cabify

PyData Meetup
2023-03-23

Intro: Toni Prada - Senior Data Science & Data Engineering Manager

Talk: Luis Redondo - Senior Data Engineer
Ignacio Peletier - Senior Data Scientist @ Pricing

Cabify: We are a multi-mobility platform

Cabify

A driver will come pick you up wherever you are.

Logistics

A fast delivery service in the city.

Scooter

Move freely by electric scooter and pay per minute.

Taxi

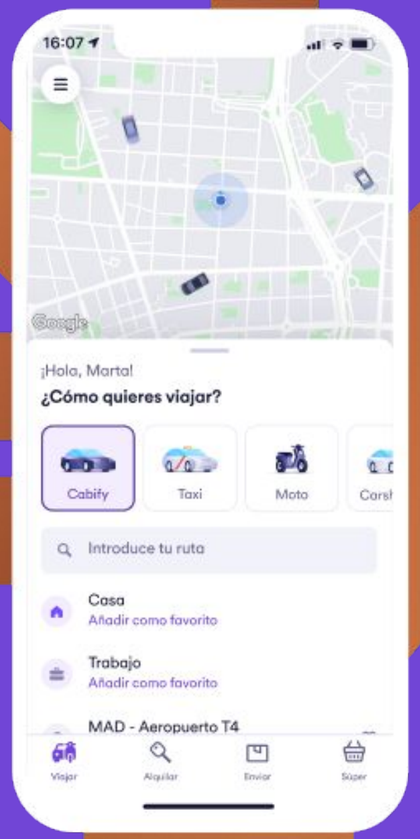
All the good things about the taxi, but with a fixed price.

Motorbike

An electric motorcycle, ready to keep up with the pace of your favorite city.

Carsharing

Rent a zero emissions car by the minute.



More than 10 years transforming the way we move

+33.000.000

registered users

+65.000

business move around with us

+1.000

employees, with a 47% female talent rate

+400.000

drivers and taxi drivers

We receive up to 50k data points per second, in real time. 98% of them include geolocation metadata.


Lima Feb 20 - 14:00 to 15:00
legacy states
Lima Feb 20 - 14:00 to 15:00
assetLocations

Al formar parte de Cabify disfrutas de:

Teletrabajo

Tenemos posiciones 100% full remote y posiciones con modelo híbrido: 2 días en casa, 3 en la oficina y 6 semanas de teletrabajo al año.

Medio día por cumpleaños

¡Esto hay que celebrarlo! Para que lo disfrutes, tómate la tarde libre.

Celebraciones, fiestas y team buildings

Para que te diviertas conociendo mejor a tus compañeros. ¡No todo es trabajar!

Formación in-house

Disfruta de programas especiales para fomentar tu desarrollo y aprendizaje.

Crédito en Cabify

Un importe mensual para usar nuestra app.

Recharge Days

¡12 viernes libres al año! El tercer viernes de cada mes es para ti.

Revisiones salariales

Porque si tú creces, tu sueldo también.

cabify.careers

¿Te unes?

cabify

Ajustando precios con un buen MLOps

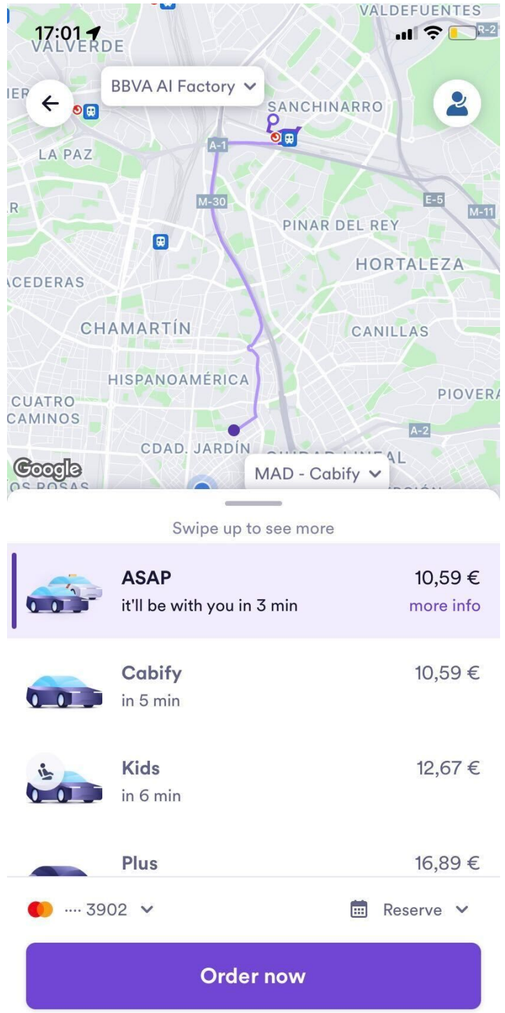
Luis Redondo | Data Engineer
Ignacio Peletier | Data Scientist @ Pricing

Agenda

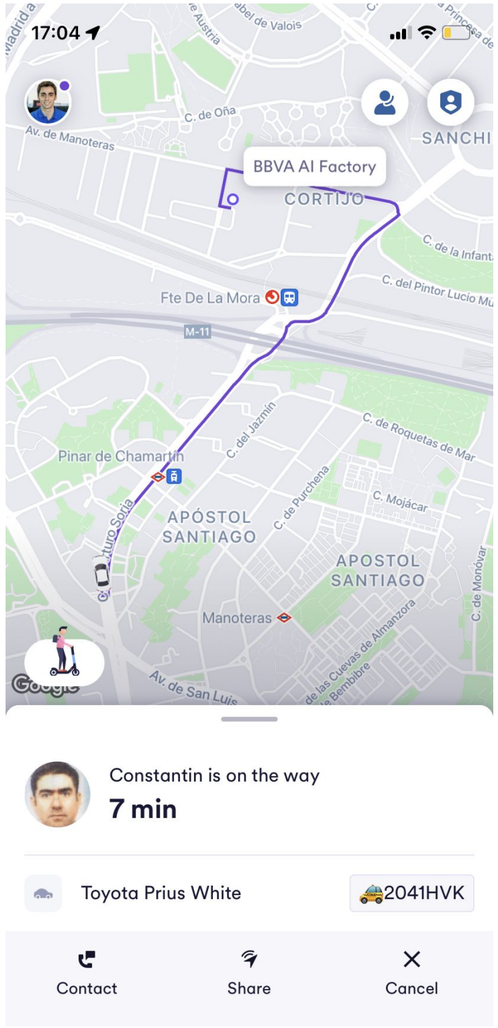
- The Funnel
  - Steps
  - Metrics
- The Challenge
  - Our Marketplace
  - Why Dynamic Pricing?
- Machine Learning
  - Optimus Price
  - The Training Data
- CatBoost Utilities
- MLOps
  - Lykeion
- Experimentation
- Next steps

The Funnel

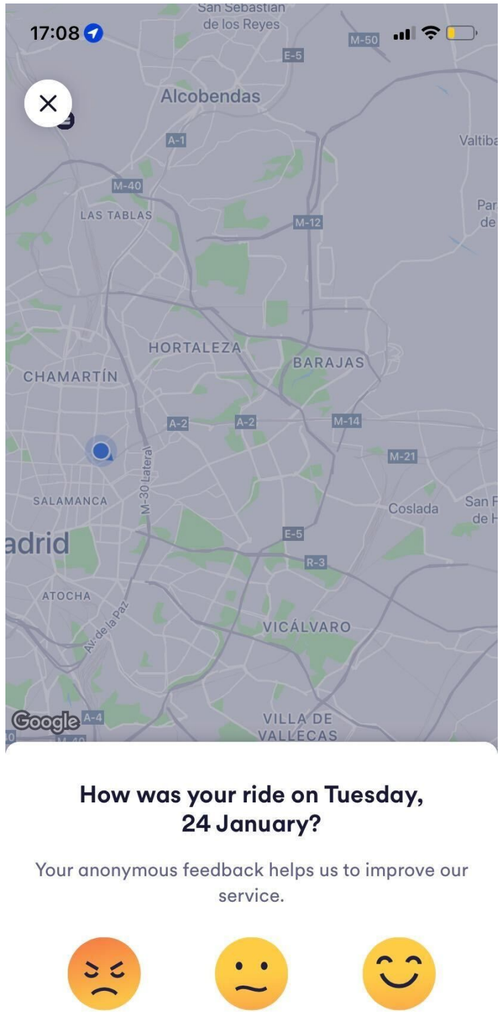
Estimation



Request



Drop Off



Somos neutros en carbono

Metrics

- **Conversion to Request:**
  - Number of Requests / Number of Estimations
- **Success Rate:**
  - Number of Drop Offs / Number of Requests
- **Conversion to Drop Off:**
  - Number of Drop Offs / Number of Estimations

