Scaling Machine Learning @ Careem

VITALII DUK

October, 2019

Careem is the Middle East's leading platform in transportation, delivery & payments.

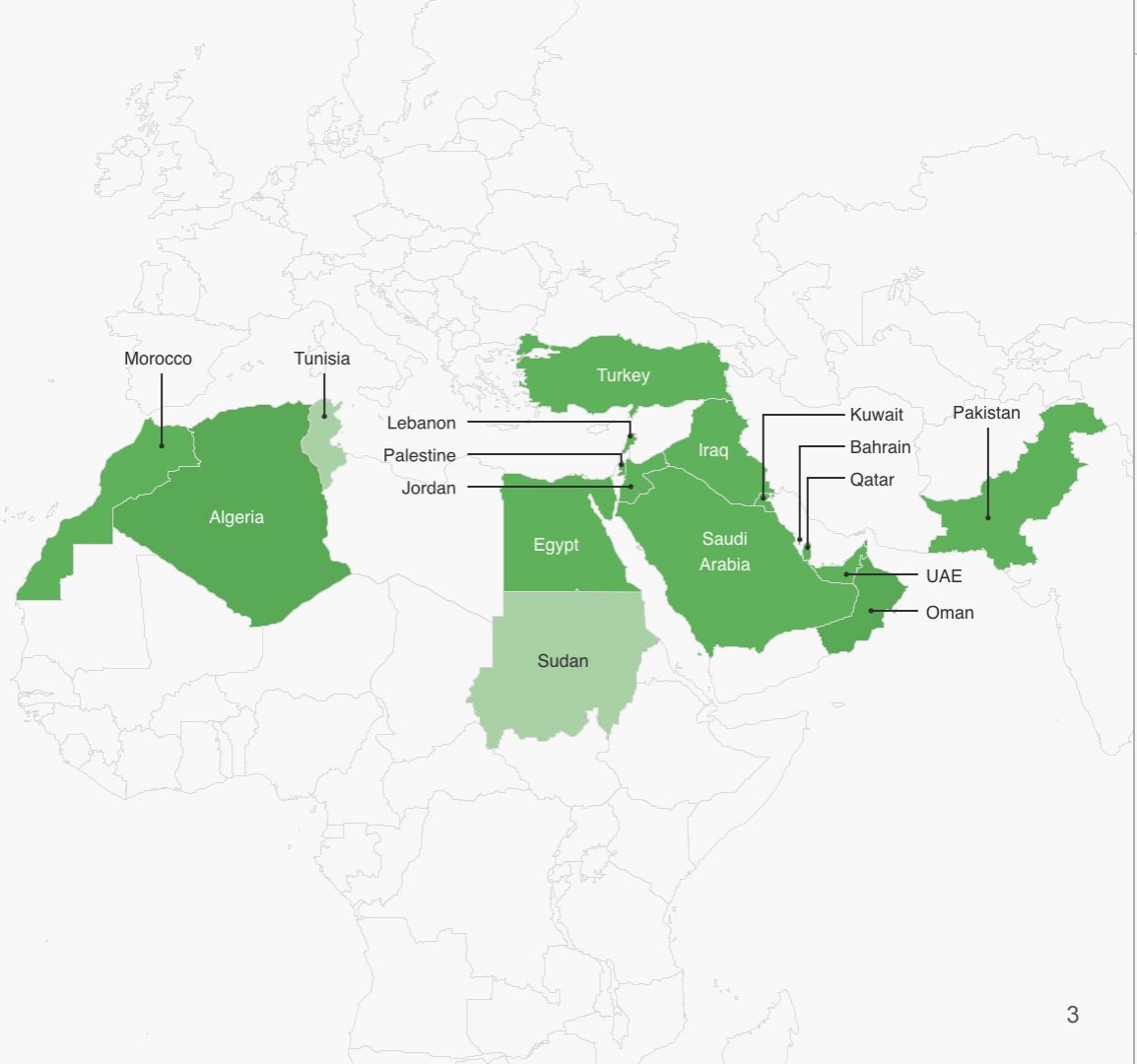
Our presence

125+
Cities

1M+
Captains

33M+
Users

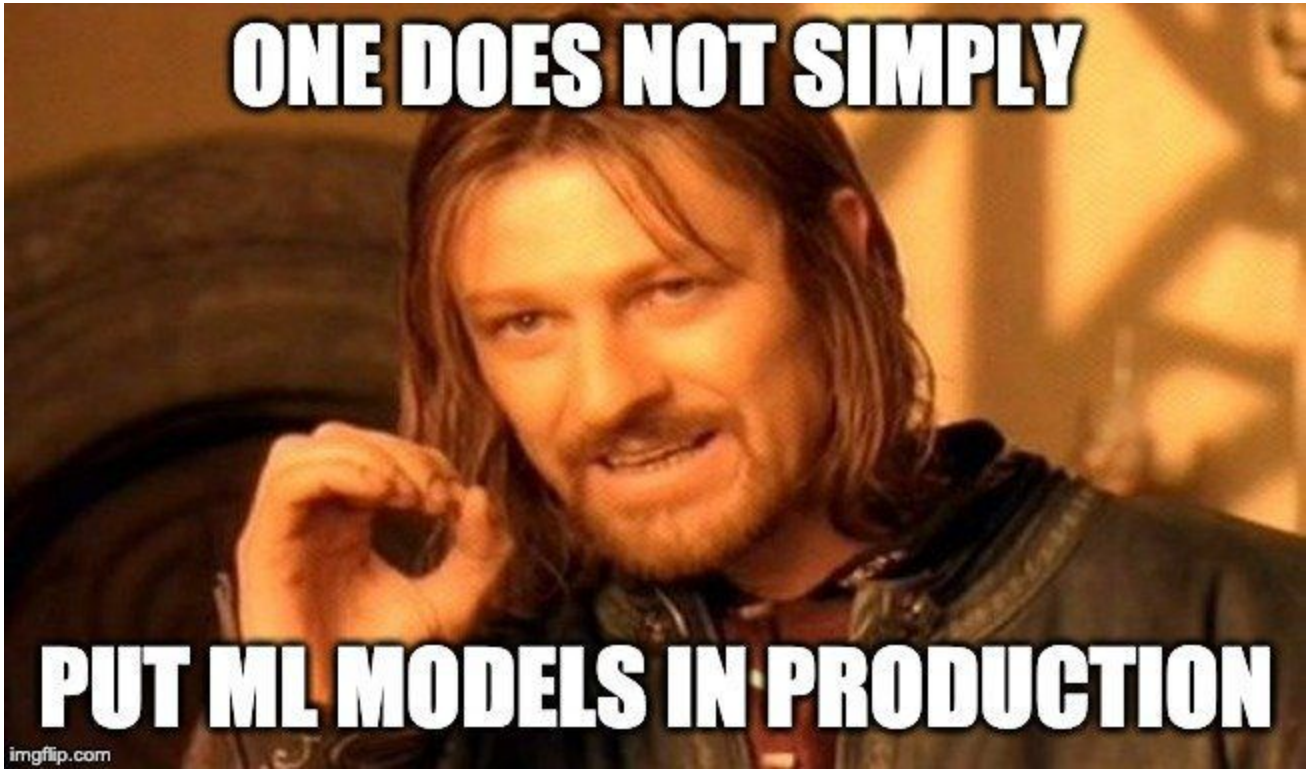
2000+
Full-time
Employees



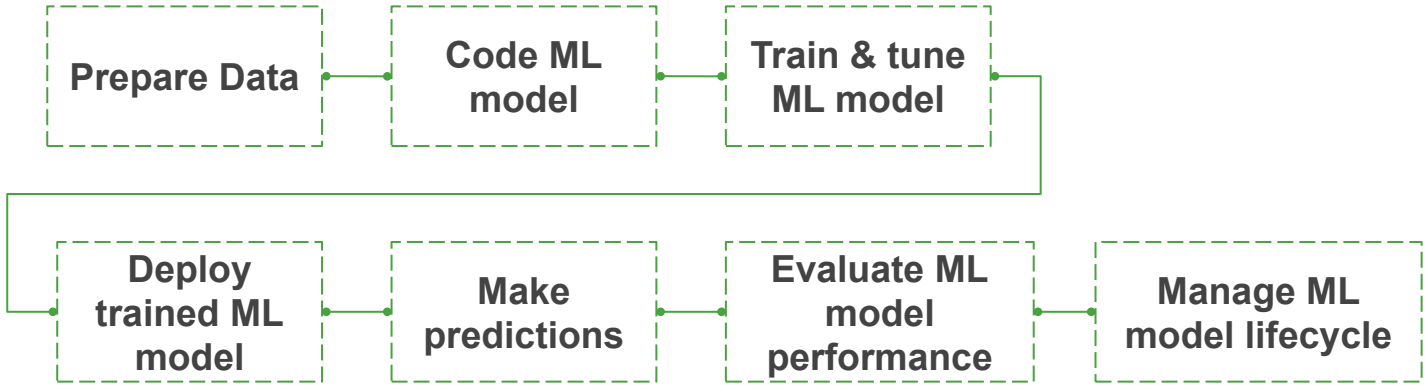
What’s hard about scaling Machine Learning?

Problem Statement

Problem Statement



Production ML workflow



Our Challenges

At Careem we've faced several issues while productizing ML models in the past:

- How to train & deploy a lot of **city-specific ML models** fast?
- How to **A/B test** the performance of several ML models running in parallel?
- How to perform **real-time monitoring** of hundreds of models running in production?

What is CMLP?

CMLP is an in-house built **Machine Learning Platform** designed to:

- **Automate** and **accelerate** the adoption of Machine Learning to support Careem's Platform vision
- **Reduce** the amount of **time** Data Scientists and Engineers need to deploy ML models into production
- Make Data Scientists **spend time more effectively**, working on more exciting tasks
- Ensure that ML models perform the same in **R&D and production environments**

