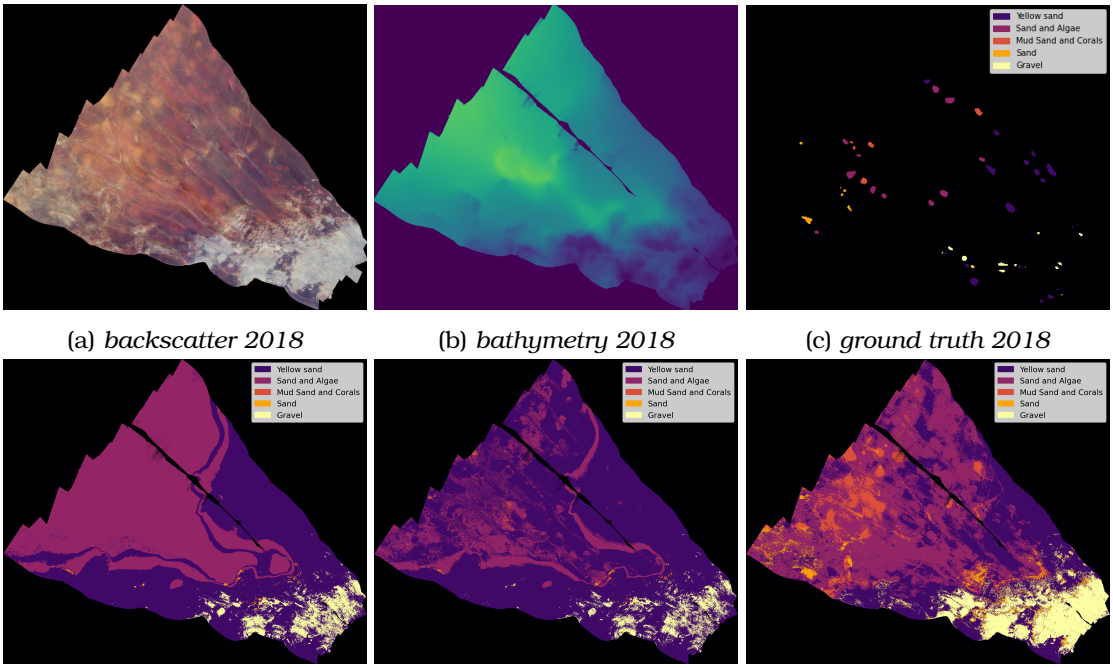
Figure 8: *Bedford 2016 dataset classification*



(a) *backscatter 2017* (b) *bathymetry 2017* (c) *ground truth 2017*

(d) *model '16* (e) *model '17* (f) *model '18*

Figure 9: *Bedford 2017 dataset classification*



(a) *backscatter 2018* (b) *bathymetry 2018* (c) *ground truth 2018*

(d) *model '16* (e) *model '17* (f) *model '18*

Figure 10: *Bedford 2018 dataset classification*

The overall assumption is that Kubeflow is a very powerful platform trying to simplify MLOps and mostly succeeds it. It manages to make an agnostic Kubernetes environment for data scientists, who can by only describing some steps in a pipeline create complex machine learning operations. It has integrated with most of the tools used by data scientists, from developing a model to deploying and monitoring in production. It contributes a lot in the decoupling and collaboration of data science teams and DevOps engineers. Kubeflow benefits of Kubernetes features of scaling and orchestration.

Some aspects of the implementation of this thesis could be further explored. User management with RBAC should be enforced in production, as well as resource management based on user or group. In this case, user isolation was enforced, but had not limitations in cluster. When tackling complex ML systems with Kubeflow and Kubernetes, infrastructure should be monitored regarding resource usage. Grafana and Prometheus, would be a great fit, as they are the most commonly used tools. Further usage and exploration should be done regarding Kubeflow training operators, tf-operator, pytorch-operator and mpi-operator (Horovod [9]), that provide out of the box distributed training. An aspect of great importance is model and data monitoring. They should be implemented, exploring Katib options of Alibi Detector, AIF Bias Detector and ART Adversial Detector, as well as ModelMesh. Rollback should also be implemented, as well as gradually replacing a model with Canary rollout. Finally, an interesting tool, when tackling neural networks and deep learning algorithms, is Katib's NAS [10], which performs Neural Architecture Search.

στους γονείς μου

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή κ. Κωνσταντίνο Καράντζαλο, που μου έδωσε την ευκαιρία να συνεργαστώ μαζί του σε ένα πολύ ενδιαφέρον αντικείμενο. Με το αντικείμενο αυτό ασχολούμαι επαγγελματικά αρκετά χρόνια και η παρούσα εργασία με βοήθησε να εξελιχθώ σε αρκετούς τομείς. Επίσης, ευχαριστώ ιδιαίτερα τον μεταδιδακτορικό ερευνητή κ. Βαλσάμη Ντούσκο για την καθοδήγηση και βοήθεια που μου προσέφερε κατά την διάρκεια εκπόνησης αυτής της διατριβής. Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου, για όλα όσα μου προσέφεραν κατά τη διάρκεια των μεταπτυχιακών σπουδών μου.

Αθήνα, Δεκέμβριος 2022

Δημήτριος Ν. Τριανταφύλλου

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

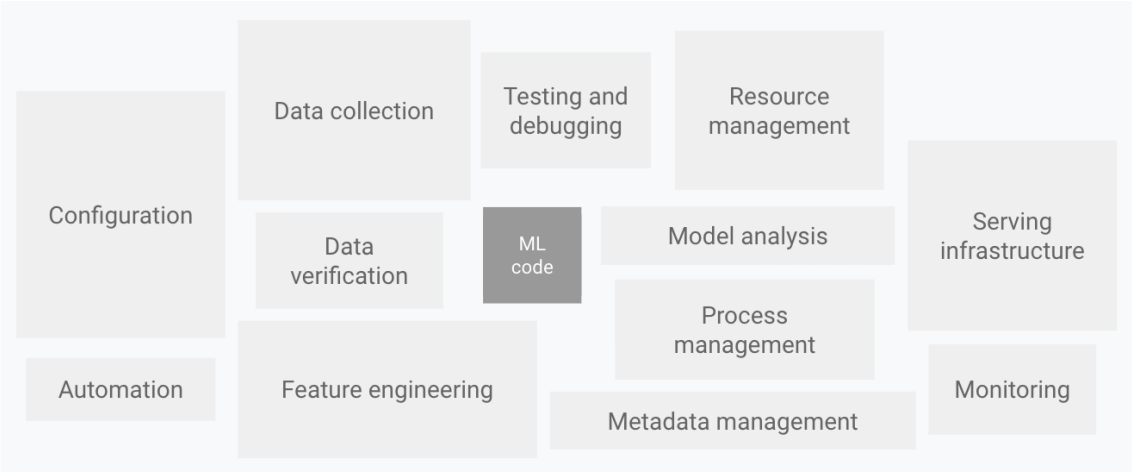
Σε αυτή την διπλωματική θα γίνει μια προσέγγιση της υλοποίησης αυτοματοποιημένων διαδικασιών μηχανικής μάθησης (MLOps) σε συστήματα που βρίσκονται στην παραγωγή. Σήμερα, η μηχανική μάθηση χρησιμοποιείται σε ένα ευρύ φάσμα εφαρμογών και αποτελεί το επίκεντρο του ενδιαφέροντος πολλών επιχειρήσεων. Το ενδιαφέρον έγκειται τόσο στην ανάπτυξη προϊόντων για άμεση χρήση μηχανικής μάθησης για πρόβλεψη και ταξινόμηση, όσο και στην αναγνώριση των τάσεων στη συμπεριφορά πελατών και επιχειρήσεων για υποστήριξη υπαρχόντων προϊόντων ή και την ανάπτυξη νέων. Κατά αυτόν το τρόπο χρειάζονται κάποιες διαδικασίες που θα αναλαμβάνουν να αναπτύσσουν συνεχώς τα μοντέλα μηχανικής μάθησης, να ελέγχουν την αποτελεσματικότητα τους, να τα εξελίσσουν και να τα ενσωματώνουν σε νέα ή υπάρχοντα προϊόντα.

1.1 Μηχανική μάθηση

1.1.1 Ανάπτυξη συστημάτων

Για την ανάπτυξη ενός συστήματος μηχανικής μάθησης, οι επιστήμονες δεδομένων καλούνται να εφαρμόσουν και να εκπαιδεύσουν ένα μοντέλο μηχανικής μάθησης με προγνωστική απόδοση σε ένα σύνολο δεδομένων, λαμβάνοντας υπόψη σχετικά δεδομένα εκπαίδευσης για την περίπτωση χρήσης τους. Ωστόσο, η πραγματική πρόκληση δεν είναι η δημιουργία ενός μοντέλου, η πρόκληση είναι η οικοδόμηση ενός ολοκληρωμένου συστήματος μηχανικής μάθησης και η συνεχής λειτουργία του στην παραγωγή.

Όπως φαίνεται στο παρακάτω διάγραμμα, μόνο ένα μικρό κλάσμα ενός πραγματικού συστήματος μηχανικής μάθησης αποτελείται από τον κώδικα. Τα απαιτούμενα περιβάλλοντα στοιχεία είναι τεράστια και πολύπλοκα.



Σχήμα 1.1: *Στοιχεία για συστήματα μηχανικής μάθησης [2]*

Σε κάθε σύστημα μηχανικής μάθησης, αφού προσδιοριστεί η επιχειρηματική χρήση και καθοριστούν τα κριτήρια επιτυχίας, η διαδικασία παράδοσης ενός μοντέλου μηχανικής μάθησης στην παραγωγή περιλαμβάνει τα ακόλουθα βήματα.

Αυτά τα βήματα μπορούν να ολοκληρωθούν χειροκίνητα ή μπορούν να ολοκληρωθούν με αυτόματη σωλήνωση.

- Εξαγωγή δεδομένων: Επιλογή και ενσωμάτωση των σχετικών δεδομένων από διάφορες πηγές.
- Ανάλυση δεδομένων: Εκτέλεση διερευνητικής ανάλυσης δεδομένων (EDA) για κατανόηση των διαθέσιμων δεδομένων για τη δημιουργία του μοντέλου. Αυτή η διαδικασία οδηγεί στα εξής:
  - Κατανόηση του σχήματος δεδομένων και των χαρακτηριστικών που αναμένονται από το μοντέλο.
  - Προσδιορισμός της προετοιμασίας δεδομένων και της μηχανικής χαρακτηριστικών που απαιτούνται για το μοντέλο.
- Προετοιμασία δεδομένων: Αυτή η προετοιμασία περιλαμβάνει καθαρισμό δεδομένων, όπου χωρίζονται τα δεδομένα σε σετ εκπαίδευσης, επικύρωσης και δοκιμών. Περιλαμβάνει επίσης μετασχηματισμό δεδομένων και εξαγωγή χαρακτηριστικών στο μοντέλο. Η έξοδος αυτού του βήματος είναι τα διαχωρισμένα σετ δεδομένων στην προετοιμασμένη μορφή.
- Εκπαίδευση μοντέλων: Ο επιστήμονας δεδομένων εφαρμόζει διαφορετικούς αλγόριθμους με τα προετοιμασμένα δεδομένα για να εκπαιδεύσει διάφορα μοντέλα μηχανικής μάθησης. Επιπλέον, σε αυτούς του αλγορίθμους αναζητά τις βέτλιστες υπερπαραμέτρους για να βρει το μοντέλο με την καλύτερη απόδοση. Η έξοδος αυτού του βήματος είναι ένα εκπαιδευμένο μοντέλο.
- Αξιολόγηση μοντέλου: Το μοντέλο αξιολογείται σε ένα σετ δοκιμών για την αξιολόγηση της ποιότητας του μοντέλου. Το αποτέλεσμα αυτού του βήματος είναι ένα σύνολο μετρήσεων για την αξιολόγηση της ποιότητας του μοντέλου.

- Επικύρωση μοντέλου: Το μοντέλο επιβεβαιώνεται ότι είναι επαρκές για προώθηση στην παραγωγή, ότι η προγνωστική του απόδοση είναι καλύτερη από ένα συγκεκριμένο κατώφλι.

- Προώθηση μοντέλου στην παραγωγή: Το επικυρωμένο μοντέλο γίνεται διαθέσιμο στην παραγωγή για την παροχή προβλέψεων. Αυτή η Προώθηση μπορεί να είναι ένα από τα ακόλουθα:

  - Microservices με REST API για την παροχή διαδικτυακών προβλέψεων.
  - Ένα ενσωματωμένο μοντέλο σε μια συσκευή edge ή κινητή συσκευή.
  - Μέρος ενός συστήματος πρόβλεψης σε παρτίδες.

- Παρακολούθηση μοντέλου: Η προγνωστική απόδοση του μοντέλου παρακολουθείται για πιθανή επίκληση μιας νέας επανάληψης στη διαδικασία μηχανικής μάθησης.

1.1.2 Διαδικασίες μηχανικής μάθησης (MLOps)

Οι διαδικασίες μηχανικής μάθησης, ή MLOps για συντομία, είναι μια βασική πτυχή της μηχανικής μηχανικής μάθησης που επικεντρώνεται στην απλοποίηση και στην επιτάχυνση της διαδικασίας παράδοσης μοντέλων στην παραγωγή και στη συντήρηση και παρακολούθηση τους. Περιλαμβάνουν τη συνεργασία μεταξύ διαφορετικών ομάδων, συμπεριλαμβανομένων επιστημόνων δεδομένων, μηχανικών DevOps, ειδικών πληροφορικής και άλλων. Οι διαδικασίες αυτές στοχεύουν στην ενοποίηση της ανάπτυξης συστήματος και της λειτουργίας ενός συστήματος μηχανικής μάθησης.

Τα MLOps [1] περιλαμβάνουν τις ακόλουθες πρακτικές:

- Continuous Integration (CI): Όπου επεκτείνεται ο έλεγχος και η επικύρωση του κώδικα και των στοιχείων προσθέτοντας δεδομένα και μοντέλα δοκιμών και επικύρωσης.

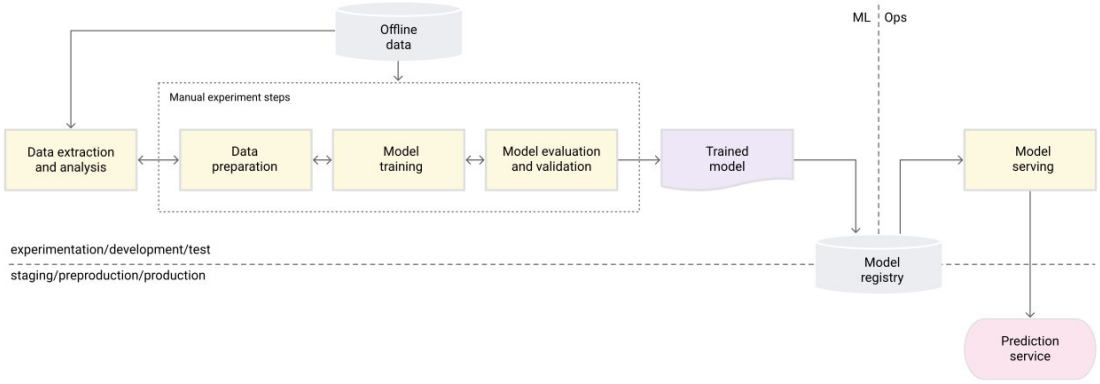
- Continuous Delivery (CD): Όπου αφορά την παράδοση μιας σωλήνωσης εκπαίδευσης ενός συστήματος μηχανικής μάθησης που κάνει διαθέσιμη αυτόματα μια άλλη υπηρεσία πρόβλεψης μοντέλου.

- Continuous Training (CT): Συναντάται μόνο σε MLOps και όχι σε DevOps, όπου επανεκπαιδεύονται αυτόματα τα μοντέλα για εκ νέου ανάπτυξη.

- Continuous Monitoring (CM): Αφορά την παρακολούθηση των δεδομένων παραγωγής και των μετρήσεων απόδοσης μοντέλων, οι οποίες συνδέονται με τις επιχειρηματικές μετρήσεις.

Μη αυτόματη διαδικασία

Πολλές ομάδες έχουν επιστήμονες δεδομένων και ερευνητές που μπορούν να κατασκευάσουν μοντέλα τελευταίας τεχνολογίας, αλλά η διαδικασία ανάπτυξης και προώθησης στην παραγωγή των μοντέλων είναι εντελώς χειροκίνητη.



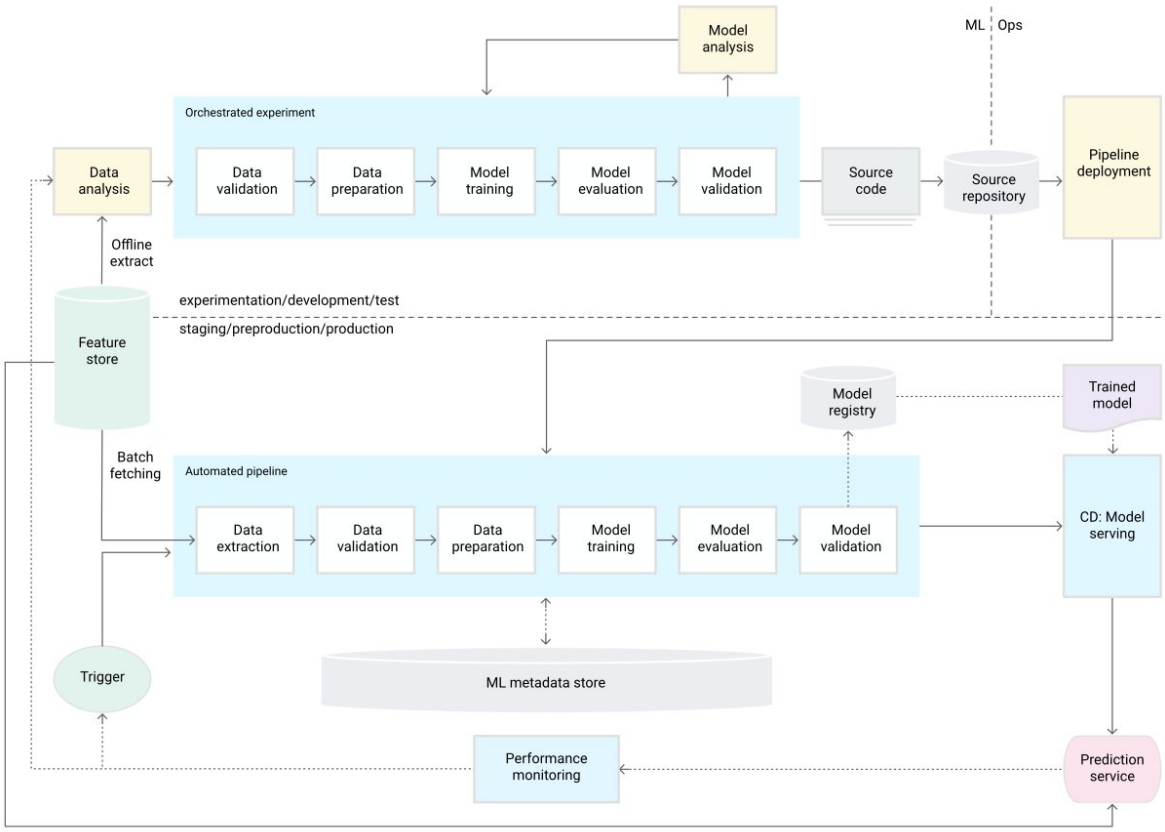
Σχήμα 1.2: *Μη αυτόματη διαδικασία MLOps*

Η διαδικασία αυτή είναι διαδραστική και βασισμένη σε σενάρια. Κάθε βήμα είναι χειροκίνητο, συμπεριλαμβανομένης της ανάλυσης δεδομένων, της προετοιμασίας δεδομένων, της εκπαίδευσης μοντέλων και της επικύρωσης. Απαιτεί χειροκίνητη εκτέλεση και χειροκίνητη μετάβαση από το ένα βήμα στο άλλο. Αυτή η διαδικασία συνήθως καθοδηγείται από πειραματικό κώδικα που γράφεται και εκτελείται από τον επιστήμονα δεδομένων διαδραστικά, μέχρι να παραχθεί ένα λειτουργικό μοντέλο. Η διαδικασία διαχωρίζει τους επιστήμονες δεδομένων που δημιουργούν το μοντέλο και τους μηχανικούς που κάνουν διαθέσιμο το μοντέλο ως ένα service πρόβλεψης. Οι επιστήμονες δεδομένων παραδίδουν ένα εκπαιδευμένο μοντέλο στην ομάδα μηχανικών για να το ενσωματώσουν στην υποδομή τους. Αυτή η μεταβίβαση μπορεί να περιλαμβάνει την τοποθέτηση του εκπαιδευμένου μοντέλου σε μια θέση αποθήκευσης ή σε ένα αποθετήριο κώδικα ή τη μεταφόρτωσή του σε ένα μητρώο μοντέλων (model registry).

Κατά αυτόν τον τρόπο, αυτή η διαδικασία δεν έχει συχνά νέες ανανεώσεις του μοντέλου, καθώς και δεν παρακολουθείται ενεργά η απόδοση του μοντέλου.

Αυτοματοποιημένη σωλήνωση

Σε αυτή την διαδικασία επιτυγχάνεται συνεχής εκπαίδευση του μοντέλου με την αυτοματοποίηση της σωλήνωσης. Αυτό επιτρέπει την συνεχή παράδοση της υπηρεσίας πρόβλεψης μοντέλου (Continuous Delivery - CD). Για να αυτοματοποιηθεί η διαδικασία χρήσης νέων δεδομένων για την επανεκπαίδευση μοντέλων στην παραγωγή (Continuous Training - CT), θα πρέπει να εισάγονται αυτοματοποιημένα δεδομένα και κάποια βήματα επικύρωσης μοντέλων στη σωλήνωση. Επίσης πρέπει να υπάρξουν κάποιοι ενεργοποιητές για την σωλήνωση και να γίνει διαχείριση των μεταδεδομένων που προκύπτουν σε κάθε στάδιο.



