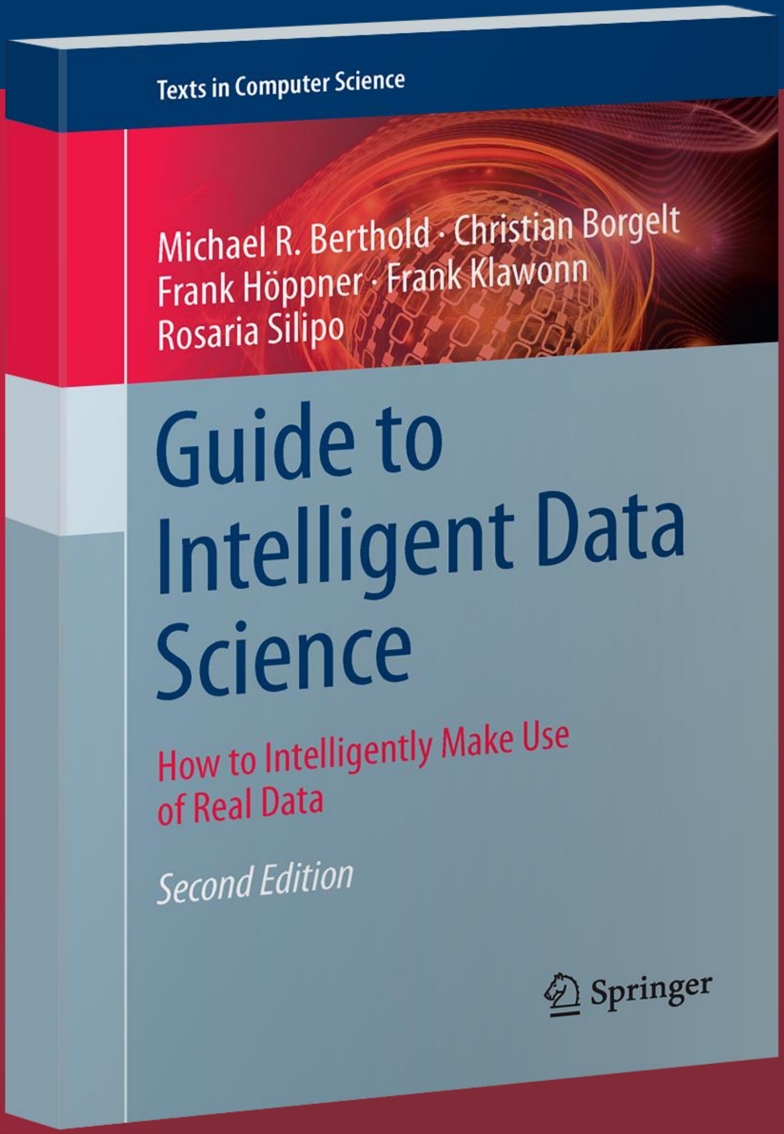
Deployment


Texts in Computer Science
Michael R. Berthold · Christian Borgelt
Frank Höppner · Frank Klawonn
Rosaria Silipo
Guide to Intelligent Data Science
How to Intelligently Make Use of Real Data
Second Edition
Springer

Summary of this lesson

„Data Scientist is just a sexed up word for Statistician“

-Nate Silver

How do we move the models to production?

