SPARK+AI SUMMIT EUROPE

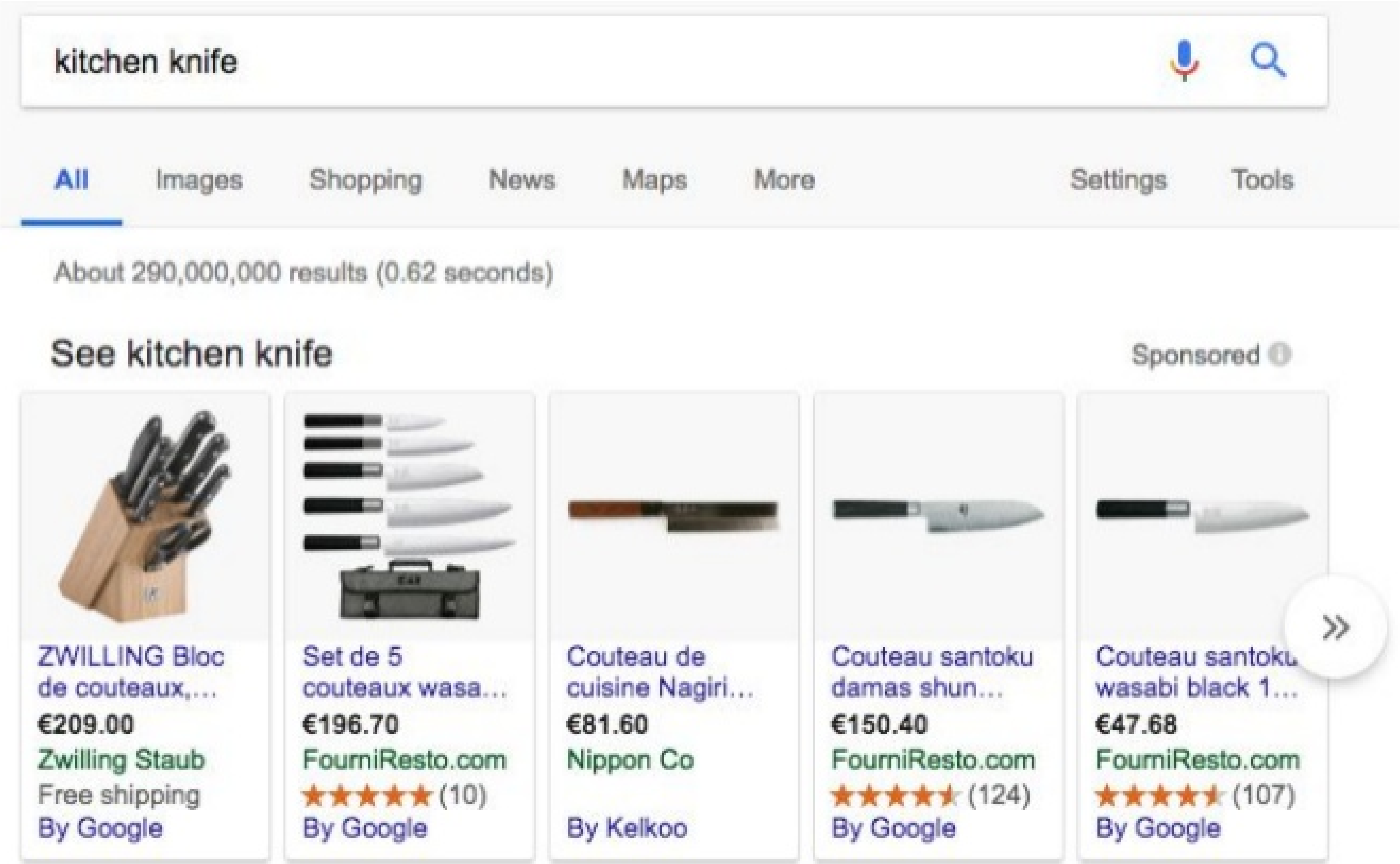
Machine Learning for AdTech in Action

Cyrille Dubarry
Han Ju

#SAISML11

AdTech introduction

Search and Performance

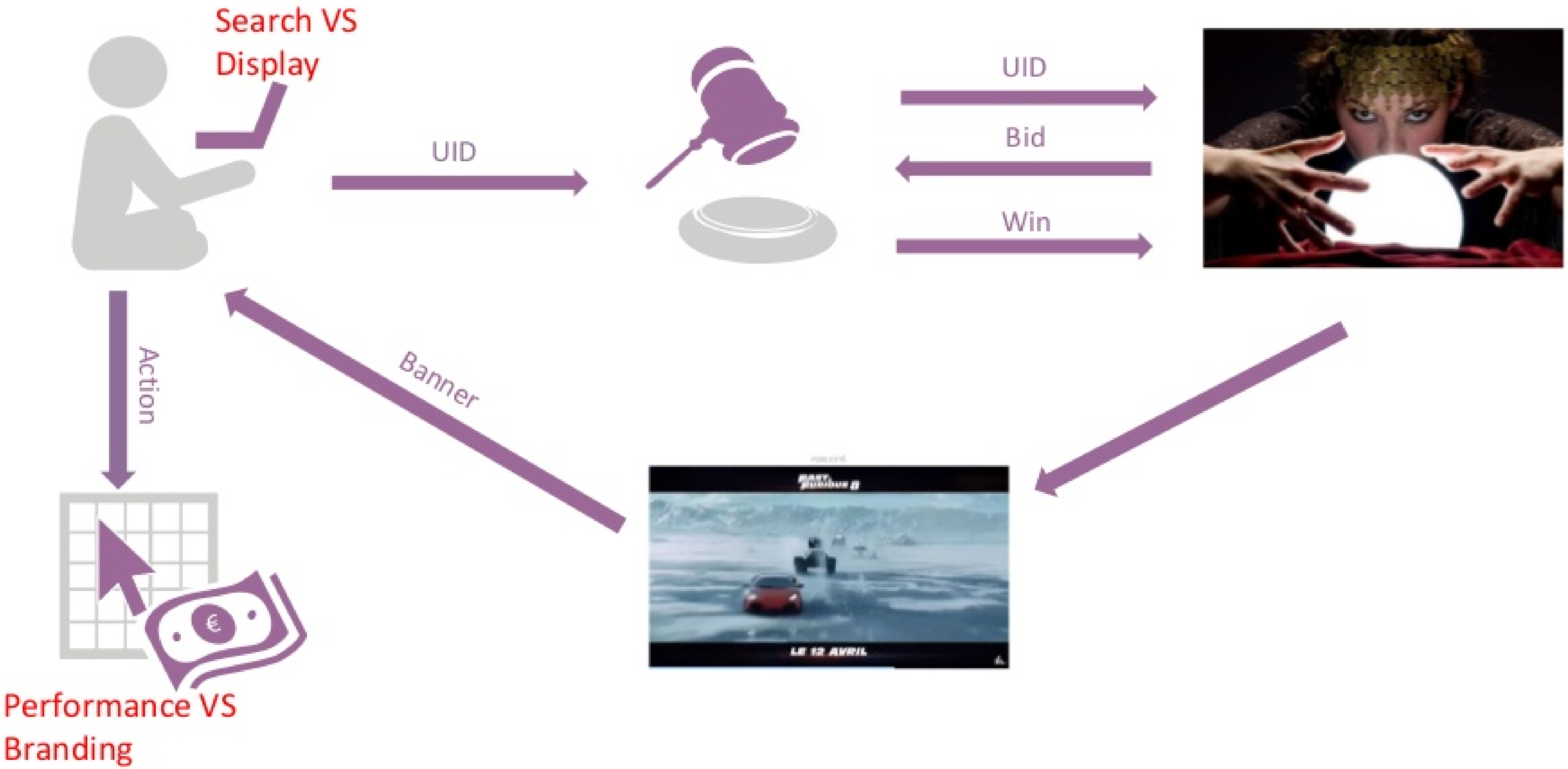


Video/Display and Branding

Cette première phase des négociations fait l'objet d'un *« rapport conjoint »* de quinze pages qui revient sur les éléments concernant les trois dossiers jugés prioritaires par Bruxelles et Londres : le règlement financier de la séparation, les droits des citoyens expatriés et la gestion de la frontière entre la république d'Irlande et la province britannique d'Irlande du Nord.