✓ **P(DO) = P(RE) · SR**

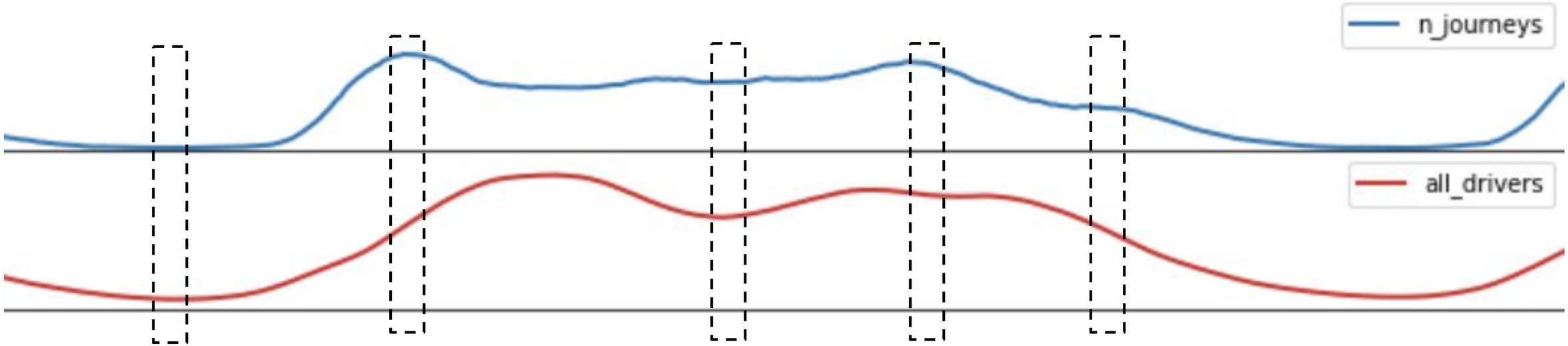
The Challenge

Our Marketplace

- Cabify’s Marketplace is two-sided:
  - **Passengers:** they want cheap journeys. Among other things, of course.
  - **Drivers:** they higher bounties. Among other things, of course.
- Our competitors are in both sides:
  - They could offer cheaper journeys to passengers, who may choose the cheapest ride.
  - They could offer higher rewards to drivers, who may choose to use their apps instead of ours.

Why Dynamic Pricing?

- Periodic variations in supply and demand.
- Price positioning might not be optimal.
- **Small price changes** around our price positioning.
- Different from spurious spikes.



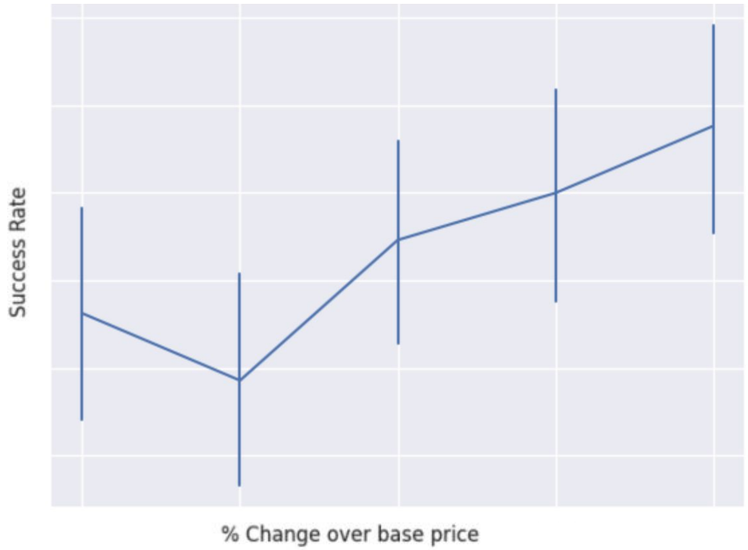
Machine Learning

Optimus Price

- Our dynamic pricing system.
- Automatically scales up or down our base prices.
- There is a set of human constraints.
- Set to optimize a specific metric (or metrics).

The Training Data

- We have past information from different price strategies and promotions.
- From these learnings, we train a prediction algorithm.



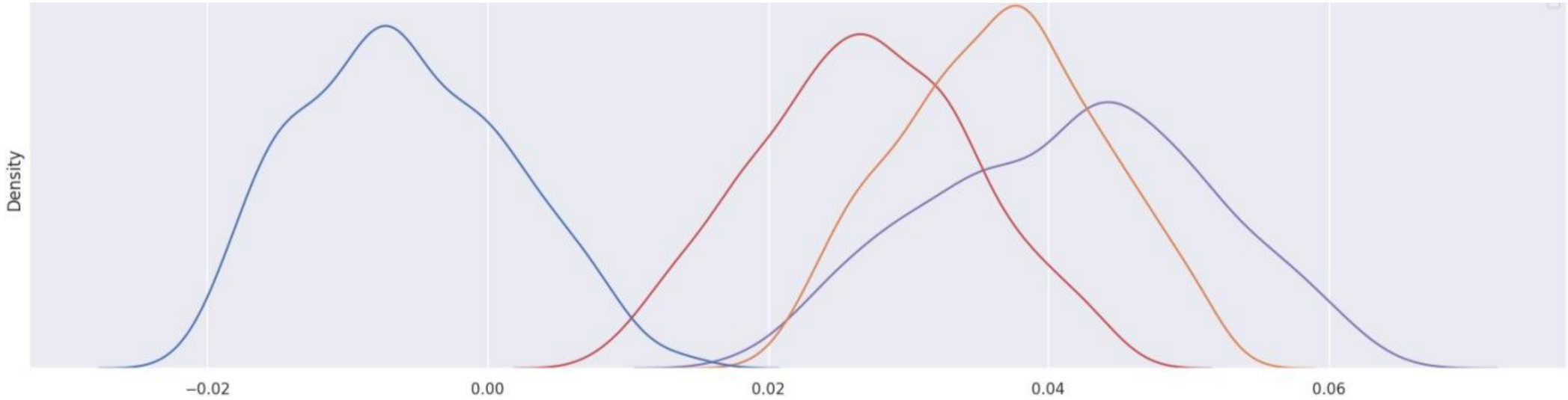
CatBoost Utilities

- **Multi-Quantile Loss**: *cool!* We just train one model:

```
q_model = CatBoostRegressor(loss_function='MultiQuantile:alpha=0.2, 0.5, 0.8')
```

- **Posterior Sampling**:

```
ps_mod = CatBoostRegressor(posterior_sampling=True).fit(train[features], train[target])
ens_preds = ps_mod.virtual_ensembles_predict(test_df, prediction_type='VirtEnsembles', virtual_ensembles_count=N)
```

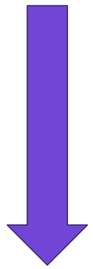


- **Monotonic Constraints**:

```
mc_mod = CatBoostRegressor(monotone_constraints = '(-1,0,0,1,0)')
```

Now what?

- We now have a beautiful notebook.
- But we need this in production.
- Training & Evaluation.
- Sustain hundreds of requests per second.
- Low response times.
- Auto-scaling.
- Iterate without involving other teams.



MLOps

Goals

- **Reduce time to market** of ML backed solutions.

- Empower Data Scientists **autonomy** to put them in production.

- Provide a **resilient** and **scalable** environment to cope up with any load.

- Integrate them as **first class citizen services**: tracing and monitoring.