Σχήμα 1.3: *Αυτοματοποιημένη σωλήνωση MLOps (CD-CT)*

Οι ενεργοποιητές της σωλήνωσης μπορούν να αυτοματοποιηθούν ανάλογα με τις ανάγκες. Μπορούν να ενεργοποιούν την σωλήνωση κατά απαίτηση, είτε ανά τακτά χρονικά διαστήματα, είτε όποτε υπάρχουν διαθέσιμα νέα δεδομένα, είτε ακόμα όταν υπάρχει αισθητή υποβάθμιση της απόδοσης, προκαλώντας την επανεκπαίδευση του μοντέλου.

Πολύ σημαντική είναι και η διαχείριση των μεταδεδομένων. Οι πληροφορίες σχετικά με κάθε εκτέλεση της σωλήνωσης όπου καταγράφονται τα συγκεκριμένα δεδομένα, κώδικες που χρησιμοποιήθηκαν, καθώς και τα artifacts που δημιουργήθηκαν, για την αναπαραγωγιμότητα και τις συγκρίσεις. Αυτό βοηθά στην διόρθωση σφαλμάτων και ανωμαλιών.

Επίσης κομβικό ρόλο στην διαδικασία έχει το "versioning". Ο στόχος του είναι να αντιμετωπίζονται τα σενάρια εκπαίδευσης, τα μοντέλα και τα σύνολα δεδομένων για εκπαίδευση μοντέλων όπως στις διαδικασίες DevOps, παρακολουθώντας τα μοντέλα και τα σύνολα δεδομένων με συστήματα ελέγχου εκδόσεων. Με αυτό τον τρόπο για κάθε εκτέλεση στην διαδικασία εκπαίδευσης του μοντέλου, θα υπάρχει καταγραφή/έκδοση τόσο του κώδικα, όσο και των δεδομένων που χρησιμοποιήθηκαν.

Το στάδιο του CD είναι η διαδικασία ανάπτυξης του κώδικα για την αντικατάσταση των προηγούμενων εκδόσεων. Αυτό πραγματοποιείται μέσω μιας στρατηγικής, συνήθως μία εκ των δύο, της "blue-green" ή της "canary". Ουσιαστικά είναι μεθοδολογίες για τη σταδιακή κυκλοφορία εκδόσεων. Η "blue-green" προσέγγιση απαιτεί δύο περιβάλλοντα παραγωγής, όσο το δυνατόν πανομοιότυπα. Ανά πάσα στιγμή ένα από αυτά, το μπλε για παράδειγμα, είναι εν λειτουργία. Καθώς ετοιμάζεται μια νέα έκδοση του μοντέλου, το τελικό στάδιο

της δοκιμής γίνεται στο πράσινο περιβάλλον. Μόλις το μοντέλο λειτουργεί στο πράσινο περιβάλλον, δρομολογούνται όλα τα εισερχόμενα αιτήματα να πηγαίνουν σε αυτό, οπότε και το μπλε είναι πλέον αδρανές. Εάν κάτι δεν λειτουργήσει σωστά, όλα δρομολογούνται ξανά στο μπλε περιβάλλον. Παρόμοια, στην "canary" στρατηγική, η αλλαγή αναπτύσσεται σε ένα μικρό υποσύνολο περιπτώσεων, εξυπηρετώντας ένα μικρό ποσοστό της κίνησης στην παραγωγή. Μόλις διαπιστωθεί ότι το καινούριο μοντέλο λειτουργεί καλά τότε δρομολογείται να εξυπηρετεί όλη την κίνηση.

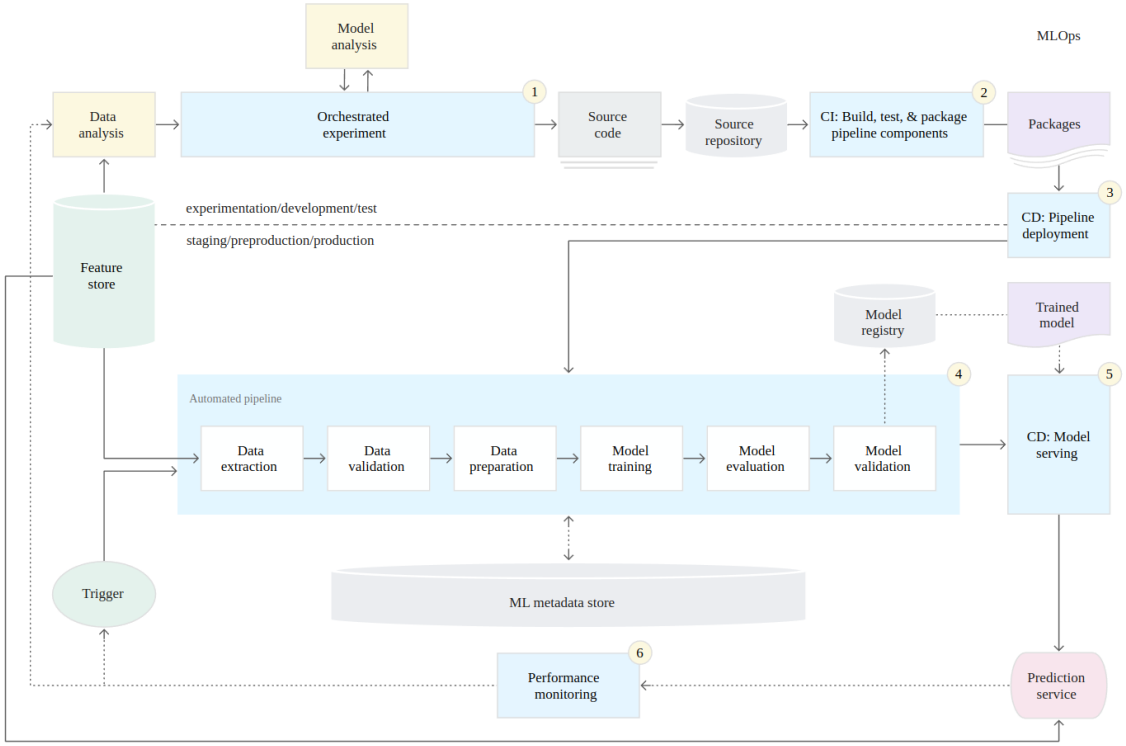
Μια επίσης στρατηγική είναι το "shadow evaluation" (Dark launch). Η στρατηγική αυτή είναι μια πολύ διαισθητική και ασφαλής. Κατά την στρατηγική αυτή, το νέο μοντέλο προστίθεται στο μητρώο ως υποψήφιο μοντέλο. Οποιοδήποτε νέο αίτημα πρόβλεψης εκτελείται τόσο από τις εκδόσεις του μοντέλου που χρησιμοποιείται επί του παρόντος στην παραγωγή, όσο και από τη νέα υποψήφια έκδοση που είναι σε shadow evaluation. Έτσι η απόδοση της νέας έκδοσης αξιολογείται πριν γίνει διαθέσιμη στην παραγωγή. Όταν θεωρηθεί ότι το υποψήφιο μοντέλο είναι επαρκές, αντικαθιστά το υπάρχων μοντέλο στην παραγωγή.

Η διαδικασία CD θα πρέπει να είναι σταδιακή για την επικύρωση της ορθότητας και της ποιότητας του μοντέλου σε ¨ζωντανά¨ δεδομένα, χρησιμοποιώντας συστήματα παραγωγής, προτού προωθηθεί στην παραγωγή και ξεκινήσει να λαμβάνει πραγματικές αυτοματοποιημένες αποφάσεις. Αυτή η αξιολόγηση ονομάζεται συχνά ¨διαδικτυακή αξιολόγηση¨, σε αντίθεση με τις δοκιμές που πραγματοποιήθηκαν κατά τη φάση CI που βασίζονται σε ιστορικά σύνολα δεδομένων και μπορεί να θεωρηθούν ¨αξιολόγηση εκτός σύνδεσης¨.

Αυτοματοποιημένη σωλήνωση με CI/CD

Για γρήγορη και αξιόπιστη ενημέρωση των σωληνώσεων στην παραγωγή, χρειάζεται ένα αυτοματοποιημένο σύστημα CI/CD. Αυτό το σύστημα επιτρέπει στους επιστήμονες δεδομένων να εξερευνούν γρήγορα νέες ιδέες σχετικά με τη μηχανική χαρακτηριστικών, την αρχιτεκτονική μοντέλων και τις υπερπαραμέτρους. Μπορούν να εφαρμόσουν αυτές τις ιδέες και να δημιουργήσουν αυτόματα, να δοκιμάσουν και να αναπτύξουν τα νέα στοιχεία της σωλήνωσης στο περιβάλλον στόχο.

Το παρακάτω διάγραμμα δείχνει την υλοποίηση της σωλήνωσης χρησιμοποιώντας CI/CD, το οποίο έχει τα χαρακτηριστικά της ρύθμισης των αυτοματοποιημένων σωληνώσεων που αναφέρθηκαν παραπάνω συν τις αυτοματοποιημένες ρουτίνες CI/CD.

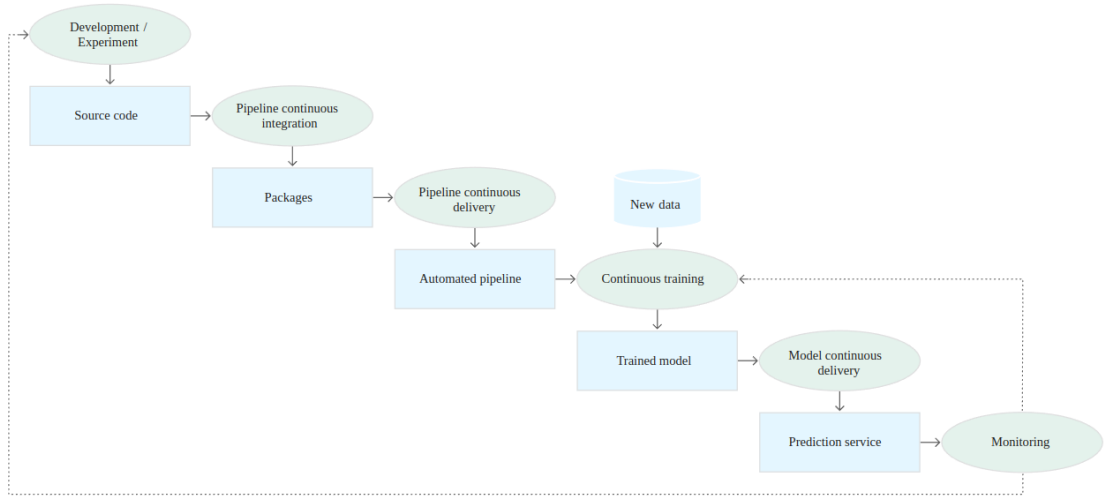


Σχήμα 1.4: *CI/CD σε αυτοματοποιημένη σωλήνωση*

Όπως φαίνεται στην παραπάνω εικόνα, για την πραγματοποίηση αυτοματοποιημένης σωλήνωσης με CI/CD χρειάζονται τα παρακάτω στοιχεία:

- Σύστημα ελέγχου εκδόσεων
- Services για δοκιμές και κατασκευή των πακέτων
- Services για εγκατάσταση
- Μητρώο για την αποθήκευση των μοντέλων
- Χώρος για την αποθήκευση των χαρακτηριστικών
- Χώρος για την αποθήκευση των μεταδεδομένων
- Ενορχηστρωτής των σωληνώσεων

Πιο συγκεκριμένα, το παρακάτω διάγραμμα δείχνει τα στάδια της αυτοματοποιημένης σωλήνωσης με CI/CD:



Σχήμα 1.5: *Τα στάδια CI/CD της αυτοματοποιημένη σωλήνωση*

Η σωλήνωση αποτελείται από τα ακόλουθα στάδια:

- Ανάπτυξη και πειραματισμός: Δοκιμάζονται επαναληπτικά νέοι αλγόριθμοι μηχανικής μάθησης και νέα μοντέλα κατά την ενορχήστρωση των βημάτων του πειράματος. Η έξοδος αυτού του σταδίου είναι ο πηγαίος κώδικας των βημάτων σωλήνωσης που στη συνέχεια προωθούνται σε ένα αποθετήριο.

- Pipeline CI: Γίνεται το build του κώδικα και εκτελούνται κάποιες δοκιμές. Οι έξοδοι αυτού του σταδίου είναι στοιχεία σωλήνωσης (πακέτα, εκτελέσιμα και artifacts) που θα αναπτυχθούν σε μεταγενέστερο στάδιο.

- Pipeline CD: Αναπτύσσονται τα artifacts που παράγονται από το στάδιο CI στο περιβάλλον στόχο. Η έξοδος αυτού του σταδίου είναι μία σωλήνωση με τη νέα εφαρμογή του μοντέλου.

- Αυτοματοποιημένη ενεργοποίηση: Η σωλήνωση εκτελείται αυτόματα στην παραγωγή με βάση ένα χρονοδιάγραμμα ή ως απόκριση σε έναν ενεργοποιητή. Η έξοδος αυτού του σταδίου είναι ένα εκπαιδευμένο μοντέλο που προωθείται στο μητρώο μοντέλων.

- Μοντέλο CD: Το εκπαιδευμένο μοντέλο αναπτύσσεται ως service προβλέψεων.

- Παρακολούθηση: Συλλέγονται στατιστικά στοιχεία για την απόδοση του μοντέλου με βάση ζωντανά δεδομένα. Η έξοδος αυτού του σταδίου είναι ένα έναυσμα για την εκτέλεση της σωλήνωσης ή για την εκτέλεση ενός νέου κύκλου πειράματος.

1.2 Αντικείμενο της διπλωματικής

Σε αυτή την διπλωματική προσεγγίζεται η υλοποίηση υποδομών και διαδικασιών μηχανικής μάθησης MLOps στην παραγωγή. Η προσέγγιση αυτή εφαρμόζεται σε ένα ζήτημα μηχανικής μάθησης για σημασιολογική κατάτμηση σε γεωχωρικά δεδομένα. Συγκεκριμένα κατηγοριοποιούνται στοιχεία από υποθαλάσσιες εικόνες, για την αναγνώριση του είδους του εδάφους. Υλοποιείται μια λύση με Kubernetes και την πλατφόρμα Kubeflow. Στις διαδικασίες μηχανικής μάθησης, θα περιγραφούν κάποιες τεχνικές που αποσκοπούν στην αυτοματοποίηση των συστημάτων για συνεχή εκπαίδευση των μοντέλων και συνεχή εξυπηρέτηση στην παραγωγή.

1.3 Δομή της διπλωματικής

Η παρούσα εργασία οργανώνεται ως εξής:

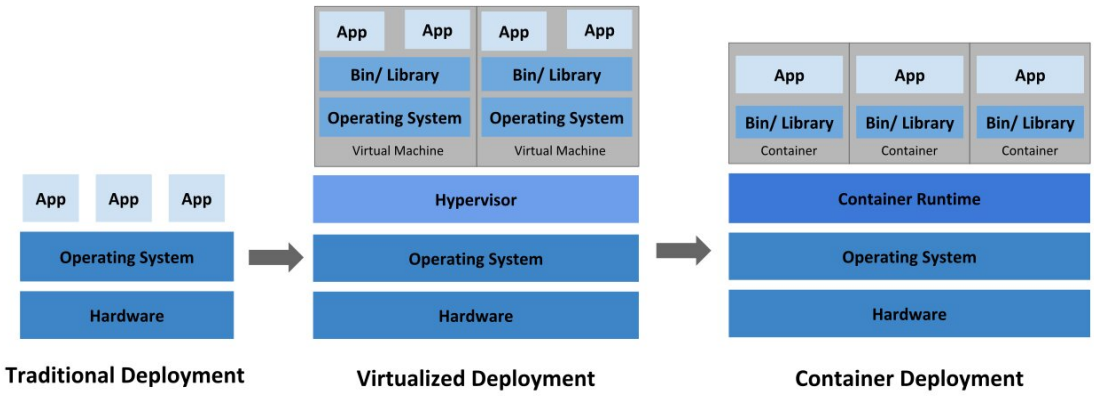
- **Στο κεφάλαιο 2** περιγράφεται το θεωρητικό υπόβαθρο αναφορικά με τα εργαλεία Kubernetes και Kubeflow.
- **Στο κεφάλαιο 3** περιγράφονται οι επιλογές που γίνονται κατά την σχεδίαση ενός συστήματος μηχανικής μάθησης στην παραγωγή. Επίσης παρουσιάζονται οι υλοποιήσεις σε μικρότερη κλίμακα, οι οποίες έγιναν στα πλαίσια της εργασίας.
- **Στο κεφάλαιο 4** προσεγγίζεται και επιλύεται ένα ζήτημα μηχανικής μάθησης με διαδικασίες MLOps με στόχο την αυτοματοποίηση στην παραγωγή. Αναλύονται όλα τα στάδια της διαδικασίας, συλλογής δεδομένων, εκπαίδευσης του μοντέλου και προώθηση του μοντέλου στην παραγωγή. Καθώς επίσης, αναλύεται η διαδικασία επανεκπαίδευσης και αντικατάστασης μοντέλων στην παραγωγή.
- **Στο κεφάλαιο 5** γίνεται μία σύνοψη σχετικά με τα οφέλη της χρήσης των MLOps, όπως υλοποιήθηκαν στην παρούσα εργασία και τις διαφορές που μπορεί να έχει από ένα σύστημα στην παραγωγή. Επίσης δίνονται σύντομα, κάποιες πιθανές μελλοντικές επεκτάσεις και εφαρμογές των MLOps διαδικασιών που υλοποιήθηκαν.

Κεφάλαιο 2

Υποδομή και εργαλεία

2.1 Kubernetes

Το Kubernetes [4] είναι μια επεκτάσιμη πλατφόρμα ανοιχτού κώδικα για τη διαχείριση φόρτου εργασίας και services με χρήση containers, που διευκολύνει τόσο το "Infrustructure as Code" (IaC) όσο και τον αυτοματισμό. Έχει ένα μεγάλο, ταχέως αναπτυσσόμενο οικοσύστημα. Οι υπηρεσίες, η υποστήριξη και τα εργαλεία Kubernetes είναι ευρέως διαθέσιμα. Η Google δημιούργησε το πρότζεκτ Kubernetes ανοιχτού κώδικα το 2014. Το Kubernetes εστιάζει στη διαχείριση φόρτου εργασίας παραγωγής σε μεγάλη κλίμακα με τις καλύτερες ιδέες και πρακτικές από την κοινότητα.



Σχήμα 2.1: *Ανάπτυξη εφαρμογών στην παραγωγή*

Όπως φαίνεται στο σχήμα 2.1, παλαιότερα, στην ¨παραδοσιακή εποχή¨ όπως αναφέρεται, οι οργανισμοί εκτελούσαν εφαρμογές απευθείας σε φυσικά μηχανήματα (servers). Δεν υπήρχε τρόπος να καθοριστούν τα όρια πόρων για εφαρμογές σε ένα φυσικό μηχάνημα και αυτό προκάλεσε προβλήματα κατανομής πόρων. Για παράδειγμα, εάν πολλές εφαρμογές εκτελούνται σε ένα φυσικό μηχάνημα, μπορεί να υπάρξουν περιπτώσεις όπου μια εφαρμογή θα καταλάμβανε τους περισσότερους πόρους και ως εκ τούτου, οι άλλες εφαρμογές θα είχαν χαμηλή απόδοση. Μια λύση για αυτό θα ήταν η εκτέλεση κάθε εφαρμογής σε διαφορετικό φυσικό μηχάνημα. Αλλά αυτή η λύση δεν μπορεί να έχει κλιμάκωση, καθώς οι πόροι μπορεί να χρησιμοποιούνταν ελάχιστα και ήταν ακριβό για τους οργανισμούς να διατηρούν πολλά φυσικά μηχανήματα.

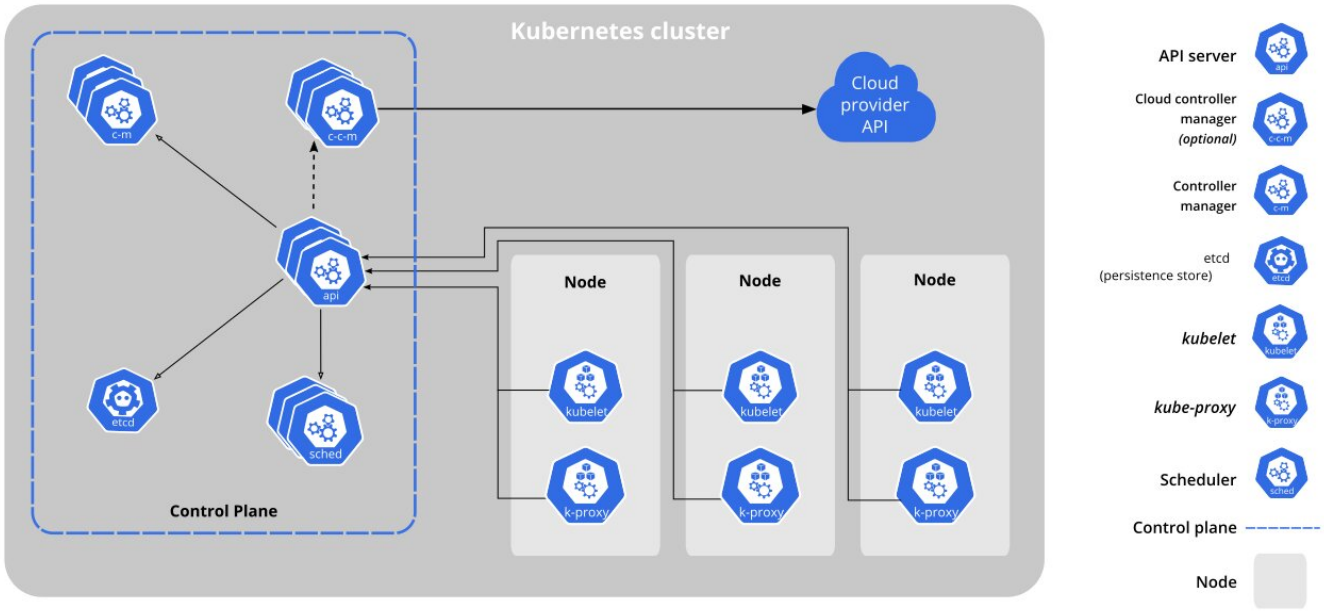
Σαν εξέλιξη, έρχεται μια ¨εποχή¨ εικονικής ανάπτυξης. Η εικονικοποίηση (virtualization) επιτρέπει να υπάρχουν πολλαπλές εικονικές μηχανές VM σε ένα φυσικό μηχάνημα. Επίσης επιτρέπει την απομόνωση των εφαρμογών μεταξύ των VM και παρέχει ένα επίπεδο ασφάλειας καθώς οι πληροφορίες μιας εφαρμογής δεν μπορούν να είναι ελεύθερα προσβάσιμες από άλλη εφαρμογή. Κατά αυτόν τον τρόπο γίνεται καλύτερη χρήση των πόρων σε ένα φυσικό μηχάνημα και διευκολύνεται η επεκτασιμότητα. Μία εφαρμογή μπορεί να προστεθεί ή να ενημερωθεί εύκολα. Με την εικονικοποίηση μπορεί να παρουσιαστεί ένα σύνολο φυσικών πόρων ως ένα cluster εικονικών μηχανών. Κάθε VM είναι ένα πλήρες μηχάνημα που τρέχει όλα τα στοιχεία, συμπεριλαμβανομένου του δικού του λειτουργικού συστήματος, πάνω από το εικονικοποιημένο υλικό.