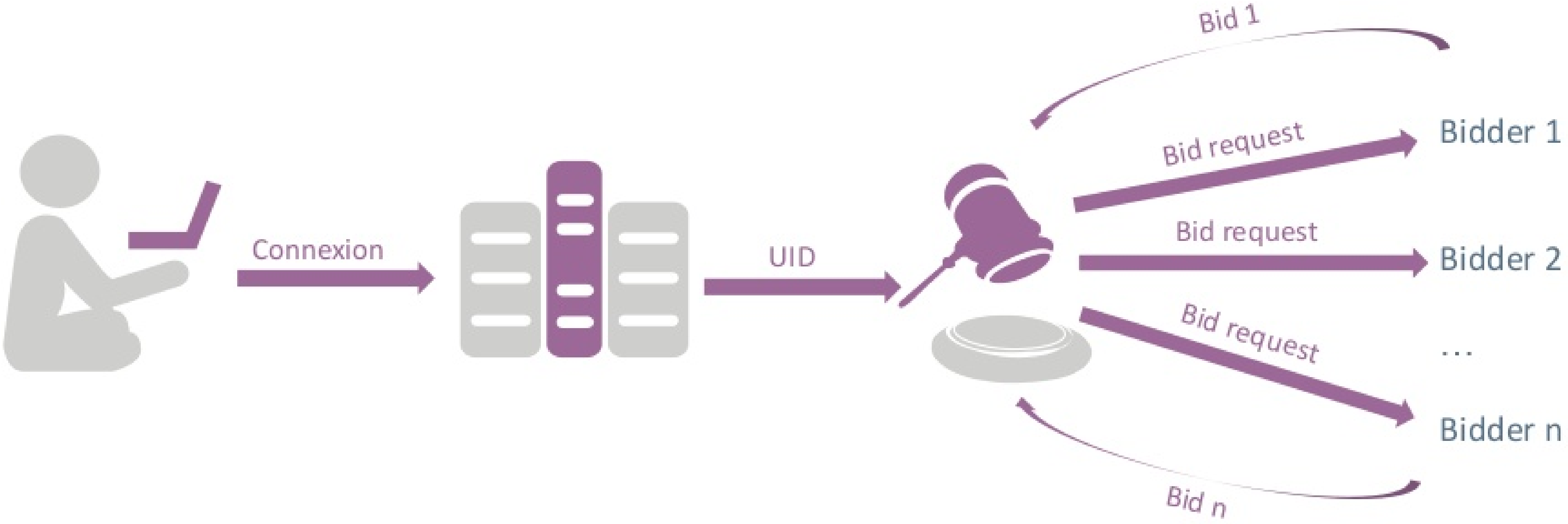
Le dossier est désormais entre les mains du Conseil européen, l'instance qui regroupe les dirigeants des Etats membres : ces derniers devront valider l'accord lors d'un sommet à Bruxelles, les 14 et 15 décembre.

Workflow of a user browsing



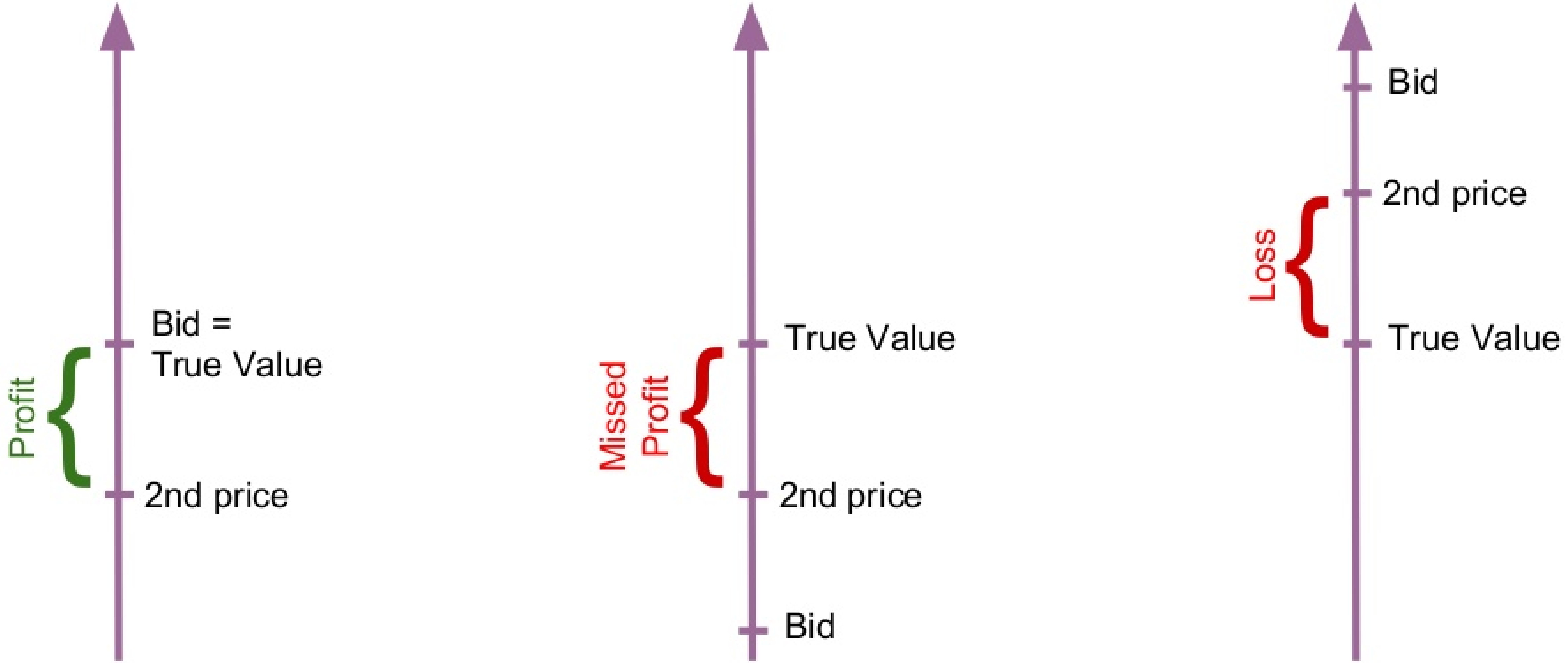
Real Time Bidding

RTB workflow



What does the winner pay?

2nd price auction



True Value for Video Branding

Assumption: the bidder is paid a fixed amount CPV if and only if the video ad is watched entirely.

True Value?

$$\mathbb{E}[Revenue] = \mathbb{E}[CPV \times 1_{watched}] = CPV \times \mathbb{P}\{Watched\}$$

Estimating the True Value

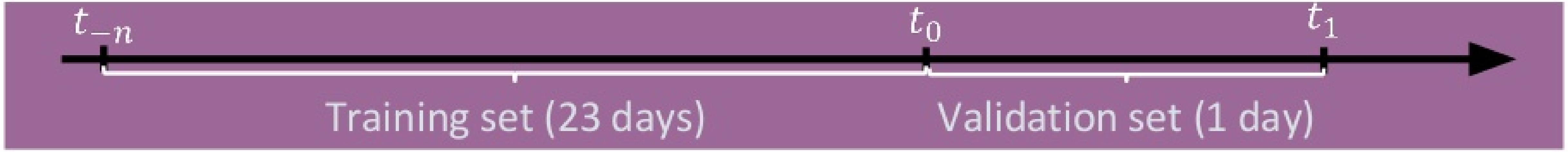
Which constraints?