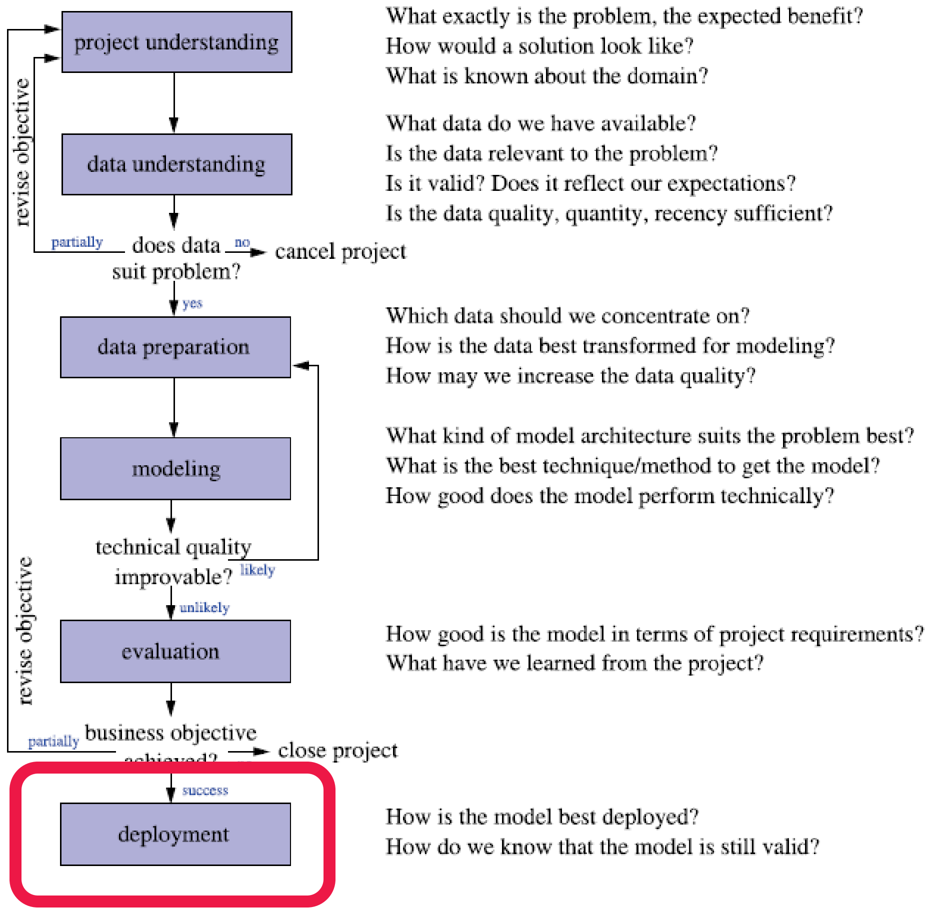
**This lesson refers to chapter 10 of the GIDS book*

Content of this Lesson

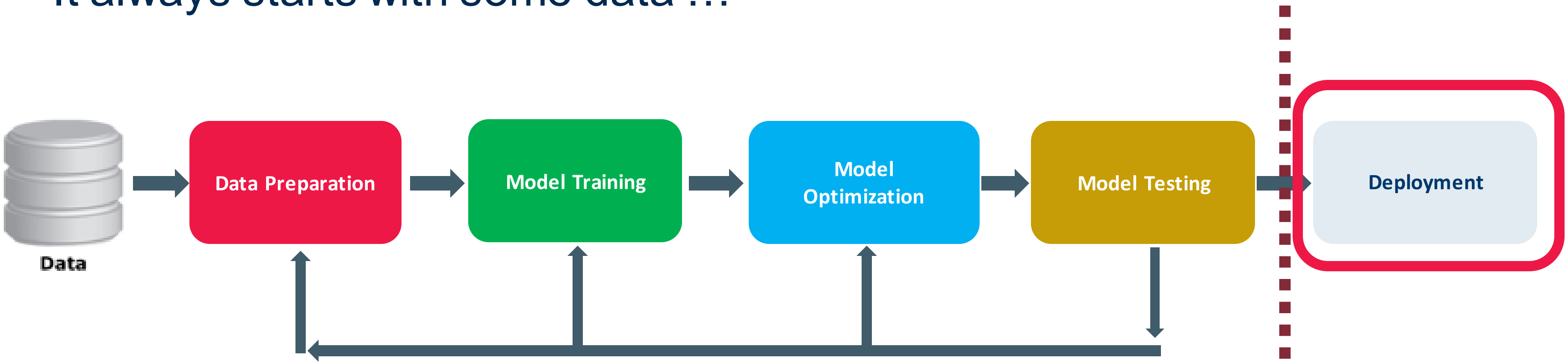
- Deployment
- Model Deployment
- Model Management
- Practical Example

Deployment

- SEMMA
  - Sample, Explore, Modify, Model, Assess
- CRISP-DM
  - Cross Industry Standard Process for Data Mining
- KDD
  - Knowledge Discovery in Databases