Έπειτα έρχεται η εποχή ανάπτυξης σε containers. Τα containers είναι παρόμοια με τα VM, αλλά έχουν λιγότερες ιδιότητες απομόνωσης για κοινή χρήση του λειτουργικού συστήματος (OS) μεταξύ των εφαρμογών. Επομένως, τα containers θεωρούνται ελαφριά ως προς τον φόρτο που δημιουργούν στο φυσικό μηχάνημα. Παρόμοια με ένα VM, ένα container έχει το δικό του σύστημα αρχείων, μερίδιο CPU, μνήμη, χώρο διεργασιών και πολλά άλλα. Επειδή είναι αποσυνδεδεμένα από την υποκείμενη υποδομή, καταφέρνουν να είναι φορητά σε cloud και διάφορα λειτουργικά συστήματα.

2.1.1 Δομικά στοιχεία

Ένα Kubernetes cluster αποτελείται από ένα σύνολο μηχανών, που ονομάζονται κόμβοι nodes, όπου εκτελούνται εφαρμογές με κοντέινερ. Κάθε cluster έχει τουλάχιστον έναν κόμβο εργασίας. Ένας κόμβος μπορεί να είναι ένα VM ή ένα φυσικό μηχάνημα.

Οι κόμβοι περιέχουν τα pods που είναι τα στοιχεία που εξυπηρετούν τον φόρτο εργασίας της εφαρμογής. Στα pods τρέχουν τα κοντέινερ της εφαρμογής.



Σχήμα 2.2: *Δομικά στοιχεία του Kubernetes [11]*

Στοιχεία του Control Plane

Το control plane διαχειρίζεται τους κόμβους και τα pods στο cluster. Σε περιβάλλοντα παραγωγής, το control plane συνήθως εκτελείται σε πολλούς υπολογιστές και ένα cluster συνήθως εκτελεί πολλούς κόμβους, παρέχοντας ανοχή σε σφάλματα και υψηλή διαθεσιμότητα.

- kube-apiserver

  Η διεπαφή για το Control Plane. Εξυπηρετεί τις λειτουργίες REST, μέσω της οποίας αλληλεπιδρούν όλα τα άλλα στοιχεία του cluster.

- etcd

  Χώρος αποθήκευσης της μορφής key-value που χρησιμοποιείται ως αντίγραφο ασφαλείας για όλα τα δεδομένα του cluster.

- kube-scheduler

  Παρακολουθεί τα νέα pods που δημιουργήθηκαν και δεν ανατέθηκαν σε κάποιο κόμβο και επιλέγει έναν κόμβο στον οποίο θα εκτελούνται.

- kube-controller-manager

  Περιέχει διαφόρων τύπων ελεγκτές που διασφαλίζουν την σωστή λειτουργία διαφόρων στοιχείων στο cluster.

- cloud-controller-manager

  Επιτρέπει την σύνδεση ενός cluster στο API ενός παρόχου cloud και διαχωρίζει τα στοιχεία που αλληλεπιδρούν με αυτήν την πλατφόρμα cloud από τα στοιχεία που αλληλεπιδρούν μόνο με το cluster.

Στοιχεία των κόμβων

- kubelet

  Ένας agent που τρέχει σε κάθε κόμβο στο cluster. Διασφαλίζει ότι είναι σε λειτουργία τα κοντέινερ στα pods.

- kube-proxy

  Ένας διαμεσολαβητής δικτύου που εκτελείται σε καθένα από τους κόμβους σε ένα cluster. Διατηρεί κανόνες δικτύου στους κόμβους. Αυτοί οι κανόνες δικτύου επιτρέπουν την επικοινωνία δικτύου με τα pods από συνδέσεις δικτύου εντός ή εκτός του cluster.

2.1.2 Λειτουργίες

Τα containers είναι ένας καλός τρόπος για να ομαδοποιηθούν και να εκτελεστούν οι εφαρμογές. Σε ένα περιβάλλον παραγωγής, πρέπει να υπάρχει διαχείριση των κοντέινερ που εκτελούν τις εφαρμογές και διασφάλιση ότι δεν υπάρχει χρόνος διακοπής λειτουργίας. Για παράδειγμα, εάν ένα κοντέινερ τερματίσει, ένα άλλο κοντέινερ πρέπει να ξεκινήσει. Αυτές τις λειτουργίες αναλαμβάνει το Kubernetes, παρέχοντας ένα πλαίσιο για να εκτελούνται τα κατανεμημένα συστήματα με ευελιξία. Φροντίζει για την κλιμάκωση και το failover της εφαρμογής, παρέχει μοτίβα ανάπτυξης, καθώς και άλλες λειτουργίες.

Κάποιες από αυτές τις λειτουργίες είναι οι παρακάτω.

- Ανακάλυψη υπηρεσίας (service discovery)

  Το Kubernetes μπορεί να δίνει διαθέσιμο ένα κοντέινερ χρησιμοποιώντας το όνομα μέσω DNS (Domain Name System) ή χρησιμοποιώντας τη δική του διεύθυνση IP.

- Εξισορρόπηση φορτίου (load balancing)

  Εάν η κίνηση σε ένα κοντέινερ είναι υψηλή, τότε γίνεται εξισορρόπηση φορτίου και διανέμεται η κίνηση στο υπόλοιπο δίκτυο έτσι ώστε η εφαρμογή να λειτουργεί σταθερά.

- Ενορχήστρωση αποθήκευσης

  Το Kubernetes επιτρέπει να αυτόματη προσάρτηση ένα συστήματος αποθήκευσης, όπως από τοπικούς αποθηκευτικούς χώρους ή ao δημόσιους παρόχους cloud..

- Αυτοματοποιημένα rollouts και rollbacks

  Δύναται να υπάρχει περιγραφή της επιθυμητής κατάστασης για τα κοντέινερ που έχουν αναπτυχθεί. Χρησιμοποιώντας το Kubernetes διασφαλίζεται ότι μπορεί να αλλάξει η πραγματική κατάσταση στην επιθυμητή κατάσταση με ελεγχόμενο ρυθμό. Για παράδειγμα, υπάρχει δυνατότητα να αυτοματοποιηθεί το Kubernetes για να δημιουργηθούν νέα κοντέινερ για μια εφαρμογή, να αφαιρεθούν τα υπάρχοντα κοντέινερ και να υιοθετηθούν όλοι οι πόροι τους στο νέο κοντέινερ.

- Αυτοματοποιημένο bin packing

  Αρχικά παρέχεται στο Kubernetes ένα cluster από κόμβους που μπορεί να χρησιμοποιήσει για την εκτέλεση εργασιών με κοντέινερ. Έπειτα περιγράφεται πόση CPU και μνήμη (RAM) χρειάζεται κάθε κοντέινερ. Το Kubernetes αναλαμβάνει να τοποθετήσει αποδοτικά τα κοντέινερ στους κόμβους, ώστε να κάνει την καλύτερη χρήση των πόρων.

- Αυτοΐαση (self healing)

  Το Kubernetes επανεκκινεί τα κοντέινερ που αποτυγχάνουν, αντικαθιστά τα κοντέινερ, τερματίζει τα κοντέινερ που δεν ανταποκρίνονται και δεν τα έχει διαθέσιμα στην εφαρμογή μέχρι να είναι έτοιμα να εξυπηρετήσουν κίνηση.

- Διαχείριση μυστικών και ρυθμίσεων

  Το Kubernetes παρέχει τεχνικές για αποθήκευση και διαχειρίση ευαίσθητων πληροφορίων, όπως κωδικούς πρόσβασης, OAuth tokens και κλειδιά SSH. Είναι δυνατή η δημιουργία και η ενημέρωση μυστικών και ρυθμίσεων εφαρμογών χωρίς την επαναδημιουργία των εικόνων του κοντέινερ και χωρίς να αποκαλύπτονται οι ευαίσθητες πληροφορίες στην υπόλοιπη υποδομή.

2.1.3 Kubernetes objects

Τα Kubernetes objects είναι κάποιες οντότητες που περιγράφουν μια επιθυμητή κατάσταση για το cluster. Περιγράφουν ποιες εφαρμογές εκτελούνται, σε ποιους κόμβους, τι πόρους χρειάζονται και όποιες στρατηγικές χρησιμοποιούν για την ομαλή εκτέλεση τους.

Για τη δημιουργία ενός Kubernetes object, πρέπει να δοθεί το spec του αντικειμένου που περιγράφει την επιθυμητή κατάστασή του, καθώς και κάποιες βασικές πληροφορίες για το αντικείμενο, όπως ένα όνομα. Μπορεί να χρησιμοποιηθεί το Kubernetes API για τη δημιουργία του αντικειμένου. Αυτό το αίτημα στο API πρέπει να περιλαμβάνει αυτές τις πληροφορίες ως JSON στο σώμα αιτήματος. Πιο συχνά χρησιμοποιείται ένα αρχείο .yaml (manifest) με το εργαλείο kubectl (Kubernetes command-line tool). Το kubectl μετατρέπει τις πληροφορίες σε JSON κατά την υποβολή του αιτήματος API.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Σχήμα 2.3: *Παράδειγμα .yaml manifest*

Στο .yaml αρχείο τα παρακάτω πεδία είναι υποχρεωτικά.

- apiVersion: Η έκδοση του Kubernetes API που χρησιμοποιείται
- kind: Το είδος του αντικειμένου
- metadata: Το είδος του αντικειμένου
- spec: Επιθυμητή κατάσταση του αντικειμένου

Βασική στοιχείο στο Kubernetes είναι οι χώροι ονομάτων (namespaces). Οι χώροι ονομάτων παρέχουν έναν μηχανισμό για την απομόνωση ομάδων οντοτήτων σε ένα cluster. Τα ονόματα των οντοτήτων πρέπει να είναι μοναδικά εντός ενός χώρου ονομάτων, αλλά όχι μεταξύ όλων. Η εμβέλεια βάσει χώρου ονομάτων ισχύει σε κάποια αντικείμενα όπως Deployments [12], Services και άλλα. Κάποια άλλα αντικείμενα είναι διαθέσιμα σε όλο το cluster, όπως τα StorageClass, PersistentVolumes, οι κόμβοι και άλλα.

Σημαντικό επίσης στοιχείο είναι τα Services [13], που καθιστούν δυνατή την πρόσβαση σε μία εφαρμογή από ένα σετ από pods. Τα Services είναι μια έννοια που ορίζει ένα σύνολο από pods που εκτελούνται κάπου στο cluster και όλα παρέχουν την ίδια λειτουργικότητα. Όταν δημιουργείται, σε κάθε service εκχωρείται μια μοναδική διεύθυνση IP. Αυτή η διεύθυνση είναι συνδεδεμένη με τη διάρκεια ζωής του service και δεν θα αλλάξει όσο εκτελείται το service. Τα pods μπορούν να διαμορφωθούν ώστε να μιλάνε με το service και υπάρχει load-balancing με τα υπόλοιπα pods του service.

Workloads

Ένα workload είναι μια εφαρμογή που εκτελείται στο Kubernetes. Εκτελείται είτε είναι ένα μεμονωμένο στοιχείο είτε είναι πολλά που λειτουργούν μαζί, σε ένα σύνολο από pods. Κάποια ενσωματωμένα workload resources στο kubernetes είναι τα παρακάτω:

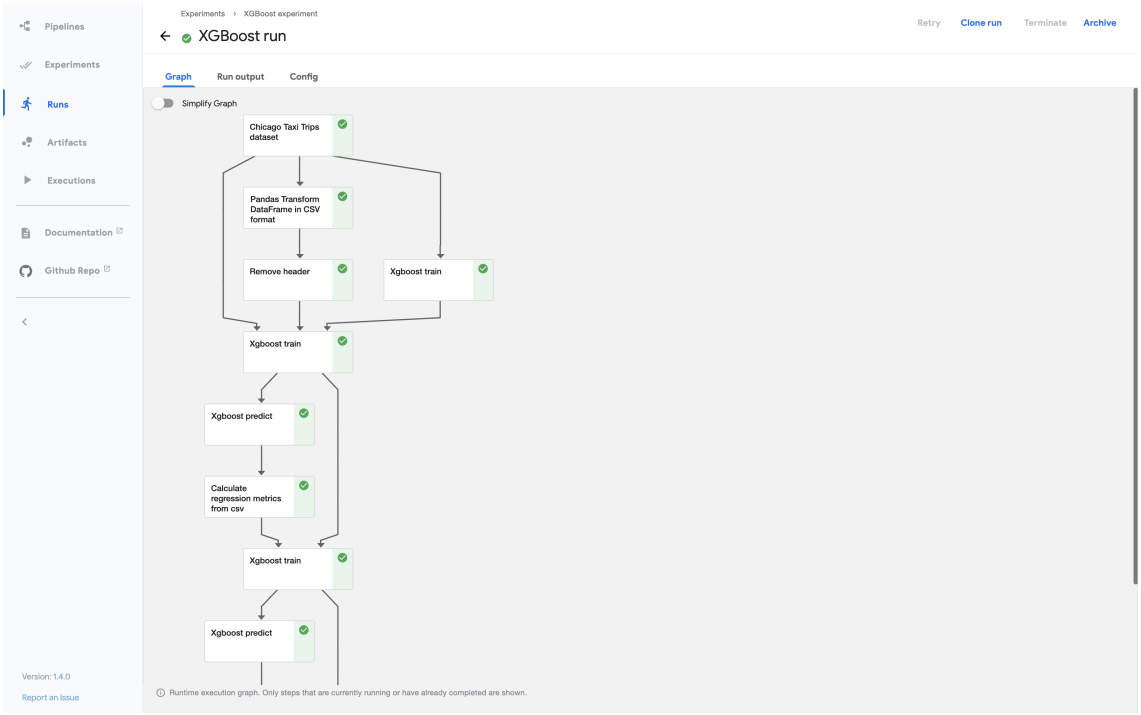
- Deployment: Στα πλαίσια μιας εφαρμογής, περιέχουν οδηγίες για την διαχείριση των pods και των ReplicaSets.
- ReplicaSet: Ο σκοπός ενός ReplicaSet [14] είναι να διατηρεί ένα σταθερό σύνολο από αντίγραφα pods που εκτελούνται ανά πάσα στιγμή.
- StatefulSet: Χρησιμοποιείται για τη διαχείριση stateful εφαρμογών.
- DaemonSet: Διασφαλίζει ότι όλοι (ή ορισμένοι) κόμβοι εκτελούν ένα αντίγραφο ενός pod. Καθώς οι κόμβοι προστίθενται στο cluster, προστίθενται και τα pods σε αυτά. Καθώς οι κόμβοι αφαιρούνται από το cluster, αυτά τα pods συλλέγονται από τον garbage collector. Η διαγραφή ενός DaemonSet θα καθαρίσει τα pods που δημιούργησε.
- Job: Μια εργασία που περιγράφει μια επιτυχημένη εκτέλεση, δημιουργεί ένα ή περισσότερα pods και θα συνεχίσει να προσπαθεί να εκτελέσει ξανά τα pods μέχρι να τερματίσει επιτυχώς.

2.2 Kubeflow

Το Kubeflow [3] είναι μια πλατφόρμα για επιστήμονες δεδομένων που θέλουν να κατασκευάσουν και να πειραματιστούν με σωληνώσεις μηχανικής μάθησης. Χρησιμοποιείται επίσης για μηχανικούς μηχανικής μάθησης που θέλουν να δημιουργήσουν συστήματα μηχανικής μάθησης σε διάφορα περιβάλλοντα για ανάπτυξη, δοκιμή και λειτουργία σε επίπεδο παραγωγής.

2.2.1 Σωληνώνεις (pipelines)

Μια σωλήνωση [15] είναι μια περιγραφή μιας ροής εργασίας μηχανικής μάθησης, που συμπεριλαμβάνει όλα τα στοιχεία της ροής εργασίας και το τρόπου με τον οποίο συνδυάζονται με τη μορφή γραφήματος. Η σωλήνωση περιλαμβάνει τον ορισμό των εισόδων (παραμέτρων) που απαιτούνται για την εκτέλεση της σωλήνωσης και τις εισόδους και εξόδους κάθε σταδίου.



Σχήμα 2.4: *Kubeflow Pipeline*

Βασικά στοιχεία των σωληνώσεων είναι τα παρακάτω:

- Component: Κάθε component της σωλήνωσης είναι ένα αυτόνομο σύνολο κώδικα από τον χρήστη, πακεταρισμένο ως εικόνα Docker , που εκτελεί μια λειτουργία στη σωλήνωση.
- Γράφος: Η οπτική περιγραφή της εκτέλεσης τη σωλήνωσης

- Εκτέλεση (Run): Μία μεμονωμένη εκτέλεση της σωλήνωσης. Κάθε εκτέλεση καταγράφει όλες τις παραμέτρους που χρησιμοποιήθηκαν, τα logs και artifacts που δημιουργήθηκαν. Βοηθώντας με αυτό τον τρόπο στην παρακολούθηση της εξέλιξης της εκτέλεσης και στην αναπαραγωγιμότητα. Στο Kubeflow υποστηρίζονται και επαναλαμβανόμενες εκτελέσεις.

- Πείραμα: Χρησιμοποιείται για την ομαδαποίηση των εκτελέσεων της σωλήνωσης με διαφορετικές παραμέτρους.

- Ενεργοποιητής εκτέλεσης (Run trigger): Ένας ενεργοποιητής εκτέλεσης ενημερώνει το σύστημα πότε να δημιουργήσει μια νέα εκτέλεση. Η ενεργοποίηση μπορεί να είναι περιοδική ή προγραμματισμένη με cron.

- Βήμα Step: Ένα βήμα είναι μία εκτέλεση ενός από τα components του αγωγού. Σε μία σωλήνωση τα components μπορεί εκτελεστούν πολλές φορές σε βρόγχους.

- Artifacts εξόδου: Παράγονται από οποιοδήποτε στάδιο της σωλήνωσης και μπορούν να δώσουν κάποιες απεικονίσεις, όπως ένας πίνακας σύγχυσης και πολλά άλλα.

- Metadata μηχανικής μάθησης: Περιέχουν πληροφορίες για την εξέλιξη της εκτέλεσης ενός πειράματος, την διαθεσιμότητα των artifacts και περαιτέρω λεπτομέρειες για τις ρυθμίσεις της κάθε λειτουργίας.

Pipelines SDK (Software Development Kit)

Το Kubeflow Pipelines SDK [6] παρέχει ένα σύνολο πακέτων Python που μπορεί να χρησιμοποιηθεί για τον καθορισμό και την εκτέλεση ροών εργασιάς μηχανικής μάθησης. Το SDK περιέχει υλοποιημένη διεπαφή με το Kubeflow Pipelines API και άλλες λειτουργίες του Kubeflow. Δίνει την δυνατότητα να δημιουργηθούν τα components που χρειάζεται η σωλήνωση εξολοκλήρου από μεθόδους της python δημιουργώντας αυτά που αποκαλεί lightweight components, καθώς και να χρησιμοποιήσει ήδη υλοποιημένα docker images για την δημιουργία ενός component. Τα docker images μπορεί να χρησιμοποιηθούν είτε σαν βάση και να υλοποιήσει ο χρήστης τον εκτελέσιμο κώδικα, είτε να εκτελεστεί ο κώδικας που υπάρχει στο docker image. Στην πράξη ο χρήστης μπορεί να δώσει ό,τι χρειάζεται για να χτιστεί ένα docker image, όπως την βάση (base_image) και τις βιβλιοθήκες που χρειάζεται να εγκατασταθούν. Έτσι χρησιμοποιώντας μόνο την python αποφεύγεται η πολυπλοκότητα της δημιουργίας Dockerfile για την δημιουργία docker image και η υλοποίηση ενός πολύπλοκου yaml manifest για την δημιουργία του Kubeflow pipeline από το Kubernetes API.

```python
from typing import NamedTuple

@component(base_image='tensorflow/tensorflow:1.11.0-py3')
def my_divmod(
  dividend: float,
  divisor: float,
  metrics: Output[Metrics]) -> NamedTuple(
    'MyDivmodOutput',
    [
      ('quotient', float),
      ('remainder', float),
    ]):
    '''Divides two numbers and calculate  the quotient and remainder'''

    # Import the numpy package inside the component function
    import numpy as np

    # Define a helper function
    def divmod_helper(dividend, divisor):
        return np.divmod(dividend, divisor)

    (quotient, remainder) = divmod_helper(dividend, divisor)

    # Export two metrics
    metrics.log_metric('quotient', float(quotient))
    metrics.log_metric('remainder', float(remainder))

    from collections import namedtuple
    divmod_output = namedtuple('MyDivmodOutput',
        ['quotient', 'remainder'])
    return divmod_output(quotient, remainder)
```

Σχήμα 2.5: *Παράδειγμα υλοποίησης Kubeflow Component με SDK*

```python
import kfp.dsl as dsl
@dsl.pipeline(
   name='calculation-pipeline',
   description='An example pipeline that performs arithmetic calculations.',
   pipeline_root='gs://my-pipeline-root/example-pipeline'
)
def calc_pipeline(
   a: float=1,
   b: float=7,
   c: float=17,
):
    # Passes a pipeline parameter and a constant value as operation arguments.
    add_task = add(a, 4) # The add_op factory function returns
                         # a dsl.ContainerOp class instance.

    # Passes the output of the add_task and a pipeline parameter as operation
    # arguments. For an operation with a single return value, the output
    # reference is accessed using `task.output` or
    # `task.outputs['output_name']`.
    divmod_task = my_divmod(add_task.output, b)

    # For an operation with multiple return values, output references are
    # accessed as `task.outputs['output_name']`.
    result_task = add(divmod_task.outputs['quotient'], c)
```

Σχήμα 2.6: *Παράδειγμα υλοποίησης Kubeflow Pipeline με SDK*

```
# Specify pipeline argument values
arguments = {'a': 7, 'b': 8}

# Submit a pipeline run
client.create_run_from_pipeline_func(
    calc_pipeline,
    arguments=arguments,
    mode=kfp.dsl.PipelineExecutionMode.V2_COMPATIBLE)
```

Σχήμα 2.7: *Παράδειγμα κλήσης στο Kubeflow API για την δημιουργία pipeline*

2.2.2 Katib

Το Katib [7] είναι ένα πρότζεκτ αναπτυγμένο για Kubernetes για αυτοματοποιημένη μηχανική μάθηση (AutoML). Πραγματοποιεί εύρεση βέλτιστων υπερπαραμέτρων, πρόωρη διακοπή (early stopping) και αναζήτηση βέλτιστης αρχιτεκτονικής νευρωνικού δικτύου (NAS). Υποστηρίζει διάφορες πλατφόρμες μηχανικής μάθησης όπως TensorFlow, MXNet, PyTorch, XGBoost και άλλες. Στην πράξη το Katib εκτελεί αρκετές δοκιμές εκπαίδευσης (trials) σε ένα πείραμα, για την εύρεση βέλτιστων υπερπαραμέτρων . Κάθε δοκιμή δοκιμάζει ένα διαφορετικό σύνολο υπερπαραμέτρων. Στο τέλος του πειράματος, το Katib εξάγει τις βέλτιστες τιμές για τις υπερπαραμέτρους.