- Features: several hundred thousands modalities

- Training frequency: several times a day

- Number of predictions: a million per second

Logistic regression VS DNN



Validation metric:

$$-LLH = -\sum_{i=1}^{N} O_i ln(p_i) + (1 - O_i) ln(1 - p_i)$$

Prediction model	Score = -LLH	Training time	Prediction complexity
Logistic regression	0.1293	70 min	40
Logistic regression with crosses	0.1272	142 min	86
Deep learning (11 layers x 1062 neurons) 1 epoch	0.1257	26h	40x1062+10x1062^2+1062 = 11 321 982
Deep learning (11 layers x 1062 neurons) 3 epochs	0.1250	78h	11 321 982

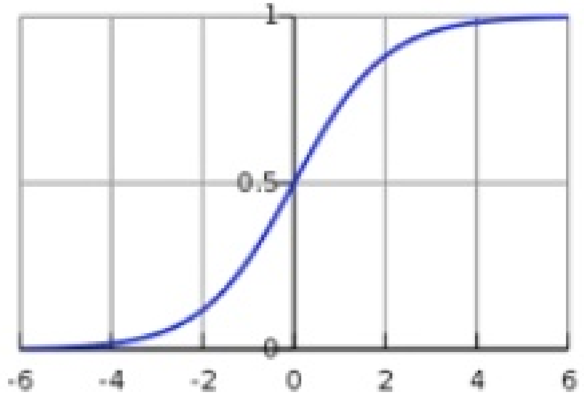
https://cloud.google.com/blog/big-data/2017/02/using-google-cloud-machine-learning-to-predict-clicks-at-scale

Reminder: Logistic Regression

$$\mathbb{P}\{Watched|x\} = \sigma(< w.x >) = \frac{1}{1 + e^{-<w.x>}}$$

x: features including historical and contextual information

w: model parameters

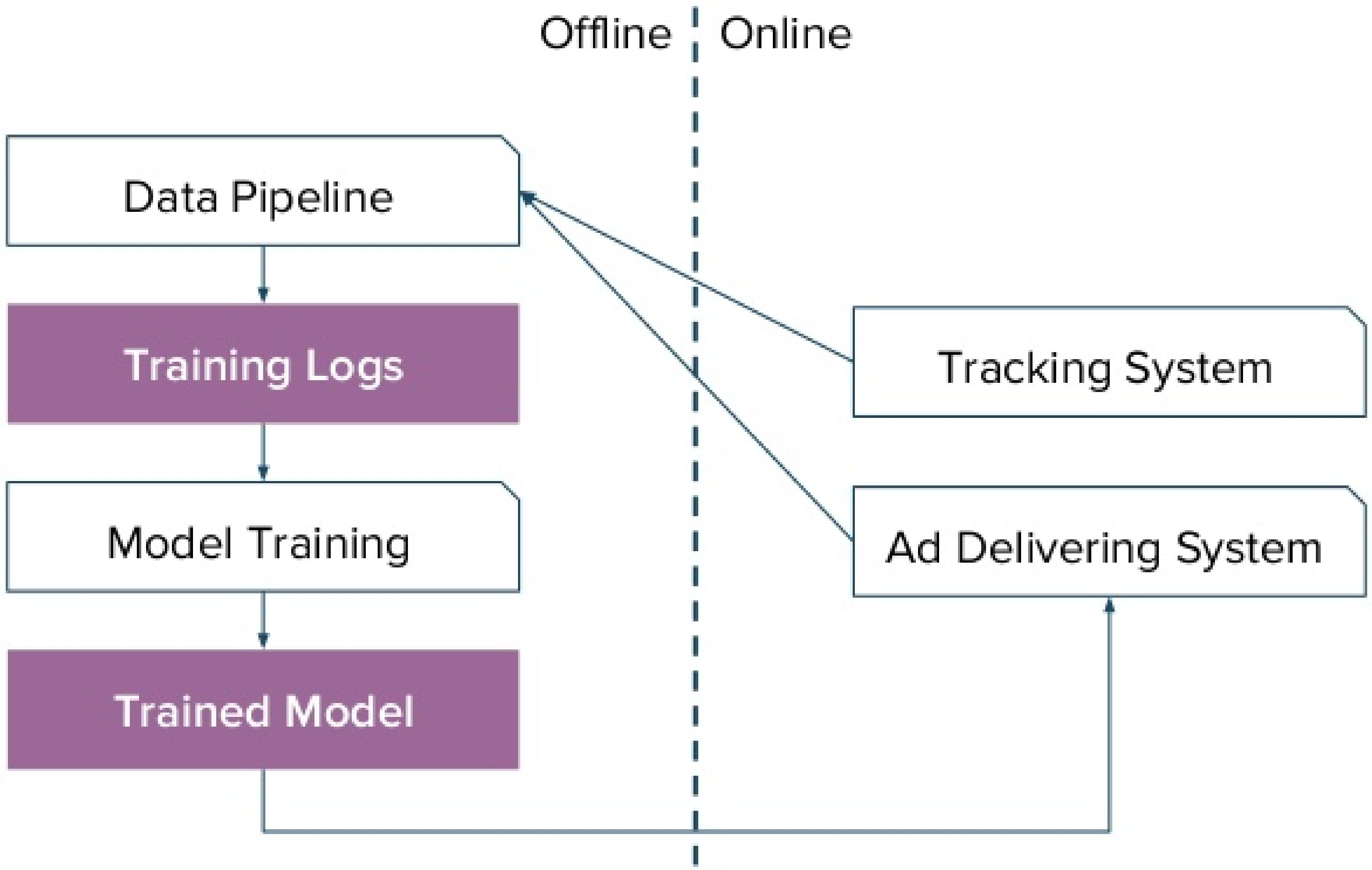


Machine Learning Stack at Teads

ML stack at Teads

- Motivation
- Model training
- Model serving
- Offline experiments

Classic setup



Challenges

Real-time prediction

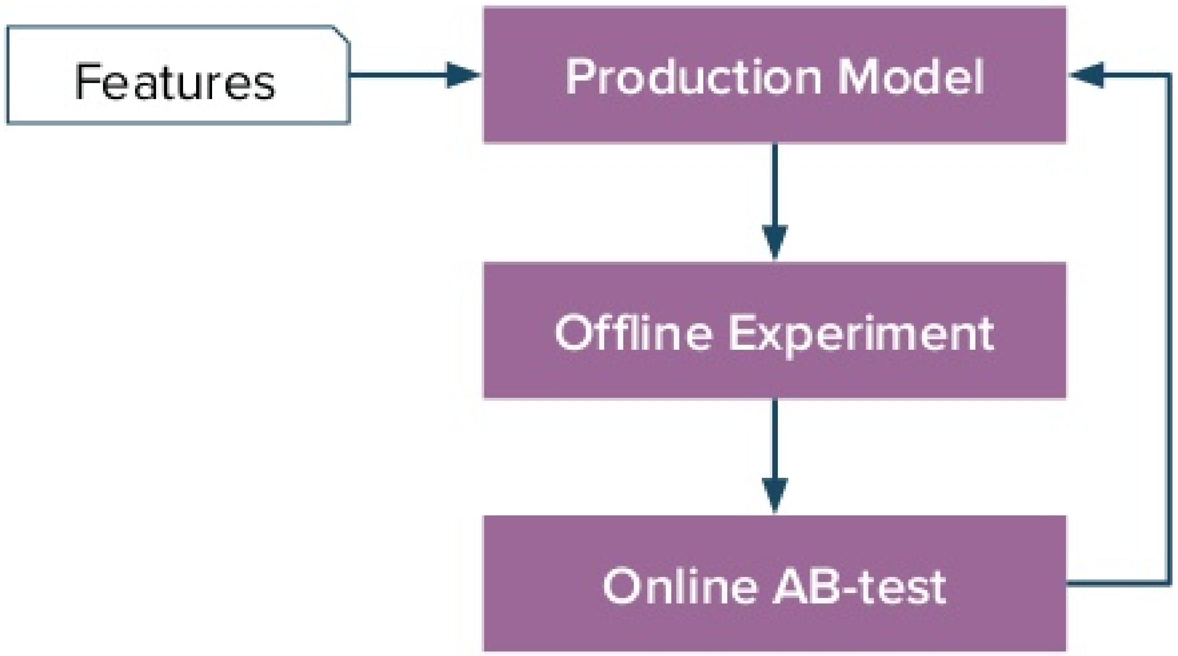
150 ms response time in RTB business

40,000 requests / second
1,000,000 model predictions / second

Model serving outside of Spark

Challenges

Flexible and reliable model improvement cycle



Why build our own stack?