CMLP today

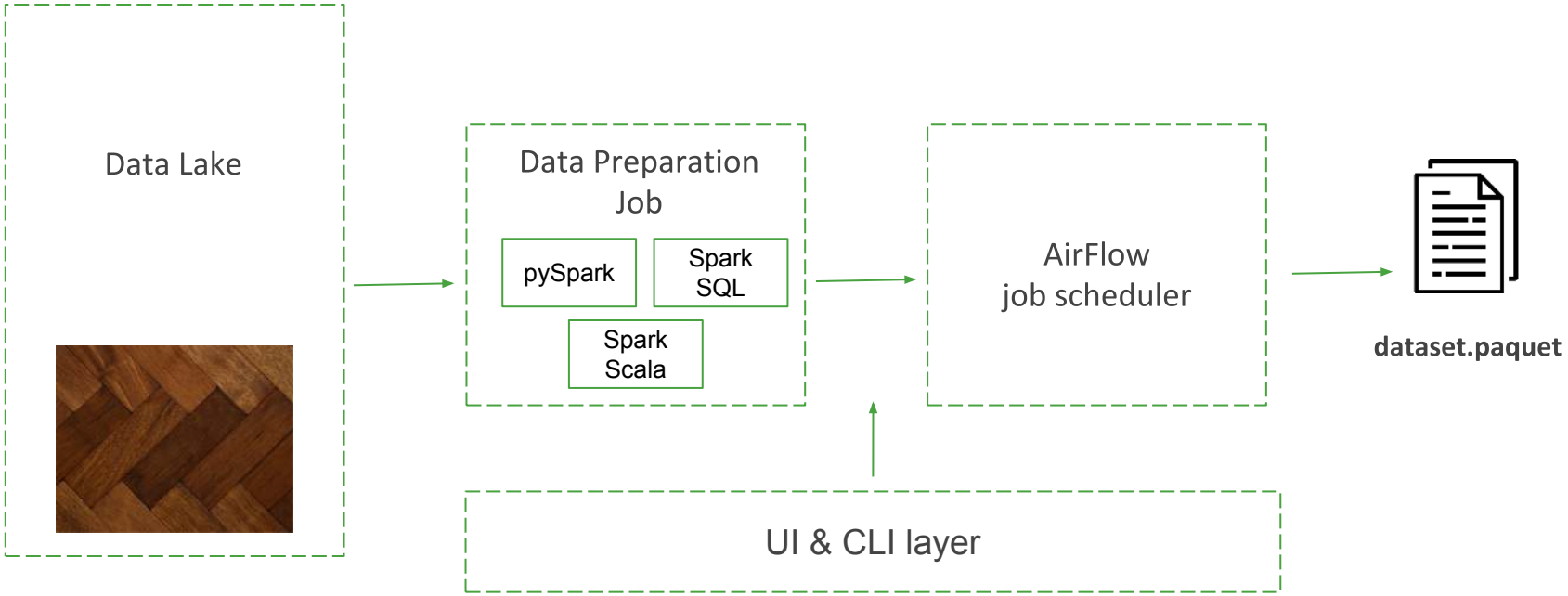
ML projects live

ML models in production

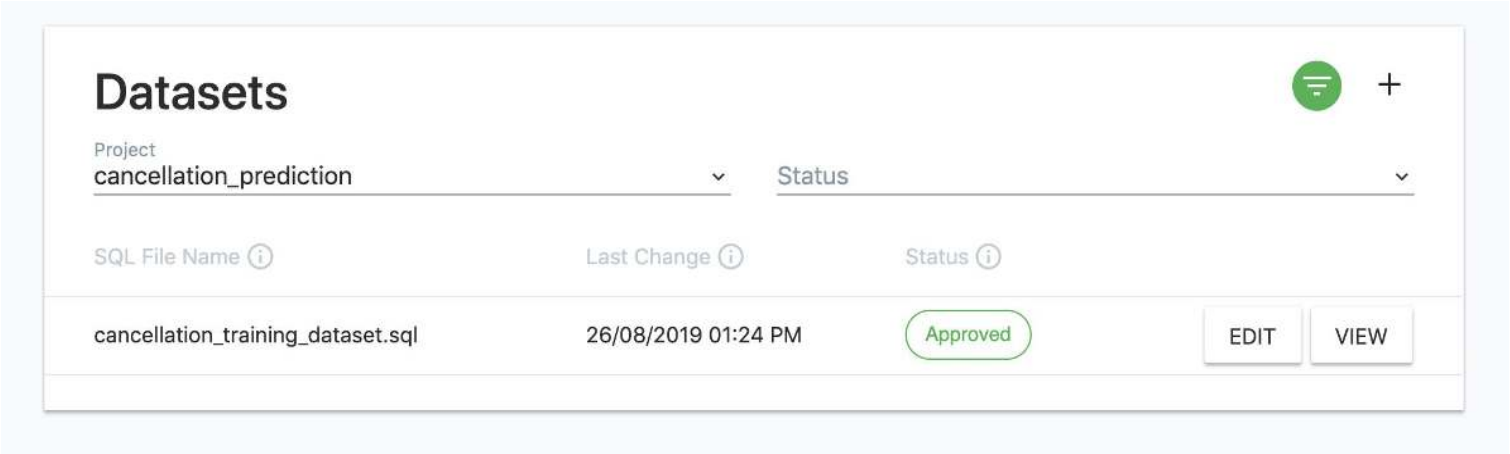
Predictions provided per day

Step 1: Data Preparation

Data Preparation



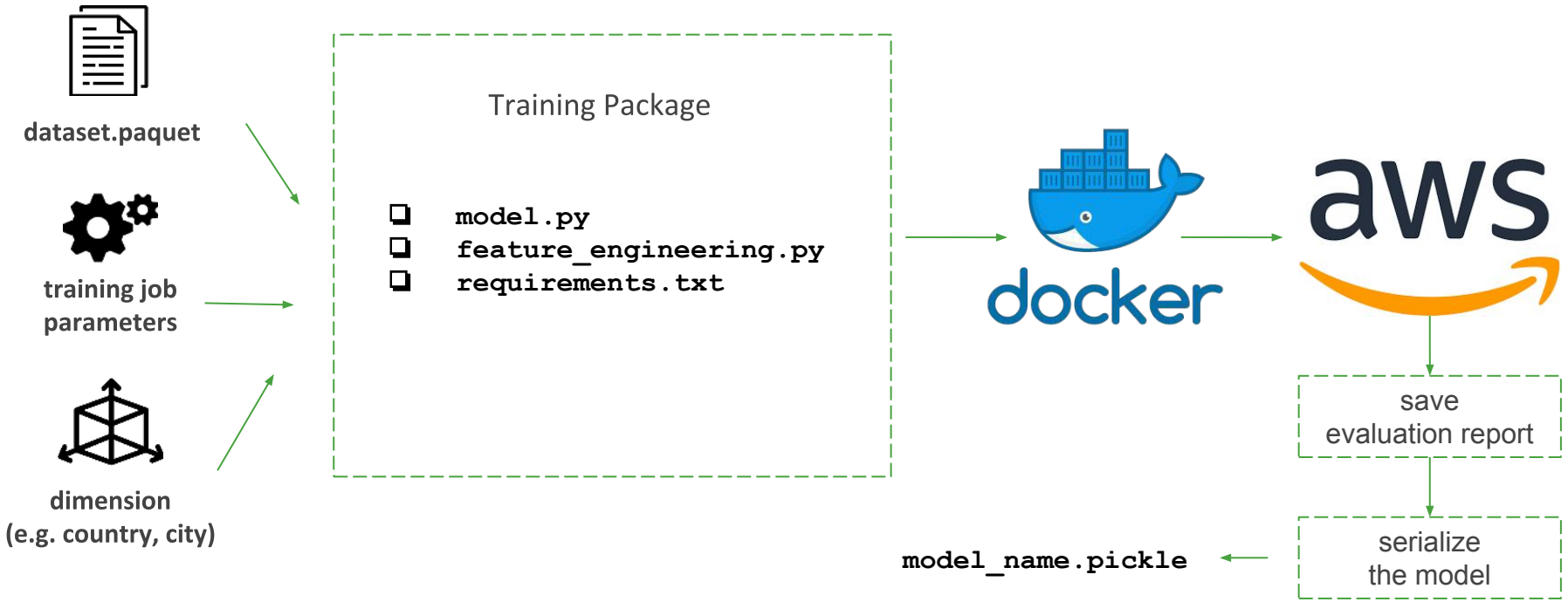
Data Preparation




```
~$ cmlp generate-dataset --project_name cancellation_prediction \
                         --sql_file cancellation_training_dataset.sql \
                         --dimensions '50,259' --options='{"days": 90}'
```