2.2.3 Notebooks

Τα Kubeflow Notebooks [16] παρέχουν έναν τρόπο εκτέλεσης περιβαλλόντων ανάπτυξης μέσα στο Kubernetes cluster, εκτελώντας τα μέσα στο pods. Υποστηρίζουν JupyterLab, RStudio και Visual Studio Code. Οι notebook servers εκτελούνται ως κοντέινερ μέσα σε ένα pod, γεγονός που σημαίνει ότι ο τύπος του server και ποια πακέτα είναι εγκατεστημένα καθορίζονται από την εικόνα Docker που θα χρησιμοποιηθεί.

2.2.4 Kserve

Το KServe [8] χρησιμοποιείται για την δημιουργία inference services με τα μοντέλα μηχανικής μάθησης σε περιβάλλοντα παραγωγής. Υποστηρίζει τις πλέον χρησιμοποιούμενες πλατφόρμες μηχανικής μάθησης TensorFlow, XGBoost, scikit-learn, PyTorch και ONNX.Επιλύει την πολυπλοκότητα της αυτόματης κλιμάκωσης (δημιουργία/διαγραφή πολλαπλών μονάδων για εξυπηρέτηση), του ελέγχου υγείας, της αυτόματης κλιμάκωσης στις GPU και τα rollouts. Επίσης παρέχει μια διεπαφή για τον χρήστη που υπάρχει και στο κεντρικό Dasboard στην σελίδα Models.

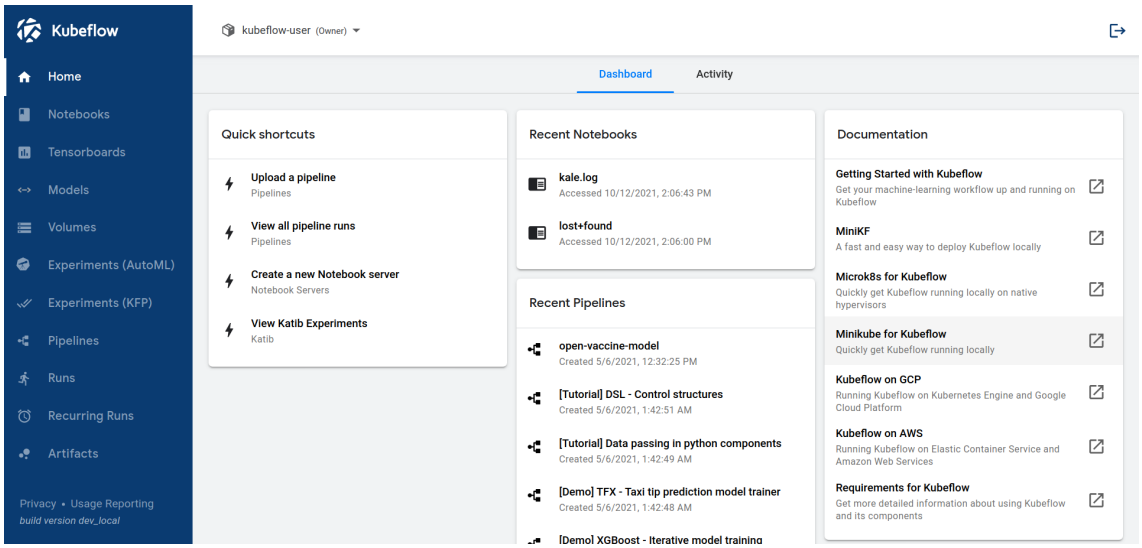
Για την αποθήκευση των μοντέλων το Kserve υποστηρίζει τους παρακάτω χώρους αποθήκευσης:

- Azure Blob
- Amazon S3
- Google Cloud Storage (GCS)

- PVC
- URI

2.2.5 Κεντρικό Dasboard

Το Dasboard [17] παρέχει γρήγορη πρόσβαση και διεπαφή στα στοιχεία του Kubeflow που έχουν εγκατασταθεί στο Kubernetes cluster.



Σχήμα 2.8: *Kubeflow Dashboard*

Κεφάλαιο 3

Σχεδίαση και υλοποίηση υποδομής

3.1 Επιλογή περιβάλλοντος

Στοχεύοντας σε μια υποδομή για MLOps στην παραγωγή προκύπτουν πολλοί παράμετροι που καθορίζουν την σχεδίαση της. Αρχικά πρέπει να επιλεγεί που θα φιλοξενηθεί η υποδομή. Μια εταιρεία μπορεί να επιλέξει να επενδύσει είτε σε εσωτερική υλοποίηση, είτε σε μια υποδομή στο cloud, είτε σε μια υβριδική λύση μεταξύ των δύο. Οι εταιρείες τεχνολογίας που θέλουν να επιβιώσουν μακροπρόθεσμα έχουν συνήθως εσωτερικές ομάδες και κατασκευάζουν προσαρμοσμένες λύσεις. Εάν έχουν τις δεξιότητες, τις γνώσεις και τα εργαλεία για να αντιμετωπίσουν σύνθετα προβλήματα, η λύση αυτή ενδείκνυται. Υπάρχουν όμως και άλλοι παράγοντες που πρέπει να ληφθούν υπόψη, όπως:

- Εργάσιμος χρόνος και προσπάθεια

  Σύμφωνα με έρευνα του cnvrg.io [18], οι επιστήμονες δεδομένων συχνά αφιερώνουν το χρόνο τους δημιουργώντας λύσεις για να προσθέσουν στην υπάρχουσα υποδομή τους προκειμένου να ολοκληρώσουν έργα. Το 65% του χρόνου τους αφιερώθηκε σε βαριές εργασίες μηχανικής, μη επιστήμης δεδομένων, όπως η παρακολούθηση, η διαμόρφωση, η διαχείριση υπολογιστικών πόρων, η υποδομή εξυπηρέτησης, η εξαγωγή χαρακτηριστικών και η ανάπτυξη μοντέλων.

  Αυτός ο χαμένος χρόνος αναφέρεται συχνά ως «κρυφό τεχνικό χρέος» (hidden technical debt) και είναι ένα κοινό σημείο συμφόρησης για τις ομάδες μηχανικής μάθησης. Η δημιουργία μιας εσωτερικής λύσης ή η διατήρηση μιας λύσης μη αποδοτικά μπορεί να διαρκέσει από 6 μήνες έως 1 χρόνο. Ακόμη και όταν έχει δημιουργηθεί μια λειτουργική υποδομή, για να συντηρηθεί και να είναι διαρκώς ενημερωμένη με την τελευταία τεχνολογία απαιτείται διαχείριση του κύκλου ζωής και μια ομάδα που ασχολείται αποκλειστικά με αυτό.

- Ανθρώπινο δυναμικό

  Για την δημιουργία των MLOps απαιτείται αρκετή δουλειά από την ομάδα των μηχανικών μηχανικής μάθησης. Για μια ομαλή ροή εργασιών μηχανικής μάθησης, κάθε ομάδα επιστήμης δεδομένων πρέπει να έχει μια ομάδα μηχανικών που κατανοεί τις μοναδικές απαιτήσεις ανάπτυξης μοντέλων μηχανικής εκμάθησης.

  Επενδύοντας σε μια πλατφόρμα MLOps αποκλειστικά στο cloud, αυτές οι διαδικασίες

μπορούν να αυτοματοποιηθούν πλήρως, διευκολύνοντας τις ομάδες μηχανικών MLOps να επικεντρωθούν στη βελτιστοποίηση της υποδομής.

- Κόστος

  Η ύπαρξη μιας αποκλειστικής ομάδας μηχανικών για τη διαχείριση μοντέλων μπορεί να είναι δαπανηρή από μόνη της. Εάν χρειαστεί να κλιμακωθούν τα πειράματα και οι αναπτύξεις, θα πρέπει να προσληφθούν περισσότεροι μηχανικοί για να διαχειριστούν αυτήν τη διαδικασία. Είναι μια σημαντική επένδυση και μια αργή διαδικασία για να βρεθεί η κατάλληλη ομάδα.

  Μια λύση MLOps έχει κατασκευαστεί με γνώμονα την επεκτασιμότητα, με ένα κλάσμα του κόστους. Αφού υπολογιστούν όλα τα διαφορετικά κόστη που σχετίζονται με την πρόσληψη και την ενσωμάτωση μιας ολόκληρης ομάδας μηχανικών, η απόδοση της επένδυσής μειώνεται, γεγονός που οδηγεί στον επόμενο παράγοντα.

- Χρόνος για να υπάρξει κέρδος

  Μπορεί να χρειαστεί πάνω από ένα χρόνο για να δημιουργηθεί μια λειτουργική υποδομή μηχανικής εκμάθησης. Μπορεί να χρειαστεί ακόμη περισσότερος χρόνος για τη δημιουργία μιας ροής δεδομένων που μπορεί να παράγει αξία για έναν οργανισμό.

  Εταιρείες όπως η Uber, το Netflix και το Facebook έχουν αφιερώσει χρόνια και τεράστιες προσπάθειες μηχανικής για να κλιμακώσουν και να διατηρήσουν τις πλατφόρμες μηχανικής εκμάθησης τους για να παραμείνουν ανταγωνιστικές. Για τις περισσότερες εταιρείες, μια επένδυση όπως αυτή δεν είναι δυνατή, ούτε και απαραίτητη. Το τοπίο της μηχανικής μάθησης έχει ωριμάσει από τότε που η Uber, το Netflix και το Facebook δημιούργησαν αρχικά τις εσωτερικές λύσεις τους. Υπάρχουν αρκετές έτοιμες λύσεις που προσφέρουν όλα όσα μπορεί να χρειάζεται μια εταιρεία, σε ένα κλάσμα του κόστους.

- Κόστος ευκαρίας

  Όπως αναφέρθηκε παραπάνω, μια έρευνα δείχνει ότι το 65% του χρόνου ενός επιστήμονα δεδομένων αφιερώνεται σε εργασίες που δεν αφορούν την επιστήμη δεδομένων. Η χρήση μιας πλατφόρμας MLOps αυτοματοποιεί τις τεχνικές εργασίες και μειώνει τις λειτουργίες DevOps. Οι επιστήμονες δεδομένων μπορούν αφιερώνουν το χρόνο τους κάνοντας περισσότερα από αυτά που προσλήφθηκαν για να κάνουν, να παραδίδουν αποδοτικά μοντέλα, ενώ ο πάροχος cloud αναλαμβάνει τα υπόλοιπα. Η υιοθέτηση μιας πλατφόρμας MLOps στο cloud δίνει ένα σημαντικό ανταγωνιστικό πλεονέκτημα που επιτρέπει στην ανάπτυξη της μηχανικής εκμάθησης να κλιμακώνεται σημαντικά.

Εν αντιθέσει, μια υβριδική λύση έρχεται να λύσει κάποια άλλα ζητήματα. Σε ορισμένες εταιρείες έχουν δοθεί ιδιωτικά και ευαίσθητα δεδομένα. Αυτά τα δεδομένα δεν γίνεται να μεταφερθούν από τους διακομιστές τους στην πιθανότητα οποιασδήποτε ευπάθειας. Σε αυτή την περίπτωση χρησιμοποιείται η υβριδική υποδομή cloud για MLOps. Προς το παρόν, η υποδομή cloud υπάρχει παράλληλα με τοπικά συστήματα στις περισσότερες περιπτώσεις. Η

διαχείριση υβριδικού νέφους είναι πολύπλοκη, αλλά συχνά απαραίτητη. Η υποδομή cloud είναι ολοένα και πιο δημοφιλής, αλλά εξακολουθεί να είναι σπάνιο να βρεθεί μια μεγάλη εταιρεία που να έχει εγκαταλείψει εντελώς την εσωτερική υποδομή.

Όσον αφορά το MLOps αποκλειστικά στο cloud, υπάρχουν αρκετές λύσεις. Πρόκειται για πλήρως διαχειριζόμενες υπηρεσίες που παρέχουν στους προγραμματιστές και στους επιστήμονες δεδομένων τη δυνατότητα να δημιουργούν, να εκπαιδεύουν και να αναπτύσσουν γρήγορα μοντέλα μηχανικής μάθησης. Μερικές από τις κορυφαίες εμπορικές πλατφόρμες είναι το Amazon Sagemaker, το Microsoft Azure MLOps με τα επιμέρους εργαλεία Azure Machine Learning, Azure Pipelines και το Google Cloud MLOps με τα εργαλεία Dataflow, AI Platform Notebook, TFX και Kubeflow.

3.2 Χρήση Kubernetes για MLOps

Στην προηγούμενη παράγραφο περιγράφηκαν κάποιες λύσεις MLOps και τα περιβάλλοντα που που μπορούν να υπάρξουν. Σαν υπόβαθρο αρκετές από αυτές τις λύσεις χρησιμοποιούν το Kubernetes. Το Kubernetes είναι ουσιαστικά ένα σύστημα ενορχήστρωσης κοντέινερ ανοιχτού κώδικα που χρησιμοποιείται από οργανισμούς για την αυτοματοποίηση της ανάπτυξης, της κλιμάκωσης και της διαχείρισης για εφαρμογές υπολογιστών. Ως ενορχηστρωτής, το Kubernetes χρησιμοποιείται για τη δημιουργία κλιμακούμενων κατανεμημένων συστημάτων και επίσης αξιοποιείται για να επιφέρει την απαραίτητη ευελιξία σε ποικίλα πλαίσια μηχανικής μάθησης στα οποία μπορούν να εργαστούν οι επιστήμονες δεδομένων. Αυτή η ευελιξία επεκτείνεται στην επεκτασιμότητα και την επανάληψη που απαιτούνται από μονάδες που εκτελούν συστήματα μηχανικής εκμάθησης σε προϊόντα, καθώς και περισσότερο έλεγχο στην κατανομή πόρων που απαιτείται από τη μονάδα λειτουργιών. Όταν εφαρμόζεται στη μηχανική μάθηση, το Kubernetes μπορεί να διευκολύνει σημαντικά τη διαδικασία για τους επιστήμονες δεδομένων και στους διαχειριστές.

Ένα Kubernetes cluster στην παραγωγή έχει περισσότερες απαιτήσεις από ένα περιβάλλον προσωπικής χρήσης, ανάπτυξης ή δοκιμής Kubernetes. Από τις πλέον βασικότερες απαιτήσεις είναι η ασφαλής πρόσβαση από πολλούς χρήστες, υψηλή διαθεσιμότητα και πόρους για να προσαρμοστεί στις μεταβαλλόμενες απαιτήσεις.

Ένα υψηλά διαθέσιμο Kubernetes cluster είναι ένα cluster που μπορεί να αντέξει μια αστοχία σε οποιοδήποτε από τα στοιχεία του και να συνεχίσει να εξυπηρετεί φόρτο εργασίας χωρίς διακοπή. Υπάρχουν τρία στοιχεία απαραίτητα για ένα υψηλά διαθέσιμο Kubernetes cluster:

- Πρέπει να υπάρχουν περισσότεροι από ένας κόμβοι διαθέσιμοι ανά πάσα στιγμή.
- Το επίπεδο ελέγχου πρέπει να εκτελείται σε περισσότερους από έναν κόμβους, έτσι ώστε η απώλεια ενός μόνο κόμβου να μην καθιστά το σύμπλεγμα μη λειτουργικό.
- Η κατάσταση του cluster πρέπει να βρίσκεται σε ένα χώρο αποθήκευσης δεδομένων που είναι ο ίδιος υψηλά διαθέσιμος.

Ενώ υπάρχουν πολλές λύσεις Kubernetes, οι διαχειριζόμενες λύσεις όπως το Azure Kubernetes Service (AKS), το Amazon Elastic Kubernetes Service (EKS) και το Google Kubernetes Engine (GKE) είναι οι πιο δημοφιλείς. Το Kubernetes είναι μια πολύ περίπλοκη πλατφόρμα, αλλά η δημιουργία ενός cluster Kubernetes είναι αρκετά εύκολη, με μια διαχειριζόμενη λύση cloud. Ωστόσο αυτές οι λύσεις είναι επί πληρωμή και κοστίζουν με βάση την χρήση των πόρων. Οπότε αρκετές εταιρείες προχωρούν σε τοπική εγκατάσταση.

3.3 Τοπική υλοποίηση

Στα πλαίσια αυτής της εργασίας υλοποιήθηκε τοπική εγκατάσταση, δεν χρησιμοποιήθηκε κάποια λύση σε cloud. Επομένως διερευνήθηκαν εκδόσεις του Kubernetes για τοπική εγκατάσταση και έπειτα εγκαταστάθηκε το Kubeflow. Επίσης, διερευνήθηκαν υλοποιήσεις για να επιτευχθεί υψηλή διαθεσιμότητα σε ένα φυσικό μηχάνημα. Η ύπαρξη ενός μόνο φυσικού μηχανήματος, αναιρεί το νόημα της υψηλής διαθεσιμότητας, αλλά προσεγγίζεται ως προσομοίωση ενός περιβάλλοντος στην παραγωγή.

3.3.1 Λύση με εικονικά μηχανήματα

Για να επιτευχθεί η υψηλή διαθεσιμότητα, πρέπει να υπάρχουν τουλάχιστον τρεις κόμβοι. Με χρήση του εργαλείου του εργαλείου Multipass της Canonical υλοποιήθηκαν τρία VM σε ένα φυσικό μηχάνημα. Το multipass εγκαθιστά ως προεπιλογή την τελευταία έκδοση Ubuntu LTS (Long Term Support).

Αλγοριθμος 3.1: *Δημιουργία τριών VM με το multipass*

```
multipass launch --name dsml01 --cpus 2 --mem 4096M --disk 20G
multipass launch --name dsml02 --cpus 2 --mem 4096M --disk 20G
multipass launch --name dsml03 --cpus 2 --mem 4096M --disk 20G
```

Αφού δημιουργήθηκαν όλα τα εικονικά μηχανήματα, εγκαταστάθηκε το kubernetes σε όλα και έπειτα ενώθηκαν σε cluster.

Για να υπάρχει διαθέσιμη η GPU του μηχανήματος σε κάποιο εικονικό μηχάνημα, έπρεπε να γίνει PCI passthrough. Αυτό θα σήμαινε ότι η GPU δεν θα ήταν πια διαθέσιμη στο φυσικό μηχάνημα, αλλά μόνο σε ένα εικονικό. Για αυτό τον λόγο και δεν συνεχίστηκε η λύση.

3.3.2 Λύση με LXD

Το LXD είναι ένας διαχειριστής κοντέινερ συστήματος και εικονικών μηχανών. Προσφέρει μια ενοποιημένη εμπειρία χρήστη γύρω από συστήματα Linux που εκτελούνται μέσα σε κοντέινερ ή εικονικά μηχανήματα. Το LXC παρέχει τη βασική λειτουργικότητα που χρησιμοποιείται από το LXD. Το LXC είναι ένα πολύ γνωστό Linux container runtime που αποτελείται από εργαλεία, πρότυπα και βιβλιοθήκες και δεσμεύσεις γλώσσας. Ενώ τα VM παρέχουν ένα πλήρες περιβάλλον, τα κοντέινερ συστήματος προσφέρουν ένα περιβάλλον όσο

το δυνατόν πλησιέστερο σε αυτό που προσφέρει ένα VM, αλλά χωρίς την επιβάρυνση που συνοδεύει την εκτέλεση ενός ξεχωριστού kernel και την προσομοίωση όλου του υλικού. Το LXC συνδυάζει τα cgroups του Linux kernel και την υποστήριξη απομονωμένων χώρων ονομάτων για να παρέχει ένα απομονωμένο περιβάλλον για εφαρμογές.

Για την δημιουργία των LXC containers χρειάζεται να δημιουργηθούν τα απαραίτητα profile που περιέχουν πληροφορίες για τις ρυθμίσεις των containers, τους πόρους του συστήματος που θα χρησιμοποιήσουν και άλλα. Επίσης υπάρχουν ήδη δημιουργημένες LXC images διαφόρων λογισμικών linux που χρησιμοποιούνται ως βάση.

Αλγοριθμος 3.2: *Δημιουργία τριών LXC containers*

```
lxc launch -p default -p microk8s ubuntu:20.04 dsml01
lxc launch -p default -p microk8s ubuntu:20.04 dsml02
lxc launch -p default -p microk8s ubuntu:20.04 dsml03
```

Μετά την δημιουργία των κοντέινερ, εγκαταστάθηκε σε όλα ως λειτουργικό σύστημα το Ubuntu 20.04. Σε συνέχεια, όπως και στην περίπτωση των εικονικών μηχανημάτων, ενώθηκαν σε cluster.

Τα LXC δίνουν εύκολη πρόσβαση στους πόρους του μηχανήματος, δεν χρειάστηκε PCI passthrough για να μπορούν να χρησιμοποιούν την GPU του φυσικού μηχανήματος, ήταν απλά μία ρύθμιση.