No available framework/library back in 2015 that satisfied all our needs

MLlib limitations

- GLM vector size limitations
- Model serving with spark dependency

Decided to do it ourselves

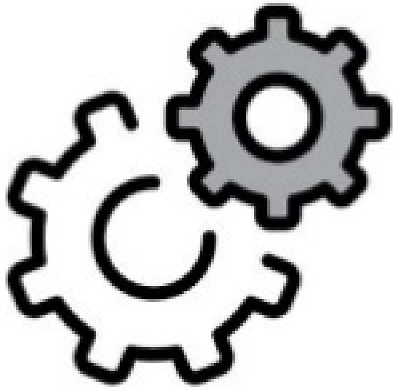
- Scala and Spark

Model Training

Model training

Different use cases in different teams

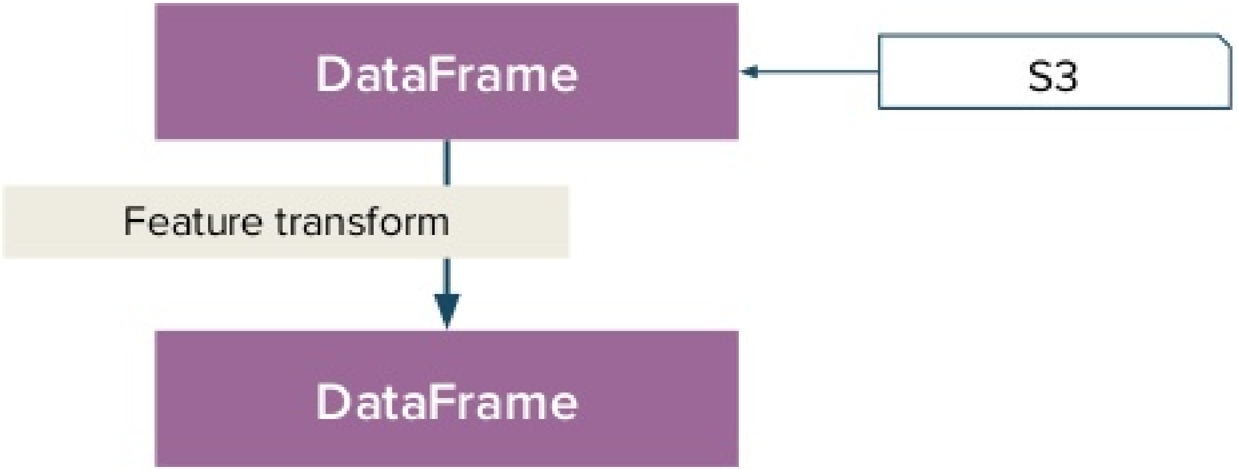
- Not the same features
- Not the same model settings



Model training steps should be configurable by using a **configuration object**

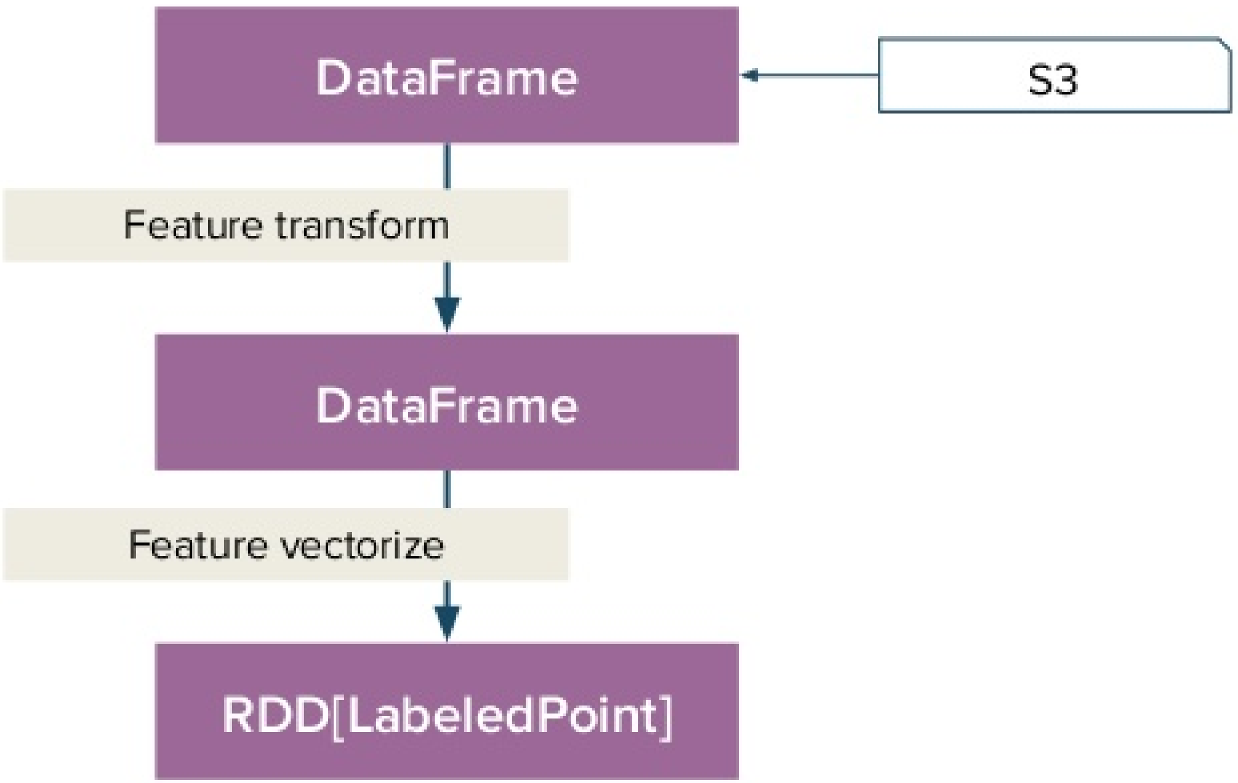
Feature transformation

```
"transform" : [
  {
    "type" : "Bucketize",
    "input" : "ad_duration",
    "output" : "ad_duration_bucket",
    "thresholds" : [
      10.0,
      15.0,
      20.0,
      25.0,
      30.0,
      45.0,
      75.0,
      120.0
    ]
  },
  ...
],
...
```



Feature vectorization

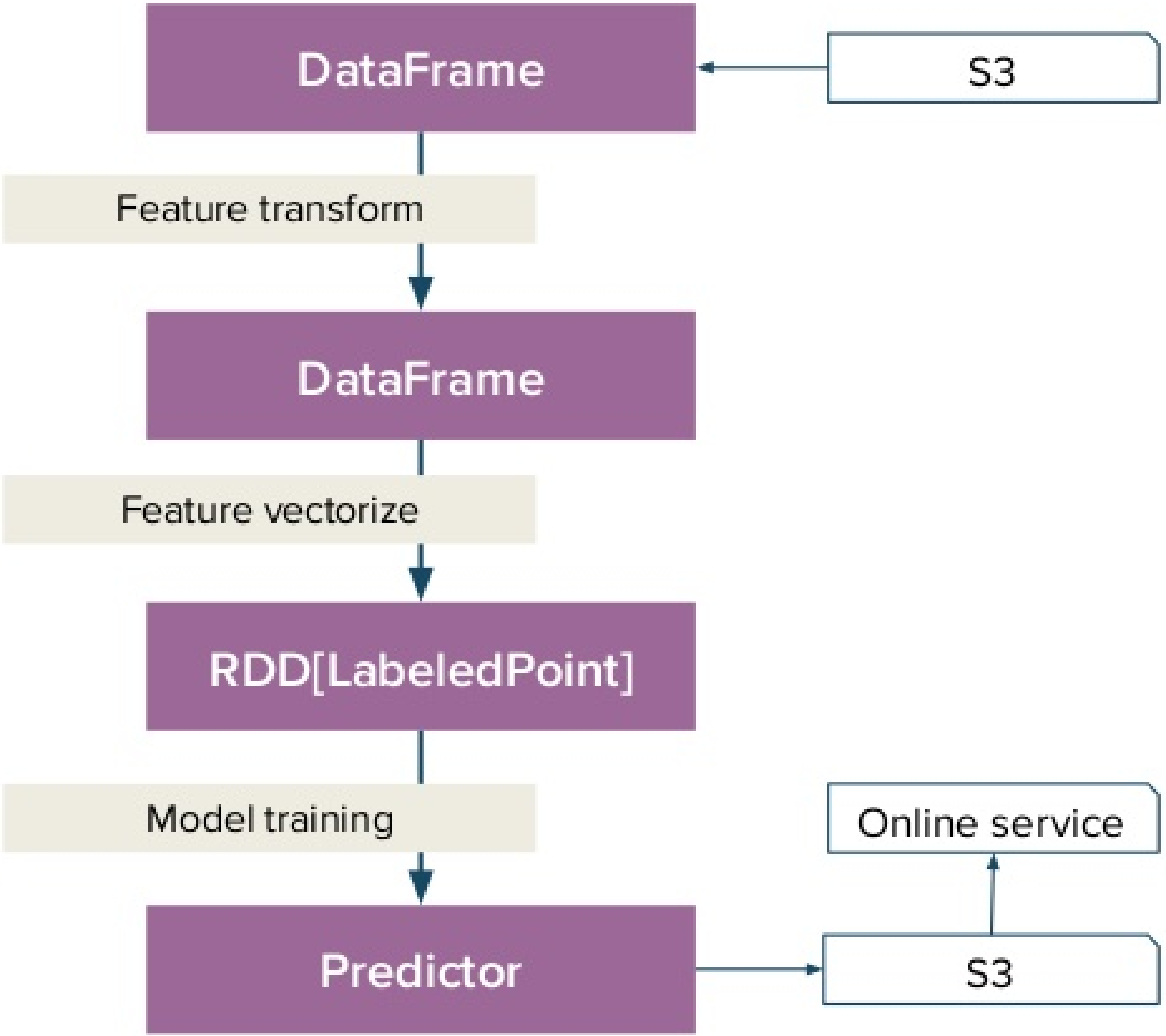
```
...,
"vectorize" : [
  {
    "type" : "ConstantVectorize",
    "value" : 1.0
  },
  {
    "type" : "HashVectorize",
    "inputs" : [
      "ad_duration_bucket",
      ...
    ],
    "inputTypes" : ...,
    ],
    "size" : 2145483000
  }
],
"label" : {
  "input" : "event_billable_dbl"
},
...
```