Step 2: Model Design & Training

Model Training

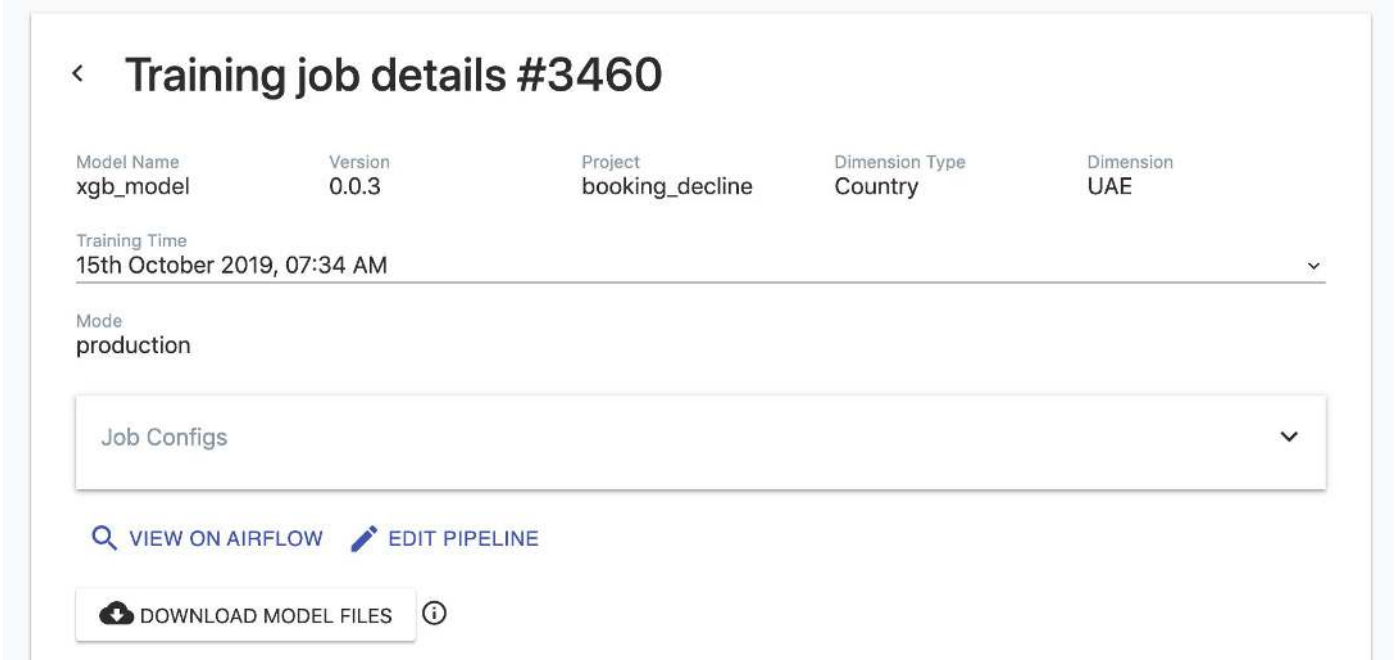


Model Training - code example

```
# Read generated dataset
training_input = TrainingInput('dataset.pickle')

# Define classification model parameters
model_parameters = {
    'algorithm':        CatBoostClassifier,
    'loss_function':    'Logloss',
    'iterations':       JobOptions.get_as_int('iterations', 1500),
    'learning_rate':    JobOptions.get_as_float('learning_rate', 0.04),
    'depth':            8,
    'use_best_model':   True,
    'one_hot_max_size': 10
}

# Job information
job_info = JobInfo(
    name=config['job_name'],
    project_name=config['project_name'],
    dimension_type=DimensionType.SERVICE_AREA,
    dimension=config['dimension']
)
```

```
# Model wrapper
catboost_model = CatBoostTrainer(**model_parameters)

trainer = Model(
    input_data=training_input,
    version='0.0.3',
    model_instance=catboost_model,
    job_info=job_info
)

# Run model training
trainer.run()

# Report metrics
reporter = trainer.get_reporter()
reporter.evaluate_metrics(
    metric_calculator_list=[roc_auc_score],
    target='fraud_probability'
)
```

