Effective MLOps and coreContol

CONTENTS

INTRODUCTION

Machine learning (ML) is increasingly used in software applications across industrial or business domains. Nowadays, machine learning (ML) algorithms are continually improving, with new solutions being developed at a very rapid pace. However, these algorithms are just a part of the overall software solution; they are part of a much larger ecosystem of technologies used for building software systems.

> "
>
> ***...data Scientists spend ~80% of their time preparing and managing data for analysis" (Forbes*)***
>
> * https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=53d634d16f63
> https://www.dataversity.net/survey-shows-data-scientists-spend-time-cleaning-data/

ML brought new challenges and added more complexity to engineering robust, resilient, and scalable systems. In this whitepaper we address some of these challenges and potential solutions that can help companies of any size mitigate the risks encountered during building, testing, validating, and deploying software systems that have a large ML component.

The solution we propose falls under a relatively new domain, namely Machine Learning Operations (MLOps), and focused on how we can adapt a DevOps mindset to facilitate the development and maintenance of ML systems.

Everyone is talking about Machine Learning Operations (MLOps) and trying to figure out how to apply it to their data processes. But what is it exactly? How easy or complicated might it be to implement MLOps? This short white paper answers these questions.

CHALLENGES OF DEVELOPING AN ML SYSTEM

Machine learning changed the paradigm used for developing software applications. When developing a ML solution engineers do not specify a step-by-step recipe with the logical operations required to solve a problem. Instead, they provide the machine with a set of examples and build algorithms that figure out the rules or steps required to solve the problem by themselves. This different approach for building software systems, means that ML solutions introduce a new plethora of challenges that developers, engineers, and domain experts need to address. Below we highlight the most critical challenges and issues encountered while developing ML solutions.

- ML requires experimentation which often results in multiple iterations or experiments.
- ML algorithms are metrics-driven which implies manual metrics tracking across experiments to determine which algorithm performs the best. This activity leads to increased complexity.
- Manually tracking metrics and other logs can lead to numerous errors.
- ML algorithm performance is tightly coupled to the data being used. Usually, the data are not versioned which greatly reduces the ability to reproduce experiments.
- Changes in the data flowing through the model often require training a new model which involves many manual processes. Moreover, once a model is trained, it needs to be deployed (again manually).
- Like the data, models are rarely not versioned, which further increases the reproducibility issues when developing a ML solution.
- In contrast to traditional software engineering, the development of a ML algorithm is often limited to individual development environments that run

on local machines. Consequently, this code is not reviewed, and no automated testing takes place. More, collaboration can be very difficult.

- The result of an ML algorithm development is a model which leads to many situations where the results cannot be reproduced or improved, with model maintenance being mostly very poorly performed.
- Minor changes to the data processing or the ML algorithm has far-reaching influences across multiple steps of the development pipeline and may even contradict previous developments.
- Finally, ML and AI pipelines are developed by Data Scientists, whom are experts in this world, but less familiar with the business and the domain in which they are active. Therefore, a representation of their research result to a domain expert will contribute to the success of the models.

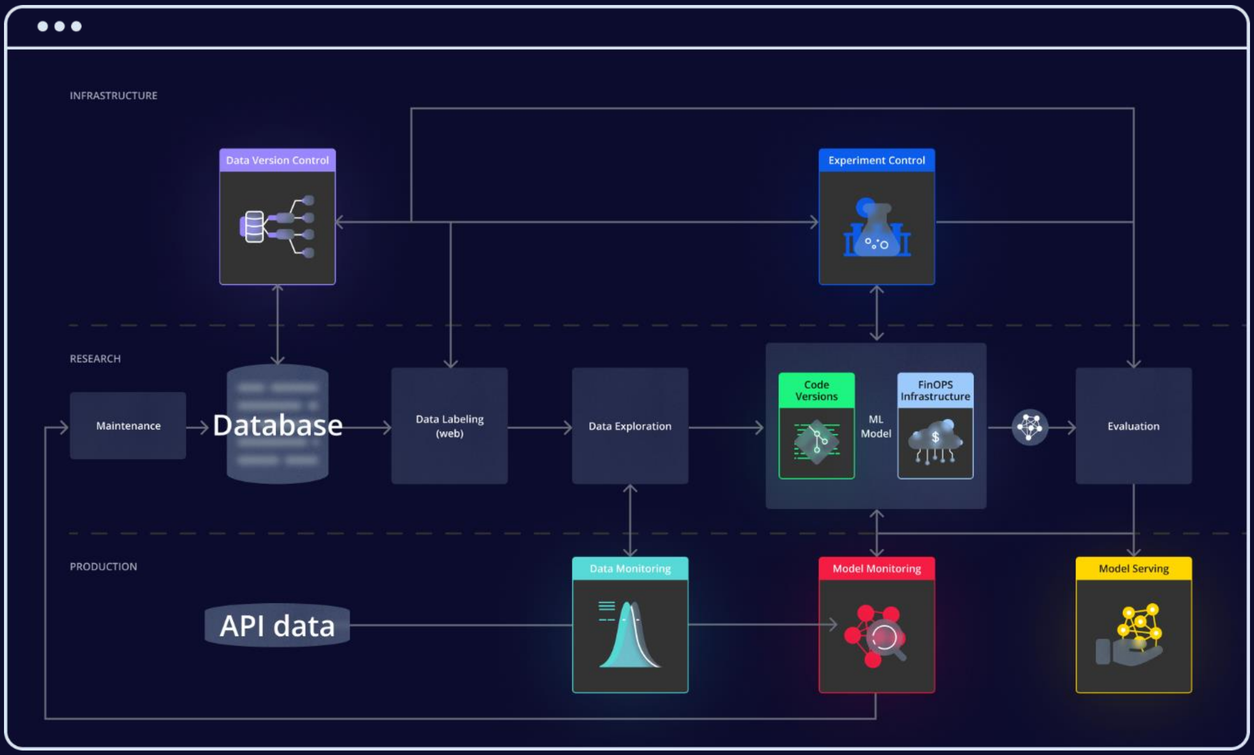
WHAT MLOPS COVERS?