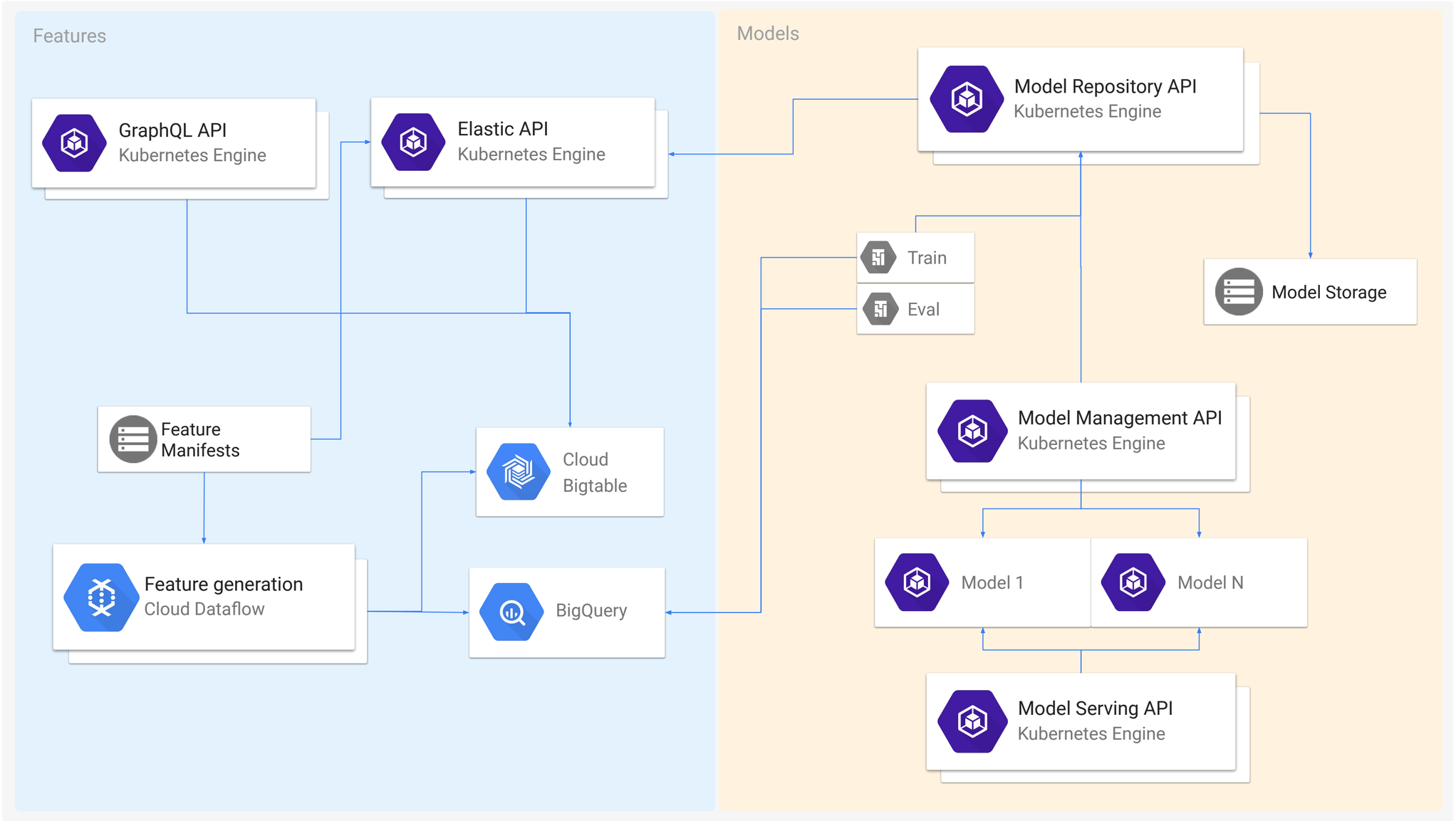
Building blocks

- Two main sides:
    - **Features**: set of tools built to compute properties or characteristics of certain entities of our platform (riders, drivers, fleets). Sometimes used as inputs to model predictions, and some of them are built also built as the result of model predictions.
    - **Models**: set of tools and services to support the whole lifecycle of models (train, deploy, evaluation).

Under the hood

- Lykeion is based on a mix of technologies:
    - **MLFlow**: open source framework to speed up ML development, manage experiments, package and trace models.
    - **Kubernetes**: runtime environment for models, that handles horizontal scalability.
    - **BigQuery**: GCP's datawarehouse that serves and stores training and evaluation data.
    - **Airflow**: orchestrator to trigger model lifecycle tasks.
    - **Flask**: Python framework to wrap and serve models as standalone applications.
    - **Prometheus+Grafana:** metrics and observability stack to track and act over models' performance.

Lykeion architecture overview



Google Cloud

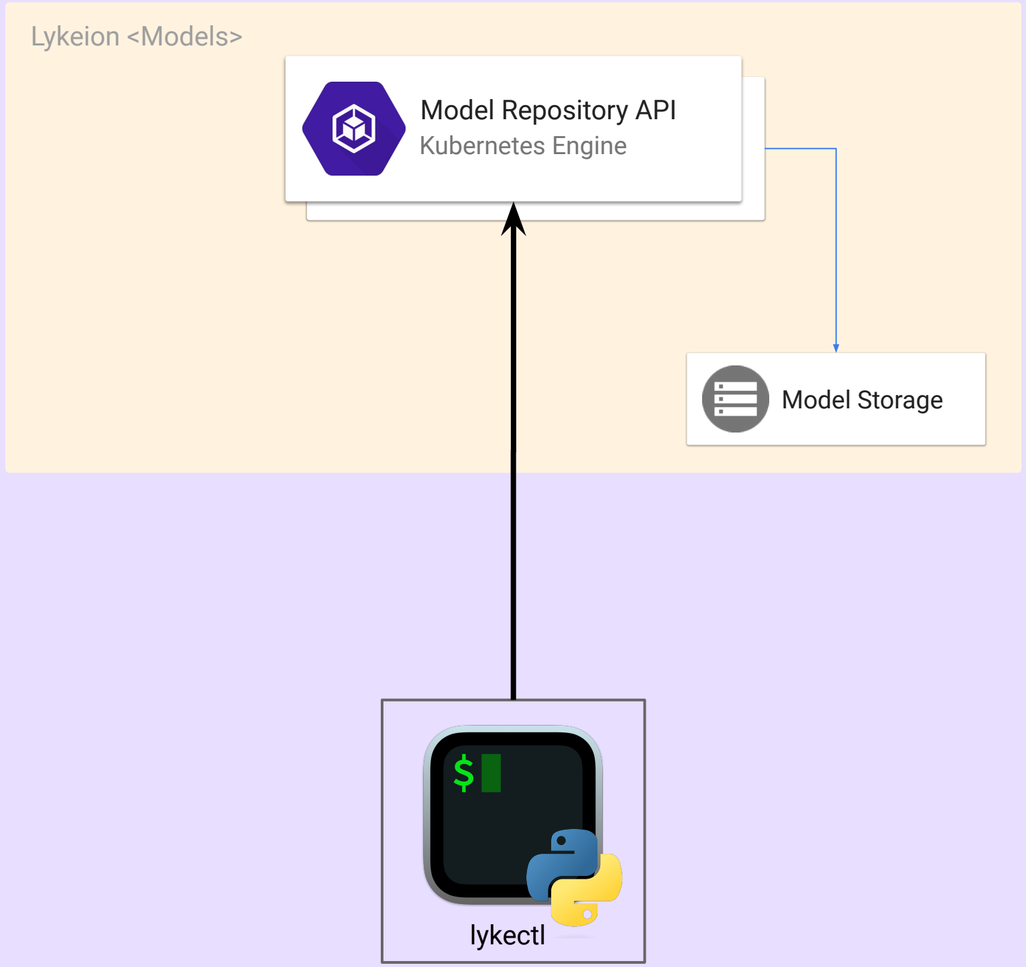
Google Product

Lykeion service

Creating a new model

Users interact with Lykeion mainly through **lykectl**, a CLI tool to trigger actions and request predictions.

- Models are grouped in families and segments
- Each model version point to:
  - Git commit hash.
  - List of input features.
- **Model Repository** stores and serves model information: status lifecycle.

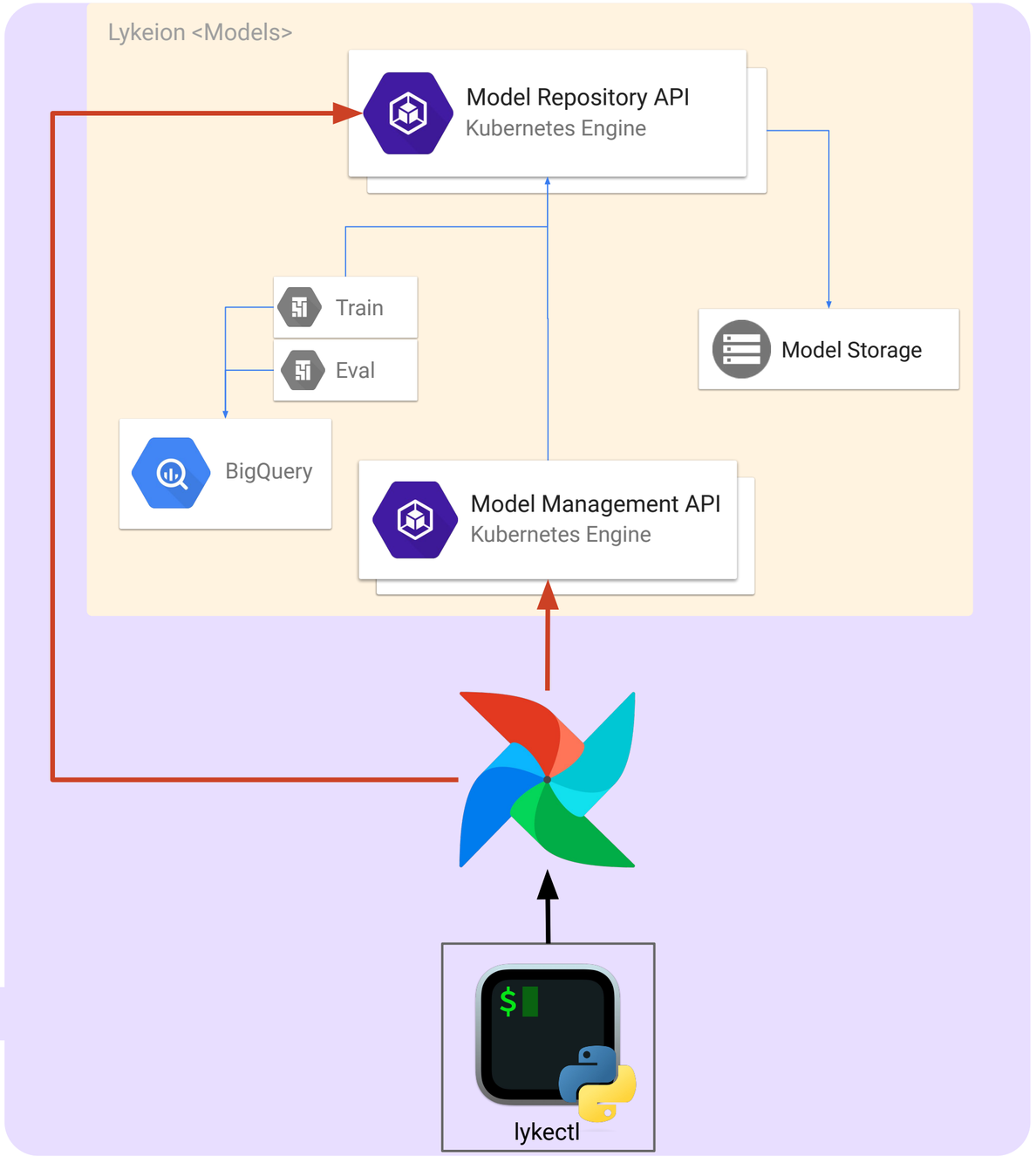



```
lykectl models create optimus-price --segment default --version 1.0
--description "Dynamic Price Model"
--source-url git@[...]@SHA1 --input-features "adhoc,lat.origin"
--input-features "adhoc,region_id" [...]
```