Model Training - UI



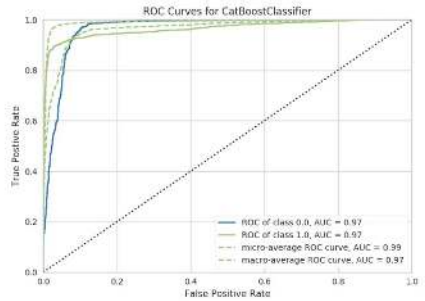
Model Training - metrics

Training Results

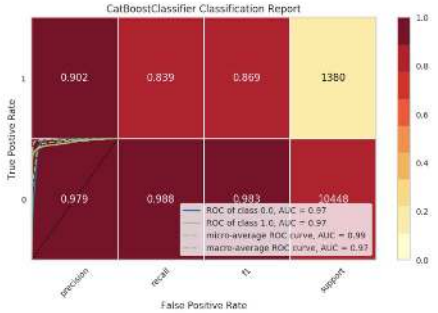
Metrics

roc_auc_score	decline_probability	0.965702	

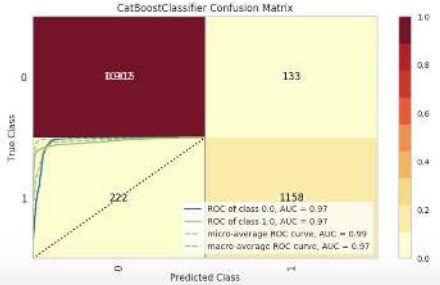
ROCAUC



Classification Report

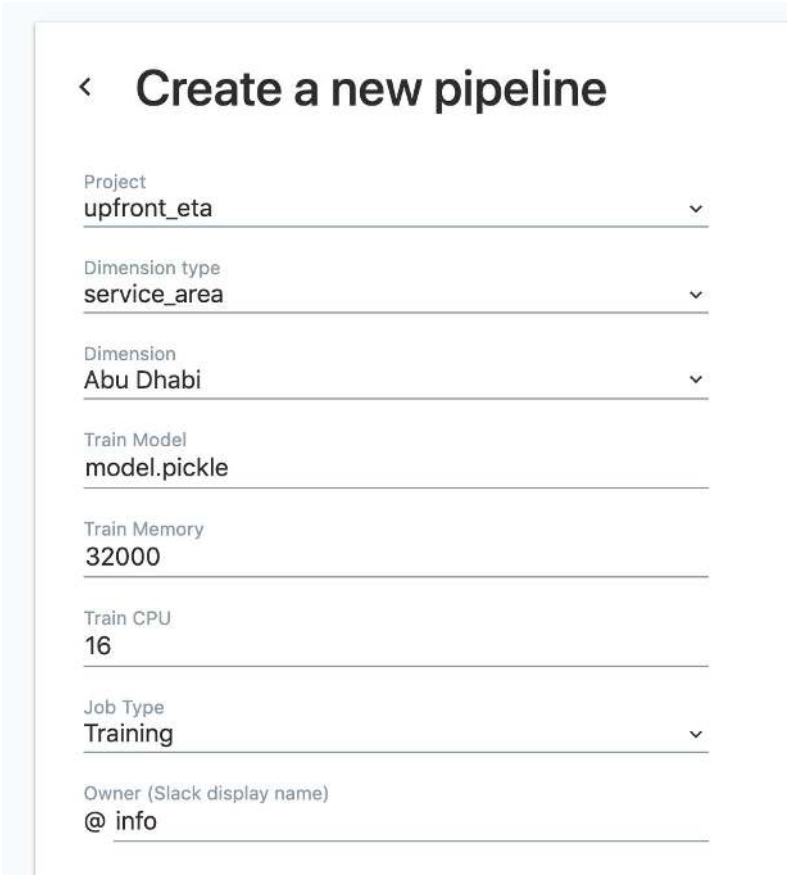


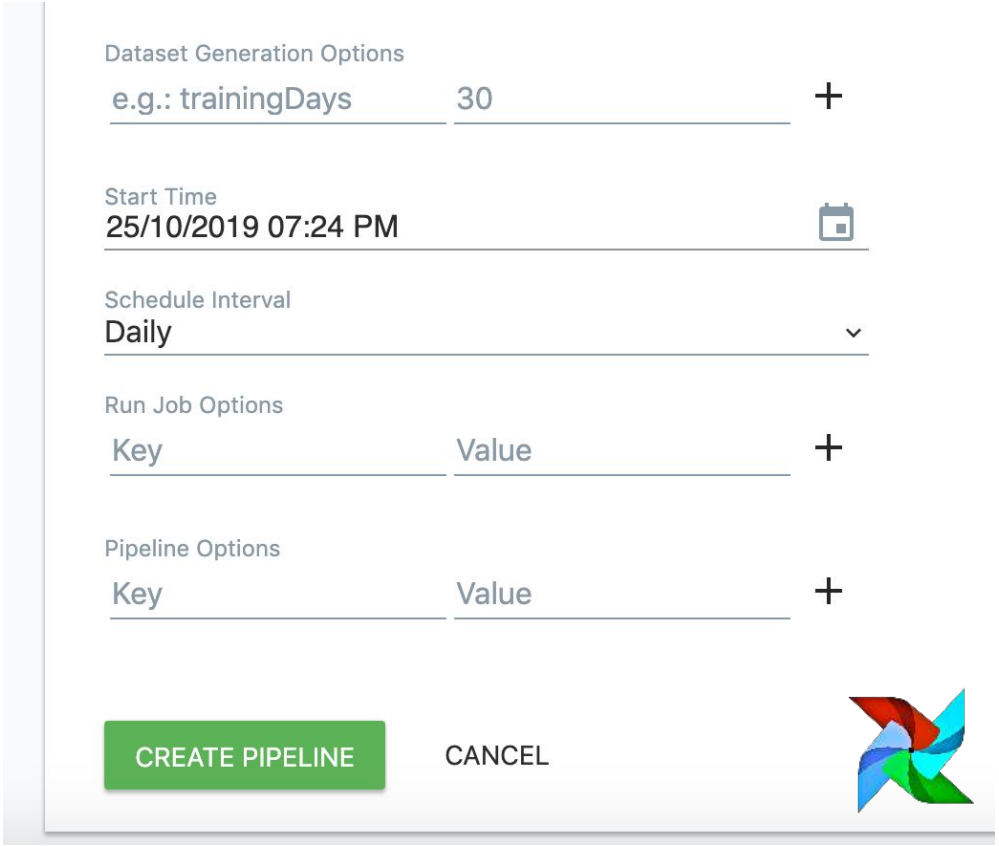
Confusion Matrix



Step 3: Scheduling Model Training

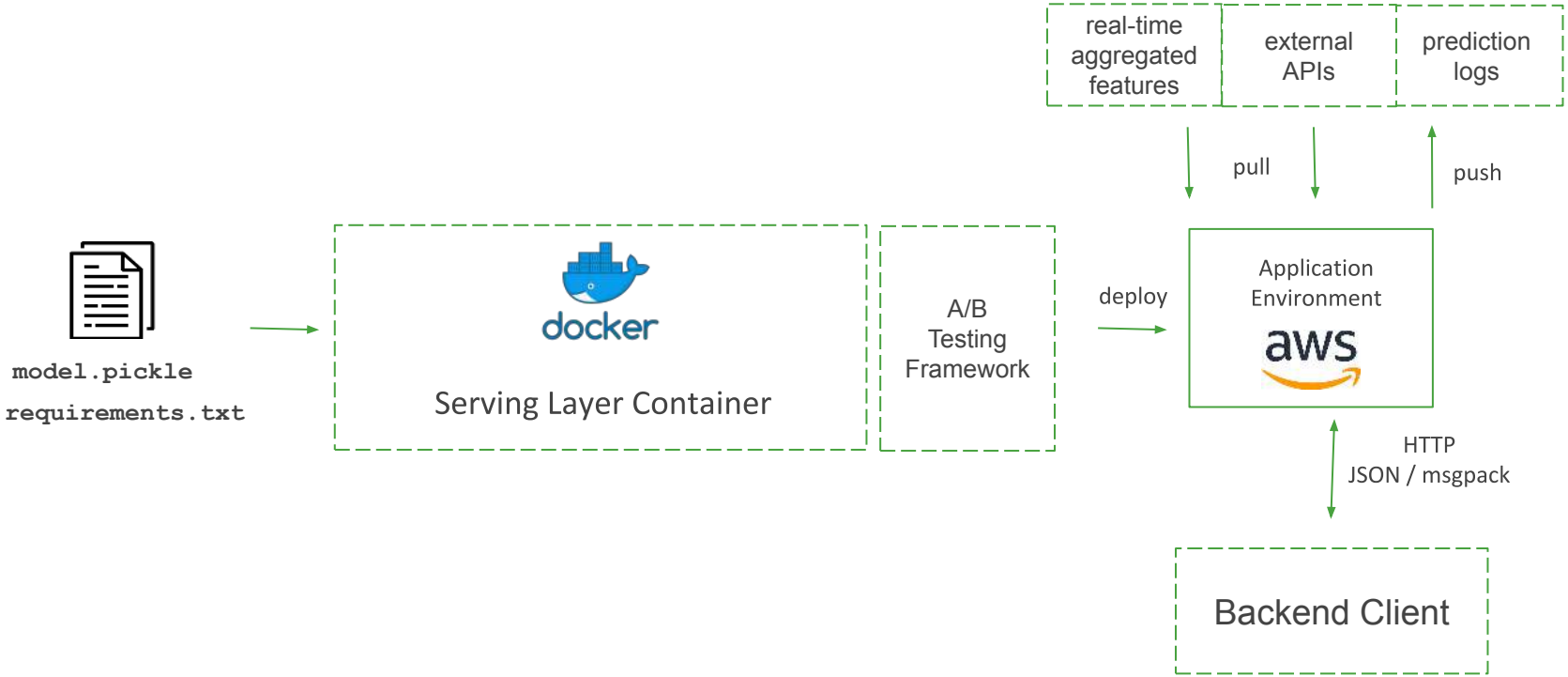
Training Scheduler - UI



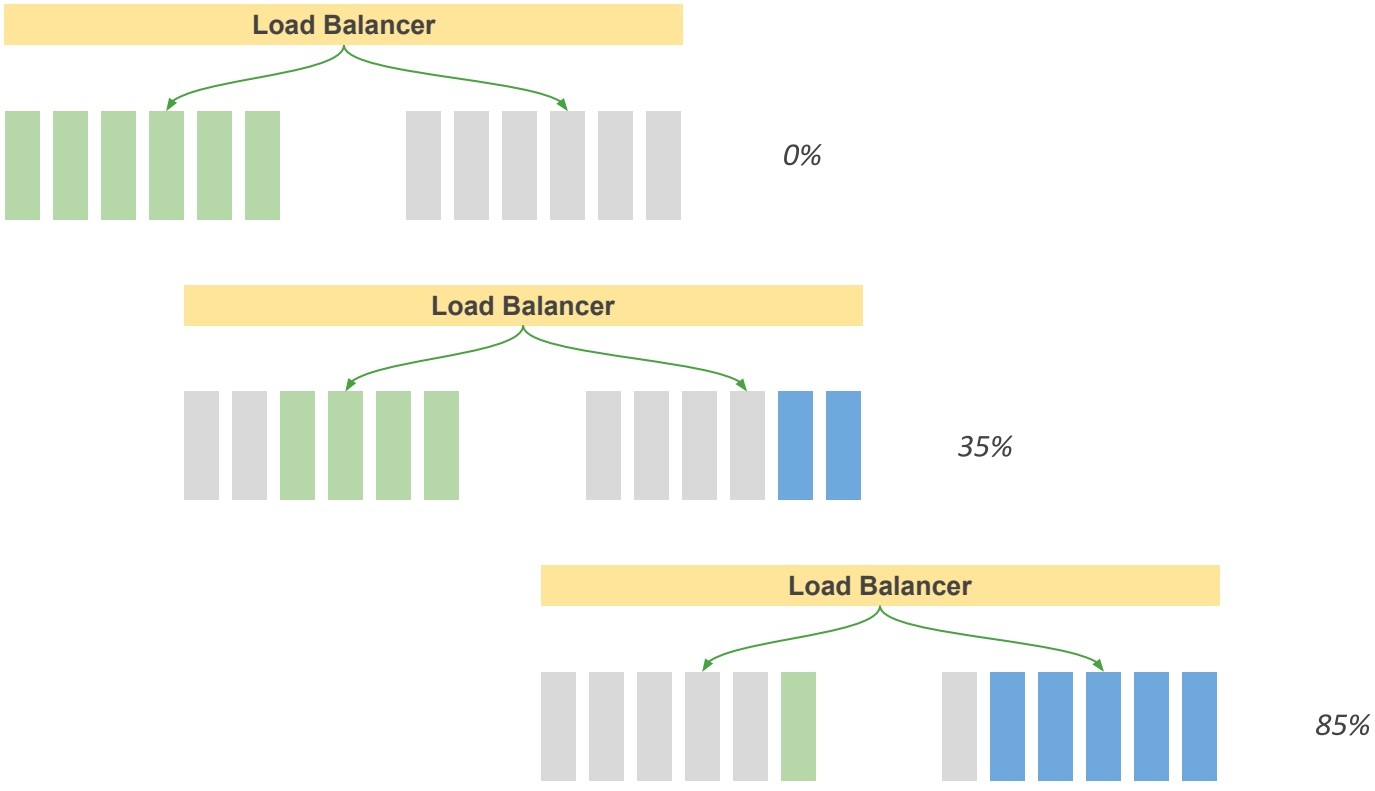


Step 4: Deploying Trained Model

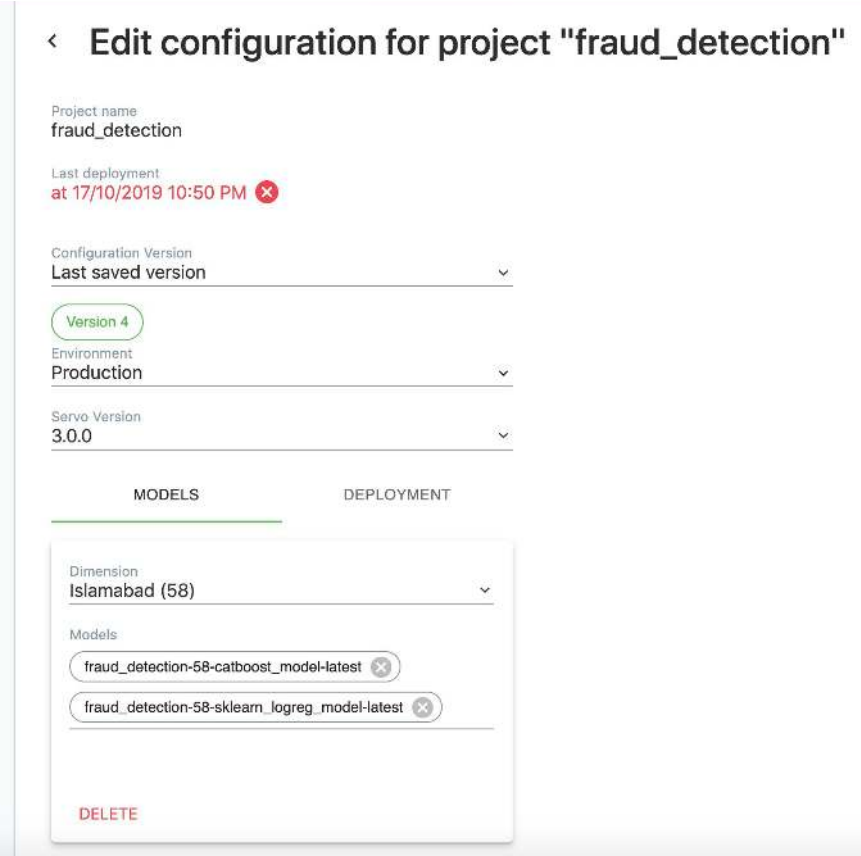
Model Serving



Model Serving - canary deployment



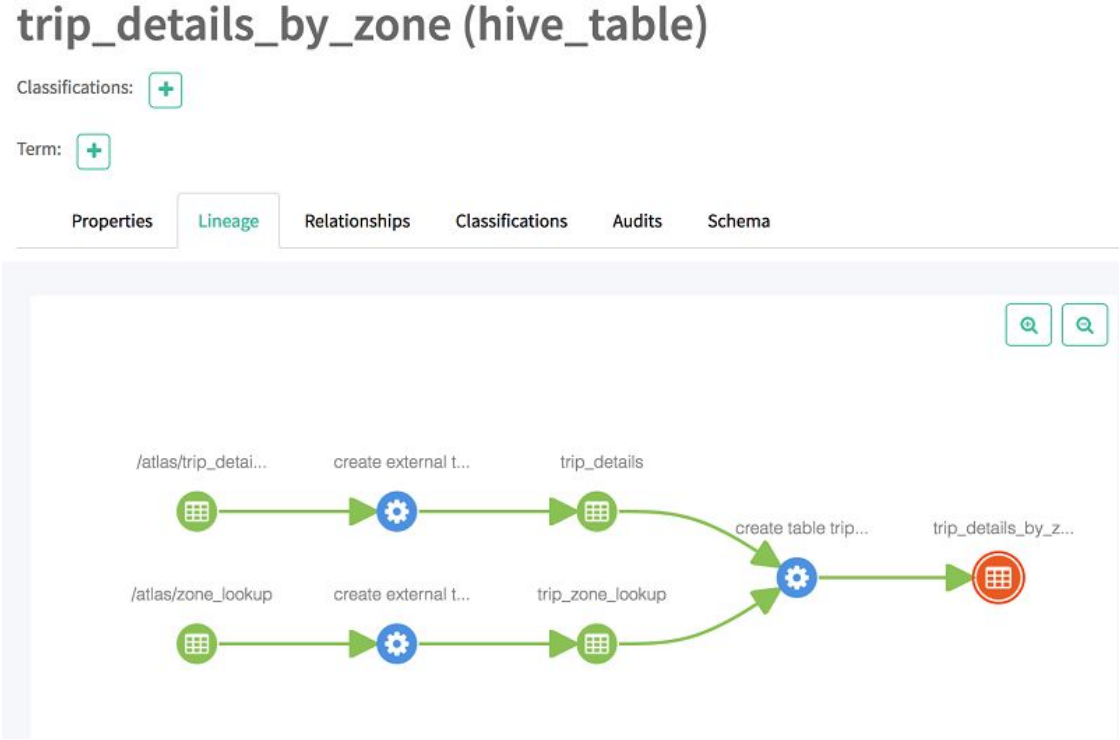
Model Serving - UI



Metadata Governance

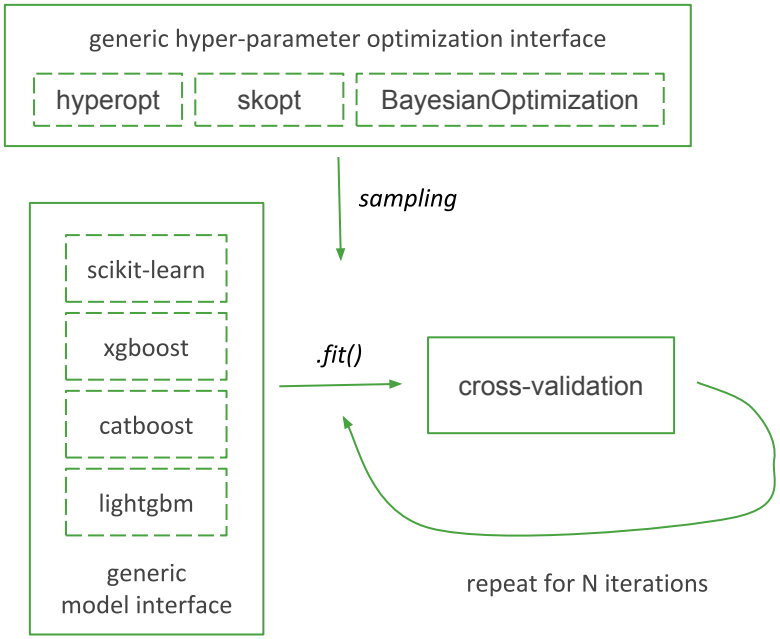
Metadata Governance

We are using Apache Atlas as the metadata management and governance tool to maintain a catalog of our data assets.



Hyper-Parameter Optimization (HPO)

Hyper-parameter optimization



HPO - code examples

```
# Read generated dataset
training_input = TrainingInput('dataset.pickle')


# Job information
job_info = JobInfo(
    name=config['job_name'],
    project_name=config['project_name'],
    dimension_type=DimensionType.SERVICE_AREA,
    dimension=config['dimension']
)

# Model wrapper
catboost_hpo_model = HpoCatBoostTrainer(