– It always starts with some data …



Data Manipulation
Data Blending
Missing Values Handling
Feature Generation
Dimensionality Reduction
Feature Selection
Outlier Removal
Normalization
Partitioning
…

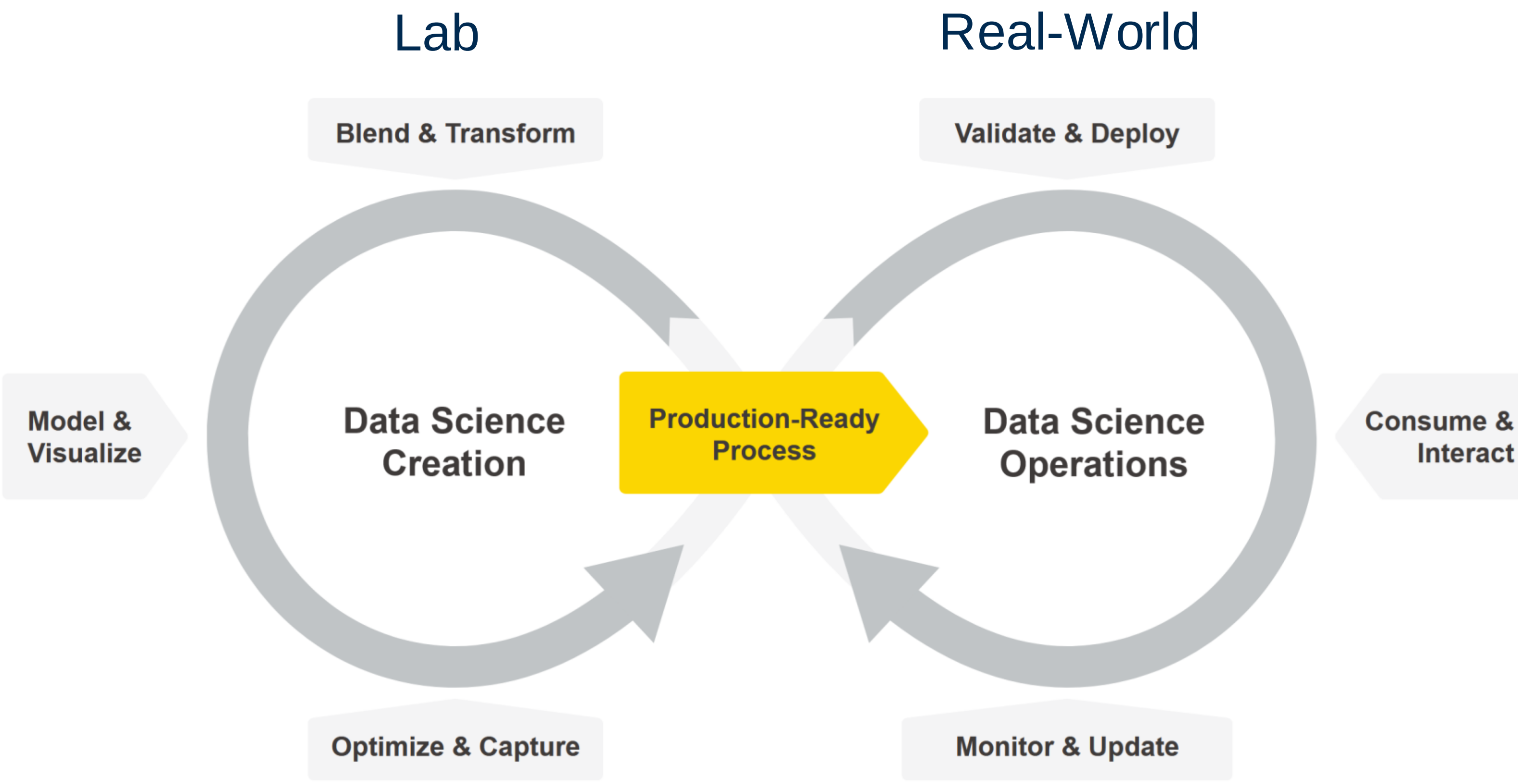
Model Training
Bag of Models
Model Selection
Ensemble Models
Own Ensemble Model
External Models
Import Existing Models
Model Factory
…

Parameter Tuning
Parameter Optimization
Regularization
Model Size
No. Iterations
…

Performance Measures
Accuracy
ROC Curve
Cross-Validation
…

Files & DBs
Dashboards
REST API
SQL Code Export
Reporting
…

What comes after Deployment?