Model training

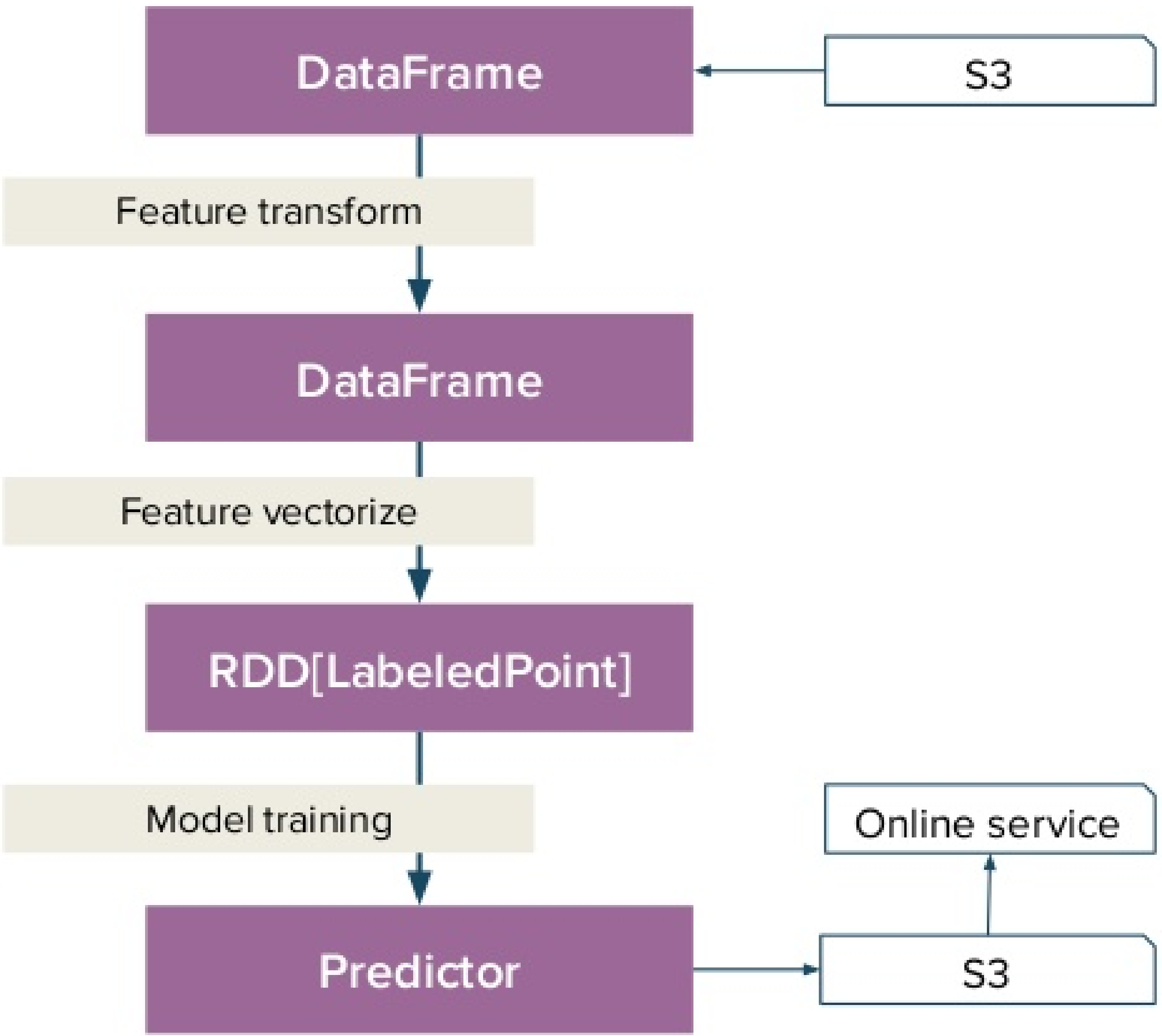
- Initial model
- Model seeding

```
...,
"initialModel" : {
  "type" : "Logistic",
  "intercept" : 0.0,
  "slopes" : {
    "type" : "Sparse",
    "indices" : [
    ],
    "values" : [
    ],
    "length" : 2145483000
  }
},
...
```



Model training
- Other settings

```
...,
"loss" : {
  "type" : "LogisticLoss"
},
"regularization" : {
  "type" : "L2",
  "alpha" : 29.0,
  ...
},
"optimization" : {
  "type" : "LBFGS",
  "maxIterations" : 150,
  "tolerance" : 1.0E-4
},
...
```



Model training

(DataFrame, TrainingConfig) => Predictor

Model training deployment

Model **name tag**

- Unique identifier of a production model

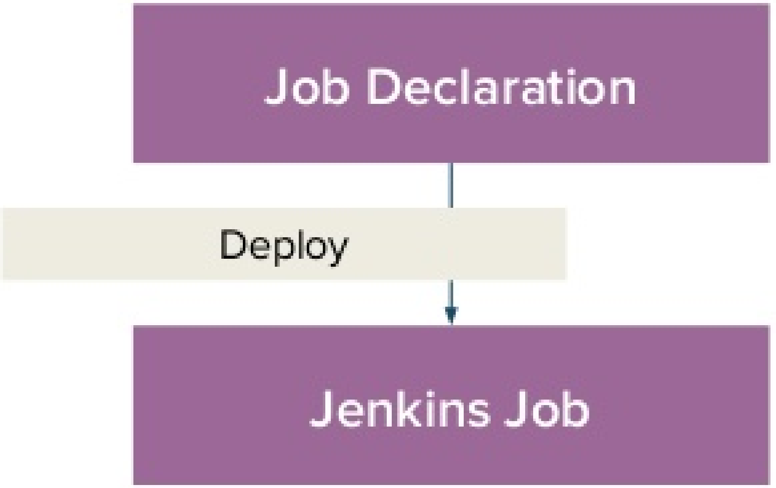
Example:

prod-team1-vtr-prediction-2018-07-15

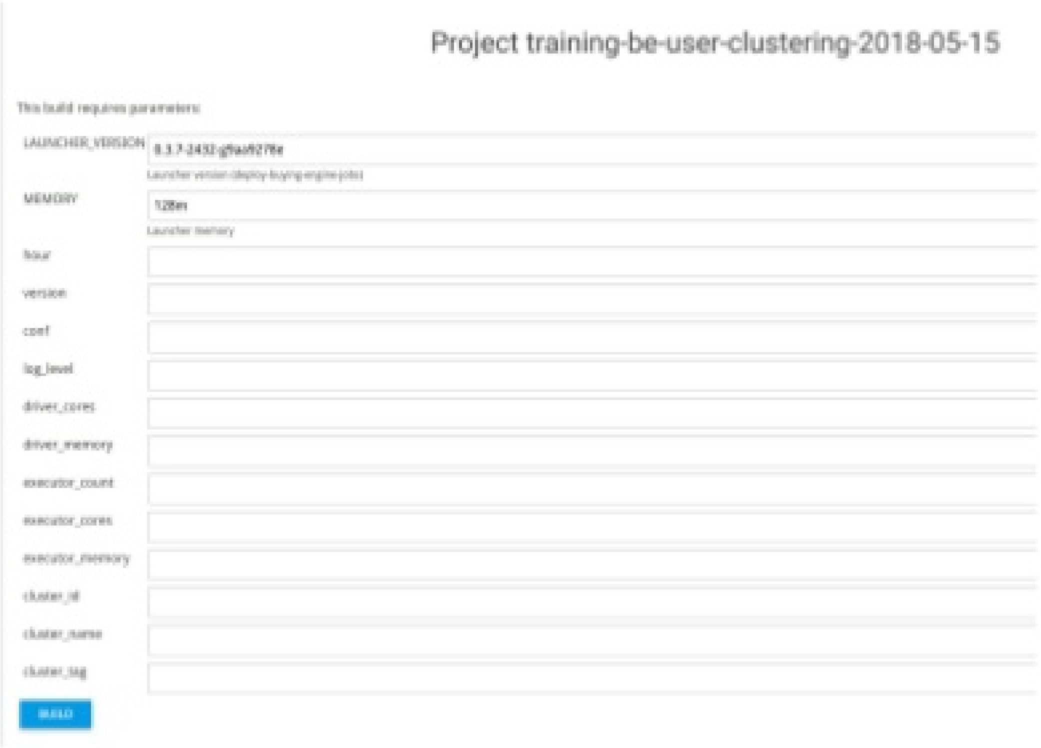
```
def getTrainingConfig(tag: String): TrainingConfig
def getLatest(tag: String): Predictor
```

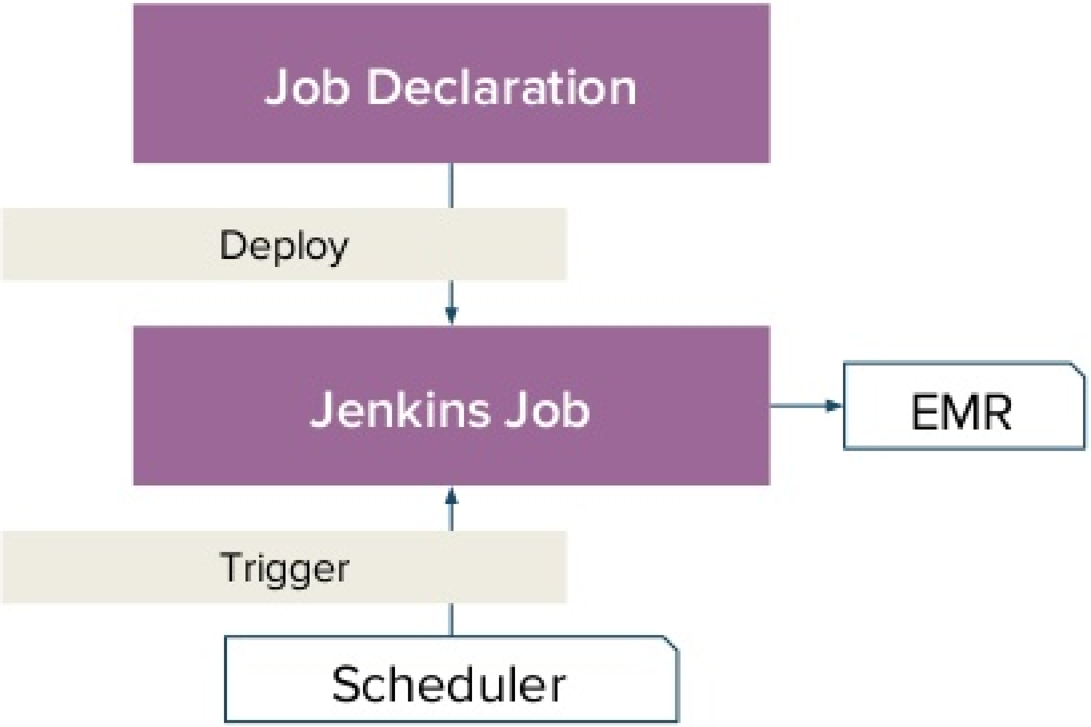
Model training deployment

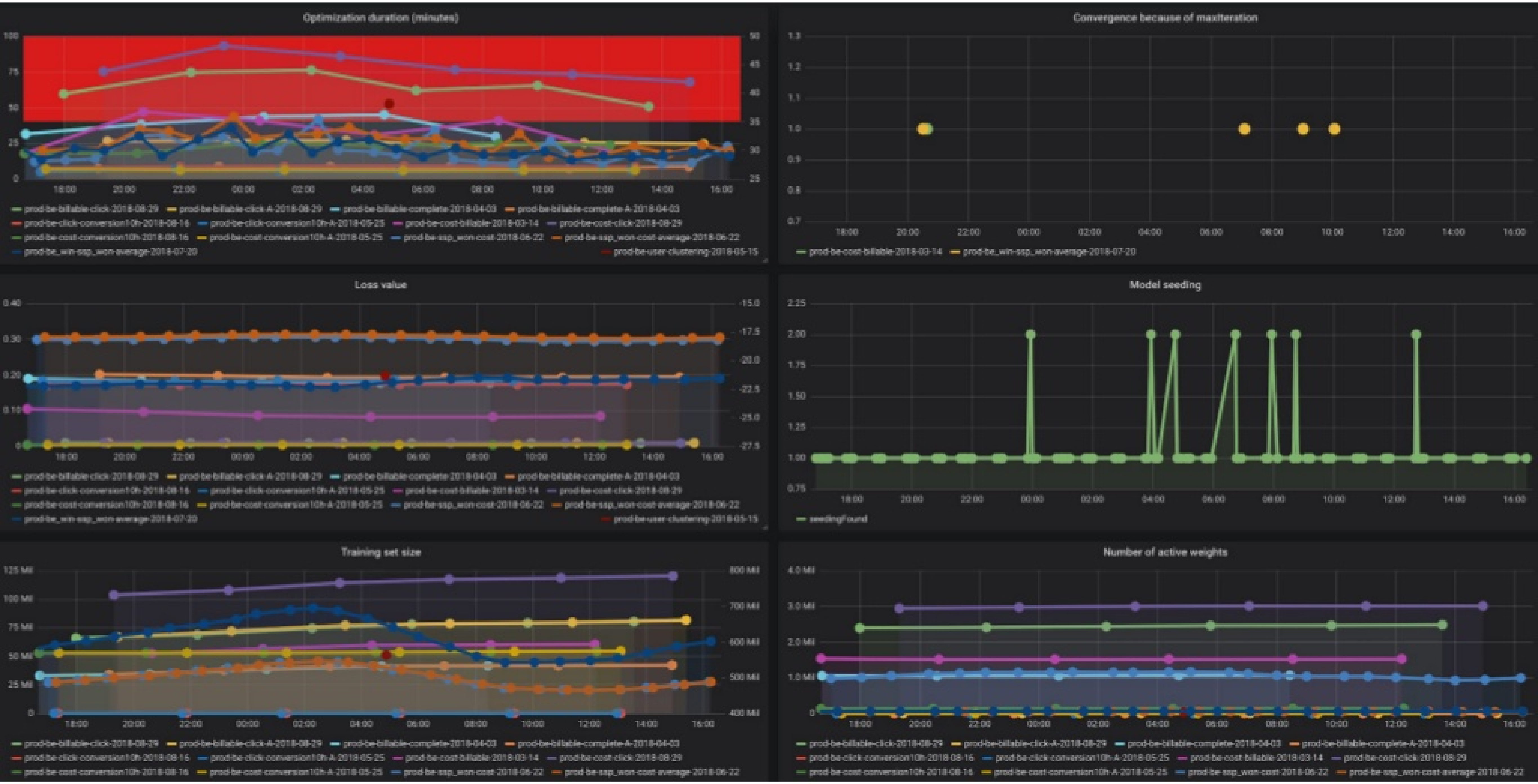
```
TrainingJob(
  modelTag = "prod-be-vtr-prediction-2018-07-15",
  jobLevel = Level.Day,
  defaultClusterOptions = EmrOptions(coreInstanceCount
= 200),
  timeout = Some(24.hours),
  group = Group.beProd,
  module = Module.libPrediction("clustering"),
  logPath = "s3a://...",
  logSize = 1,
  defaultConf = Seq(
   "spark.sql.shuffle.partitions" -> "20000"
  ),
  mainClass = Some("...")
)
```