3.4 Επιλογή εργαλείων και εκδόσεων

Η εργασία αυτή επικεντρώνεται στην υλοποίηση διαδικασιών MLOps με χρήση του εργαλείου Kubeflow. Όταν αναπτύσσονταν η υποδομή της εργασίας, το Kubeflow υποστήριζε μέχρι και την έκδοση 1.21 Kubernetes. Για το Kubernetes υπάρχουν πολλές λύσεις για τοπική εγκατάσταση, όπως το MicroK8s της εταιρείας Canonical, τα K3s, RKE2 της Rancher, το Minicube, το kind, το kubeadm και άλλα. Υλοποιήθηκε εγκατάσταση με το Microk8s και με το RKE2. Αμφότερες οι λύσεις συναντώνται στην παραγωγή, έχουν εύκολη εγκατάσταση και δημιουργία cluster. Προτιμήθηκε το Microk8s για τον λόγο ότι είχε μεγαλύτερη υποστήριξη στο Kubeflow από την open-source κοινότητα.

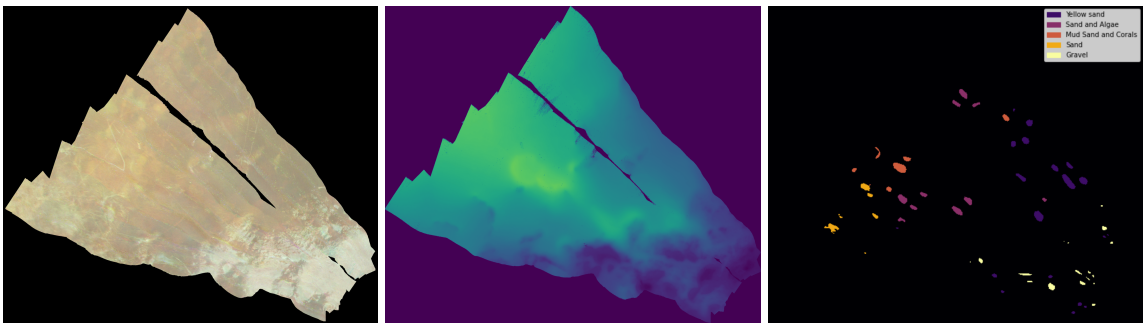
Κατά την συγγραφή της διπλωματικής, το Kubeflow ήταν σε δοκιμαστική φάση στην έκδοση 1.6. Παρόλα αυτά, εγκαταστάθηκε αυτή η έκδοση, για την χρήση των τελευταίων διαθέσιμων λειτουργιών. Συγκεκριμένα εγκαταστάθηκε η έκδοση v1.6.0-rc.0. Όπως είναι φυσικό, υπήρξαν αρκετές δυνατότητες που δεν ήταν πλήρως λειτουργικές. Η πλατφόρμα Kubeflow, όπως περιγράφηκε στο κεφάλαιο 2, συνενώνει πολλά αυτούσια δομικά στοιχεία, όπως το Kubeflow Pipelines, το KServe και αρκετά άλλα. Αυτά εγκαταστάθηκαν το καθένα ξεχωριστά, για να υπάρχουν οι τελευταίες εκδόσεις με όλες τις δυνατότητες. Χρησιμοποιήθηκαν τα yaml manifests του git αποθετηρίου του Kubeflow, με ταυτόχρονη χρήση του εργαλείου kustomize. Το εργαλείο kustomize είναι ένα εργαλείο που μπορεί να συνδυάζει κάποια αρχεία ρυθμίσεων με το yaml manifest ενός Kubernetes object, μεταβάλλοντας έτσι το παραγώμενο object.

Κεφάλαιο 4

Προσέγγιση προβλήματος

4.1 Περιγραφή προβλήματος και δεδομένων

Όπως περιγράφηκε στην εισαγωγή το πρόβλημα στο οποίο θα γίνει προσέγγιση είναι η ταξινόμηση στοιχείων του θαλάσσιου βυθού. Τα δεδομένα προέρχονται από ηχοβολιστικά μηχανήματα πολλαπλών δεσμών και πολλαπλών συχνοτήτων. Τελικό παραγόμενο είναι μία εικόνα οπισθοσκέδασης τριών διαστάσεων και μία εικόνα βαθυμετρίας δύο διαστάσεων, όπου αναπαριστούν τον θαλάσσιο πυθμένα. Παράλληλα, ύστερα από έρευνα, έχουν δημιουργηθεί εικόνες ίδιου μεγέθους αλλά μίας διάστασης όπου έχει κατηγοριοποιηθεί κάθε pixel σε μία κατηγορία εδάφους. Αυτές εικόνες αναφέρονται ως ground truth εικόνες. Η τιμή του κάθε pixel της αναπαριστά μία κατηγορία εδάφους. Τα τρία είδη των εικόνων αποτελούν τα δεδομένα, με βάση τα οποία θα εκπαιδεύσουμε ένα μοντέλο ταξινόμησης υποθαλάσσιου εδάφους. Έπειτα το μοντέλο αυτό θα μπορεί να δέχεται ως είσοδο μια εικόνα οπισθοσκέδασης και μία εικόνα βαθυμετριάς από ένα άγνωστο υποθαλάσσιο εδάφος. Τότε το μοντέλο θα ταξινομεί κάθε pixel, παράγοντας ως αποτέλεσμα μία εικόνα, που περιγράφει τις κατηγορίες του εδάφους.



(α') *backscatter* (β') *bathymeterry* (γ') *ground truth*

Σχήμα 4.1: *Εικόνες για την εκπαίδευση των μοντέλων*

Οι κατηγορίες του εδάφους που εντοπίζονται είναι οι εξής:

- Yellow Sand
- Sand and Algae
- Mud Sand and Corals

- Sand
- Gravel

Οι εικόνες οπισθοσκέδασης, βαθυμετρίας και οι ground truth εικόνες, είναι της μορφής GeoTIFF. Το GeoTIFF είναι ένα είδος εικόνας .tif το οποίο περιέχει επιπλέον γεωχωρικές πληροφορίες τις οποίες αναφέρει ως tags. Αυτά τα tags μπορούν να περιέχουν τις εξής πληροφορίες:

- Χωρική έκταση: Την περιοχή που καλύπτει αυτό το σύνολο δεδομένων.
- Σύστημα αναφοράς συντεταγμένων
- Ανάλυση: Τα δεδομένα είναι σε μορφή raster. Αυτό σημαίνει ότι αποτελείται από pixel. Η περιοχή στο έδαφος που καλύπτει κάθε εικονοστοιχείο είναι η χωρική του ανάλυση.
- Nodata: Χαρακτηρίζει τα pixels με άγνωστα χαρακτηριστικά
- Αριθμός επίπεδων που υπάρχουν στο αρχείο .tif

Στην παρούσα εργασία θα μελετηθεί η περιοχή Bedford Basin, όπου έχουν συλλεχθεί οι απαραίτητες εικόνες οπισθοσκέδασης και βαθυμετρίας και έχουν δημιουργηθεί οι αντίστοιχες ground truth εικόνες για τα έτη 2016, 2017, 2018.

4.1.1 Προεπεξεργασία των δεδομένων

Για την μετατροπή των δεδομένων και την δημιουργία των συνόλων εκπαίδευσης και επαλήθευσης χρειάστηκαν να γίνουν κάποιες ενέργειες. Από τις εικόνες οπισθοσκέδασης και ground truth εικόνες αφαιρέθηκαν όσα pixel είχαν το tag "nodata" που περιγράφει πως δεν υπάρχει γνώση για το pixel. Οπότε σαν δεδομένα εισόδου κρατήθηκαν τα pixel που βρίσκονται σε θέση όπου υπάρχει πληροφορία τόσο στις εικόνες οπισθοσκέδασης, βαθυμετρίας όσο και στην ground truth εικόνα.

Τέλος το δεδομένα χωρίζονται σε ένα σύνολο εκπαίδευσης και ένα επαλήθευσης με βάση ένα βέλτιστο ποσοστό διαχωρισμού που θα περιγραφεί ακολούθως.

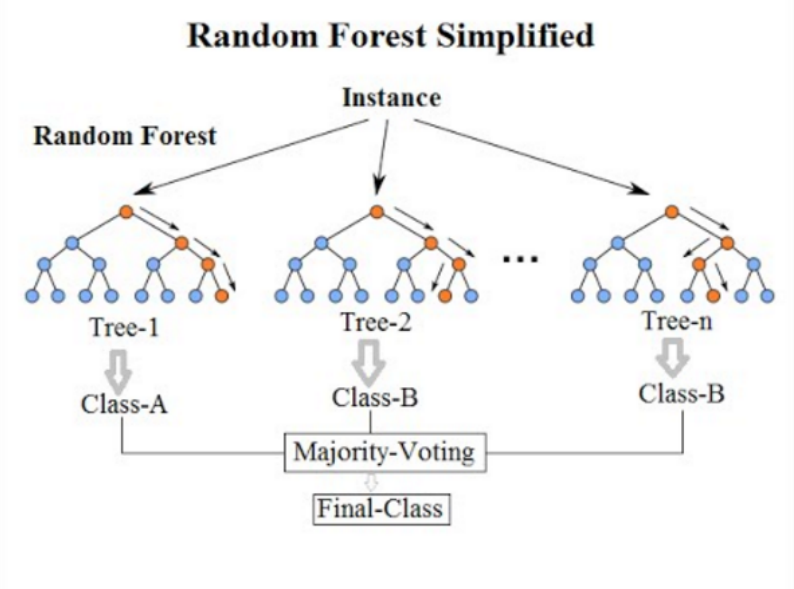
4.2 Αλγοριθμική Προσέγγιση

4.2.1 Εκπαίδευση του μοντέλου

Σε συνέχεια της προσέγγισης των Μερτίκας Π. και Καράντζαλου Κ. [5] για την δημιουργία του μοντέλου εξετάστηκαν δύο είδη είδη ταξινομητών, ο αλγόριθμος ταξινόμησης Rf (Random Forest) και ο αλγόριθμος Μηχανών διανυσμάτων υποστήριξης (SVM).

Αλγόριθμος RF

Ο αλγόριθμος RF προτάθηκε από τον Breiman (2001)[19] και είναι ένα είδος αλγορίθμου πρόβλεψης που βασίζεται σε δέντρα ταξινόμησης, σε παλινδρόμηση (CART) (L. I. Breiman, Friedman, Olshen, Stone, 1984) [20] και στην τεχνική bagging (Breiman, 1996) [21].

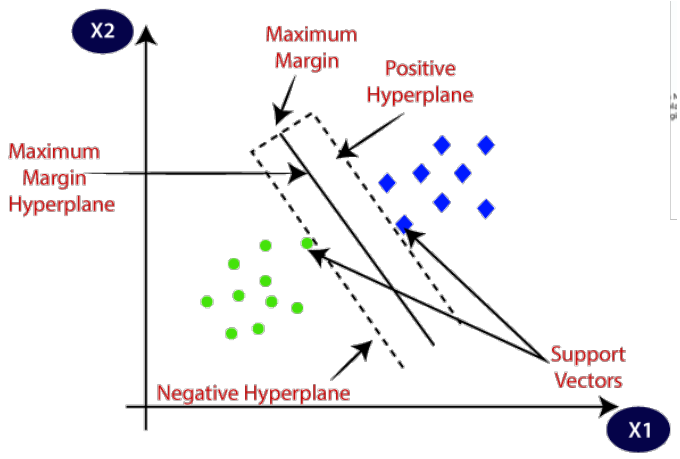


Σχήμα 4.2: *Αλγόριθμος ταξινόμησης Random Forest*

Τα τυχαία δάση λειτουργούν με βάση την αρχή ότι ένας μεγάλος αριθμός δέντρων θα έχουν καλύτερη απόδοση από ένα μόνο δέντρο στην εκπαίδευση ενός μοντέλου. Μερικά μεμονωμένα δέντρα μπορεί να είναι λάθος, αλλά εφόσον τα μεμονωμένα δέντρα δεν κάνουν εντελώς τυχαίες προβλέψεις, το άθροισμά τους θα σχηματίσει μια προσέγγιση των υποκείμενων δεδομένων. Αυτή την λογική ενσωματώνει το bagging. Το bagging είναι μία η αλγοριθμική τεχνική που χρησιμοποιείται στο σενάριο τυχαίων δασών. Εκπαιδεύει μεμονωμένα δέντρα απόφασης σε τυχαία δείγματα υποσυνόλων του συνόλου δεδομένων για να μειώσει τη συσχέτιση. Τα μεμονωμένα δέντρα αποφάσεων είναι επιρρεπή σε υπερβολική προσαρμογή και έχουν την τάση να μαθαίνουν τον θόρυβο στο σύνολο δεδομένων. Τα τυχαία δάση λαμβάνουν κατά μέσο όρο πολλά δέντρα, επομένως, εφόσον τα μεμονωμένα δέντρα αποφάσεων δεν συσχετίζονται, αυτή η στρατηγική μειώνει την υπερπροσαρμογή και την ευαισθησία στο θόρυβο στο σύνολο δεδομένων.

Μηχανές Διανυσμάτων Υποστήριξης (SVM)

Οι Μηχανές Διανυσμάτων Υποστήριξης (SVMs) είναι μία ομάδα αλγορίθμων επιτηρούμενης μάθησης που χρησιμοποιούνται για κατηγοριοποίηση και για προβλήματα παλινδρόμησης. Αναπτύχθηκαν για πρώτη φορά από τον Vapnik και τους συνεργάτες τους στο ATT Bell Labs το 1992 [22]. Απέσπασαν γρήγορα το ενδιαφέρον καθώς παρουσίασαν μεγάλη ικανότητα γενίκευσης σε σχέση με άλλες παραδοσιακές μεθόδους ταξινόμησης. Η βασική ιδέα της κατασκευής τους στηρίζεται στην αρχή ελαχιστοποίησης του κατασκευαστικού ρίσκου (SRM) που έχει αποδειχθεί πως υπερτερεί έναντι της παραδοσιακής ελαχιστοποίησης του εμπειρικού ρίσκου (ERM) στην οποία στηρίζονται τα νευρωνικά δίκτυα. Η κατηγοριοποίηση των δεδομένων στηρίζεται στην εύρεση ενός βέλτιστου υπερεπιπέδου hyperplane που διαχωρίζει τα δεδομένα δημιουργώντας το μέγιστο περιθώριο.



Σχήμα 4.3: *Μηχανές Διανυσμάτων Υποστήριξης (SVMs)*

Στην περίπτωση που ο γραμμικός διαχωρισμός είναι αδύνατος, γίνεται χρήση κατάλληλων απεικονίσεων που μεταφέρουν το σύνολο των δεδομένων σε μεγαλύτερη διάσταση ώστε να επιτευχθεί τελικά ο διαχωρισμός τους. Η ικανότητα γενίκευσης της χρήσης των SVM σε μη γραμμικά δεδομένα στηρίζεται στο τέχνασμα του πυρήνα (kernel trick). Κάθε μηχανή διανυσμάτων υποστήριξης είναι ένας δυαδικός ταξινομητής, έχει δηλαδή τη δυνατότητα κατηγοριοποίησης σε δύο κλάσεις. Εάν οι κλάσεις είναι περισσότερες, εφαρμόζονται διάφορες τεχνικές ώστε να αναλυθεί το πρόβλημα πολλών κλάσεων σε επιμέρους δυαδικά προβλήματα ταξινόμησης, χρησιμοποιώντας πολλαπλές μηχανές διανυσμάτων υποστήριξης.

4.2.2 Αποτίμηση μοντέλου

Αφού κατασκευαστεί το εκάστοτε μοντέλο από την διαδικασία εκπαίδευσης, επιλέγουμε κάποιες μετρικές ώστε να αξιολογήσουμε την ποιότητα/ικανότητα του ταξινομητή που υλοποιήθηκε. Οι μετρικές που επιλέχθηκαν είναι οι παρακάτω:

- Ακρίβεια (Accuracy): Το ποσοστό των σωστών προβλέψεων του μοντέλου
- Συνέπεια (Precision) : Το ποσοστό των θετικών ταυτοποιήσεων που ήταν πράγματι σωστό

- Ανάκληση (Recall): Το ποσοστό των πραγματικών θετικών που προσδιορίστηκε σωστά
- Confusion Matrix: Πίνακας που περιέχει μια σύνοψη των προβλέψεων, πότε έγιναν σωστά, πότε εσφαλμένα. Στην περίπτωση των εσφαλμένων υπάρχει πληροφορία και της πραγματικής κατηγορίας και της εσφαλμένα ταξινομημένης κατηγορίας. Ουσιαστικά περιέχει την πληροφορία όλων των παραπάνω μετρικών.
- Cohen's Kappa: Ο συντελεστής k Cohen είναι ένα στατιστικό μέτρο της συμφωνίας μεταξύ των αξιολογήσεων δύο βαθμολογητών όταν και οι δύο βαθμολογούν το ίδιο αντικείμενο. Λαμβάνει τιμές στο διάστημα [-1, 1], με την τιμή 1 να δηλώνει τέλεια συμφωνία. Οι χαμηλές αρνητικές τιμές (0 έως -0,10) μπορούν γενικά να ερμηνευθούν ως «καμία συμφωνία». Ένα μεγάλο αρνητικό κάπα αντιπροσωπεύει μεγάλη διαφωνία μεταξύ των αξιολογητών [23].

4.3 Αυτοματοποίηση με Kubeflow

4.3.1 Περιγραφή του pipeline

Η διαδικασία εκπαίδευσης που περιγράφηκε στην προηγούμενη παράγραφο υλοποιήθηκε σε kubeflow pipelines. Για την δημιουργία των pipelines το kubeflow παρέχει κάποιες δυνατότητες για την δημιουργία του εκάστοτε σταδίου (component) όπως περιγράφηκε στο κεφάλαιο 2. Συγκεκριμένα υλοποιήθηκαν και οι 2 εναλλακτικές για την δημιουργία των pipelines, τόσο με την δημιουργία docker image με τον εκτελέσιμο κώδικα, τόσο με χρήση μόνο της python και την δημιουργία ενός lightweigt component.

Αλγοριθμος 4.1: *Dockerfile for cpu*

```
FROM python:3.9-slim

RUN mkdir -p /opt/seabed
COPY requirements.txt /opt/seabed/
RUN pip install -r /opt/seabed/requirements.txt
COPY dtrian_seabed_classification.py /opt/seabed/
COPY dtrian_seabed_classification-multi-input.py /opt/seabed/
COPY libs/ /opt/seabed/libs/
COPY utils/ /opt/seabed/utils/
WORKDIR /opt/seabed/

RUN  chgrp -R 0 /opt/seabed \
  && chmod -R g+rwX /opt/seabed

ENTRYPOINT ["python","./dtrian_seabed_classification.py"]
```

Αλγοριθμος 4.2: *Dockerfile για χρήση GPU*

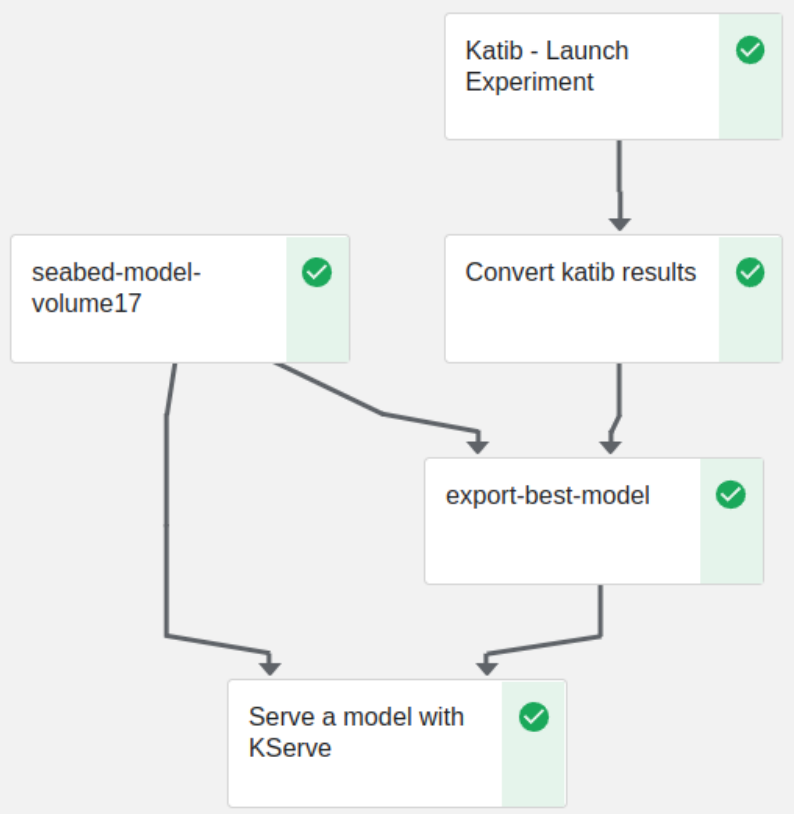
```
FROM pytorch/pytorch:1.12.1-cuda11.3-cudnn8-runtime
RUN mkdir -p /opt/seabed
RUN pip show torch
COPY requirements.txt /opt/seabed/
RUN pip install -r /opt/seabed/requirements.txt
COPY dtrian_seabed_classification.py /opt/seabed/
COPY dtrian_seabed_classification-multi-input.py /opt/seabed/
COPY libs/ /opt/seabed/libs/
COPY utils/ /opt/seabed/utils/
WORKDIR /opt/seabed/

RUN  chgrp -R 0 /opt/seabed \
  && chmod -R g+rwX /opt/seabed
ENTRYPOINT ["python","./dtrian_seabed_classification.py"]
```

Όπως φαίνεται τα παραπάνω αρχεία είναι πανομοιότυπα, με μόνη διαφορά το docker image που χρησιμοποιούν ως βάση. Η έκδοση για την GPU θα μπορεί να εκτελέσει τον κώδικα τόσο αποκλειστικά σε CPU όσο και συνδιαστικά με CPU, αντίθετα από την έκδοση σε CPU που δεν μπορεί να χρησιμοποιήσει την GPU για την εκτέλεση του κώδικα. Παρόλη αυτή την δυνατότητα της GPU έκδοσης, να μπορεί να εκτελεί σε οποιοδήποτε υλικό, δημιουργήθηκε και η έκδοση για CPU λόγω σημαντικά μικρότερου μεγέθους του παραγώμενου docker image.