Machine Learning Operations is software versioning management on steroids combined with DevOps. Reconstructing an AI / ML experiment resulting with a model, holds a high number of variables involved, such as hyper parameters, versions of the data itself, additional processing done on the inference or training software and more. Those variables are managed and stored using various tools while building the AI pipeline. Additionally, moving models to production is more than CI/CD wrapping of the model and preparing it for serving, it's also the ability to monitor the performance of the model and detect drifts in the data. Last, is the ability to bring the domain expert into the loop, and analyze the changes together with the Data Scientist.

In the diagram below, the components in the "research" path in the center of the image are the regular flow of the data scientists and ML engineers in their research. The highlighted components, wrapping that flow are the infrastructure and production tool that helps get control of this pipeline, with the ability to also involve the domain experts.

CoreControl platform will help you arrange and enter the new age of data-centric via MLOps

From research to production and back for continuous improvement.



HOW COMPLICATED IS IT TO START MANAGING MLOPS?

Most of the MLOps philosophy is to be transparent and included in the research and production pipelines, rather than being serial to an existing flow. As such, it does not require refactoring any code in the training or operational code.
With **coreControl** we help you deploy code pieces in your training and inference code, such that initially, you will get documentation of experiments, data versions and running models performance. We help your engineers to deploy MLOps methodologies easily as our experience is coming horizontally from many companies and expertise we acquired.

The initial outcome is a dashboard view that is recording the running experiments. In the next stages, a minimal user cooperation is needed in order to be able to name the experiments, describe the purpose and hypothesis behind, so it will be easy to benchmark the results.

Such results can be obtained within 30 days with CoreAI service.

Our solution also puts the feet on the ground and can work in an entirely on premise environment to maintain clients confidentiality.

After the system is up and running, **coreControl** introduces a no-code experience. Thus less experienced engineers can run new experiments directly on your cluster. Also, they will be able to easily visualize the data and the models directly from the UI, without having to write any additional code.

Most of the data science teams get stuck at the moment when they have to ship their model to production. With the help of **coreControl** you will be able to have “control” over this step. In the end, your machine learning solution can bring real value to your company.

A GOOD SOLUTION HOLDS BUSINESS VALUES

- Easy enter into Data-Centric movement
- Continuous Experimentation and CI/CD of your model
- Accelerate the path and time to a successful model
- Automation of ML research process
- Bringing the domain expert into the loop with a collaborative environment
- Easy move from MVP to production, reduce the mode to production by ~3-6 months
- Knowledge preservation, new employee training is cut by 3-5 months
- Solve real data issues from the field, not hypothesis in the lab
- Reduce data scientist time wasted on infrastructure taks by 90%
- Reduce devops support time by 80%
- FinOps - Managing model training costs and loss prevention alerts

WHY DATA-CENTRIC?

Most of the data scientist's time is spent on data related issues and not necessarily on the algorithmic side. It's not that more data can solve and improve the model, it's how the data is being improved and sorted properly to serve a better model. Improving data quality is becoming significantly more important in the production phase when the model meets reality.

The domain experts are a key player in helping data scientists to improve the data quality and KPI’s that the model is striving for.

This Data-Centric approach puts the data in the center and not the model itself as it's not only “garbage in, garbage out” to create a successful model.

These trends, data quality improvement and creating a collaborative environment that will add the domain experts into the loop, increase the number of experiments and the need to manage them and share experiments results. The more experiments you have the better the chances to strive for a better model, however, selecting a preferred model should be easy.

Training costs usually explode when running experiments at scale. Optimize the resources used to run experiments and maximize your throughput.

Taking your model to production means it's a newborn - research to production in a few steps with end-to-end visibility of the process.

Keep your model and pipeline fresh and updated with real data and issues from the production.

ROBUST ML SYSTEMS WITH CORECONTROL

The **coreControl** dashboard that we will provide will store and display the data versioning, Git versions, results, and KPIs of trained models and if necessary special preprocessing parameters unique for the research. Below are additional aspects covered by the **coreControl** dashboard.

In this section we will review the MLOps functions covered by **coreControl:**

- Experiment Management / KPIs
- Data Versioning
- Github version
- Model Serving
- Model Monitoring
- Data Monitoring
- FinOps and Infrastructure

Experiment Management / KPIs

One of the key factors of an MLOps infrastructure is to track the experiments. After dozens or even hundreds of experiments, it is difficult to remember which hyperparameters performed best and which one had the leading KPI. **coreControl** can easily be integrated into the training code of your model to keep track of your configuration, hyperparameters, and metrics. Also, it has support to log different kinds of objects like images, tables, graphs, etc.

Another issue appears after you have logged information for dozens of experiments and you want to compare them. **coreControl**'s simple interface gives you the possibility to pin a couple of key metrics that will show up directly into the table where you search for experiments. In this manner, you will be able to easily surf between experiments and pick the desired one.

One last thing to mention is that you can implement any custom KPI/metric, both model and business-wise. Hence, you will always feel in control.

Data Versioning

Data versioning can be applied to the Data Feature set Map, as well as to the new data version that is created as a result of the new settings of the encoder.

CoreAI will automate that the second dataset will be versioned by the first dataset version and the committed code of the autoencoder model.

coreControl is data centric. Therefore, we put emphasis on how to interact with the data. The main issue with data versioning systems is that they don't give you a clear way to visualize what is going on. After you version your data, it is hard to see what is the actual difference between those versions. With current solutions you have to download every different version and manually take a look. This is very tedious and time consuming.

With **coreControl**, for every version of the data, you will be able to see an overall description of the data like the number of samples & features, the storage it uses and the newest samples that were added. You can easily take a look at statistics properties like mean, median, quantiles and more. Also, for every feature we compute the outliers, histogram and distribution.

The icing on the cake is the fact that you can directly compare multiple versions of the data. Thus, you can easily navigate through different versions and understand what has changed from one to another.

Github Versions

Your model training code versioning is critical as well, as changes like preprocessing of the data, thresholds are affecting the model performance.

As part of any experiment the version of the Github commit is stored and reported on the dashboard. Such information can in later stages serve for investigating model KPIs and to be able to spot contributing factors to the model's success or its productional behavior.