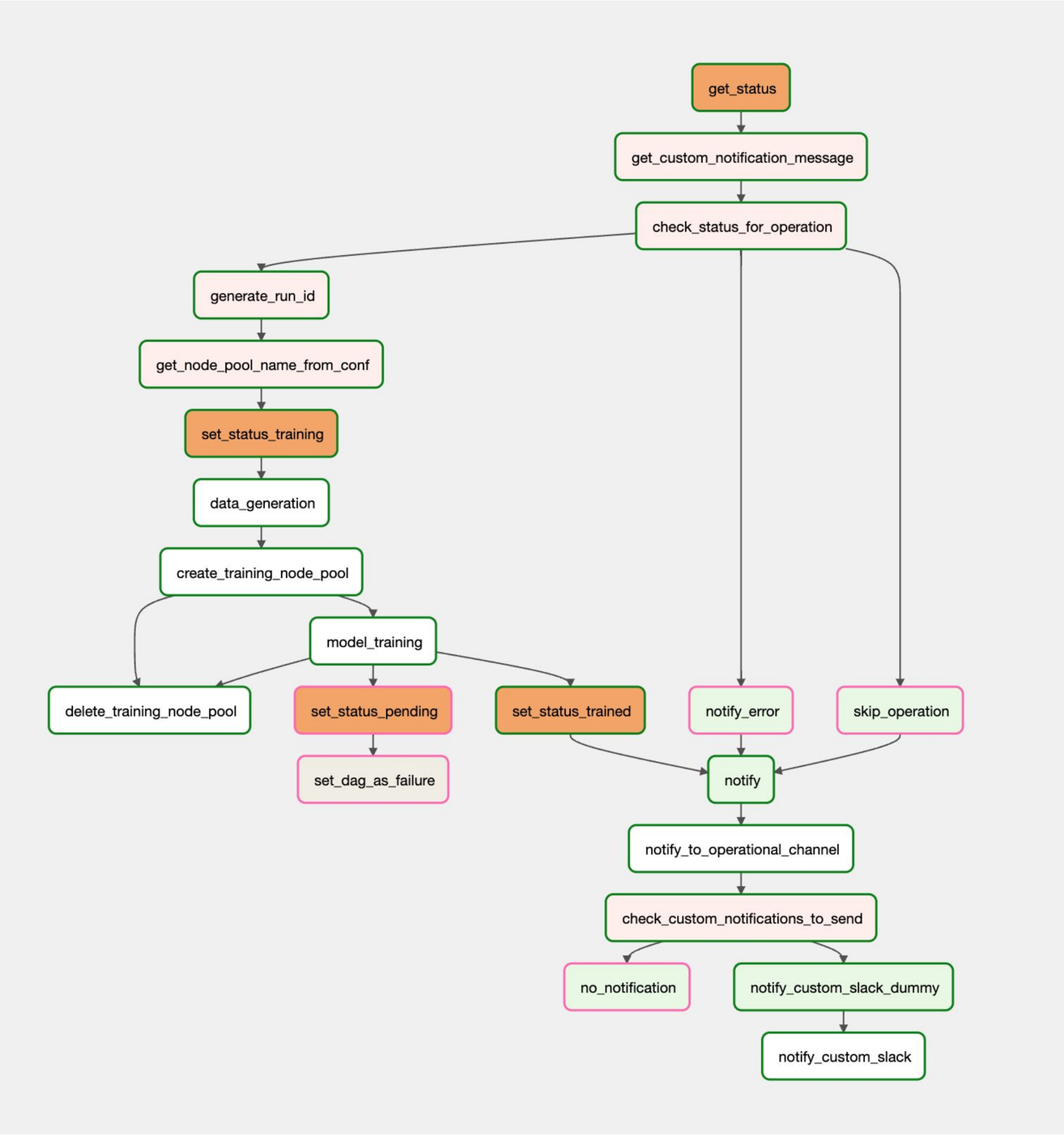
Training

To train a model, the user just needs to invoke a lykectl command:

- By triggering an Airflow DAG that, combining different **Model Management** tools will:
  - Create a Kubernetes Node Pool to execute all the workloads.
  - Build the training dataset in BigQuery.
  - Launch the model training.
- The output artifact is stored by the Model Repository.



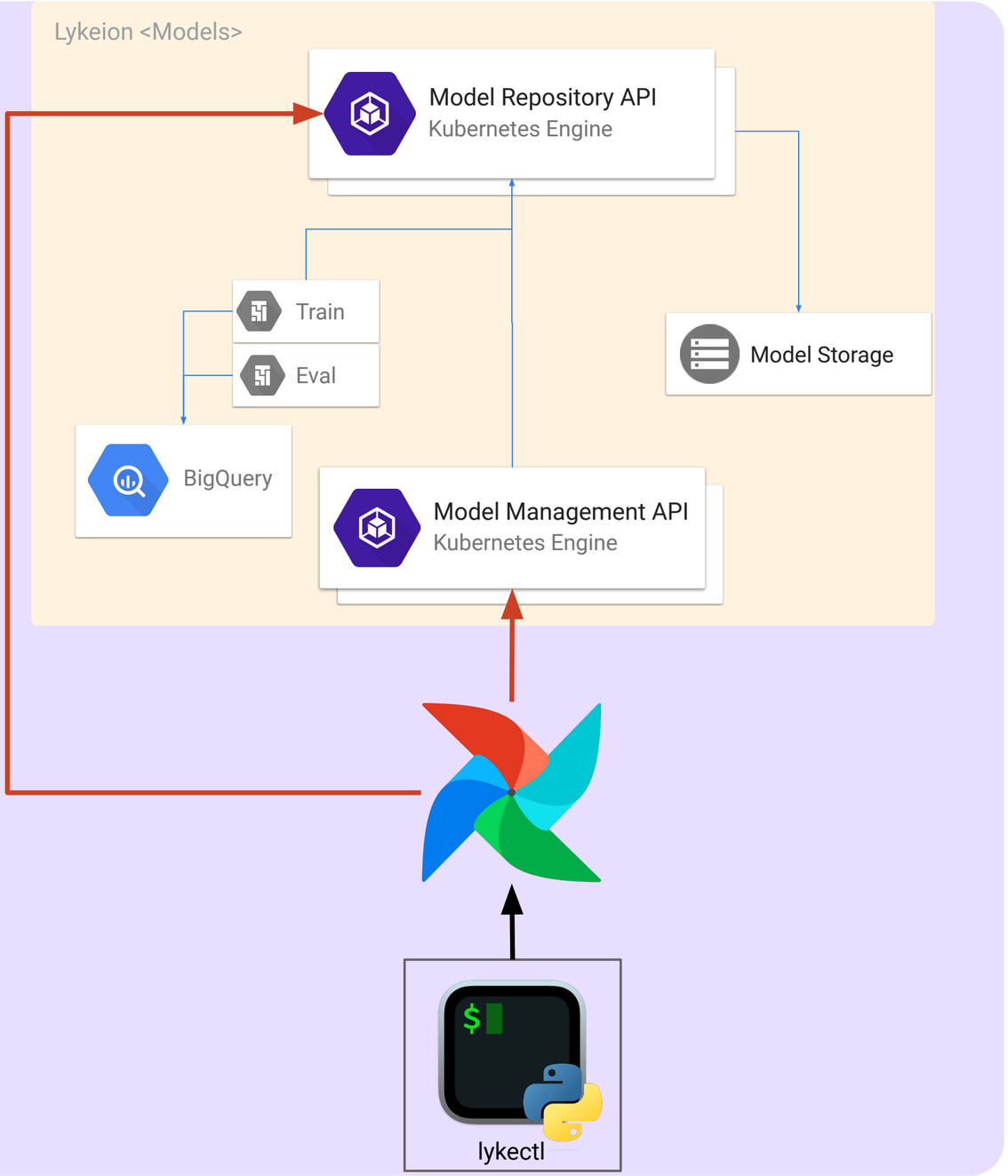

```
lykectl models train optimus-price --segment default --version 1.0
```


get_status
get_custom_notification_message
check_status_for_operation
generate_run_id
get_node_pool_name_from_conf
set_status_training
data_generation
create_training_node_pool
model_training
delete_training_node_pool
set_status_pending
set_status_trained
notify_error
skip_operation
set_dag_as_failure
notify
notify_to_operational_channel
check_custom_notifications_to_send
no_notification
notify_custom_slack_dummy
notify_custom_slack