Model training deployment







Optimization duration (minutes)
Convergence because of maxIteration
Loss value
Model seeding
Training set size
Number of active weights
prod-be-billable-click-2018-08-29
prod-be-billable-click-A-2018-08-29
prod-be-billable-complete-2018-04-03
prod-be-billable-complete-A-2018-04-03
prod-be-click-conversion10h-2018-08-16
prod-be-click-conversion10h-A-2018-05-25
prod-be-cost-billable-2018-03-14
prod-be-cost-click-2018-08-29
prod-be-cost-conversion10h-2018-08-16
prod-be-cost-conversion10h-A-2018-05-25
prod-be-ssp_won-cost-2018-06-22
prod-be-ssp_won-cost-average-2018-06-22
prod-be_win-ssp_won-average-2018-07-20
prod-be-user-clustering-2018-05-15
seedingFound

Model Serving

Model serving

Impl as a scala library

- Dependency of online system codebase

Fast and lightweight

- Prefer native code than a prediction service
- Avoid http calls
- No Spark dependency

Predictor

Configuration of model scoring

Components

- Feature transform
- Feature vectorization
- Trained model

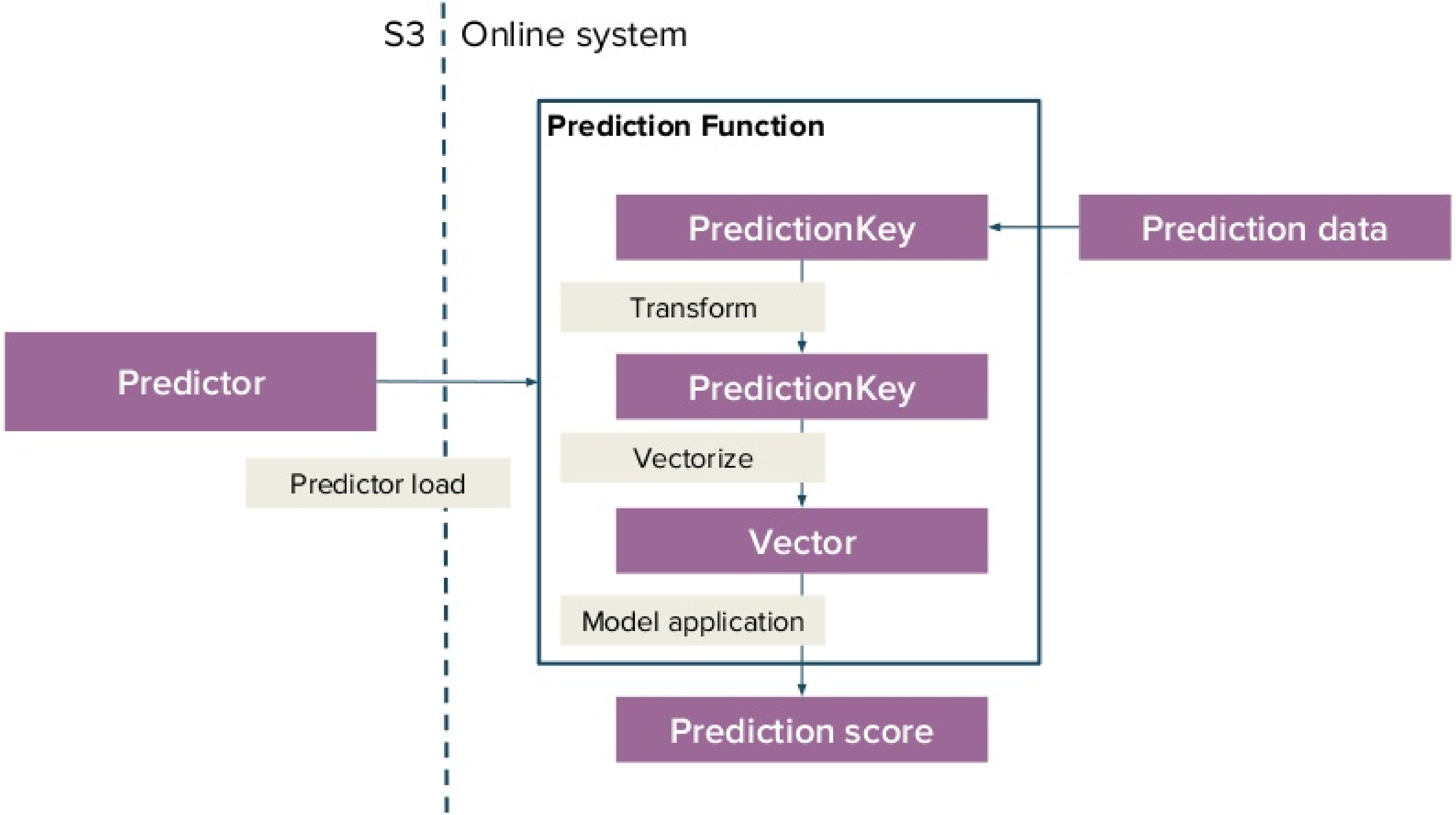
Same feature generation as in training

- Vectors are the same online and offline

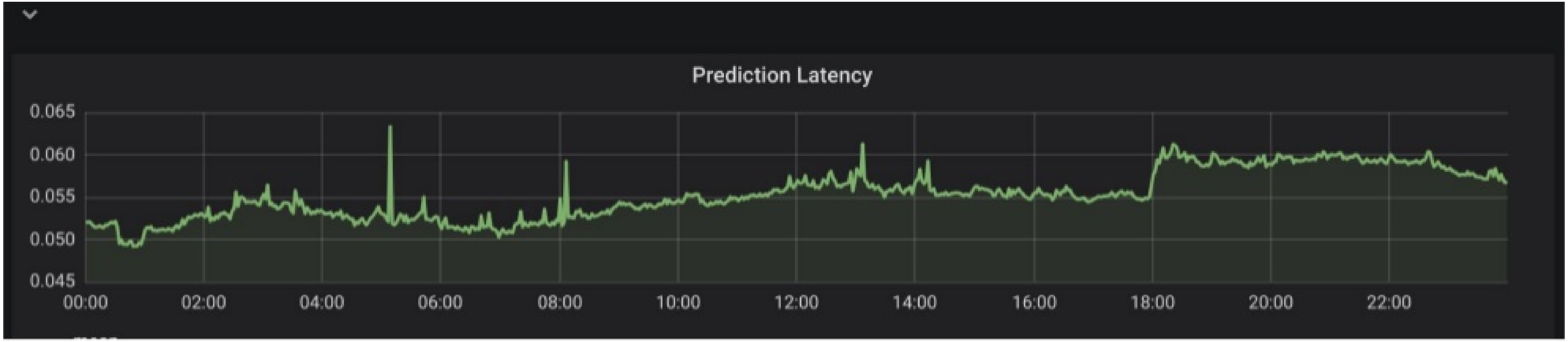
```
{
  "transform" : [...],
  "vectorize" : [...],
  "model" : {
    "type" : "Logistic",
    "intercept" : -0.59580942,
    "slopes" : {
      "type" : "Sparse",
      "indices" : [
        5385, 6271, ...
      ],
      "values" : [
        -0.002536684,
        0.026973965,
        ...
      ],
      "length" : 2145483000
    }
  }
}
```

Online feature transform

```
case class PredictionKey private (values: Vector[Any], schema: Type) {
  def add[T: Convert](name: String, t: T): PredictionKey = {...}

  def value(idx: Int): Any = {...}

  def get[T: Convert](name: String): T = {...}

  def get[T: Convert](idx: Int): T = {...}
}
```


S3
Online system
Prediction Function
PredictionKey
Prediction data
Transform
Predictor
PredictionKey
Vectorize
Predictor load
Vector
Model application
Prediction score

Average model prediction scoring takes **55 microseconds** in our production system (prediction function application time)