By versioning the code you will be able to have full control over different variables like the configuration of your system and the hyperparameters. Also, one key factor that is hard to spot while debugging is the preprocessing and postprocessing pipeline. It is submitted to changes over time, therefore a bug can easily be introduced. With the help of **coreControl** those issues can be easily reproduced, then you will be able to go back to your working version of the code.

Model Serving

CoreAI will help in wrapping the model serving with a container and if necessary add a basic service API.

The model can be served in production using different methods like a Model-as-Service, where the model is exposed on the internet through a RESTful API, or like a Model-as-Package, where the model is shipped as a Python package to be directly integrated into other systems.

The models from production are integrated with **coreControl**. Therefore, without any effort, you can leverage all of its features like monitoring your model and data, or serving a different version of the algorithm.

Model serving will include obviously the additions of monitoring code to the model performance.

Model Monitoring

While your model is in production, it is the subject to degradation. Your model is prepared to handle situations similar to those it was trained on. Hence it won’t respond well to data that is different from the one used for training.

Moreover, as the model degrades, it requires careful monitoring to ensure that the prediction performance is not affected by the passage of time or other unexpected events. Therefore, robust production models require advanced monitoring that is usually divided into data monitoring and concept drift. Additionally, success criterias are monitored as the model is in service, which will help validate better model versions success.

Data Monitoring

Many of the ML/AI models are based on time series data and require continuous training of models to keep it up to date. As your model is already in production, an automated model update or a manual one, require monitoring of the data to make sure it is keeping the same assumption you made in your research phase.

Graphical representation of the various data, as well as valuable out of distribution events. Notification of drift, anomaly, and untrusted situations.

With a robust data monitoring system, you are able to visualize all the data that is different from your training scenario. **coreControl** offers various methods that detect this drift like KL divergence, KS statistical test, tree methods, etc. Also, with an alarm system in place, you will be able to catch those scenarios in time and change them properly.

In the end, with a proper monitoring system, you will always be prepared to adapt to new changes, quickly react to unexpected events, mitigate risks, and constantly bring value to your clients.

Concept drift

Effective for monitoring the performance of the models during production run time and can detect model drifting, benchmarking for the engineering features, and more. CoreAI provides the relevant code for implementing on the running model serving code in order for it to report the performance. Additional benchmark building is feasible as well per the client needs.

In this step, predictions from the production model are continually validated against the outputs from the test set. The end goal is for the data to be distributed according to the same statistical distribution. In other words, the values predicted by the model in production and the values predicted by the model on the test set should have similar (or even the same) statistical properties. Thus, the concept drift monitoring system will confirm that the model's behavior is consistent and persists over time.

coreControl implements this monitoring component, which will continually run to check the similarity between the real-time and test prediction values. Therefore the system will swiftly detect a drift in the model's predictions, and you will be warned to appropriately respond to this event.

Concept monitoring is performed using similar techniques as for the data monitoring step (e.g., KL divergence, KS statistical test, tree methods, etc.). Therefore, the outcomes of the model monitoring follows the same pattern as for the data monitoring. Using coreControl’s model monitoring component you will get real-time insights into the behavior of your models, in production which allow you to rapidly react to any unexpected events. Together, these tools help you adapt your data and ML strategy to assure control over your IT infrastructure and software applications.

Infrastructure / FinOps

The last piece of the puzzle is the actual infrastructure that will sustain the model ecosystem. **coreControl** facilitates the possibility to monitor all your machines (CPU, memory, GPU, I/O, etc.) and turn them on/off on demand. It can be used both on-premise and on different cloud providers (e.g. AWS). The point of **coreControl** is to hide the complexity of your infrastructure and to unify different cloud providers or on-premise machines into a single view. Thus, it will be accessible to both technical and non-technical persons.

The icing on the cake is the fact that **coreControl** will monitor all the costs that the infrastructure generated, this is what we call FinOps. The possibility of having control over your infrastructure costs. You can visualize all your expenditures over time. Also, you can set price thresholds for every specific experiment or even for a project. You will get notifications when you are close to finishing your credits and when you will pass your predefined threshold. Your current expenditures and predefined thresholds are displayed, in a single view, in your experiments or projects table.

In the end, such a dashboard will help you understand the costs of your development and maintenance. Also, it will facilitate you with a forecasting mechanism to see how many resources you still need to achieve your goal.

BENEFITS IN USE CASES

In the following two examples we will review the benefits of using coreControl and start getting control over MLOps process's.

1) Manage and Reconstruct your experiments
2) Clear and easy to use data versioning and preprocessing - keep track of what data is used for each experiment
3) End-to-end process visibility - view what is happening with the data, models, and infrastructure across multiple environments and project phases.
4) Cost management - explore how much your ML is costing across all phases of your projects
5) Benchmarking of experiments, model performance and data health
6) Data Monitoring

Use case 1 - Startup in the Medical Field

One of our clients is a start-up in the medical field, which uses ML to increase efficiency and speed up time-consuming tasks, such as keypoint detection and segmentation in x-ray images.

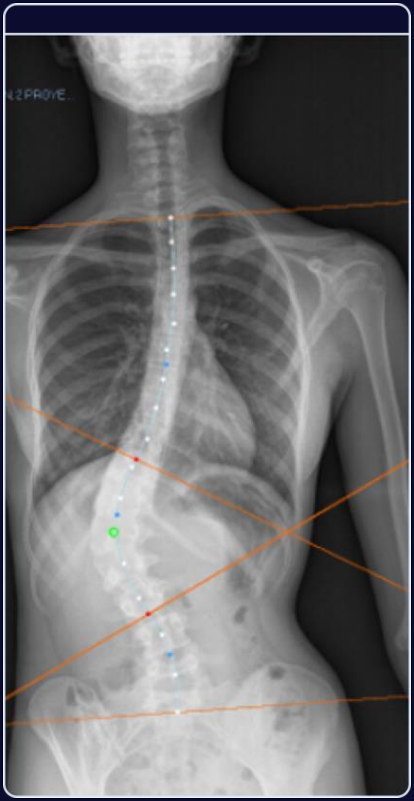
At the beginning, the ML components were developed in a local environment, by data scientists with good domain expertise, but minimal knowledge in the MLOps field. When the data scientists left the company, all the knowledge about the reasoning, processes, modeling and all the tracks of the experiments were lost.