Αλγοριθμος 4.3: *Απόσπασμα κώδικα δημιουργίας του Pipeline*

```
@dsl.pipeline(
    name="Seabed classification",
    description="Seabed classification with early stopping"
)

def seabed_thesis_final():
    katib_op = katib_experiment_launcher_op(
        experiment_name=experiment_name,
        experiment_namespace=experiment_namespace,
        experiment_spec=ApiClient().sanitize_for_serialization(experiment_spec),
        experiment_timeout_minutes=600,
        delete_finished_experiment=False)

    model_volume_op = dsl.VolumeOp(
        name="seabed-model-volume-final",
        resource_name="seabed-model-volume",
        size="1Gi",
        modes=dsl.VOLUME_MODE_RWO
    )
    convert_katib_results_op = components.func_to_container_op(convert_katib_results)(katib_op.output).
      after(katib_op)

    best_model_op = dsl.ContainerOp(
        name="export best model",
        image="localhost:32200/dtrian_seabed-classification:0.0.5-cpu",
        command=["sh","-c"],
        arguments=["python ./dtrian_seabed_classification.py --track --gt=GT_CAT.tif
        --dataset-dir=./samples/Bedford\\ Basin/2016 --model-export-path=/mnt/models/model.
      joblib {}"
        .format(str(convert_katib_results_op.output))],
        pvolumes={"/mnt/models/":model_volume_op.volume}
        ).after(convert_katib_results_op)

    kserve_op = kserve_launcher_op(
        action="apply",
        model_uri=f"pvc://{model_volume_op.outputs['name']}/",
        model_name="seabed-clf",
        namespace=experiment_namespace,
        framework="sklearn",
        watch_timeout="300",
    ).set_image_pull_policy('Always').after(best_model_op)

kfp_client.create_run_from_pipeline_func(seabed_thesis_final,  experiment_name='seabed', namespace=
      experiment_namespace, arguments={})
```

Κατά συνέπεια, δημιουργήθηκε το παρακάτω pipeline ως περιγραφή της ροής εργασίας που ακολουθήθηκε.

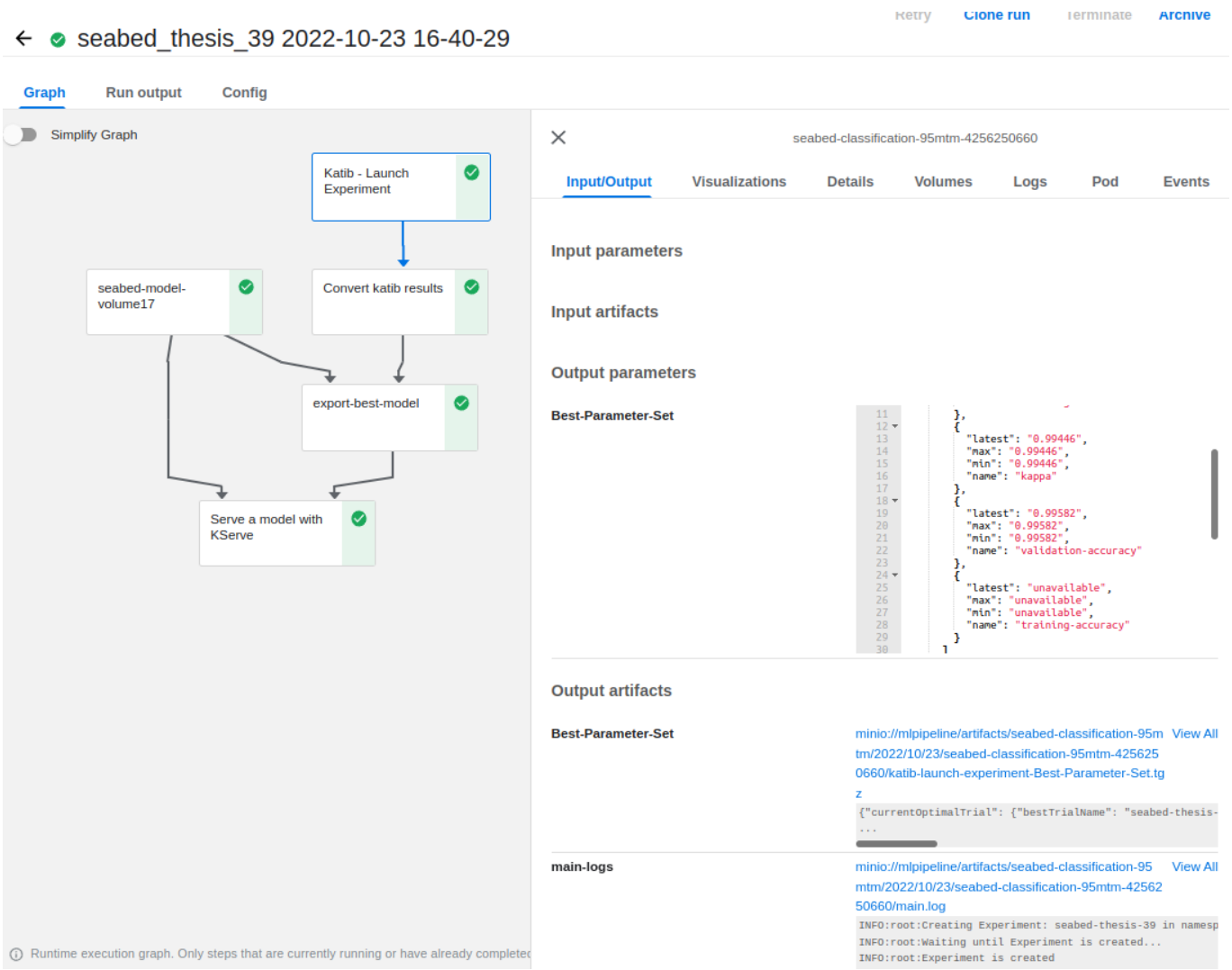


Σχήμα 4.4: *Pipeline για εκπαίδευση και προώθηση στην παραγωγή ταξινομητή υποθαλάσσιου εδάφους*

Όπως φαίνεται στην παραπάνω εικόνα το pipeline καλείται αρχικά να κάνει εύρεση βέλτιστων υπερπαραμέτρων. Έπειτα καλείται να χρησιμοποιήσει ένα μεταβατικό στάδιο για να μετρατρέψει τις βέλτιστες παραμέτρους σε είσοδο για το επόμενο στάδιο που είναι η εκπαίδευση του μοντέλου με τις βέλτιστες υπερπαραμέτρους και να δημιουργήσει το μοντέλο. Για την αποθήκευση του μοντέλου δημιουργήθηκε σε ένα στάδιο ένα pvc. Στο τελικό στάδιο το μοντέλο γίνεται διαθέσιμο για εξυπηρέτηση.

Κατά την εκτέλεση αυτού του pipeline δημιουργείται, αν δεν υπάρχει, το πείραμα στο οποίο θέλουμε να πραγματοποιηθεί αυτή η εκτέλεση. Η εκτέλεση είναι επίσης μία οντότητα του Kubeflow που δημιουργείται. Παρακάτω δίνεται η διεπαφή του Kubeflow για το πείραμα και την εκτέλεση που έγιναν. Η διεπαφή της εκτέλεσης παρέχει πληροφίες για όλα τα στάδια του pipeline, με τις εισόδους/εξόδους αυτών, τις θέσεις των artifacts αυτών, τα logs των pods που εκτέλεσαν τις διαδικασίες, κάποιες οπτικοποιήσεις που μπορεί να δημιουργήσει ο χρήστης και πολλά άλλα.

Σχήμα 4.5: *Σελίδα πειραμάτων*



Σχήμα 4.6: *Σελίδα εκτελέσεων*

Στην εκτέλεση μπορούν να υπάρχουν και οπτικοποιήσεις σε μορφή γραφημάτων, πινάκων ή ένα προσαρμοσμένο HTML. Συγκεκριμένα επιλέχθηκε να οπτικοποιηθεί ο πίνακας σύγχυσης για εποπτεία της απόδοσης του μοντέλου στην εκάστοτε κατηγορία ταξινόμησης.

Σχήμα 4.7: *Οπτικοποίησεις αποτελεσμάτων των πειραμάτων*

4.3.2 Στάδιο εύρεσης βέλτιστων υπερπαραμέτρων

Στο στάδιο αυτό, με χρήση του εργαλείου Katib, γίνονται δοκιμές εκπαίδευσης του ταξινομητή με πολλούς συνδυασμούς υπερπαραμέτρων μέχρι να βρεθούν οι βέλτιστες με βάση κάποια κριτήρια. Χρησιμοποιείται ένα ¨χτισμένο¨ docker image που υπάρχει στην Docker registry και είναι προσβάσιμη στο Kubernetes Cluster, το "localhost:32200/dtrian_seabed-classification:0.0.5-cpu". Παρακάτω δίνεται ο τρόπος με τον οποίο δίνονται οι παράμετροι που πρέπει να εξερευνήσει το Katib, οι δυνατές τιμές αυτών και τα κριτήρια επιλογής βέλτιστου συνόλου υπερπαραμέτρων και ολοκλήρωσης του πειράματος.

Αλγοριθμος 4.4: *Παραμετροποίηση του Katib 1/3*

```
# Trial count specification.
max_trial_count = 30
max_failed_trial_count = 3
parallel_trial_count = 3

# Objective specification.
objective = V1beta1ObjectiveSpec(
    type="maximize",
    goal=0.99,
    objective_metric_name="accuracy",
    additional_metric_names=[
        "kappa",
        "precision_1",
        "precision_2",
        "precision_3",
        "precision_4",
        "precision_5",
        "recall_1",
        "recall_2",
        "recall_3",
        "recall_4",
        "recall_5",
    ],
)

algorithm = V1beta1AlgorithmSpec(
    algorithm_name="bayesianoptimization",
    algorithm_settings=[V1beta1AlgorithmSetting(name="random_state", value="10")],
)

parameters = [
    V1beta1ParameterSpec(
        name="model",
        parameter_type="categorical",
        feasible_space=V1beta1FeasibleSpace(list=["svm", "rf"]),
    ),
    V1beta1ParameterSpec(
        name="perc",
        parameter_type="categorical",
        feasible_space=V1beta1FeasibleSpace(list=["0.3", "0.5", "0.7"]),
    ),
    V1beta1ParameterSpec(
        name="embedding_sigma",
        parameter_type="double",
        feasible_space=V1beta1FeasibleSpace(min="0.000001", max="10.0"),
    ),
    V1beta1ParameterSpec(
        name="embedding_dim",
        parameter_type="categorical",
        feasible_space=V1beta1FeasibleSpace(list=["0", "2", "3", "6", "7"]),
    ),
]
# Early Stopping specification.
early_stopping = V1beta1EarlyStoppingSpec(
    algorithm_name="medianstop",
    algorithm_settings=[
        V1beta1EarlyStoppingSetting(name="min_trials_required", value="2")
    ],
)
```

Αλγοριθμος 4.5: *Παραμετροποίηση του Katib 2/3*

```
# JSON template specification for the Trial's Worker Kubernetes Job.
trial_spec = {
    "apiVersion": "batch/v1",
    "kind": "Job",
    "spec": {
        "template": {
            "metadata": {"annotations": {"sidecar.istio.io/inject": "false"}},
            "spec": {
                "containers": [
                    {
                        "name": "training-container",
                        "image": "localhost:32200/dtrian_seabed-classification:0.0.5-cpu",
                        "command": [
                            "python",
                            "./dtrian_seabed_classification.py",
                            "--model=${trialParameters.model}",
                            "--perc=${trialParameters.perc}",
                            "--gt=GT_CAT.tif",
                            "--dataset-dir=./samples/Bedford\\ Basin/2016",
                            "--embedding-sigma=${trialParameters.embeddingSigma}",
                            "--embedding-dim=${trialParameters.embeddingDim}",
                        ],
                    }
                ],
                "restartPolicy": "Never",
            },
        }
    },
}
# Configure parameters for the Trial template.
trial_template = V1beta1TrialTemplate(
    retain=True,
    primary_container_name="training-container",
    trial_parameters=[
        V1beta1TrialParameterSpec(
            name="model", description="Model (svm, rf)", reference="model"
        ),
        V1beta1TrialParameterSpec(
            name="perc", description="Percentage of training set", reference="perc"
        ),
        V1beta1TrialParameterSpec(
            name="embeddingSigma",
            description="Positional embedding sigma",
            reference="embedding-sigma",
        ),
        V1beta1TrialParameterSpec(
            name="embeddingDim",
            description="Specify positional embedding dimension",
            reference="embedding-dim",
        ),
    ],
    trial_spec=trial_spec,
)
```

Αλγοριθμος 4.6: *Παραμετροποίηση του Katib 3/3*

```
experiment_spec = V1beta1ExperimentSpec(
    max_trial_count=max_trial_count,
    max_failed_trial_count=max_failed_trial_count,
    parallel_trial_count=parallel_trial_count,
    objective=objective,
    algorithm=algorithm,
    early_stopping=early_stopping,
    parameters=parameters,
    trial_template=trial_template,
)
katib_op = katib_experiment_launcher_op(
    experiment_name=experiment_name,
    experiment_namespace=experiment_namespace,
    experiment_spec=ApiClient().sanitize_for_serialization(experiment_spec),
    experiment_timeout_minutes=600,
    delete_finished_experiment=False,
)
```

Όπως φαίνεται στον παραπάνω κώδικα για την αναζήτηση των βέλτιστων υπερπαραμέτρων χρησιμοποιείται Μπεϋζιανή βελτιστοποίηση. Η εκπαίδευση μοντέλων είναι μια δαπανηρή διαδικασία και κάθε φορά που πρέπει να αξιολογηθεί ένα διάνυσμα υπερπαραμέτρων, πρέπει να εκτελείται αυτή η διαδικασία. Αυτό καθιστά την αναζήτηση πλέγματος πολύ ακριβή καθώς είναι εκθετική ως προς τον αριθμό των υπερπαραμέτρων. Η τυχαία αναζήτηση μπορεί επίσης να χρειάζεται πολλές επαναλήψεις για να φτάσει σε ένα καλό διάνυσμα υπερπαραμέτρων, καθώς δοκιμάζει τυχαία διαφορετικές επιλογές. Προτού διαδοθεί ευρέως ο αυτόματος συντονισμός υπερπαραμέτρων, ο κοινός μηχανισμός για την εύρεση ενός καλού συνόλου υπερπαραμέτρων ήταν χειροκίνητη επιλογή, γνωστή ως Grad Student Descent. Η ανθρώπινη λογική ακολουθεί συχνά ένα Μπεϋζιανό μοντέλο, όπου δοκιμάζεται κάτι και στη συνέχεια επιλέγεται επαναληπτικά αυτό που πιστεύεται ότι είναι το επόμενο καλό σύνολο τιμών προς δοκιμή. Τα συστήματα στον πραγματικό κόσμο συχνά ταιριάζουν σε μια κατανομή πιθανοτήτων, όπως μια κανονική κατανομή. Η Μπεϋζιανή βελτιστοποίηση μοντελοποιεί την απόδοση του διανύσματος υπερπαραμέτρου ως κατανομή, συχνά μια διαδικασία Gauss. Στη συνέχεια γίνεται προσπάθεια να βελτιστοποιηθεί η απόδοση αυτής της λειτουργίας.

Αυτό το στάδιο παράγει ως αποτέλεσμα το σύνολο των βέλτιστων υπερπαραμέτρων, καθώς επίσης και τα αποτελέσματα όλων των δοκιμών με μία περιεκτική οπτικοποίηση αυτών. Η οπτικοποίηση θα δοθεί παρακάτω, στην παράγραφο των αποτελεσμάτων.

4.3.3 Βοηθητικά στάδια

Δημιουργία persistent volume

Στην παράγραφο 2.2.4 αναφέρθηκε πως το Kserve υποστηρίζει ως χώρους αποθήκευσης των μοντέλων τους Azure Blob, S3, GCS, PVC, URI. Στα πλαίσια αυτής της εργασίας χρησιμοποιήθηκε PVC, οπότε ήταν και απαραίτητη η δημιουργία του. Σε άλλη περίπτωση όπου υπήρχε κάποιος από τους παραπάνω επί πληρωμή χώρους αποθήκευσης δεν θα χρειαζόταν η δημιουργία.

Μετατροπή αποτελεσμάτων του Katib

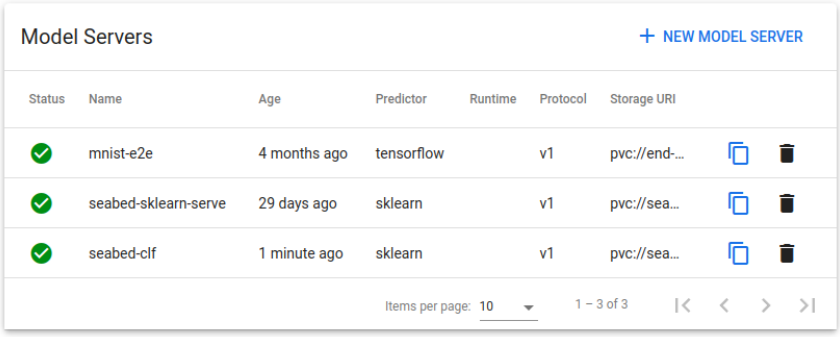
Το στάδιο αυτό μετατρέπει την έξοδο του προηγούμενου σταδίου, της εύρεσης των βέλτιστων υπερπαραμέτρων, στη μορφή που δέχεται το επόμενο στάδιο της εκπαίδευσης του βέλτιστου ταξινομητή.

4.3.4 Δημιουργία βέλτιστου μοντέλου

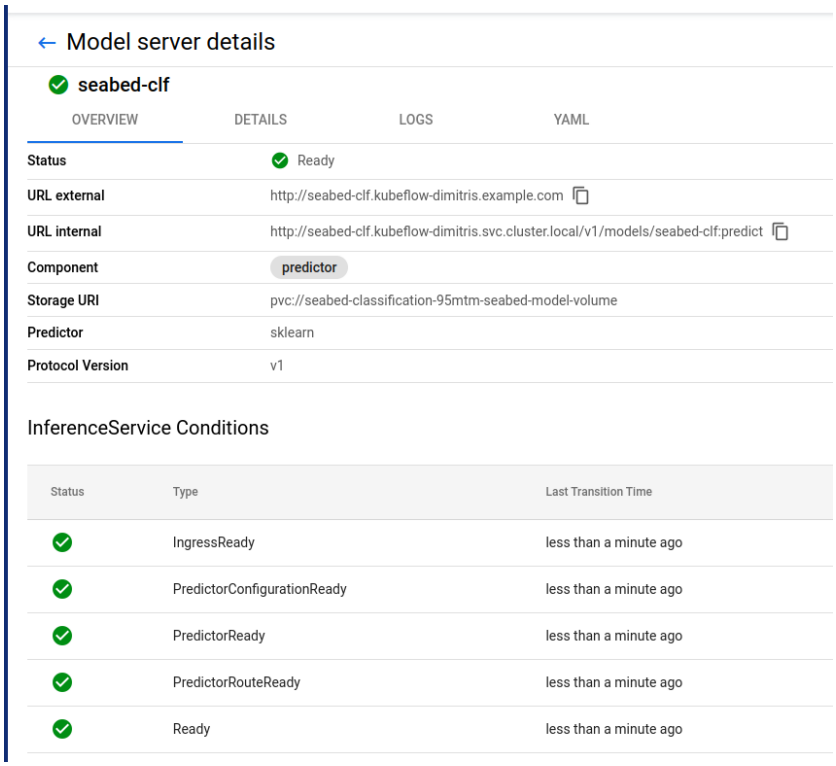
Στο στάδιο αυτό εκπαιδεύεται το μοντέλο με βάση τις βέλτιστες υπερπαραμέτρους του μοντέλου. Σημειώνεται πως θα μπορούσε κατά την διάρκεια της εξερεύνησης των υπερπαραμέτρων να αποθηκεύονται τα μοντέλα, δεδομένου ότι γίνεται η διαδικασία της εκπαίδευσης για κάθε σύνολο υπερπαραμέτρων. Αν όμως γίνοταν αυτό, σε περιπτώσεις όπου υπάρχει μεγάλος αριθμός δοκιμών, θα μπορούσε ο αριθμός των αποθηκευμένων μοντέλων και ο χώρος τον οποίο καταλαμβάνουν να είναι μη διαχειρίσιμα.

4.3.5 Προώθηση του μοντέλου για εξυπηρέτηση

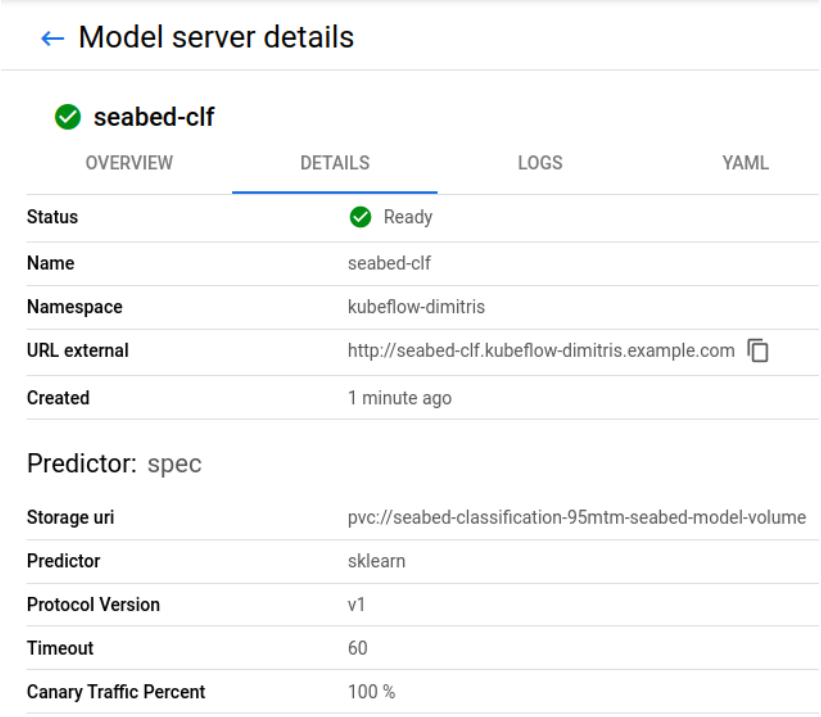
Με χρήση του Kserve το βέλτιστο μοντέλο που εκπαιδεύτηκε στο προηγούμενο στάδιο γίνεται διαθέσιμο στην παραγωγή. Παρακάτω απεικονίζεται η διεπαφή του KServe που προσφέρει στο κεντρικό dashboard του Kubeflow.