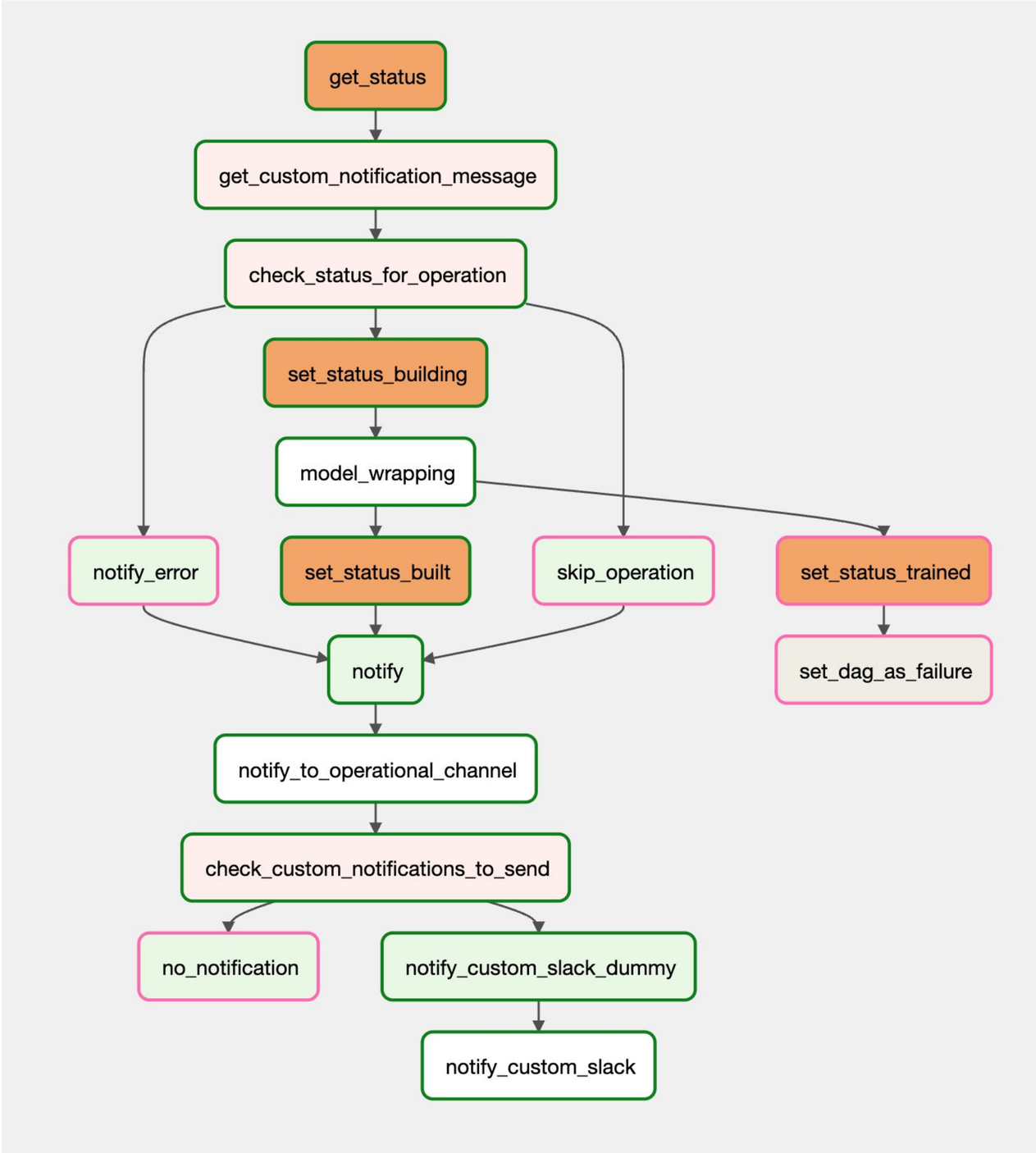
Building

To be able to put the model into action, we need to build (or wrap) it:

- On another Airflow DAG, we will trigger a Kaniko build, to create the Flask application that wraps the MLFlow model.
- The Flask application offers a common interface to request predictions and will take care of sending basic prediction metrics.
- The resulting Docker image will be pushed to our registry.



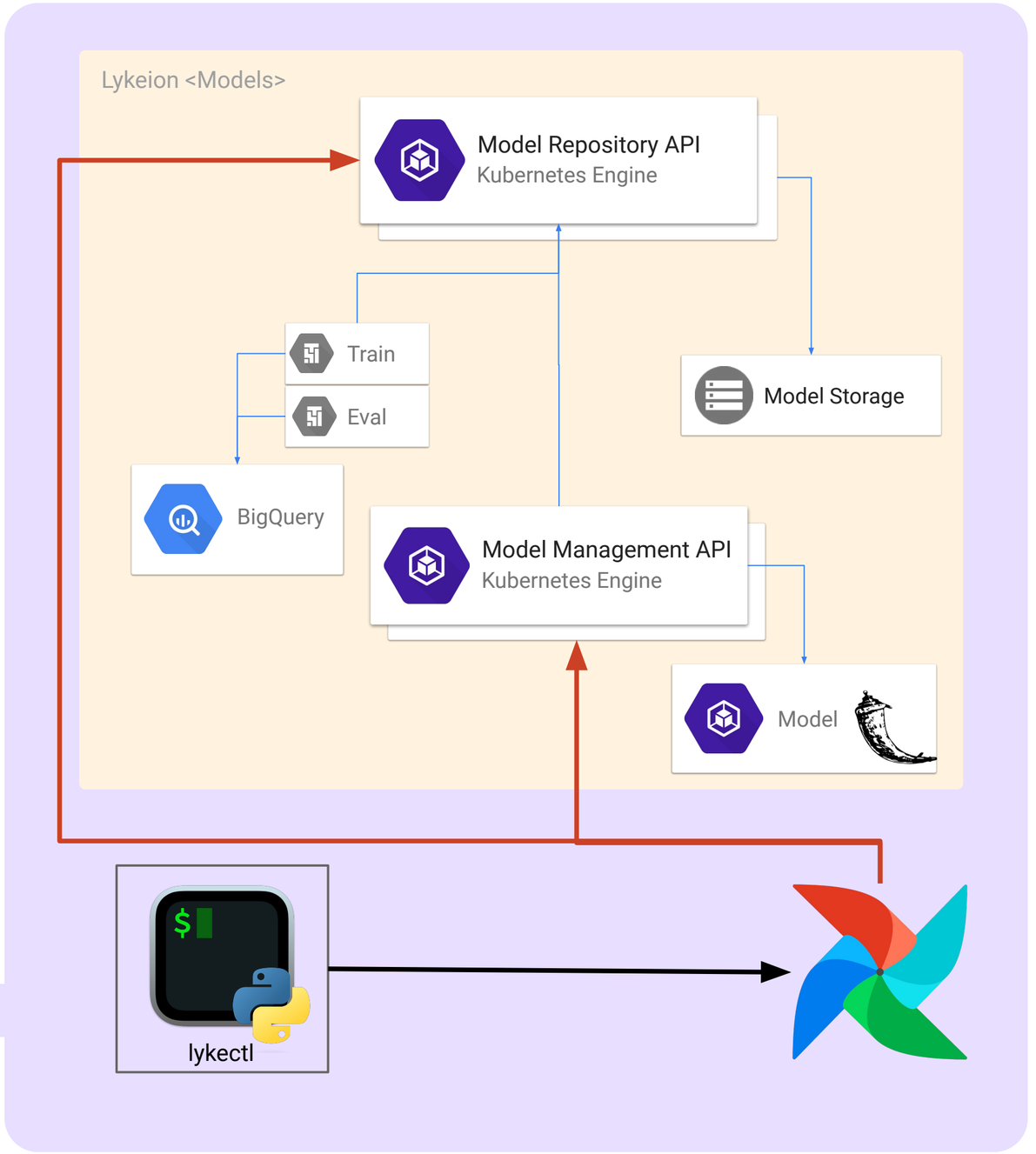

```
lykectl models build optimus-price --segment default --version 1.0
```


get_status
get_custom_notification_message
check_status_for_operation
set_status_building
model_wrapping
notify_error
set_status_built
skip_operation
set_status_trained
notify
set_dag_as_failure
notify_to_operational_channel
check_custom_notifications_to_send
no_notification
notify_custom_slack_dummy
notify_custom_slack