This has caused a massive technical debt, as the developers that came after had to reproduce their thought process and experiments to be able to reach the baseline performances.
All the knowledge that had to be regained was documented and stored in a way such that any new team members can pick it up from that exact point, without any loss or technical debt.

Using **coreControl** and a pragmatic approach, we built pipelines that allow us to keep track of all the changes in software, data, or model, so that all the knowledge behind an experiment is preserved, in a collaboration-centric system. Moreover, as the client is involved in the medical domain, we also integrated automatic report generation for FDA compliance.

The approach resulted in fast experimentation, with minimal code addition. The results are quantified by the following numbers.

- **150+** total experiments, each of them having the following attached to it
  - Traceable code, data and model versions.
  - Hyperparameters and other configurations.
  - A complete report for FDA compliance.
  - Qualitative and quantitative performance reports.
- **30+** experiments on augmentation strategies only, done in a systematic way.
- **20+** of the best models stored and backed up. (along with everything that is required to reproduce them…)



At the moment, the engineers and scientists do not spend any time collecting and logging metadata about experiments and models; Instead, coreControl automatically takes care of this, making the whole process more time-efficient.

Use case 2 - Demand forecasting

A client in a traditional business domain looking to adopt ML for their operations. They wanted to pilot an end-to-end solution for predicting demand at multiple levels, starting with individual items and working their way up to regional outputs.

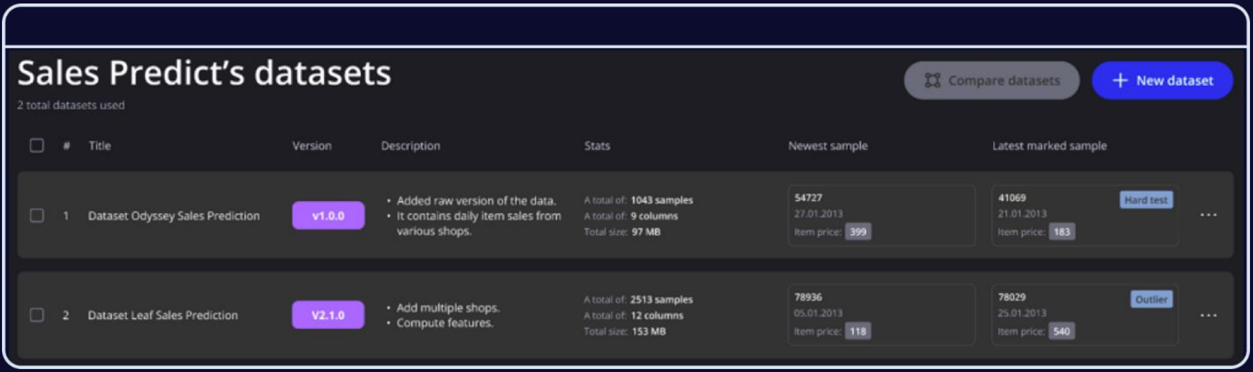
Most ML projects get stuck at the development stage. When they get to the operationalization, deployment, and monitoring steps, they realize the complexity of the whole process, which is intimidating. It is essential to mention that all those components are mandatory to have a successful production ML application.

Features Extraction <> Data Versioning - One of the greatest challenges was to extract the features that will be used in the training in this time series data. In this case, each additional feature created a data version. This helped to quickly identify the contributing features to the accuracy of the predicting model.

By using coreControl, we easily bypassed those impediments. With its operationalization functionalities, we quickly ran +100 experiments. Using its core comparison methods, we quickly picked the best model out of the experiments. Also, with the help of coreControl, we deployed and set up the monitoring infrastructure in less than two weeks. In the end, we had a ready-for-production pilot, with all its components in place, in less than two months.

With the help of coreControl, we could quickly ingest the raw CSV files containing demand data into a dedicated *data warehouse*. From this point, the dataset was versioned, and any changes were reflected in the commits. As shown in the image below, we analyzed two data sources within the demand forecasting project. “Dataset Odyssey Sales Prediction” was newly added and initialized with version v1.0.0 and “Dataset Leaf Sales Prediction” was the subject of multiple changes ending with version v2.1.0. With an emphasis on an intuitive UI, coreControl displays an overview of the dataset, such as the description of the latest commit, automatically inferred statistics, and the newest samples added to the dataset. Hence, anybody can easily search any dataset existing within the company.

Moreover, wecan tag any sample of the dataset with custom labels to make it more accessible for the users to recognize specific issues. For example, within the demand forecasting use case, we marked the examples that are difficult for the ML model to estimate or represent an outlier. Therefore, we could quickly test our model on those hand-picked samples and see how well it behaved.