    problem_type='classification'
)

trainer = Model(
    input_data=training_input,
    version='0.0.4',
    model_instance=catboost_hpo_model,
    job_info=job_info
)

# Run model training
trainer.run()

# Report metrics
reporter = trainer.get_reporter()
reporter.evaluate_metrics(
    metric_calculator_list=[roc_auc_score],
    target='fraud_probability'
)
```

HPO - code examples

```python
# Read generated dataset
training_input = TrainingInput('dataset.pickle')


# Job information
job_info = JobInfo(
    name=config['job_name'],
    project_name=config['project_name'],
    dimension_type=DimensionType.SERVICE_AREA,
    dimension=config['dimension']
)

# Model wrapper
catboost_hpo_model = HpoCatBoostTrainer(
    problem_type='classification'
)

trainer = Model(
    input_data=training_input,
    version='0.0.4',
    model_instance=catboost_hpo_model,
    job_info=job_info
)

# Run model training
trainer.run()

# Report metrics
reporter = trainer.get_reporter()
reporter.evaluate_metrics(
    metric_calculator_list=[roc_auc_score],
    target='fraud_probability'
)
```

```python
# Read generated dataset
training_input = TrainingInput('dataset.pickle')

# Define classification model parameters
model_parameters = {
    'algorithm':        CatBoostClassifier,