Σχήμα 4.8: *Model servers*



Σχήμα 4.9: *Model server του ταξινομητή*



Σχήμα 4.10: *Λεπτομέρειες του Model server του ταξινομητή*

Από τις παραπάνω εικόνες παρέχεται η εποπτεία των ταξινομητών που είναι στην παραγωγή. Μερικές από τις σημαντικές πληροφορίες που δίνονται είναι τα εσωτερικά και εξωτερικά (του kubernetes cluster) urls, η κατάσταση των services, το ποσοστό της κίνησης που εξυπηρετούν στην παραγωγή, σύμφωνα με το canary rollout.

4.3.6 Επανεκπαίδευση

Για την επανεκπαίδευση ενός μοντέλου που βρίσκεται στην παραγωγή, όπως προαναφέρθηκε, το kubeflow δίνει την δυνατότητα να εκτελείται προγραμματισμένα κάποιο pipeline ανά τακτά χρονικά διαστήματα ή όπως το ορίζει ο χρήστης. Καθώς επίσης, χωρίς να συνιστάται, ο χρήστης μπορεί να εκτελεί χειροκίνητα το pipeline. Σαν γενική εφαρμογή, σε κάποιο στάδιο του pipeline συλλέγονται τα δεδομένα. Τα δεδομένα αυτά θα είναι τα πλέον ενημερωμένα και πλήρη την εκάστοτε χρονική στιγμή που γίνεται μία επανεκπαίδευση. Επομένως, δεν θα αλλάξει ο αλγόριθμος εκπαίδευσης του μοντέλου, ούτε το pipeline για την εκπαίδευση του μοντέλου. Σε αυτή την λογική θεωρείται δεδομένο ότι ο αλγόριθμος της εκπαίδευσης του μοντέλου είναι αποδοτικός και απλά επανεκπαιδεύεται το μοντέλο με ενημερωμένα δεδομένα, καθώς επίσης βρίσκονται οι καινούριες βέλτιστοι υπερπαράμετροι.

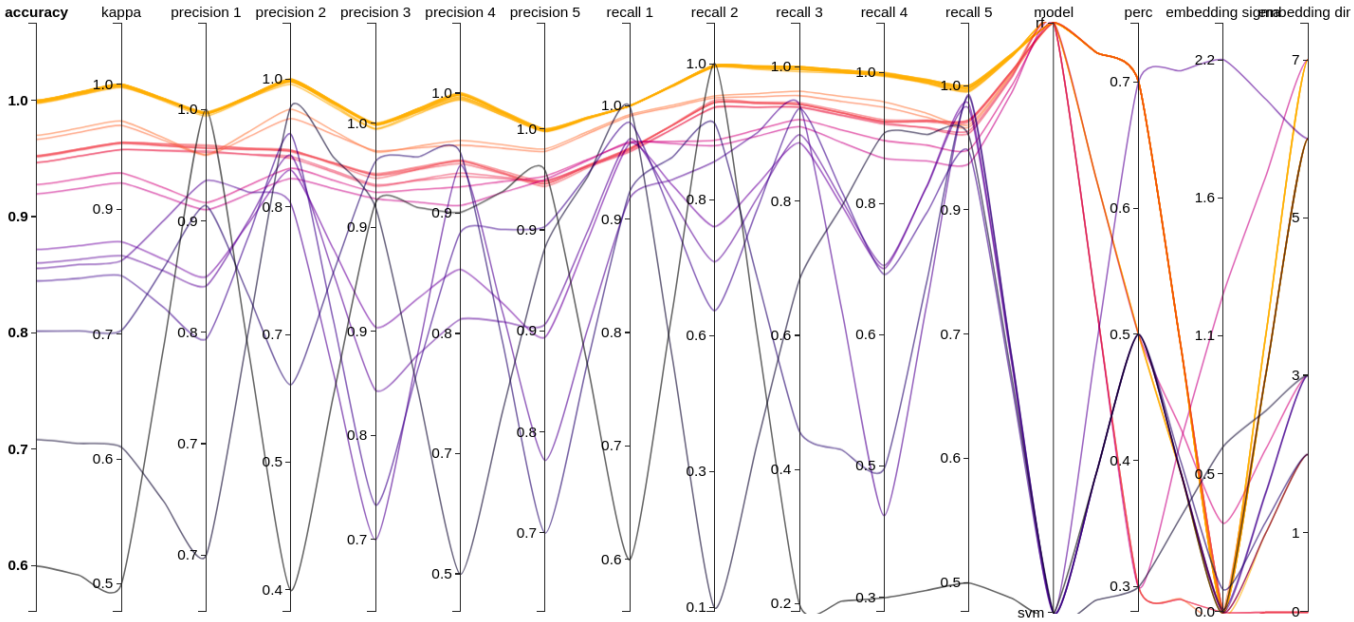
Στην περίπτωση που εξετάζει η παρούσα εργασία, υπάρχουν δεδομένα για την υποθαλάσσια περιοχή Bedford Basin για τα έτη 2016, 2017 και 2018. Θα προσωμοιωθεί η επανεκπαίδευση σε τρεις εκτελέσεις, μία για κάθε έτος. Στην πρώτη εκτέλεση, που θα γίνοταν θεωρητικά το 2016, δίνονται ως είσοδος στο pipeline για εκπαίδευση τα δεδομένα του έτους 2016. Στην δεύτερη εκπαίδευση, του έτους 2017, δίνονται ως είσοδος τα δεδομένα των 2016 και 2017. Τέλος στην τρίτη, του έτους 2018, δίνονται ως είσοδος τα δεδομένα των 2016, 2017, 2018.

4.4 Αποτελέσματα

4.4.1 Σχολιασμός εκπαίδευσης και συντονισμού παραμέτρων

Όπως περιγράφηκε στις παραπάνω παραγράφους αυτού του κεφαλαίου, για την εκπαίδευση του μοντέλου διερευνήθηκαν οι αλγόριθμοι ταξινόμησης RF, SVM και ένα σύνολο υπερπαραμέτρων. Επίσης αναφέρθηκαν οι μετρικές με τις οποίες εκτιμάται η απόδοση του μοντέλου.

Χρονικά προηγείται η εκπαίδευση του μοντέλου με χρήση των δεδομένων του έτους 2016. Όπου γίνεται και η εύρεση των βέλτιστων υπερπαραμέτρων του μοντέλου. Από το katib προκύπτει το παρακάτω διάγραμμα με τον αντίστοιχο πίνακα.



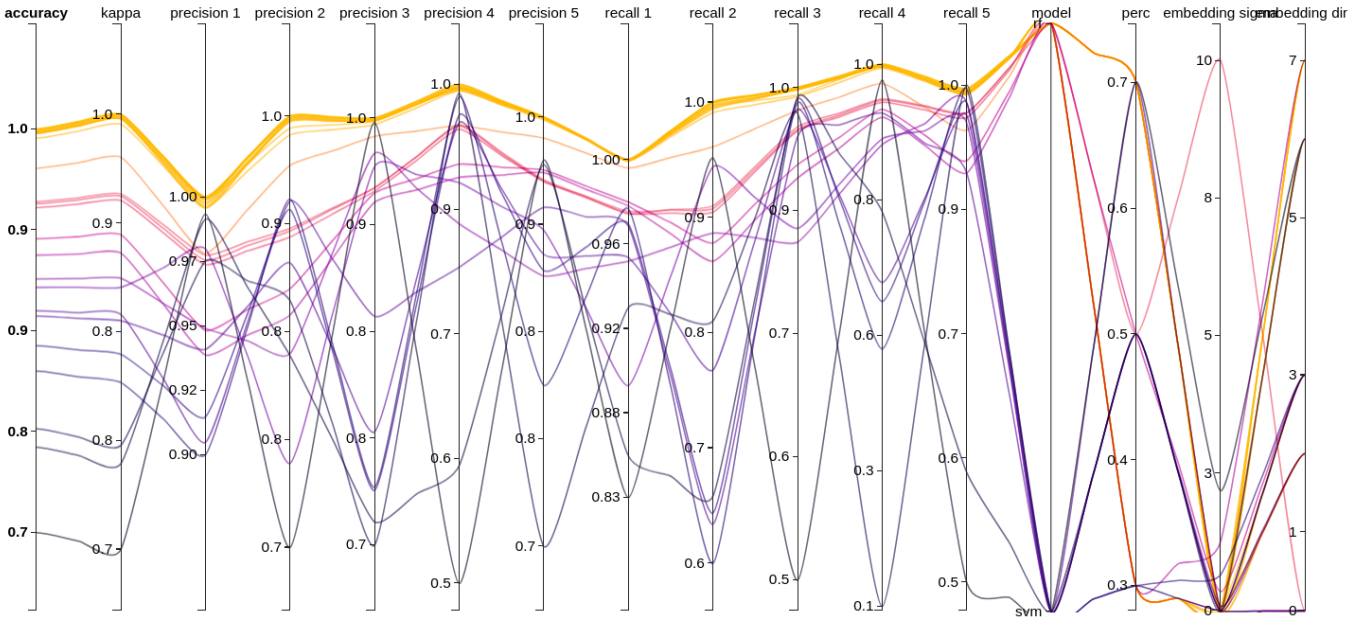
Σχήμα 4.11: *Αναζήτηση βέλτιστων υπερπαραμέτρων 2016 (Katib)*

Status	Accuracy	Kappa	Precision 1	Precision 2	Precision 3	Precision 4	Precision 5	Recall 1	Recall 2	Recall 3	Recall 4	Recall 5	Model	Perc	Embedding sigma	Embedding dir
Succeeded	0.99796	0.9974	0.99815	0.99702	0.99813	0.99691	0.9997	0.99969	0.99758	0.99816	0.99668	0.99579	rf	0.7	1.79734e-5	7
Succeeded	0.97306	0.96554	0.97668	0.96128	0.98521	0.95134	0.98447	0.99544	0.95933	0.97448	0.96482	0.95031	rf	0.3	7.00753e-3	6
Succeeded	0.86093	0.8207	0.91391	0.92941	0.73521	0.96297	0.88991	0.96811	0.64809	0.98613	0.60768	0.98028	svm	0.3	0.65418	3
Succeeded	0.88297	0.85003	0.90409	0.93559	0.85817	0.79646	0.93495	0.96691	0.63868	0.94697	0.91652	0.9041	svm	0.5	8.92547e-2	2
Succeeded	0.90313	0.87655	0.94798	0.82543	0.97341	0.84401	0.88609	0.94985	0.88602	0.83645	0.86477	0.97987	svm	0.5	6.9168e-6	2
Succeeded	0.80861	0.75375	0.97381	0.86384	0.69652	0.98707	0.69319	0.92711	0.81134	0.97126	0.10737	0.99937	svm	0.5	4.46636e-6	6
Succeeded	0.92857	0.9086	0.94716	0.87236	0.94521	0.9092	0.96161	0.97875	0.87766	0.91755	0.9226	0.91743	rf	0.5	0.3514	3
Succeeded	0.91812	0.89519	0.93772	0.85341	0.93873	0.89475	0.9598	0.97684	0.86214	0.9035	0.90973	0.90425	rf	0.3	1.25708	7
Succeeded	0.99757	0.99689	0.99773	0.99732	0.99845	0.99428	0.99949	0.99957	0.9968	0.9988	0.99636	0.99349	rf	0.7	7.28999e-6	6

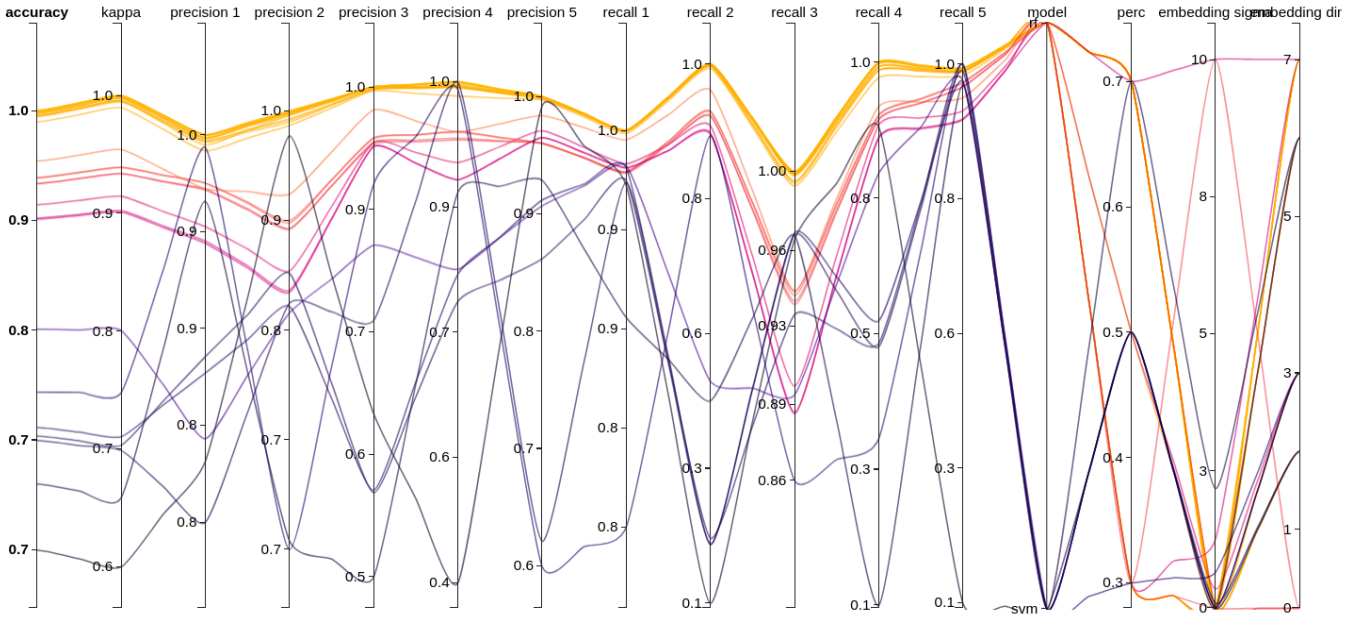
Σχήμα 4.12: *Πίνακας δοκιμών αναζήτησης βέλτιστων υπερπαραμέτρων 2016 (Katib)*

Οι υπερπαράμετροι που διερευνεί το Katib είναι το ο αλγόριθμος ταξινόμησης, με επιλογές τον RF και τα SVM, το embedding sigma και το embedding dimensions. Στην εικόνα 4.11 αναπαριστώνται γραφικά τα αποτελέσματα όλων των δοκιμών που διενεργήθηκαν, όπως προέκυψαν από το Katib. Με ανοιχτόχρωμα χρώμα φαίνονται οι καμπύλες όπου το εκπαιδευόμενο μοντέλο απέδωσε καλύτερα με κριτήριο την μέγιστη ακρίβεια. Φαίνεται ξεκάθαρα ότι ο αλγόριθμος RF υπερτερεί έναντι των SVM για όλους τους συνδιασμούς παραμέτρων. Επίσης φαίνεται ότι ο ταξινομητής μπορεί να προβλέπει ικανοποιητικά και στην περίπτωση που εκπαιδευτεί με το 30% των δεδομένων και να επαληθευθεί στο υπόλοιπο 70%. Συχνά είναι επιθυμητό ένα μικρότερο ποσοστό των δεδομένων στο σύνολο εκπαίδευσης, για την αποφυγή υπερπροσαρμογής του μοντέλου στα δεδομένα. Στην εικόνα 4.12 με κίτρινο χρώμα φαίνεται η δοκιμή με το βέλτιστο αποτέλεσμα, κατά το Katib, και κατά συνέπεια το βέλτιστο σύνολο υπερπαραμέτρων. Όμως το katib μπορεί να ελέγχει μία μετρική ως κριτήριο για την απόδοση του μοντέλου. Οπότε, αν η ακρίβεια μιας δοκιμής με το 70% των δεδομένων ως σύνολο εκπαίδευσης έχει ως αποτέλεσμα ένα μοντέλο με ακρίβεια ελάχιστα μεγαλύτερη, θα προτιμηθεί ενός μοντέλου που εκπαιδεύτηκε με σύνολο εκπαίδευσης το 50% δεδομένων. Στην πράξη μπορεί να μην επιθυμητό αυτό.

Παρόμοια ικανοποιητικά είναι τα αποτελέσματα για την αναζήτηση βέλτιστων υπερπαραμέτρων και τις αντίστοιχες εκπαιδεύσεις των επόμενων χρόνων 2017, 2018.



Σχήμα 4.13: *Αναζήτηση βέλτιστων υπερπαραμέτρων 2017 (Katib)*



Σχήμα 4.14: *Αναζήτηση βέλτιστων υπερπαραμέτρων 2018 (Katib)*

4.4.2 Σχολιασμός απόδοσης των ταξινομητών

Επομένως, έχουν προκύψει τρεις ταξινομητές για τα τρία έτη. Μελετήθηκε η απόδοση των ταξινομητών αυτών έναντι των δεδομένων από κάθε έτος.

	accuracy	**kappa**	**prec_1**	**prec_2**	**prec_3**	**prec_4**	**prec_5**	**recall_1**	**recall_2**	**recall_3**	**recall_4**	**recall_5**
model '16 at 2016	0.99983	0.99978	0.99988	0.99962	0.99984	0.99994	1.0	1.0	1.0	1.0	0.99893	0.9999
model '16 at 2017	0.60898	0.49144	0.65907	0.24476	0.55603	0.87916	0.99995	0.99949	0.255690	0.53317	0.09311	0.96927
model '16 at 2018	0.72483	0.58081	0.69983	0.71013	0.04479	0.93878	0.99195	0.94077	0.86175	0.00108	0.00106	0.75573
model '17 at 2016	0.99978	0.99971	0.99994	0.99939	0.99988	0.99978	0.99999	0.99998	0.9999	0.99994	0.9985	0.99995
model '17 at 2017	0.99968	0.9996	0.99907	0.99987	0.99987	0.99997	1.0	0.99996	0.99866	0.99993	0.99976	0.9999
model '17 at 2018	0.6699	0.48917	0.61697	0.71738	0.04586	0.99714	0.98888	0.99937	0.62684	0.00148	0.02406	0.69297
model '18 at 2016	0.99969	0.99959	0.9999	0.99935	0.9999	0.99922	0.99997	0.99982	0.99979	0.99994	0.99845	0.99999
model '18 at 2017	0.99967	0.99959	0.99911	0.99986	0.9998	0.99997	1.0	0.99993	0.9987	0.99997	0.99965	0.99989
model '18 at 2018	0.99686	0.99559	0.99634	0.99729	0.99662	0.99425	0.99972	1.0	0.99983	0.99783	0.97006	0.9947

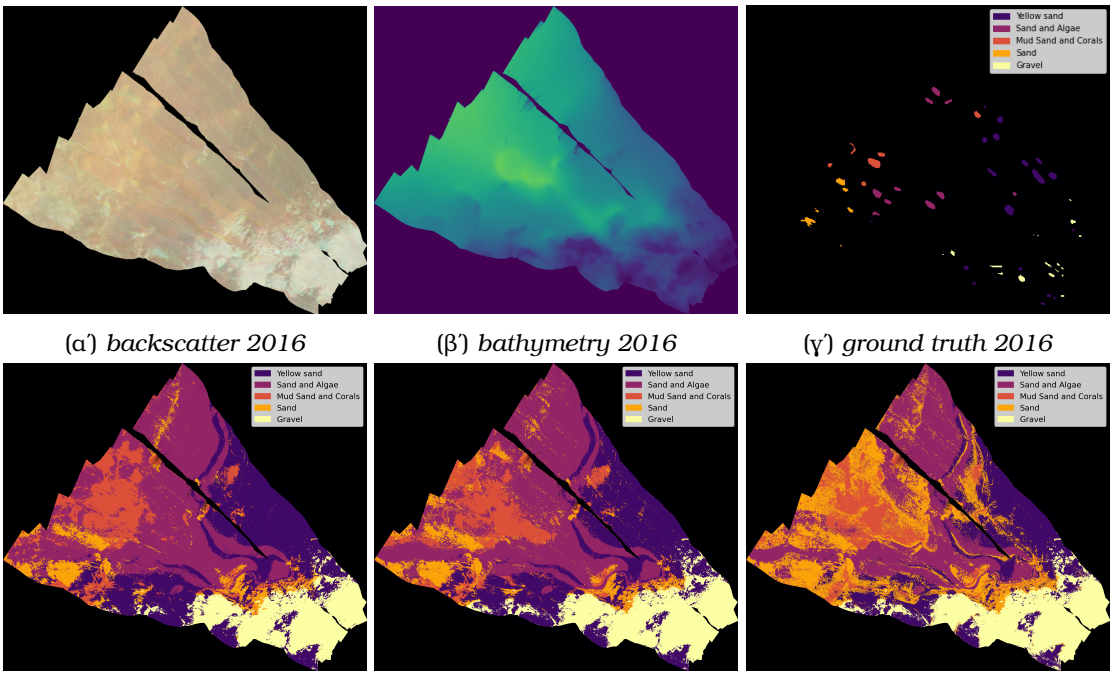
Table 4.1: *Αποτίμηση των μοντέλων των 2016, 2017, 2018*

Το μοντέλου του 2016 όπως φαίνεται στον πίνακα 4.1, παρουσιάζει εξαιρετική ακρίβεια και συνέπεια και ανάκληση σε κάθε κατηγορία όταν εκτιμάται έναντι των δεδομένων του 2016, με τα οποία εκπαιδεύτηκε. Όμως είναι επιθυμητό το μοντέλο να μπορεί να γενικεύει στον πληθυσμό και να προβλέπει σωστά σε άγνωστα δεδομένα. Εκτιμώντας το μοντέλο αυτό ως προς τα άγνωστα δεδομένα του 2017, όπως είναι λογικό δεν αποδίδει ισάξια καλά. Φαίνεται πως είναι ικανό να ταξινομεί ικανοποιητικά την κατηγορία 5 (gravel) με καλή συνέπεια και ανάκληση, την ξεχωρίζει έναντι των άλλων. Την κατηγορία 1 (yellow sand) φαίνεται ότι την εντόπισε σε όλα τα σημεία που υπήρχε, έχοντας καλή ανάκληση, όμως την εντόπισε λανθασμένα σε πολλά άλλα σημεία, έχοντας κακή συνέπεια. Την κατηγορία 4 (sand), δεν την εντόπισε επαρκώς σε αριθμό των σημείων που εμφανίζεται, αλλά είχε καλή συνέπεια όπου ταξινομήθηκε. Στις κατηγορίες 2 (sand and algae), 3 (mud sand and corals) το μοντέλο έδειξε αδυναμία να ταξινομήσει σωστά. Καλύτερη απόδοση συνολικά φαίνεται να έχει το μοντέλο όταν εκτιμάται έναντι των δεδομένων του 2018. Αυτό μπορεί να σημαίνει ότι τα δεδομένα των ετών 2016, 2018 μπορεί να παρουσιάζουν μια ομοιότητα έναντι του 2017 ή πιθανών να υπάρχει κάποιος θόρυβος, ίσως μεγαλύτερος, στα δεδομένα του 2017. Όποια και αν είναι η περίπτωση φαίνεται πως και στην περίπτωση των δεδομένων του 2018, αναγνωρίζεται ικανοποιητικά η κατηγορία 5, αλλά με μικρότερη ανάκληση την εκτίμηση στα δεδομένα του 2017. Η κατηγορία 2 αναγνωρίζεται αρκετά καλύτερα, με σημαντικά καλύτερη ανάκληση και συνέπεια από την εκτίμηση στα δεδομένα του 2017. Αντίθετα, η κατηγορία 3 ταξινομείται καλύτερα, όχι επαρκώς, στα δεδομένα του έτους 2017 έναντι στα δεδομένα του 18.