What is model deployment?

- Notice the dashed line between model testing and model deployment?
- This is where the jump **from the lab to the real world** happens
- Eventually a trained model must be included in a final application to be used **by external applications and/or end users**
- The final application is the deployment application
- The step of building the application around the trained model is called **deployment**
- Notice that the deployment application must be developed and finally put into production like all pieces of software
- When the deployment application is moved into production, so is the trained model

Deployment requirements

- Easy
  - It must be easy for the application developer to include the trained model into the deployment application
  - Easy to use for end users
  - Easy to integrate in a Service Oriented Architecture

- Safe
  - At the same time it must be correct. For example, it must include the whole data preparation part.
  - Most reasons of deployment failures are in the not faithful export of the pre-processing and post-processing steps from the training application into the deployment application.
  - Think of a model trained on normalized data and of a deployment application where normalization has been forgotten.

Once in the real world, the deployment application and the trained model must oblige to the laws of IT

- **Automation**
  - On demand & scheduled execution
  - Monitoring and Updating
- **Auditing**
  - Justify decisions
  - Store previous executions
  - Reproducibility
- **Security**
  - Protection of sensitive data
  - Protection of sensitive applications
  - Versioning & Disaster Recovery

Deployment Options

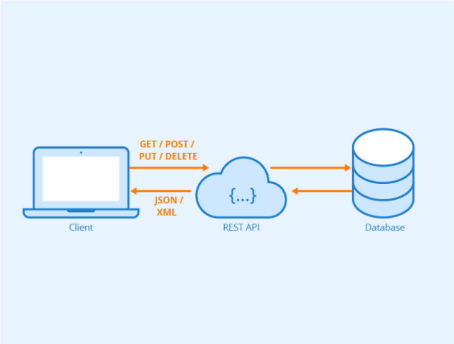
Deployment

Usage of a trained model in an application to provide answers for a real-world use case

- In its own application
  - Easy to use for end users (as a web application)
  - Easy to integrate in a Service Oriented Architecture (as a web service)

- Consumed by external Applications
  - As a file in standard format
  - As a software library

Deploying the ML Model



In a web service

Inside its own application

in a web application

ML model

as a file in a standard format

as a software library

```
<?xml version="1.0"
      encoding="iso-8859-1" ?>
<languages>
  <language id="fr">
    <name lang="fr">Français</name>
    <name lang="en">French</name>
    <name lang="es">Frances</name>
    <name lang="de">Französisch</name>
    <name lang="eo">Franca</name>
  </language>
```