    'loss_function':    'Logloss',
    'iterations':       JobOptions.get_as_int('iterations', 1500),
    'learning_rate':    JobOptions.get_as_float('learning_rate', 0.04),
    'depth':            8,
    'use_best_model':   True,
    'one_hot_max_size': 10
}

# Job information
job_info = JobInfo(
    name=config['job_name'],
    project_name=config['project_name'],
    dimension_type=DimensionType.SERVICE_AREA,
    dimension=config['dimension']
)

# Model wrapper
catboost_model = CatBoostTrainer(**model_parameters)

trainer = Model(
    input_data=training_input,
    version='0.0.3',
    model_instance=catboost_model,
    job_info=job_info
)

# Run model training
trainer.run()

# Report metrics
reporter = trainer.get_reporter()
reporter.evaluate_metrics(
    metric_calculator_list=[roc_auc_score],
    target='fraud_probability'
)
```

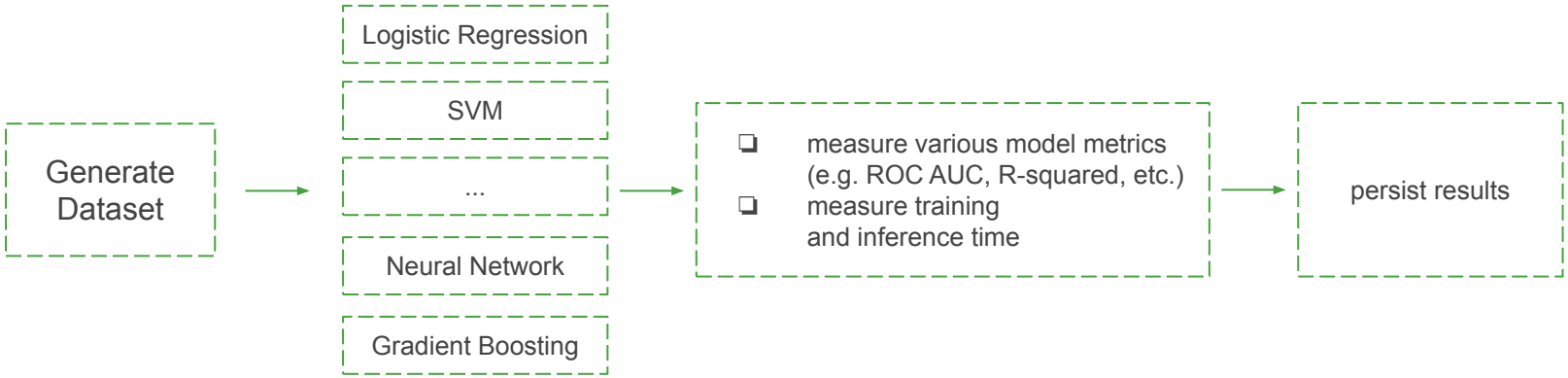
HPO - code examples

```python
# Read generated dataset
training_input = TrainingInput('dataset.pickle')


# Job information
job_info = JobInfo(
    name=config['job_name'],
    project_name=config['project_name'],
    dimension_type=DimensionType.SERVICE_AREA,
    dimension=config['dimension']
)

# Model wrapper
catboost_hpo_model = HpoCatBoostTrainer(
    problem_type='classification'
)

trainer = Model(
    input_data=training_input,