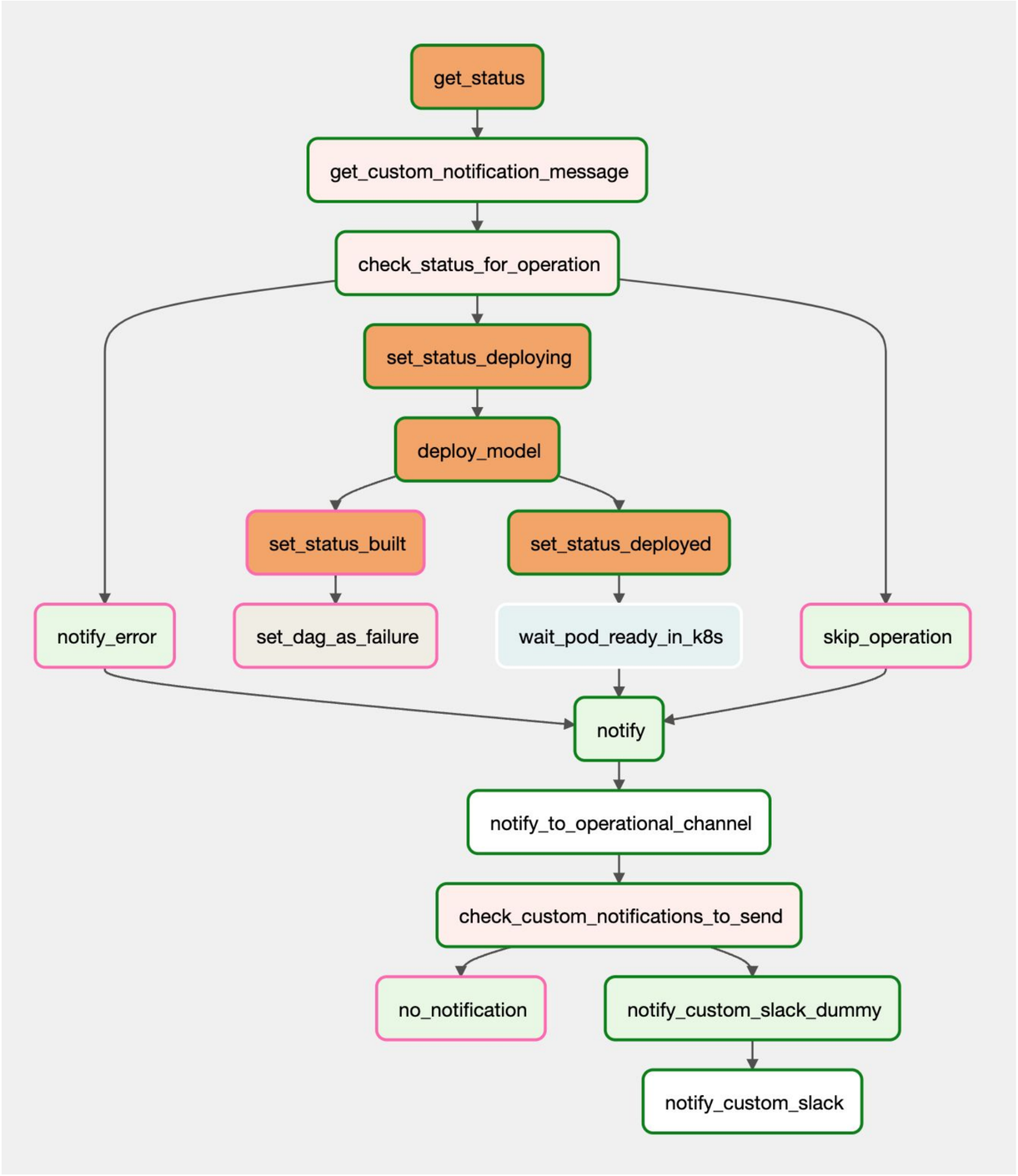
Deploying

In the final stage to put the model in production, we will use (you guessed it!) an Airflow DAG to:

- Create all the necessary Kubernetes components to make it reachable: service, autoscaling configuration and deployment.
- Once the Flask application load the Model in memory, it is ready to start attending prediction requests.

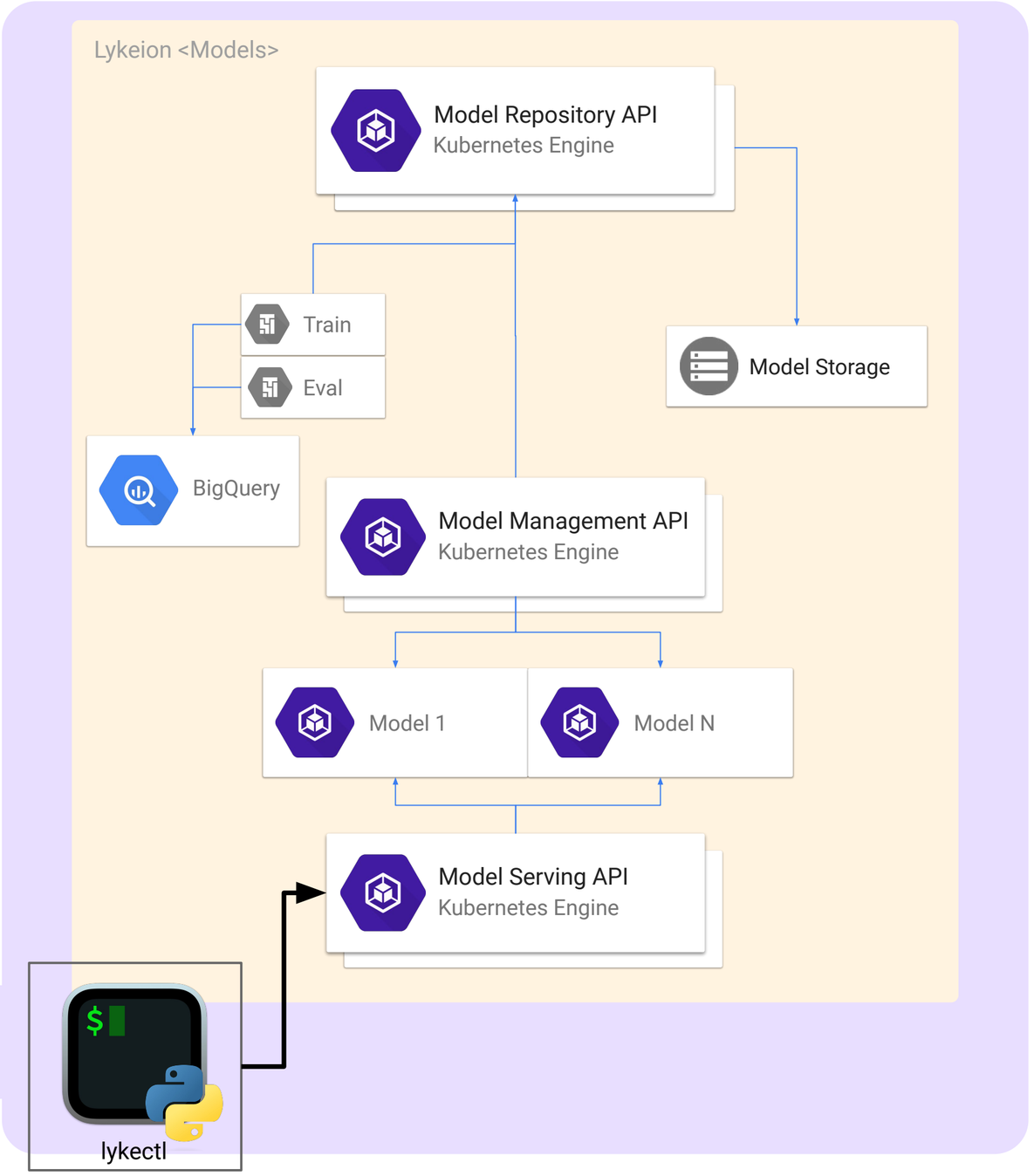



```
lykectl models build deploy optimus-price --segment default --version 1.0
```


get_status
get_custom_notification_message
check_status_for_operation
set_status_deploying
deploy_model
set_status_built
set_status_deployed
notify_error
set_dag_as_failure
wait_pod_ready_in_k8s
skip_operation
notify
notify_to_operational_channel
check_custom_notifications_to_send
no_notification
notify_custom_slack_dummy
notify_custom_slack

Requesting predictions

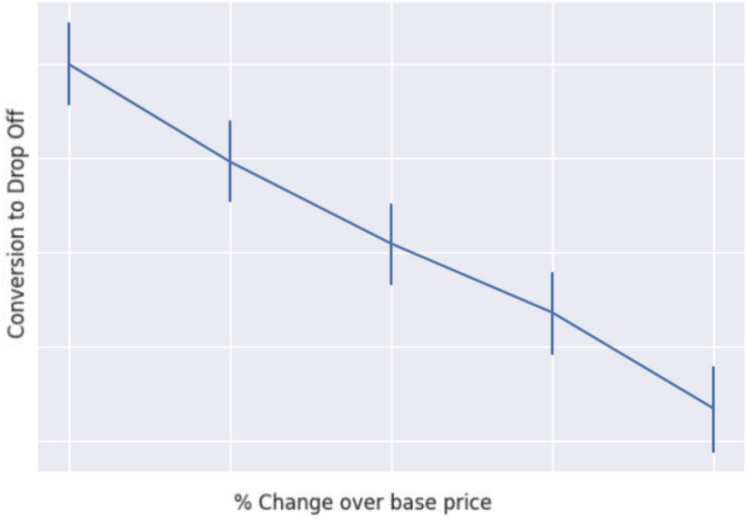
- Once a model is deployed, users can start requesting predictions using:
  - **lykectl**: good for testing or punctual usage.
  - **predict-api** (Model Serving): gateway service between all the deployed models and the outer world:
    - Standardize inputs.
    - Metric services and teams usage.
    - Store prediction audits.