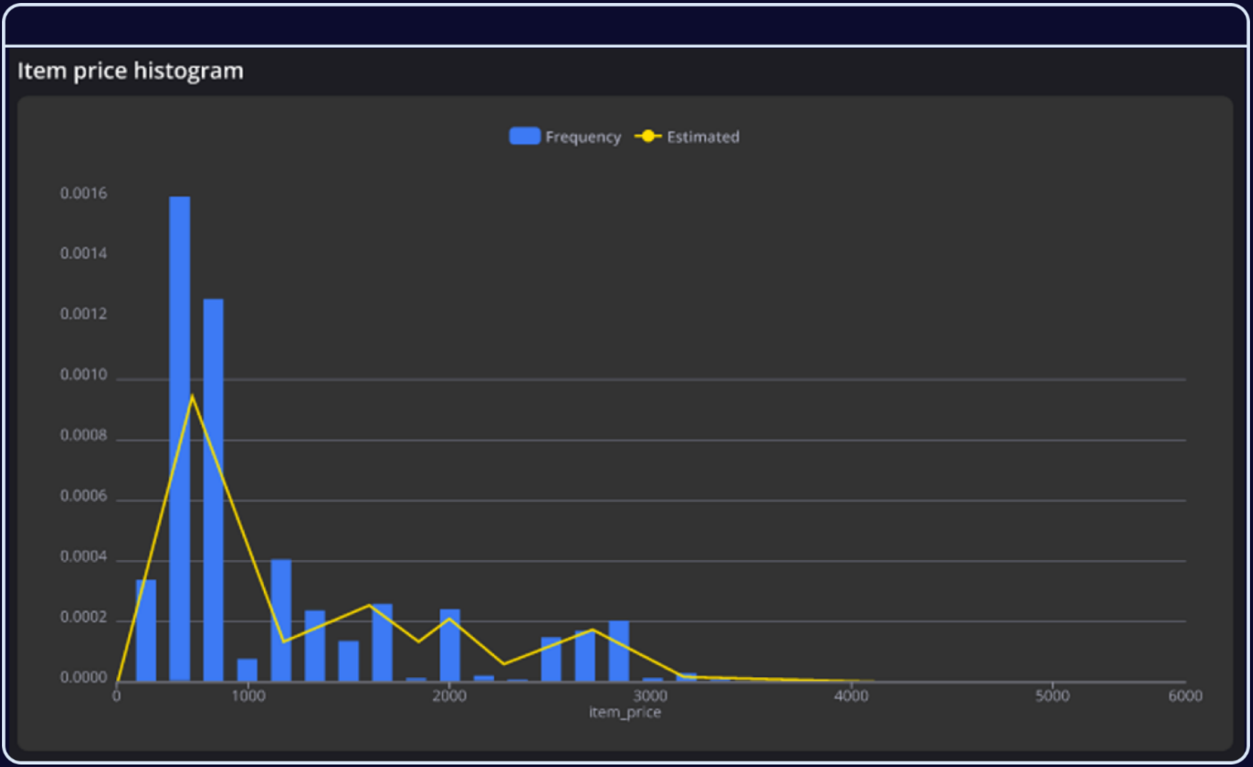
Sales Predict's datasets

2 total datasets used

#	Title	Version	Description	Stats	Newest sample	Latest marked sample
1	Dataset Odyssey Sales Prediction	v1.0.0	• Added raw version of the data. • It contains daily item sales from various shops.	A total of: 1043 samples A total of: 9 columns Total size: 97 MB	54727 27.01.2013 Item price: 399	41069 Hard test 21.01.2013 Item price: 183
2	Dataset Leaf Sales Prediction	V2.1.0	• Add multiple shops. • Compute features.	A total of: 2513 samples A total of: 12 columns Total size: 153 MB	78936 05.01.2013 Item price: 118	78029 Outlier 25.01.2013 Item price: 540

After the data is uploaded, coreControl automatically computes a set of statistics and displays them via our interactive dashboard. Here we can quickly see every feature's histogram, probability distribution, outliers, and more. Thus, we can quickly analyze our dataset without needing to write any code. The most powerful part of our technology is that now we can quickly compare different datasets and versions without writing any additional code. With the help of our comparison tool, we can easily see the differences between multiple datasets, such as the number of samples, required storage size, statistical properties, and much more.

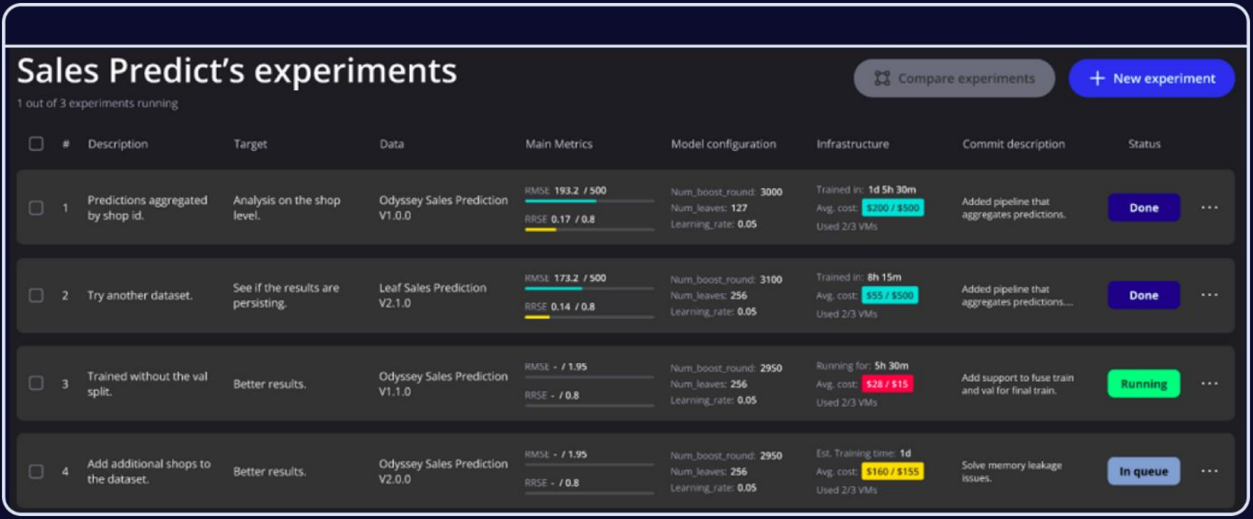
After coreControl ingested the raw data into the newly created data lake, we were able to develop a set of custom data pipelines to clean and normalize the data, to engineer new features, etc. The newly created features are stored in a feature store, and can be quickly accessed from the cached features. This option is available from anywhere inside the coreControl ecosystem. Thus, the features can be shared between teams, machines, and even ML models.



Also, with the help of coreControl, we refined the whole *experimentation process*. For every experiment, we track all the variables that are relevant for a particular model together with its KPIs. This helps us compare different solutions statistically to ensure that the model with the best performance is pushed to production. More concretely, for every experiment, we follow all the core components of a training process, such as hyper-parameters, git commit, metrics, dataset version, etc. Within coreControl, we can see the state of the experiment, such as its main metrics, business KPIs and even how much it costs to run it on a given infrastructure. Our light Python SDK can easily log all those variables from our custom training code. With a click of a button, we can compare multiple experiments and decide on the best solution.

Within only two weeks, we could run hundreds of experiments. With the pipeline orchestration infrastructure, we quickly engineered dozens of new features. coreControl comparing system helped us decide on the best model hyper-parameters and optimal feature subset. Otherwise, picking the proper experiment would have gotten out of hand quickly and introduced errors in the decision process.

Therefore, we have an optimal tradeoff between an accurate ML model and the correct number of features. In the end, the model from production is accurate, fast, and memory efficient.