    version='0.0.4',
    model_instance=catboost_hpo_model,
    job_info=job_info
)

# Run model training
trainer.run()

# Report metrics
reporter = trainer.get_reporter()
reporter.evaluate_metrics(
    metric_calculator_list=[roc_auc_score],
    target='fraud_probability'
)
```

```python
# Read generated dataset
training_input = TrainingInput('dataset.pickle')

# Define classification model parameters
model_parameters = {
    'algorithm':        CatBoostClassifier,
    'loss_function':    'Logloss',
    'iterations':       JobOptions.get_as_int('iterations', 1500),
    'learning_rate':    JobOptions.get_as_float('learning_rate', 0.04),
    'depth':            8,
    'use_best_model':   True,
    'one_hot_max_size': 10
}

# Job information
job_info = JobInfo(
    name=config['job_name'],
    project_name=config['project_name'],
    dimension_type=DimensionType.SERVICE_AREA,
    dimension=config['dimension']
)

# Model wrapper
catboost_model = CatBoostTrainer(**model_parameters)

trainer = Model(
    input_data=training_input,
    version='0.0.3',
    model_instance=catboost_model,
    job_info=job_info
)

# Run model training
trainer.run()

# Report metrics
reporter = trainer.get_reporter()
reporter.evaluate_metrics(
    metric_calculator_list=[roc_auc_score],
    target='fraud_probability'
)
```