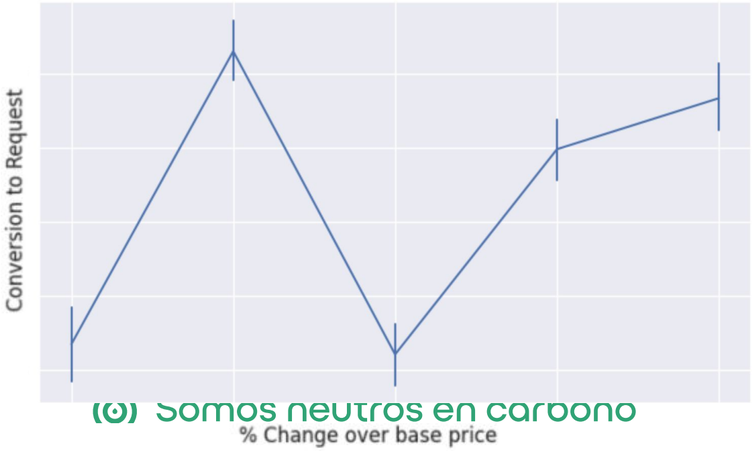

```
lykectl models predict optimus --segment default --version 1.0 --features
'{"lat_origin": -34.61315, "lat_destination": -34.61315, "region_id":
"ar_buenos_aires"}'
```

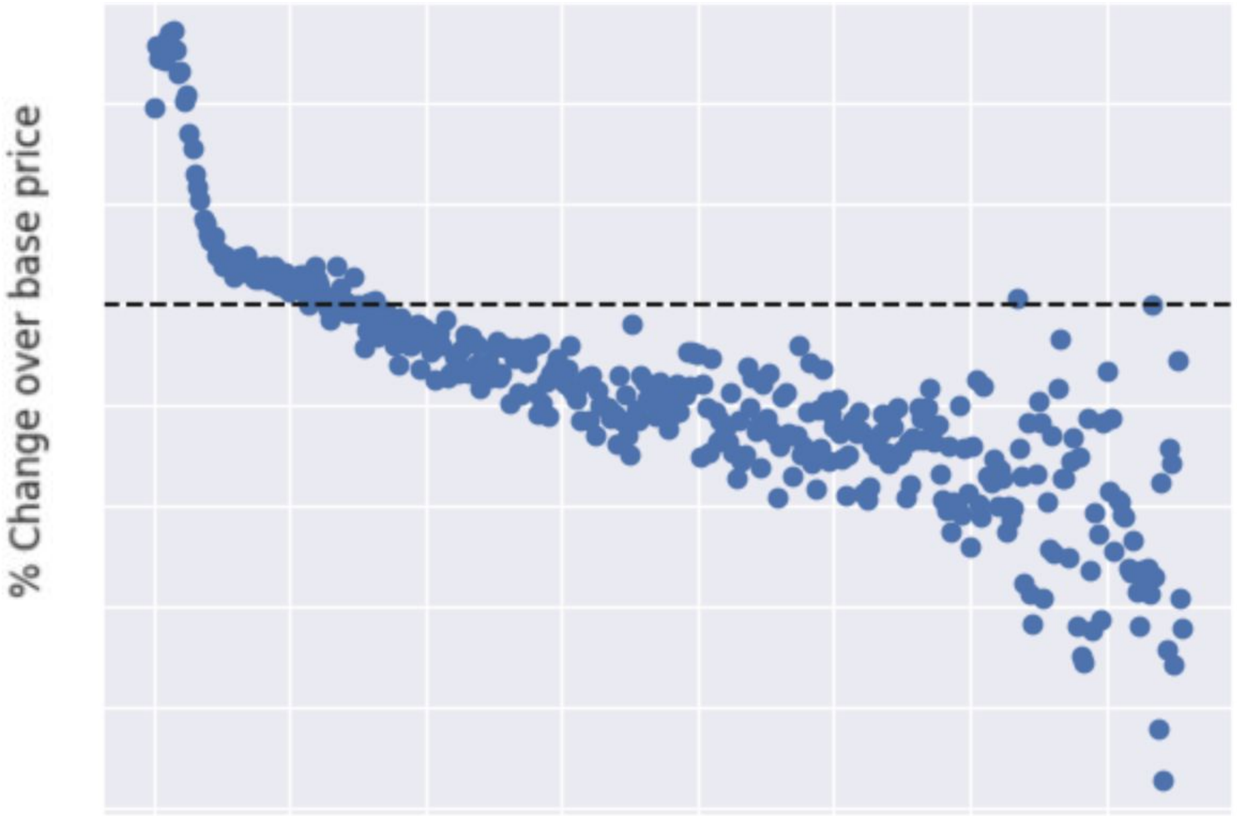
Experimental Results

Training

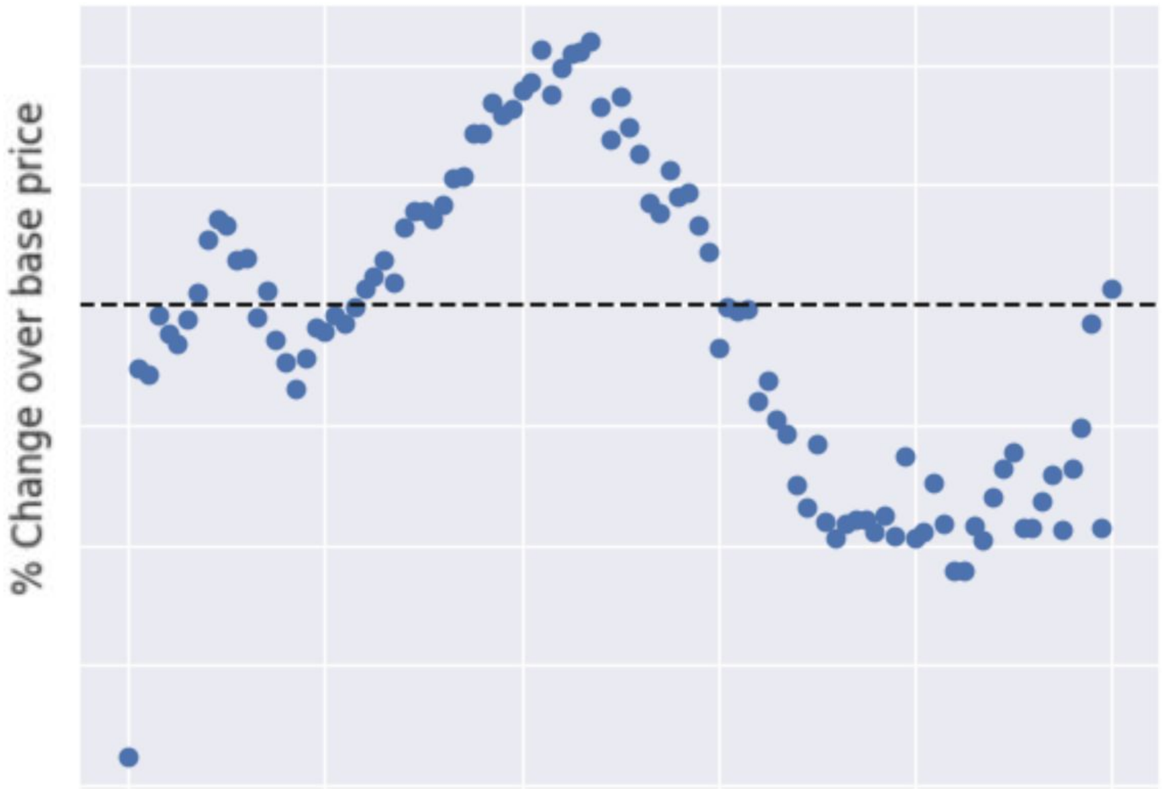


Algorithm





Distance

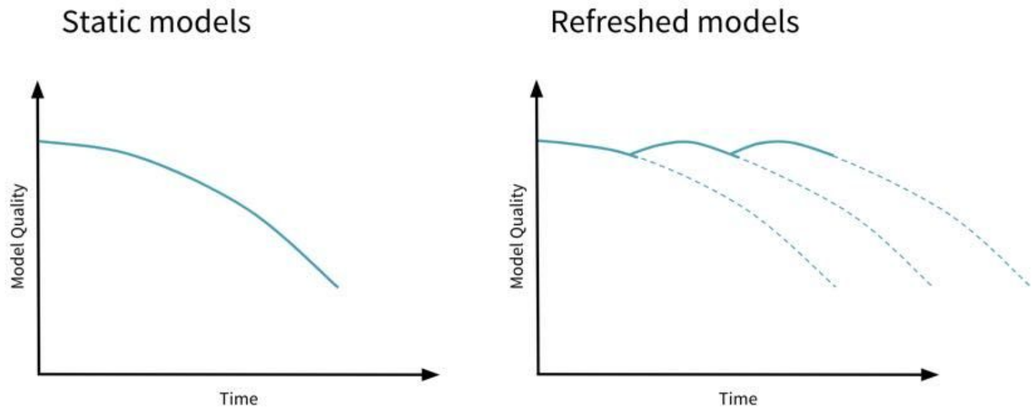


Saturation Index

Next steps

Next steps

- We want to stay on top of the performance of our models, to avoid any **degradation**. These are the main types of degradation in ML models:
  - **Concept drift**: changes in statistical properties of the target variable (e.g. new ways of fraud).
  - **Upstream data changes**: changes in the units, formats used in the clients of the model.
  - **Data drift**: changes in the training data source (e.g. seasonality, trends change).