Sales Predict's experiments

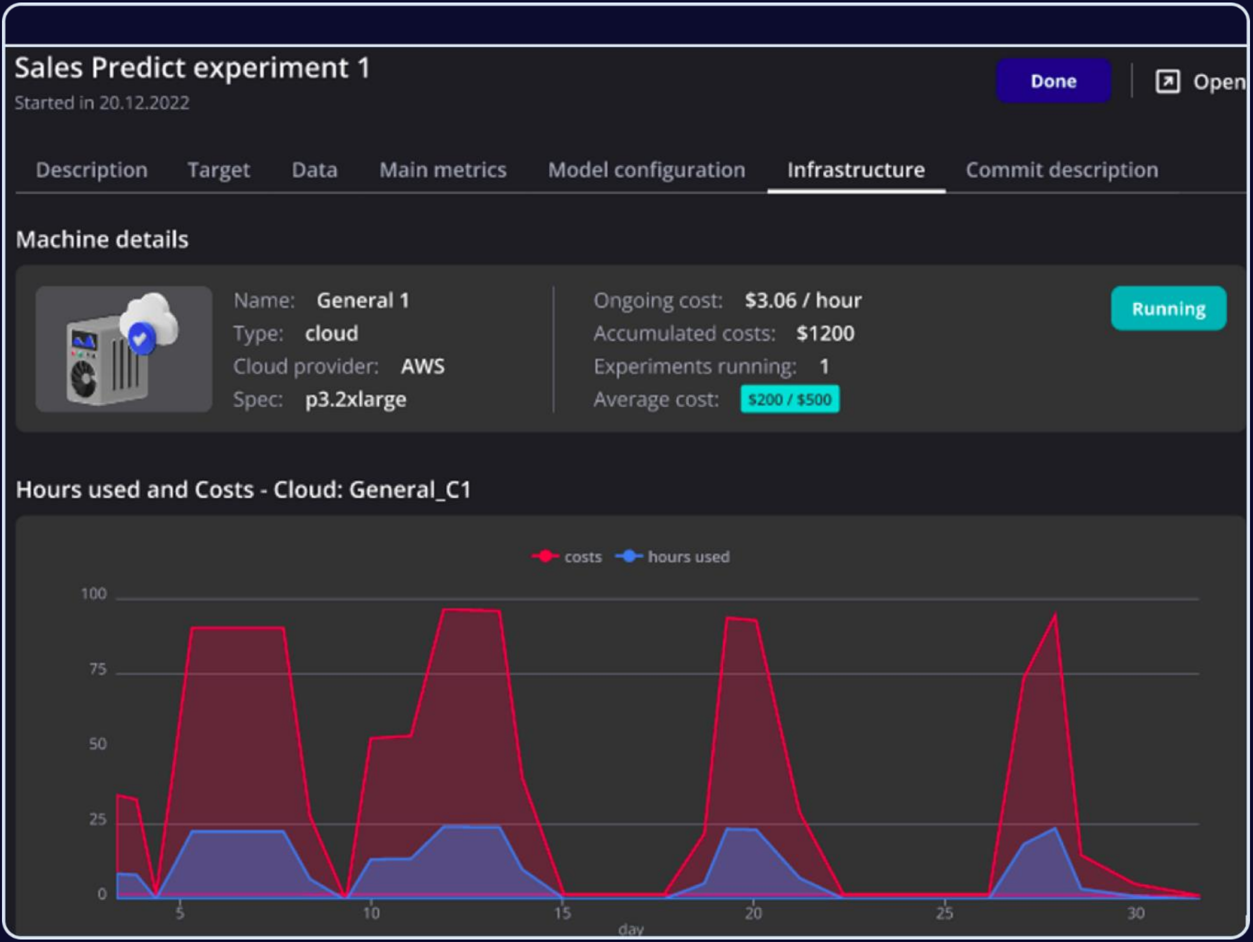
1 out of 3 experiments running

#	Description	Target	Data	Main Metrics	Model configuration	Infrastructure	Commit description	Status
1	Predictions aggregated by shop id.	Analysis on the shop level.	Odyssey Sales Prediction V1.0.0	RMSE 193.2 / 500 RRSE 0.17 / 0.8	Num_boost_round: 3000 Num_leaves: 127 Learning_rate: 0.05	Trained in: 1d 5h 30m Avg. cost: $200 / $500 Used 2/3 VMs	Added pipeline that aggregates predictions.	Done
2	Try another dataset.	See if the results are persisting.	Leaf Sales Prediction V2.1.0	RMSE 173.2 / 500 RRSE 0.14 / 0.8	Num_boost_round: 3100 Num_leaves: 256 Learning_rate: 0.05	Trained in: 8h 15m Avg. cost: $55 / $500 Used 2/3 VMs	Added pipeline that aggregates predictions....	Done
3	Trained without the val split.	Better results.	Odyssey Sales Prediction V1.1.0	RMSE - / 1.95 RRSE - / 0.8	Num_boost_round: 2950 Num_leaves: 256 Learning_rate: 0.05	Running for: 5h 30m Avg. cost: $28 / $15 Used 2/3 VMs	Add support to fuse train and val for final train.	Running
4	Add additional shops to the dataset.	Better results.	Odyssey Sales Prediction V2.0.0	RMSE - / 1.95 RRSE - / 0.8	Num_boost_round: 2950 Num_leaves: 256 Learning_rate: 0.05	Est. Training time: 1d Avg. cost: $160 / $155 Used 2/3 VMs	Solve memory leakage issues.	In queue

We can drill down into each experiment and examine it in detail. However, coreControl is more than an experiment tracking tool. One thing we like about it is that it includes a FinOps component that lets us see our research expenses. As shown in the image below, we can effortlessly visualize our infrastructure's daily costs and usage. Hence, we can easily forecast how much resources we will need to train a specific model in the future.

The last piece of the puzzle consists of the deployment process. coreControl leverages container technologies (i.e. Docker) to deploy the solution. This process permits us to engineer a scalable infrastructure in a very short time. We wrapped up the ML solution with a RESTful API , which makes it accessible over the internet with very little hustle. By using a RESTful API we can now integrate the model into any system, for example, you can have a client application that displays the predictions or and a server application that uses the predictions as part of more complicated process. The image below shows an example where we deployed two models. For each model, we can visualize their metrics, information about the drift status (both data and concept drift), required infrastructure, and accumulated costs.