AutoML

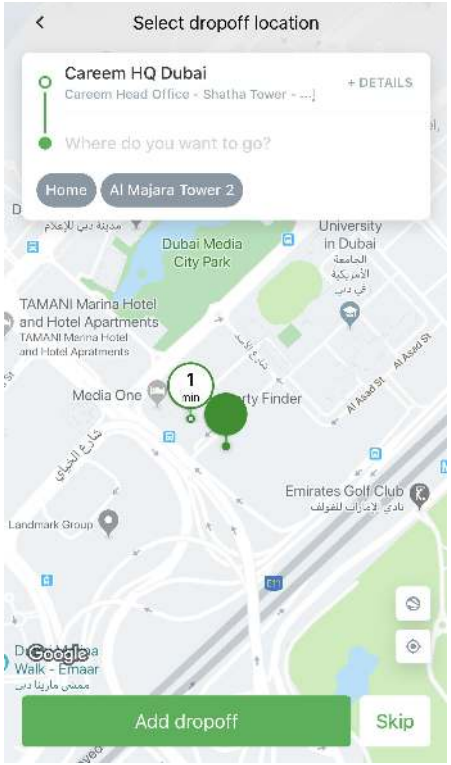
AutoML




```
~$ cmlp automl --project_name fraud_detection \
               --problem_type classification \
               --dataset fraud_dataset.sql
```

AI use-cases @ Careem

Suggested drop-off locations



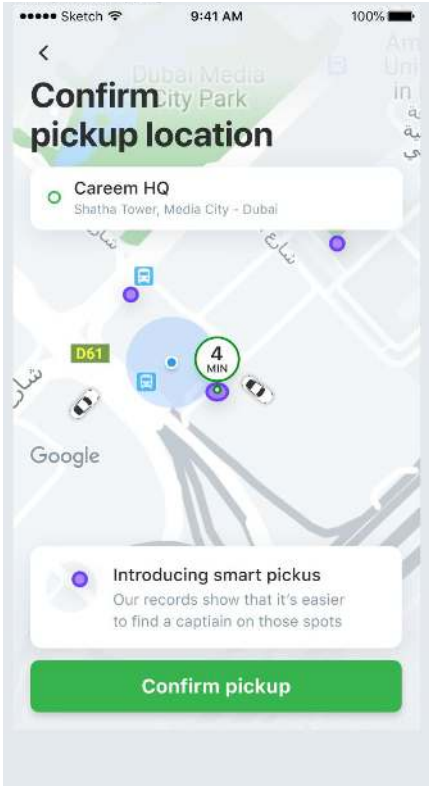
Why?

- better user experience
- reduced time to booking
- fewer cancellations

How?

- supervised ML algorithms
- trained on historical trips data
- >200 user, location and time-related factors used

Meeting points



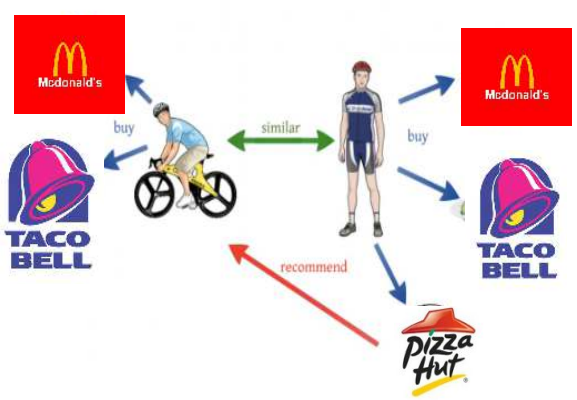
Why?
Customer and Captain can't find each other,

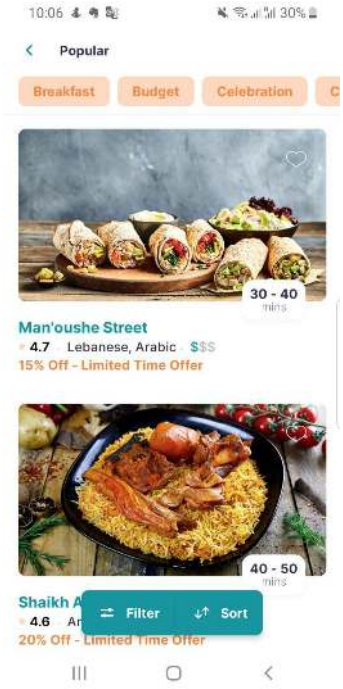
which **has a direct impact** on:
- cancellation rate
- contact rate
- the actual time of arrival (ATA)

and **indirect impact** on:
- retention
- billable availability (BA)
- customer throughput rate (CTP)

How?
- unsupervised ML algorithms
- based on historical data about “successful” pickups / drop-offs

Careem NOW - recommendation engine





Why?

- provide personalized experience: no burgers for vegetarians
- improve conversion, or secondary metrics keeping the conversion the same

How?

- collaborative filtering algorithms trained on historical user's behavior

Moving Forward

- Continue working on more sophisticated model **auto rollout** features
- Enhance **AutoML** capabilities by adding **automated feature engineering** (e.g. **featuretools**)
- Supercharge model **drift detection**
- Improve the latency of tree-based ML models using binary compilers (e.g. **treelite**)
- Move more features to the **UI**, to improve the UX
- Add **permutation importance** as a default tracked metric

Shukran! Thank you! Shukriya! Merci!