The optimus price model could be prone to **data driftness**.

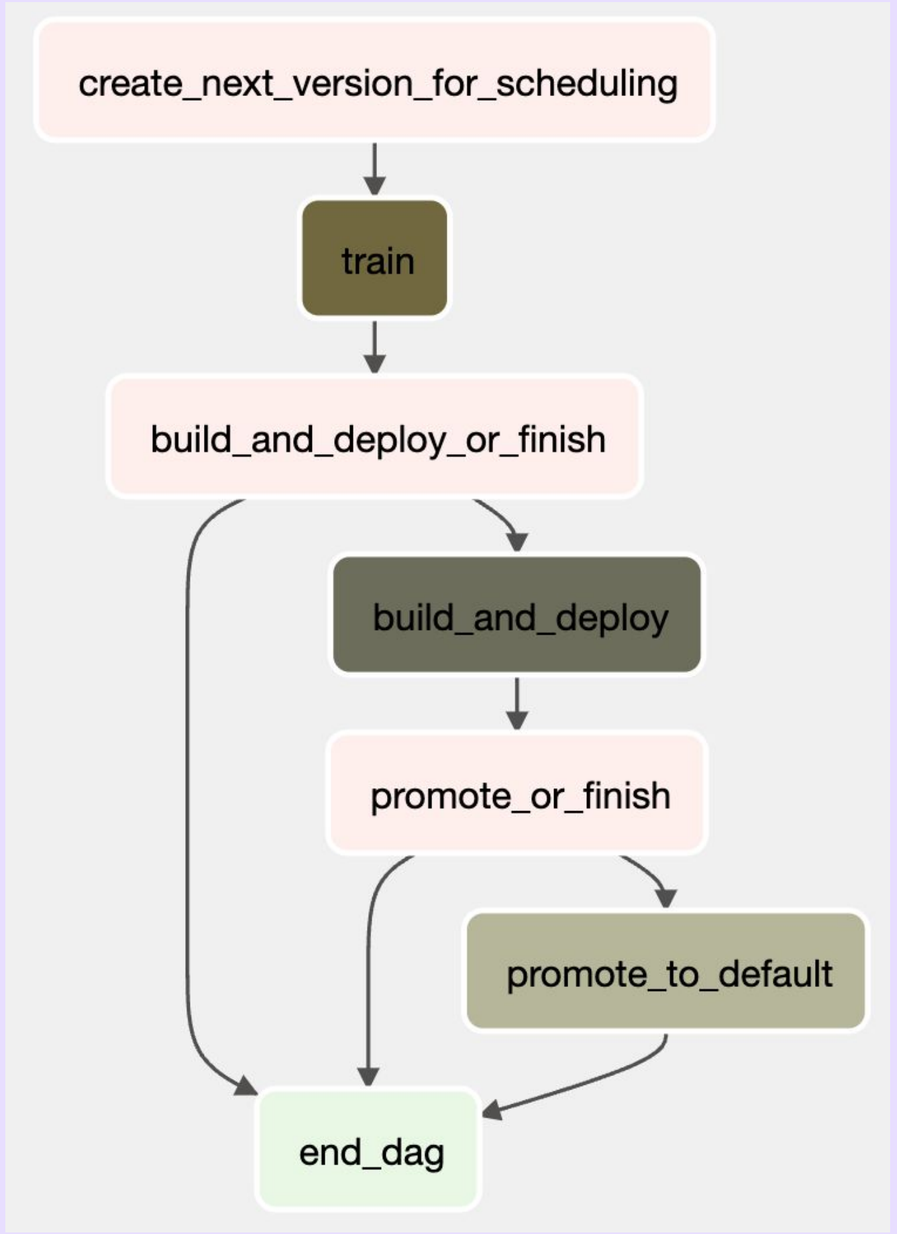
Next steps

Scheduling trainings and evaluations

Exporting metrics to Prometheus and Grafana

Schedule trainings

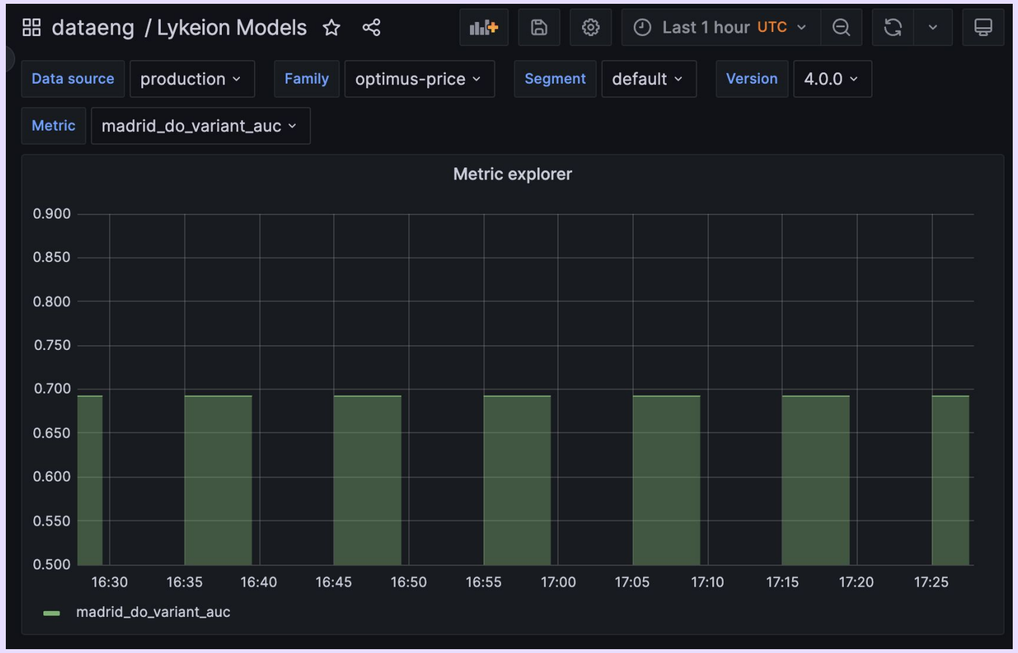
- We can set up an schedule to perform these operations periodically, going up to the deploy of new versions.
- By setting up default versions of a model, the downstream clients won't notice the *idol-swap*.
- The goal for optimus price is to train it once a day, so we can make it as close to recent scenarios as possible

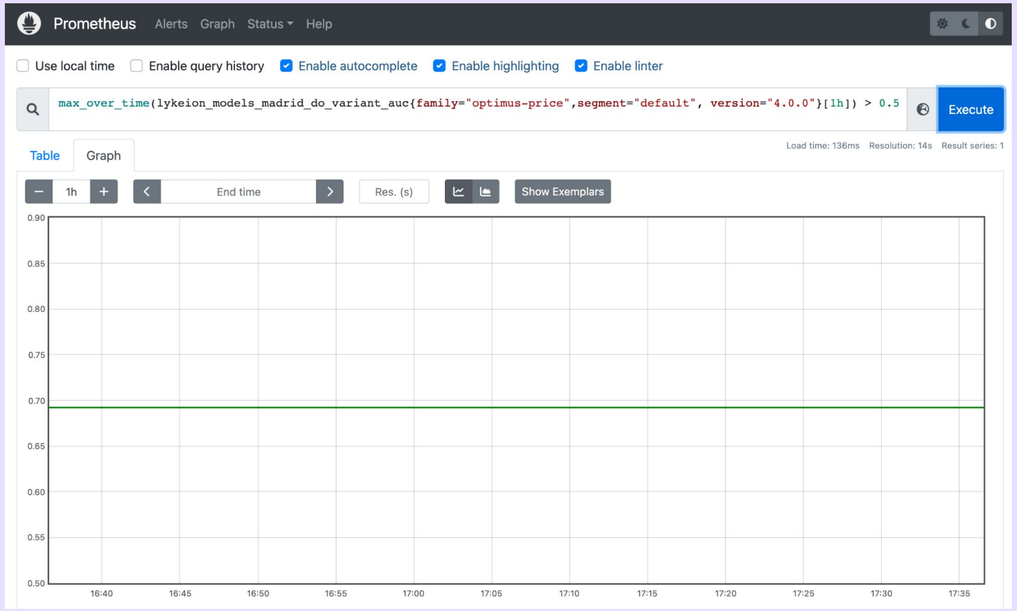


Exporting metrics

- Prometheus+Grafana are the *de facto* observability stack at Cabify.
- We have implemented an exporter, that requests data from the MLFlow Tracking Server.
- Data Scientists just need to use **mlflow.log_metric** or **mlflow.log_params** and they will reach the monitoring infra.
- This allows us to set up alerts, just as a regular production service.

- **Next big thing:** link these two -> whenever an alert gets triggered, we could launch a training.





Q&A

cabify

Thanks!

Somos neutros en carbono

References

- CatBoost:
  - Monotone Constraints: https://catboost.ai/en/docs/references/training-parameters/common#monotone_constraints
  - Multi-Quantile Loss: https://catboost.ai/en/docs/concepts/loss-functions-regression#MultiQuantile
  - Virtual Ensembles: https://arxiv.org/pdf/2006.10562.pdf