Moreover, for every project, we can deploy various versions of the model which makes A/B testing and experimentation a lot easier compared to more traditional methods.



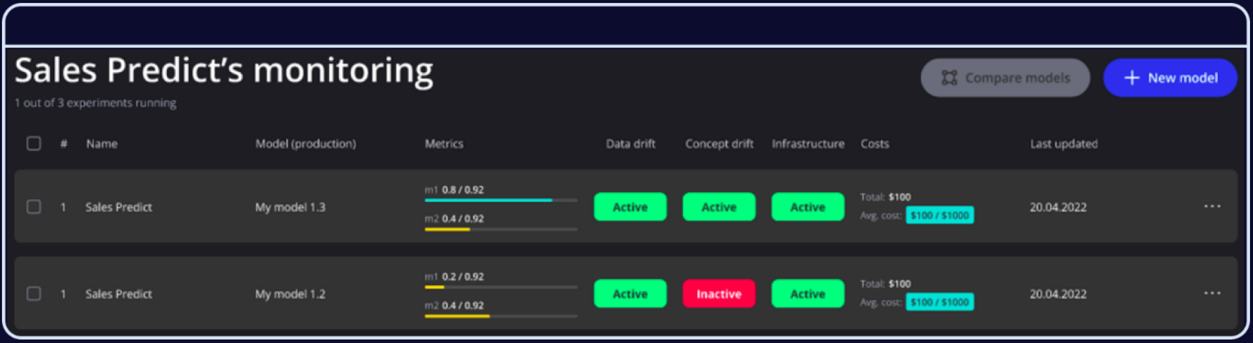
Sales Predict experiment 1

Started in 20.12.2022

Machine details

Name:	General 1
Type:	cloud
Cloud provider:	AWS
Spec:	p3.2xlarge
Ongoing cost:	$3.06 / hour
Accumulated costs:	$1200
Experiments running:	1
Average cost:	$200 / $500