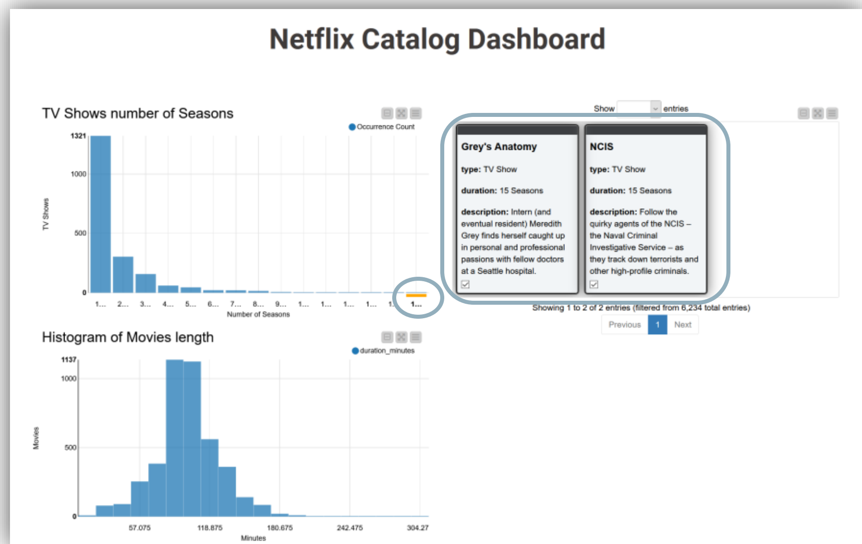
Consumed by external applications

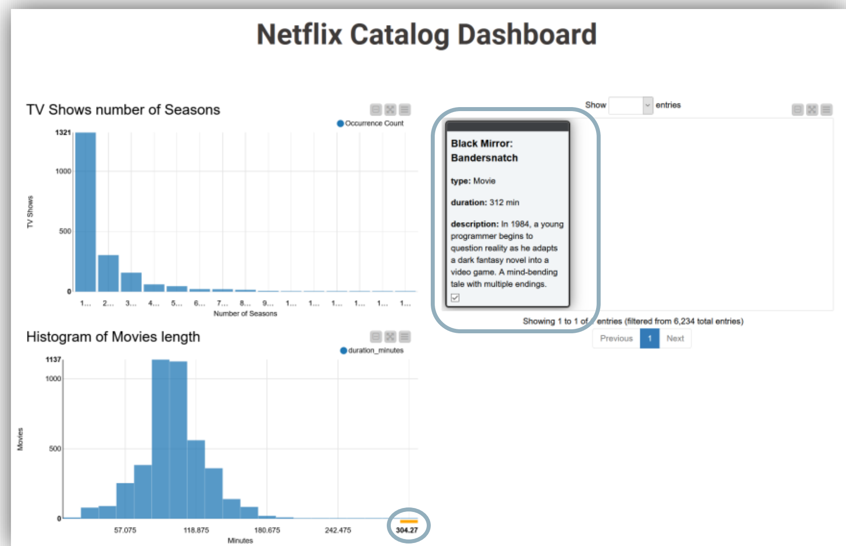
Deployment in its own application

- Easy to use for end users
  - If the model has been deployed into an application for end users, it must be easy to use also for non-experts and non-data-scientists kind of users
  - As a web application from a web browser
  - Hide model complexity
  - Offer touchpoints for exposed parameters

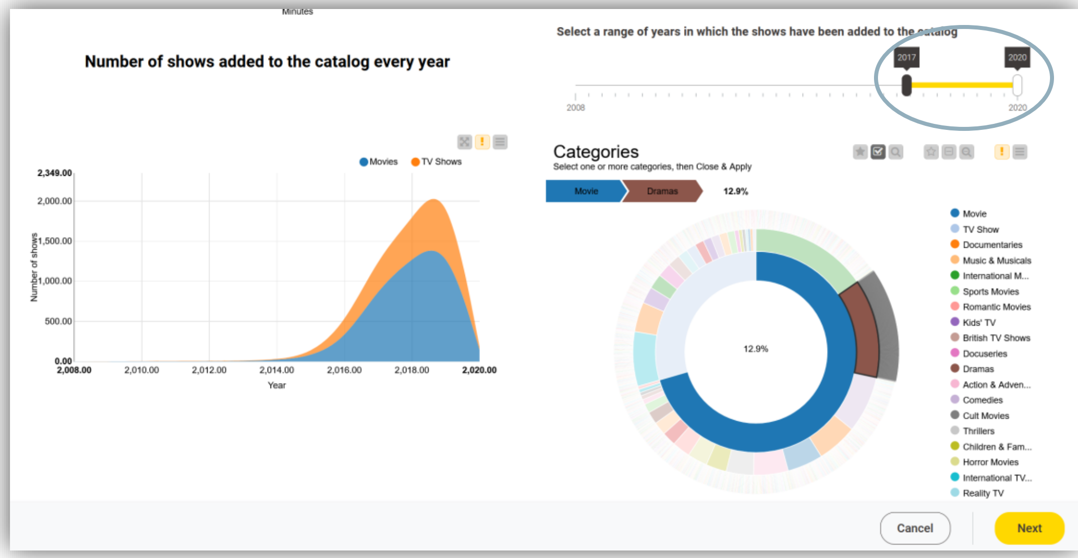
- Easy to integrate in a Service Oriented Architecture
  - As a web service
  - Via standard interfaces for web services