Το μοντέλου του 2017, εκπαιδεύτηκε με τα δεδομένα των ετών 2016-2017. Στα δεδομένα των ετών 2016, 2017 παρουσιάζει εξαιρετική απόδοση, όπως φαίνεται στον πίνακα 4.1. Στα άγνωστα δεδομένα του 2018, το μοντέλο ταξινομεί λιγότερο αποδοτικά από το μοντέλο του 2016. Όπως αναφέρθηκε στον σχολιασμό του μοντέλου του 2016, φαίνεται πως τα δεδομένα 2016, 2018 εμφανίζουν μεγαλύτερη συσχέτιση έναντι στο 2017. Επομένως στο μοντέλο του 2017 στην εκπαίδευση του, προσαρμόστηκε προς τα δεδομένα του 2018. Επομένως είναι λογικό να μειωθεί η απόδοση του μοντέλου στα δεδομένα του 2018.

Τέλος, το μοντέλο του 2018 εκπαιδεύτηκε σε στα δεδομένα όλων των ετών 2016, 2017, 2018. Όπως φαίνεται στον πίνακα 4.1, το μοντέλο αποδίδει εξαιρετικά στα δεδομένα όλων των ετών.

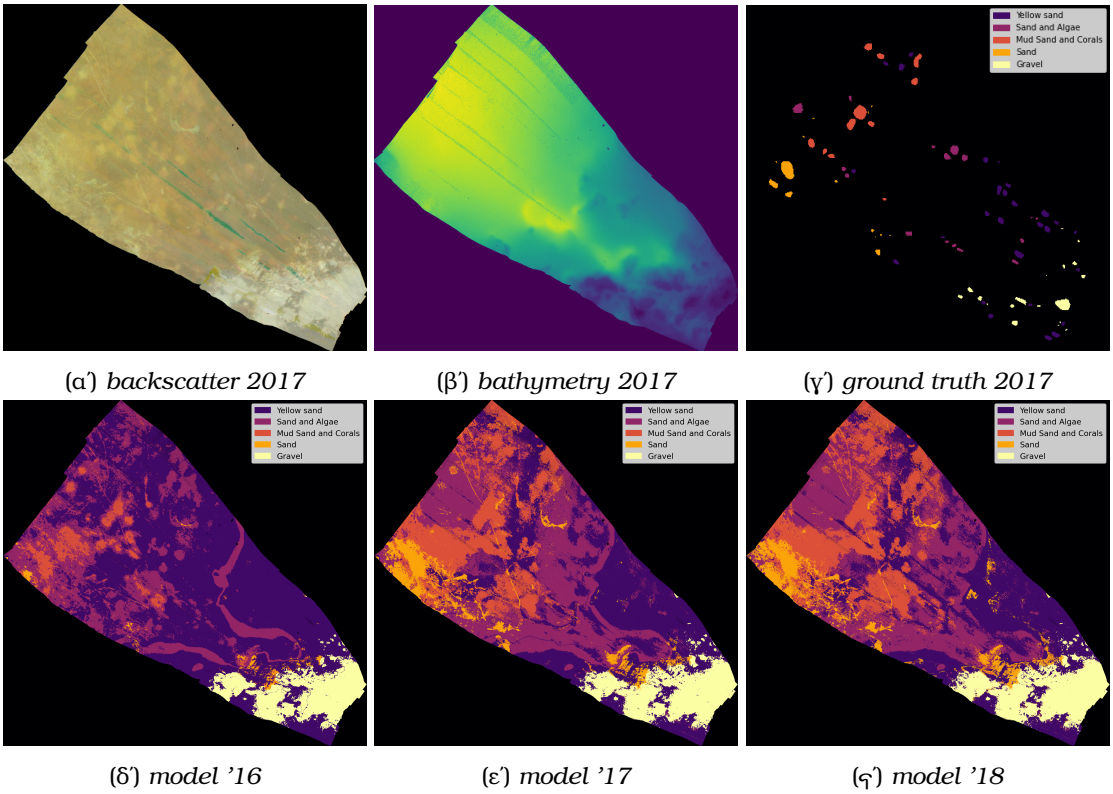
Παρακάτω δίνεται γραφικά πως ταξινομούνται τα δεδομένα κάθε έτους από κάθε ταξινομητή. Δίνεται στις πρώτες εικόνες των συμπλεγμάτων οι εικόνες αναφοράς των δεδομένων και στην συνέχεια οι εικόνες πρόβλεψης από το εκάστοτε μοντέλο.



(α') *backscatter 2016* (β') *bathymetry 2016* (γ') *ground truth 2016*

(δ') *model '16* (ε') *model '17* (ϛ') *model '18*

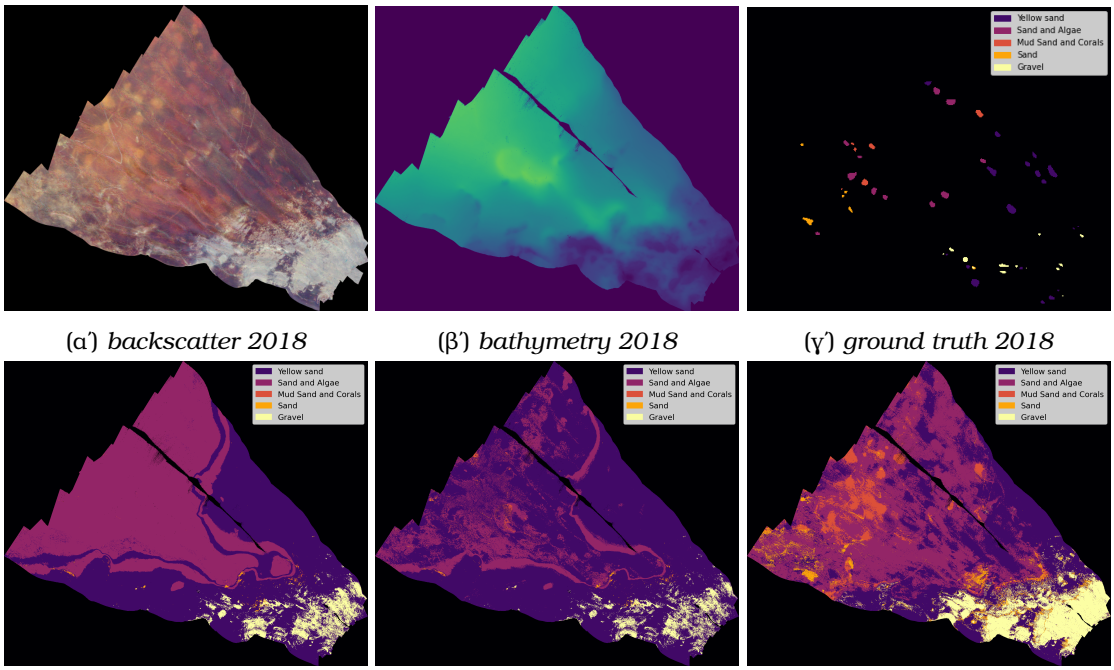
Σχήμα 4.15: *Ταξινόμηση δεδομένων Bedford 2016*



(α') *backscatter 2017* (β') *bathymetry 2017* (γ') *ground truth 2017*

(δ') *model '16* (ε') *model '17* (ϛ') *model '18*

Σχήμα 4.16: *Ταξινόμηση δεδομένων Bedford 2017*



(α') *backscatter 2018* (β') *bathymetry 2018* (γ') *ground truth 2018*

(δ') *model '16* (ε') *model '17* (ϛ') *model '18*

Σχήμα 4.17: *Ταξινόμηση δεδομένων Bedford 2018*

Κεφάλαιο 5

Συμπεράσματα και μελλοντικές ενέργειες

Στην παρούσα εργασία έγινε μία προσέγγιση σχεδιασμού και υλοποίησης ενός συστήματος μηχανικής μάθησης στην παραγωγή. Σκοπός και επίκεντρο της προσέγγισης αυτής, ήταν η αυτοματοποίηση των διαδικασιών που είναι αναγκαία σε τέτοιου είδους συστήματα. Οι διαδιακασίες αυτές (MLops) αφορούν την συνεχή παρακολούθηση, εκπαίδευση και διαθεσιμότητα των μοντέλων που είναι στην παραγωγή. Επιλέχθηκε και χρησιμοποιήθηκε η πλατφόρμα Kubeflow, για να δημιουργηθούν οι διαδικασίες MLOps. Ταυτόχρονα, δόθηκε και μια εικόνα, των ομάδων εργασίας που χρειάζονται σε τέτοια συστήματα και πως μεταβάλλεται με την εξέλιξη των τεχνολογιών μηχανικής μάθησης. Περιγράφηκε πως με την χρήση των διαδικασιών MLOps, διαμορφώνεται ο ρόλος τόσο των επιστημόνων δεδομένων, όσο και των μηχανικών μηχανικής μάθησης. Η διαμόρφωση αυτή ευνοεί την συνεργασία των δύο αυτών ομάδων, όντας πια λιγότερο εξαρτημένες. Πρακτικά, υλοποιήθηκε μια υποδομή που προσομοιώνει τις συνθήκες στην παραγωγή. Καθώς επίσης, έγινε εφαρμογή σε ένα ζήτημα μηχανικής μάθησης, για ταξινόμηση στοιχείων από υποθαλάσσιες εικόνες, για την αναγνώριση του είδους του εδάφους. Για την επίλυση του ζητήματος, χρησιμοποιήθηκε και επεκτάθηκε υπάρχουσα αλγοριθμική λύση [5].

5.1 Συμπεράσματα

Για την υποδομή των διαδικασιών μηχανικής επιλέχθηκε η πλατφόρμα Kubeflow που στηρίζεται στο Kubernetes. Το Kubeflow υπόσχεται να απλοποιήσει τις διαδικασίες μηχανικής μάθησης όπου υπάρχει το Kubernetes. Όπως αναλύθηκε στην παράγραφο 2, η εκπαίδευση, η παρακολούθηση και η διαθεσιμότητα μοντέλων στην παραγωγή χωρίς διαδικασίες MLOps, είναι μια περίπλοκη διαδικασία. Αυτή η μη αυτοματοποιημένη διαδικασία, επιβάλλει την εξαρτημένη συνεργασία των επιστημόνων δεδομένων και των μηχανικών DevOps. Πράγματι το Kubeflow, καταφέρνει να δώσει την δυνατότητα σε έναν επιστήμονα δεδομένων, απλά περιγράφοντας κάποιες διαδικασίες στα Kubeflow Pipelines γράφοντας python, να υλοποιούνται και να μπορούν αυτόματα να συντηρούν ένα μοντέλο στην παραγωγή. Κατά αυτό τον τρόπο δεν χρειάζεται πια η διαρκής ενασχόληση του επιστήμονα δεδομένων με την παρακολούθηση του μοντέλου στην παραγωγή και η εξειδίκευση του μηχανικού DevOps σε ζητήματα μηχανικής μάθησης.

Το Kubeflow, ενσωματώνει διάφορα εργαλεία που συνεισφέρουν στον επιστήμονα δεδομένων. Η συνεισφορά αυτή αφορά τόσο το πρώιμο στάδιο της εξερεύνησης των δεδομένων και αναζήτησης βέλτιστης λύσης σε ένα ζήτημα μηχανικής μάθησης, όσο και την διαμορφωμένη λύση pipeline, που θα συντηρεί ένα μοντέλο στην παραγωγή. Παρέχει εργαλεία που έχουν καθιερωθεί στην εργασία του επιστήμονα δεδομένων, όπως τα jupyter notebooks για ανάπτυξη κώδικα, οπτικοποιήσεις με tensorboards, καθώς και όποια εργαλεία χρειάζονται στον κύκλο εργασία ενός συστήματος μηχανικής μάθησης.

5.2 Μελλοντικές εργασίες

Ανάλογα με τις ανάγκες που καλείται να καλύψει η υποδομή, η υλοποίηση των Kubeflow και Kubernetes, μπορεί να γίνει εξαιρετικά περίπλοκη. Σε ένα παραγωγικό περιβάλλον, χρειάζεται ομάδα DevOps που να ασχολείται αποκλειστικά με την συντήρηση των δύο εργαλείων. Χρειάζεται να γίνει διαχείριση χρηστών, για την χρήση τόσο του Kubeflow όσο και πόρων του Kubernetes. Η λύση που υλοποιήθηκε παρείχε απομόνωση χρηστών στις οντότητες του Kubeflow, αλλά όλοι οι χρήστες είχαν απεριόριστη πρόσβαση στους πόρους του kubernetes, δυνατότητα που δεν επιτρέπεται στην παραγωγή. Επίσης, χρειάζεται η επιβολή ορίων στη χρήση των πόρων του kubernetes ανάλογα τον χρήστη ή το group του.

Πολύ σημαντική πτυχή, που δεν εξερευνήθηκε εκτενώς σε αυτή την εργασία, είναι η παρακολούθηση του μοντέλου στην παραγωγή, καθώς και παρακολούθηση της ποιότητας των δεδομένων. Τα μοντέλα μηχανικής μάθησης συχνά αντιμετωπίζουν κατεστραμμένα, καθυστερημένα ή ελλιπή δεδομένα. Τα ζητήματα ποιότητας δεδομένων ευθύνονται για το μεγαλύτερο μερίδιο των αστοχιών στην παραγωγή. Επίσης, ακόμα και αν η ποιότητα των δεδομένων είναι εγγυημένη, η απόδοση του μοντέλου μπορεί να φθίνει. Σε αυτή την περίπτωση, υπάρχουν δύο συνήθεις αιτίες: Η μετατόπιση δεδομένων ή η μετατόπιση εννοιών, ή και τα δύο ταυτόχρονα. Το Kserve χρησιμοποιήθηκε για την δυνατότητα να του να κάνει διαθέσιμο το μοντέλο στην παραγωγή. Εξερευνήθηκαν οι δυνατότητες για rollouts που δίνει, καθώς και σαν ελάχιστη παρακολούθηση ελέγχονταν η καταγραφή (logging) που παρέχει στην οπτική διεπαφή του. Σαν συνέχεια, θα μπορούσαν να χρησιμοποιηθούν κάποια εργαλεία που έχει για παρακολούθηση του μοντέλου όπως τα Alibi Detector, AIF Bias Detector και ART Adversial Detector. Επίσης, σημαντικά εργαλεία του Kserve είναι το ModelMesh που χρησιμοποιείται για την διαθεσιμότητα μεγάλου αριθμού μοντέλων, το Autoscaling για την αυτόματη κλιμάκωση τον services που παρέχουν το μοντέλο και άλλα.

Παρακολούθηση χρειάζεται και στην υποδομή [24]. Το Kubeflow είναι μία πλατφόρμα που ενσωματώνει εξαρχής μεγάλο αριθμό εργαλείων. Όταν έπειτα προστίθενται τα όποια πειράματα, pipelines, servers των μοντέλων, τότε οι γίνεται μεγάλη χρήση των πόρων του συστήματος. Χρήσιμη και ευρέως χρησιμοποιούμενη λύση για την παρακολούθηση της υποδομής είναι τα εργαλεία Prometheus και Grafana.

Ενδιαφέρον και μελλοντική ανάγκη παρουσιάζουν τα προβλήματα μεγάλων δεδομένων. Σε τέτοια προβλήματα, ο όγκος των δεδομένων για την εκπαίδευση ενός μοντέλου ή και το ίδιο το μοντέλο είναι αρκετά μεγάλα και χρήζουν να κατανεμηθεί η εκπαίδευση. Το Kubeflow ευνοεί εγκαινώς την κατανεμημένη εκπαίδευση, με τους operators [25] που υποστηρίζει, tf-operator, pytorch-operator και mpi-operator (Horovod [9]).

Περαιτέρω, θα μπορούσαν να υλοποιηθούν κάποια σενάρια, με τεχνικές rollout και rollbacks. Στην πράξη πραγματοποιήθηκε Canary rollout με πλήρη αντικατάσταση του μοντέλου στην παραγωγή. Σε επόμενο στάδιο, θα πρέπει να υλοποιηθεί βαθμιαία αντικατάσταση του μοντέλου και διαδικασία rollback σε περίπτωση κάποιας αστοχίας.

Τέλος, στα πλαίσια της αλγοριθμικής προσέγγισης ενός ζητήματος μηχανικής με νευρωνικά δίκτυα, το Katib παρέχει το εργαλείο NAS. Το NAS (Neural Architecture Search [10], αν και είναι ακόμα σε δοκιμαστική φάση beta, αναζητά την βέλτιστη αρχιτεκτονική ενός νευρωνικού δικτύου ενός μοντέλου μηχανικής μάθησης. Το NAS στηρίζεται στο ENAS (Efficient Neural Architecture Search όπως προτάθηκε από τους Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le and Jeff Dean[26].

Βιβλιογραφία

[1] *MLOps: Continuous delivery and automation pipelines in machine learning*. https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning. Accessed on 20-10-2022.

[2] Daniel Golovin Eugene Davydov Todd Phillips Dietmar Ebner Vinay Chaudhary Michael Young Jean Franc¸ois Crespo Dan Dennison D. Sculley, Gary Holt. *Hidden Technical Debt in Machine Learning Systems*. NIPS'15, Proceedings of the 28th International Conference on Neural Information Processing Systems, 2015.

[3] *Kubeflow*. https://www.kubeflow.org/. Accessed on 20-11-2022.

[4] *Kubernetes*. https://kubernetes.io/docs/concepts/overview/. Accessed on 20-10-2022.

[5] P. Mertikas και K. Karantzalos. *Seafloor mapping from multispectral multibeam acoustic data at the European Open Science Cloud*. ISPRS congress 2020, Remote Sensing Laboratory, National Technical University of Athens, Heroon Polytechniou 9, 15780 Zographos, Greece, 2020.

[6] *Kubeflow Pipelines SDK*. https://www.kubeflow.org/docs/components/pipelines/v1/sdk/sdk-overview/. Accessed on 20-10-2022.

[7] *Katib*. https://www.kubeflow.org/docs/components/katib/. Accessed on 20-10-2022.

[8] *KServe*. https://www.kubeflow.org/docs/external-add-ons/kserve/kserve/. Accessed on 20-10-2022.

[9] *Horovod*. https://horovod.ai/. Accessed on 20-10-2022.

[10] *Neural Architecture Search based on ENAS*. https://www.kubeflow.org/docs/components/katib/experiment/#neural-architecture-search-based-on-enas. Accessed on 20-10-2022.

[11] *Kubernetes Components*. https://kubernetes.io/docs/concepts/overview/components/. Accessed on 20-10-2022.

[12] *Kubernetes Deployments*. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/. Accessed on 20-10-2022.

[13] *Kubernetes Services*. https://kubernetes.io/docs/concepts/services-networking/service/. Accessed on 20-10-2022.

[14] *Kubernetes ReplicaSets*. https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/. Accessed on 20-10-2022.

[15] *Kubeflow Pipelines*. https://www.kubeflow.org/docs/components/pipelines/. Accessed on 20-10-2022.

[16] *Kubeflow Notebooks*. https://www.kubeflow.org/docs/components/notebooks/. Accessed on 20-10-2022.

[17] *Kubeflow Central Dashboard*. https://www.kubeflow.org/docs/components/central-dash/. Accessed on 20-10-2022.

[18] Maya Perry. *Survey Build vs Buy Decision. Should you build or buy a Data Science Platform*. cnvrg.io, https://cnvrg.io/build-vs-buy-data-science-platform, 2020.

[19] Leo Breiman. *Random Forests*. Kluwer Academic Publishers, Statistics Department, University of Cal ifornia, Berkeley , CA 94720, 2001.

[20] Friedman J.H. Stone C.J. Leo Breiman, L.I. και R.A. Olshen. *Classification and Regression Trees (CART)*. Kluwer Academic Publishers, 1984.

[21] Leo Breiman. *Bagging Predictors*. Kluwer Academic Publishers, Statistics Department, University of California Berkeley, CA 94720, 1996.

[22] Vapnik Boser, Guyon. *A Training Algorithm for Optimal Margin Classifiers*. 1992.

[23] Mary L McHugh. *Interrater reliability: the kappa statistic*. Biochem Med (Zagreb), 2012.

[24] Elisabetta Di Nitto Damian A.Tamburria, Marco Miglierina. *Cloud applications monitoring: An industrial study*. 2020.

[25] *Kubeflow Training Operators*. https://www.kubeflow.org/docs/components/training/. Accessed on 20-10-2022.

[26] Barret Zoph Quoc V. Le Jeff Dean Hieu Pham, Melody Y. Guan. *Efficient Neural Architecture Search via Parameter Sharing*. 2018.