Hours used and Costs - Cloud: General_C1



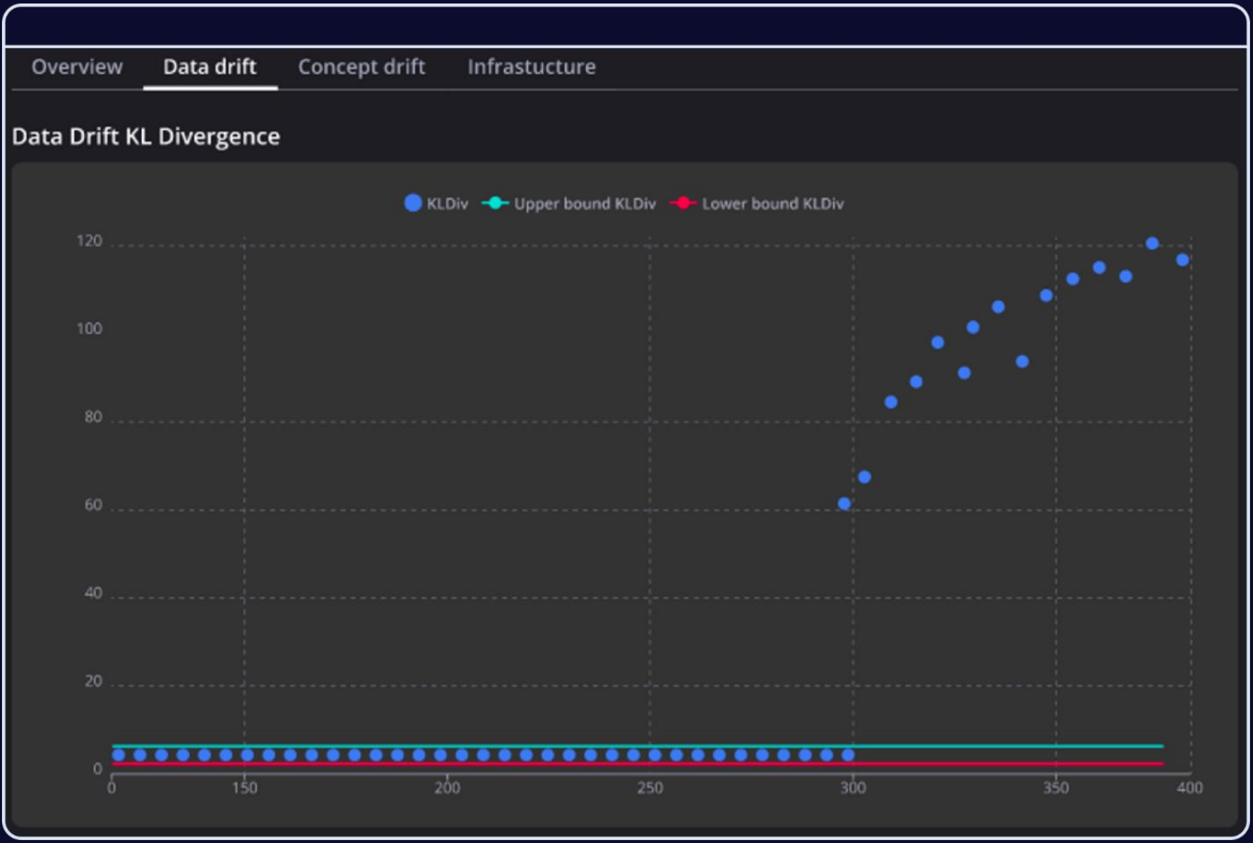
Sales Predict's monitoring

1 out of 3 experiments running

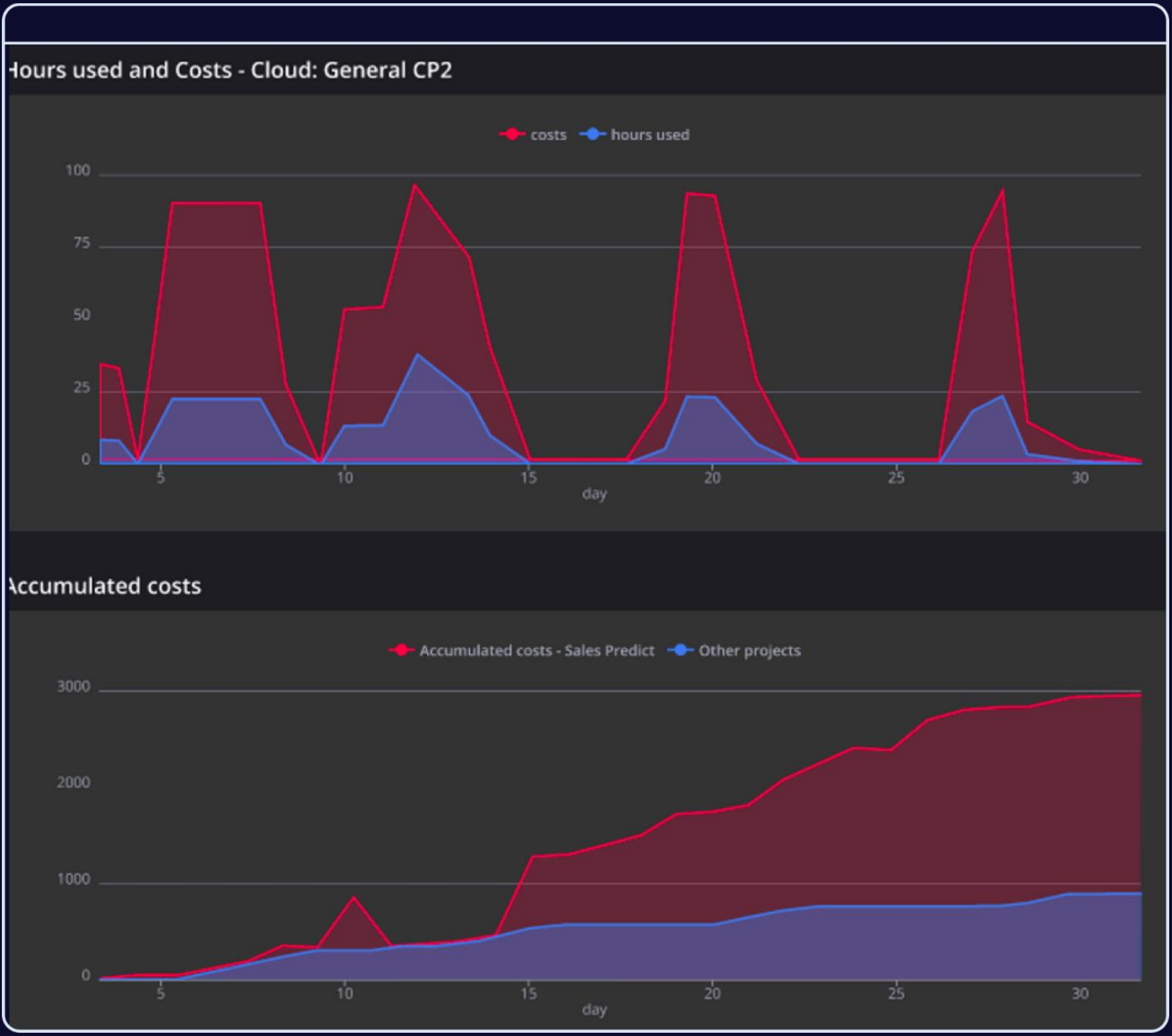
#	Name	Model (production)	Metrics	Data drift	Concept drift	Infrastructure	Costs	Last updated
1	Sales Predict	My model 1.3	m1 0.8 / 0.92 m2 0.4 / 0.92	Active	Active	Active	Total: $100 Avg. cost: $100 / $1000	20.04.2022
1	Sales Predict	My model 1.2	m1 0.2 / 0.92 m2 0.4 / 0.92	Active	Inactive	Active	Total: $100 Avg. cost: $100 / $1000	20.04.2022

coreControl is continually monitoring the deployed model with different data and concept drift metrics. In the image below, you can observe an example of our monitoring in real-life. The teal and red bars, from the image below, represent the upper and lower bounds for the metrics (i.e. the intervals within which the data and model behave as expected). coreControl has out-of-the-box statistical tools to detect drift, such as KL divergence, KS statistics, tree methods, etc. After the monitoring system detects any value outside the allowed interval, an alarm is triggered, and the coreControl monitoring system will notify you that a specific drift is detected.

What is even more powerful, is that coreControl includes a fully customisable monitoring system whereby you can pick from a series of predefined metrics or even create your own.



Finally, coreControl offers a FinOps component. Using this feature, we can visualize the daily and accumulated costs of our production infrastructure. Moreover, this feature offers us a clear view of the resources that our solutions need. Also, we can quickly analyze the sustainability and scalability of the software component to bring as much value as possible to the company. The FinOps feature ensures us that we will always be aware of the current infrastructure costs so that we will never end up with skyrocketing infrastructure bills overnight.



Using coreControl, we quickly set up the infrastructure on our cluster of computers. We immediately ingested the data into our data lake, where we automatically visualized and explored it. The data is versioned and ready to be cleaned, transformed, and modeled into our data pipeline. With the robust infrastructure in place, we efficiently run over 100 experiments. By using coreControl, we recorded all the ML experiments metadata. After using our powerful comparing tool, we picked up the best model, which we containerized and passed directly to the deployment infrastructure managed entirely by coreControl. With the FinOps component, we planned how many experiments we could run without overflowing our budget. Thus, by having a clear view of our resources, we focused on reaching our target KPI within the given restrictions. By having the monitoring component up and running, we prevented one crucial concept drift of the model. It predicted more sales to a specific shop than it should. Therefore, we avoided oversupplying a shop and saved lots of resources.

To conclude, with the help of coreControl, we versioned and modeled the dataset. We designed, built, tuned, and deployed the model. Also, we put in place the monitoring component. Using coreControl, our team achieved all those steps in less than two months, and on other projects, it took us more than one year to fulfill all those milestones. By using coreControl, we had almost a 90% increase in efficiency.

Contact

CEO Eitan Netzer

Email: eitan@coreai.ai

Phone: +972508922925

Website: coreai.ai