- One page usually includes an Interactive Dashboard to show the results
- Fast and intuitive decision support even for non expert users
- Can show model prediction and more complex interactive data visualization





- Interactive plots and charts
- Data selection across plots, charts, and tables
- Items such as: range slider, selection bullets, menus, …



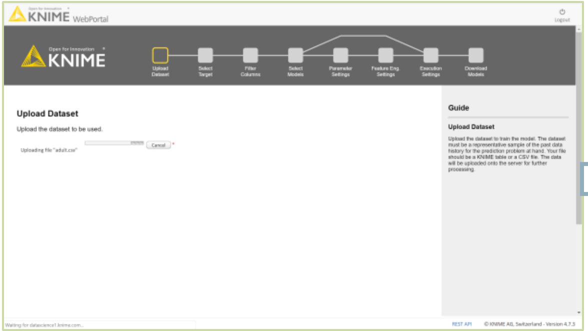
- One final dashboard page ➔ to show results
- What about having touchpoints that require end user interaction?
- Hide complexity in automated snippets
- Expose parameters interesting to the end users via touchpoints

- Example: Guided Automation.
  - Train a number of models on the selected training set
  - Sequence of Touchpoints could be:

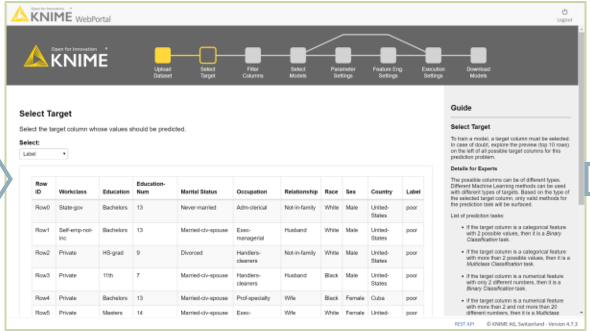
1. Load Data ⇨ 2. Select Target ⇨ 3. Filter Columns ⇨ 4. Select Models ⇨ 5. Select Execution Engine ⇨ 6. Show Results