Offline Experiments

Offline experiment

Internal tool to evaluate ML models on historical data

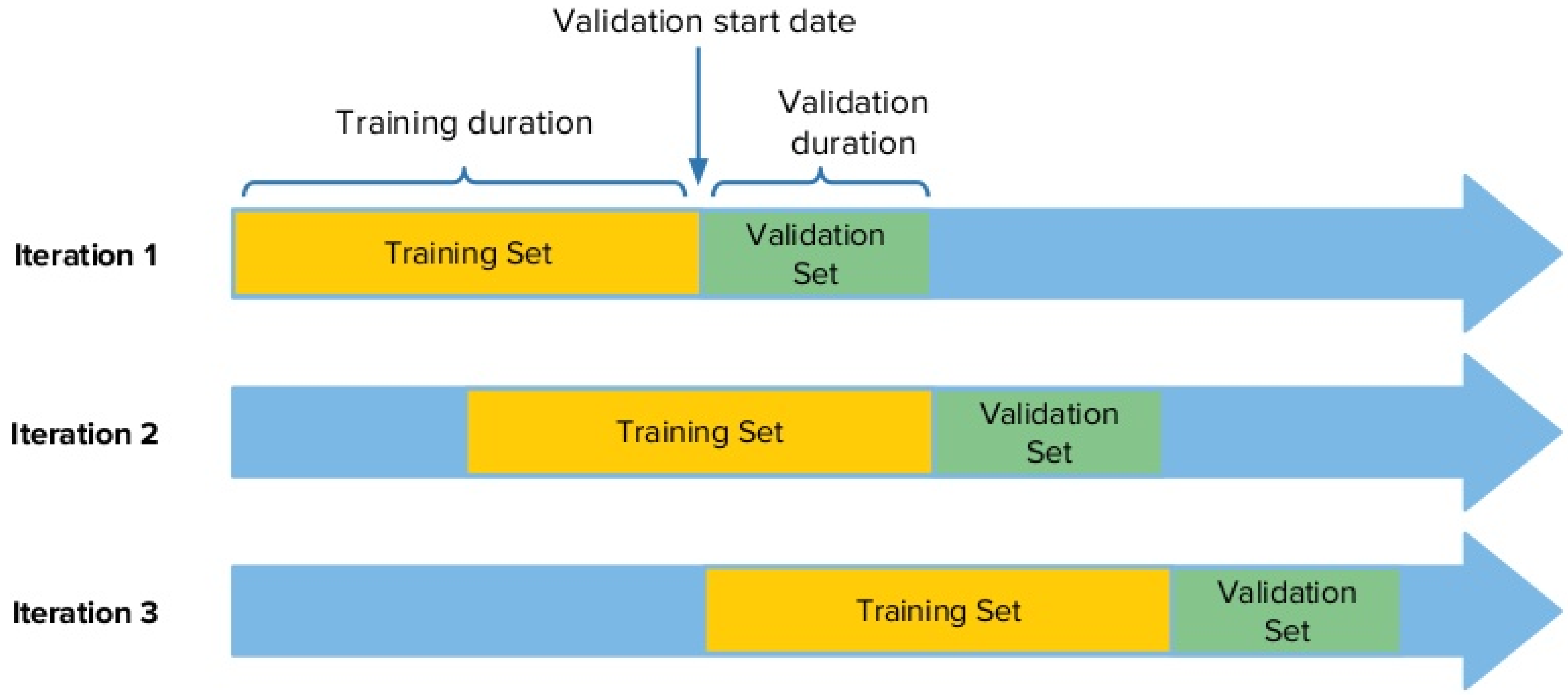
Simplify experiment process

Formalize the testing protocol

Keep track of the experiment results

Wide range of metrics with confidence interval

Offline experiment



Experiment creation

```
In [8]: datak.experiments.create(
          estimatorId = Id("b9c1cdbdb5fa2a680ba3e85aa8359a12c236689a"),
          name = "offline experiment test example",
          owner = "datakinator-team",
          trainingDuration = 4,
          trainingLogId = 43,
          validationDuration = 4,
          validationLogId = 43,
          nbIter = 1,
          validationStart = "2018-05-01 00:00:00",
          offset= 0,
          validationOnlyFeatures = "constantOne",
          metric_weights = "constantOne",
          projections = "constantOne,business_model"
        )
Out[8]: res7: Long = 3316L
```

Notebook as main interface of experiments

- **jupyter** notebook
- **jupyter-scala** kernel

Weighted metrics computation

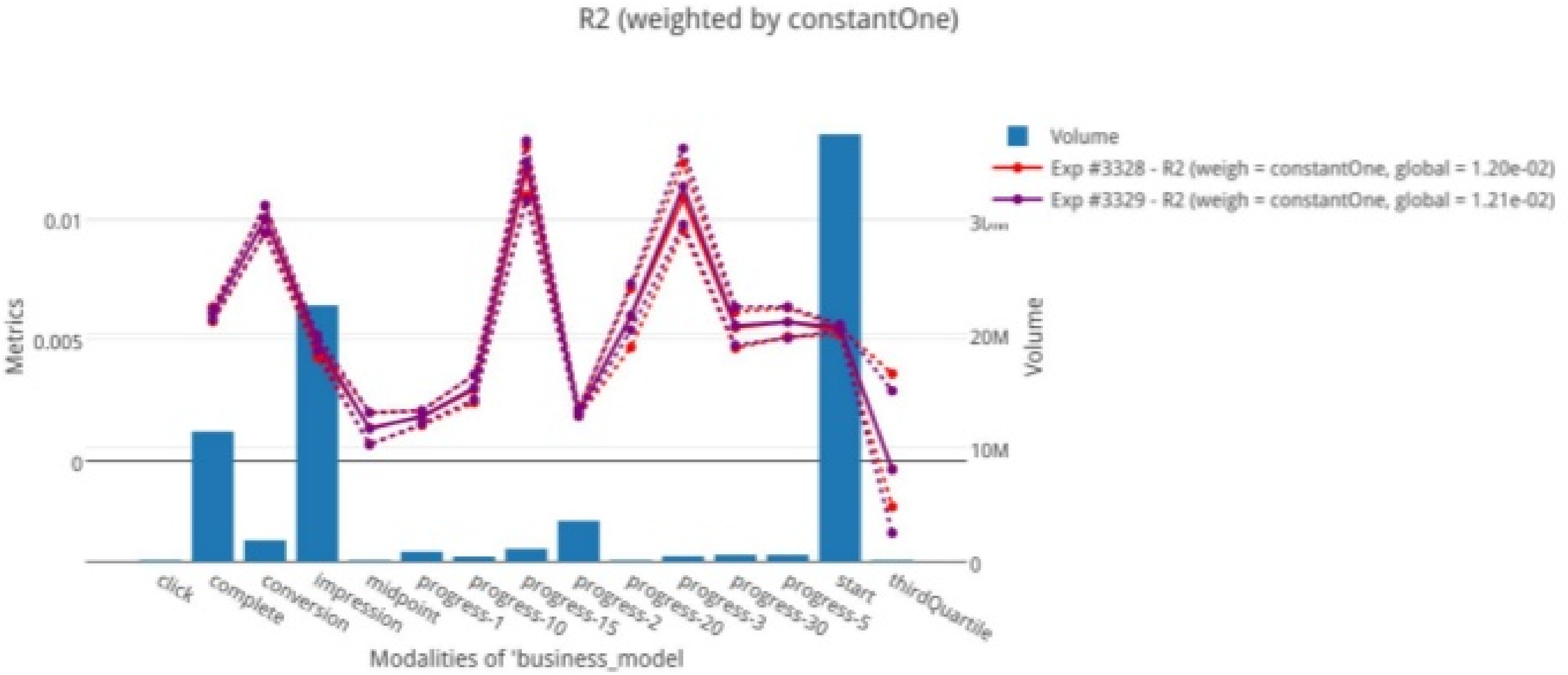
Result projections

- Aggregate the result metrics by categories (day, business model, country, etc.)

Jupyter-scala kernel: https://github.com/jupyter-scala/jupyter-scala/

Result visualization

```
viz.plot(
  subSetIds = allIds,
  metrics = Seq(RegressionMetric.R2),
  weights = Seq('constantOne),
  projection = 'business_model
)
```



Offline experiment

3000

experiments launched by different teams during last 18 months

480000

hours of logs analysed, all experiments combined

Thanks!

Follow us!

Our engineering blog on medium:

https://medium.com/teads-engineering