Guided Automation: An example

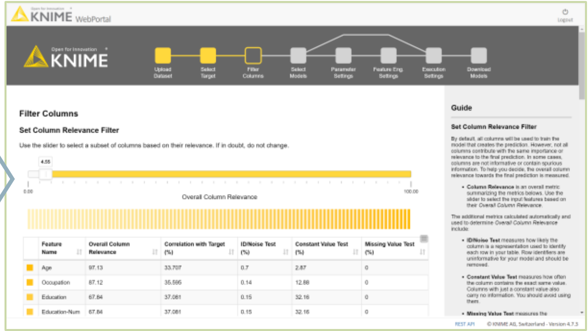
1. Load Data



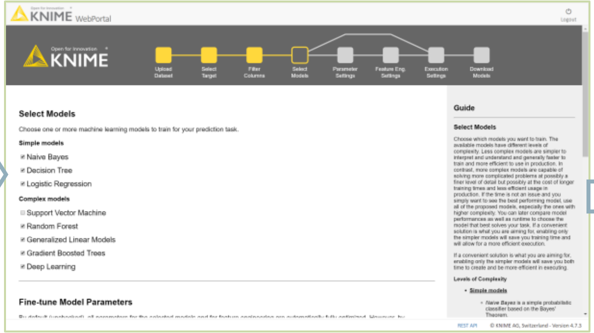
2. Select Target



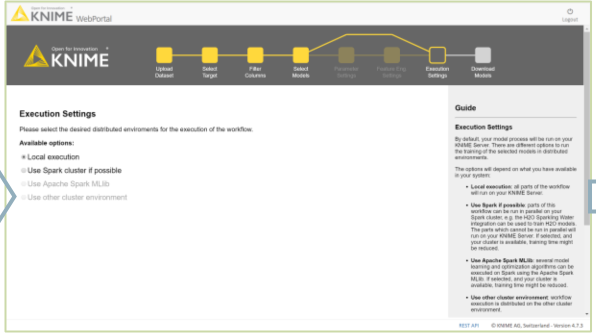
3. Filter Columns



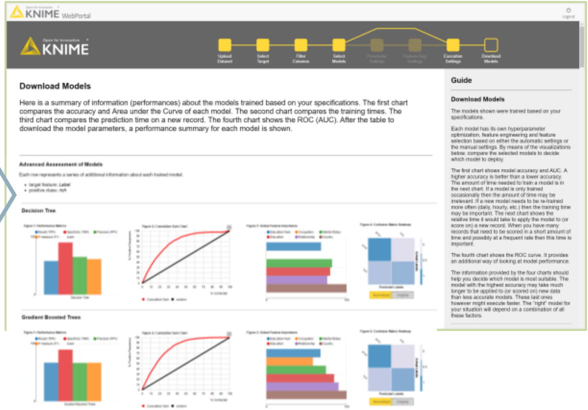
4. Select Models



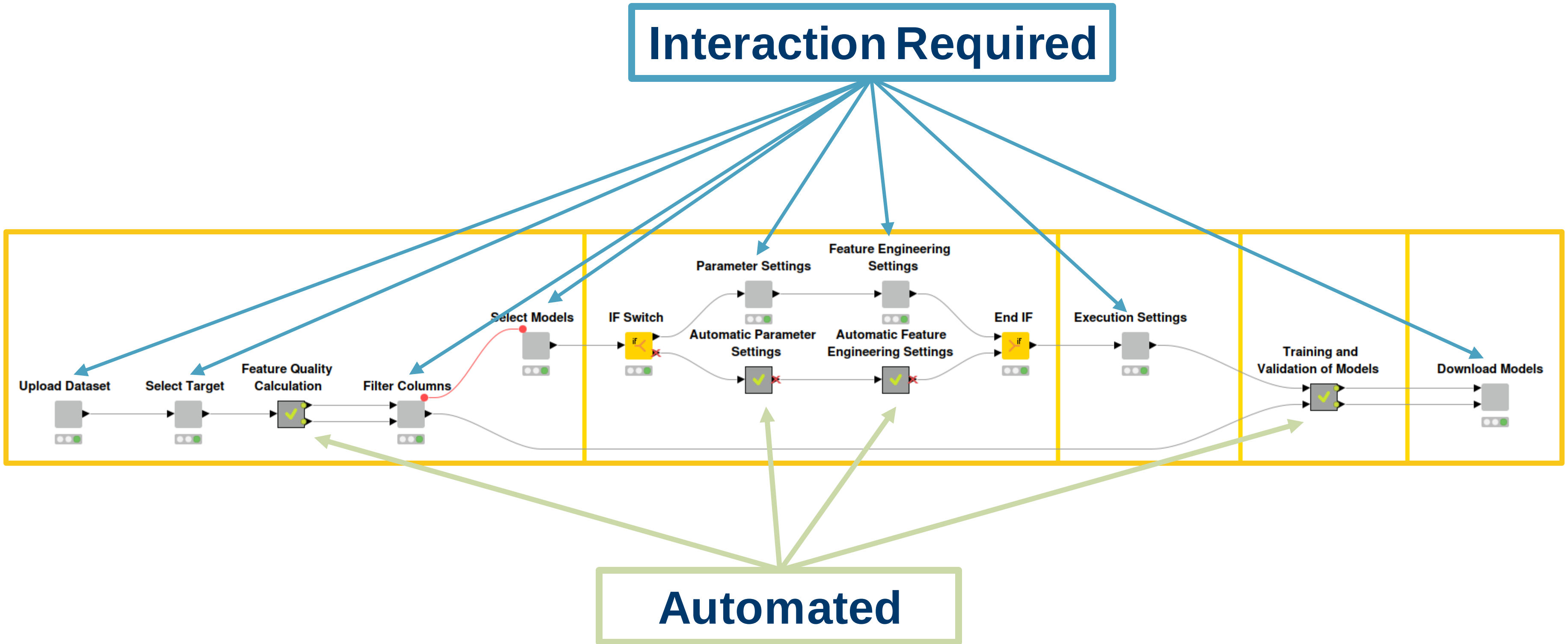
5. Select Execution Engine



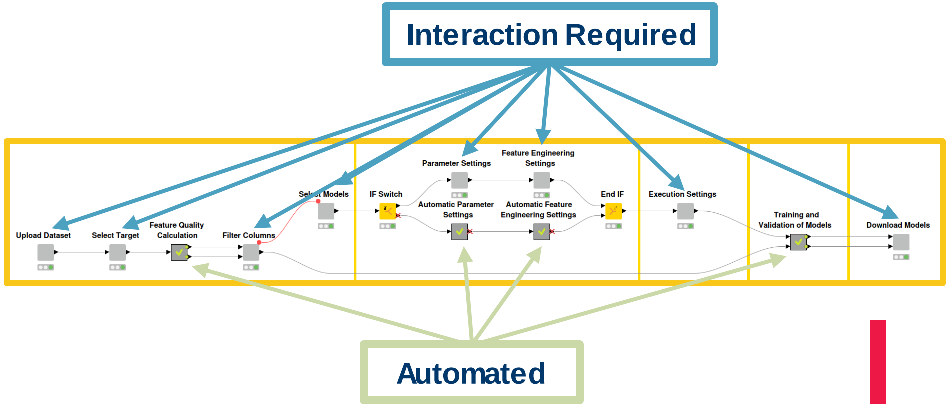
6. Show Results



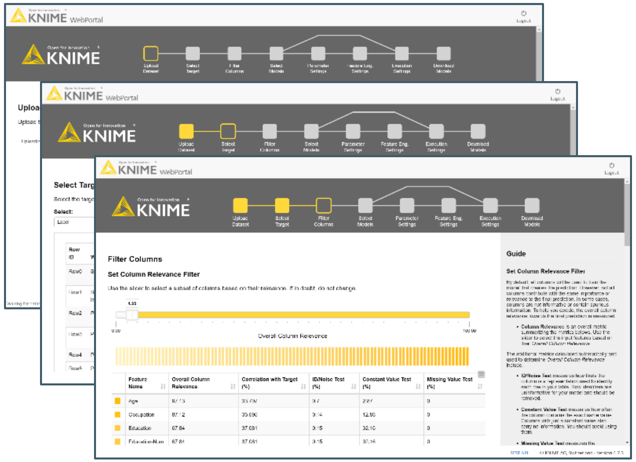
Building a Guided Automation Workflow



KNIME’s Guided Automation: Automation + Interaction



http://myserver.com



KNIME Server

analytical application for business users from web browser

Deployment in a web service

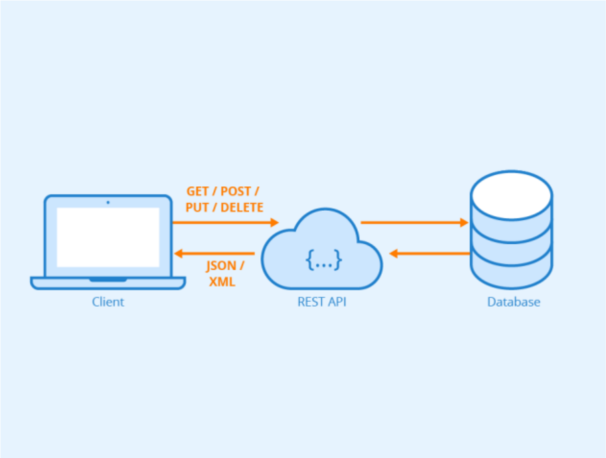
- A web service provides interoperability between computer systems
  - over the internet
  - through a web technology, such as HTTP
  - to transfer machine-readable file formats such as XML and JSON.

- Web Services with REST architecture are the current state of the art

- What is a REST architecture
  - **Representational State Transfer** (**REST**) is a software architectural style introducing a set of constraints for web services.
  - Web services that conform to the REST architectural style, are called *RESTful* (REST) web services.
  - **REST services** allow the requesting systems to access and manipulate representations of web resources by using a **uniform** and **predefined** set of stateless operations. You cannot make up your own arbitrary set of operations, as in SOAP web services.
  - Stateless protocol and standard operations => fast execution, easy to manage

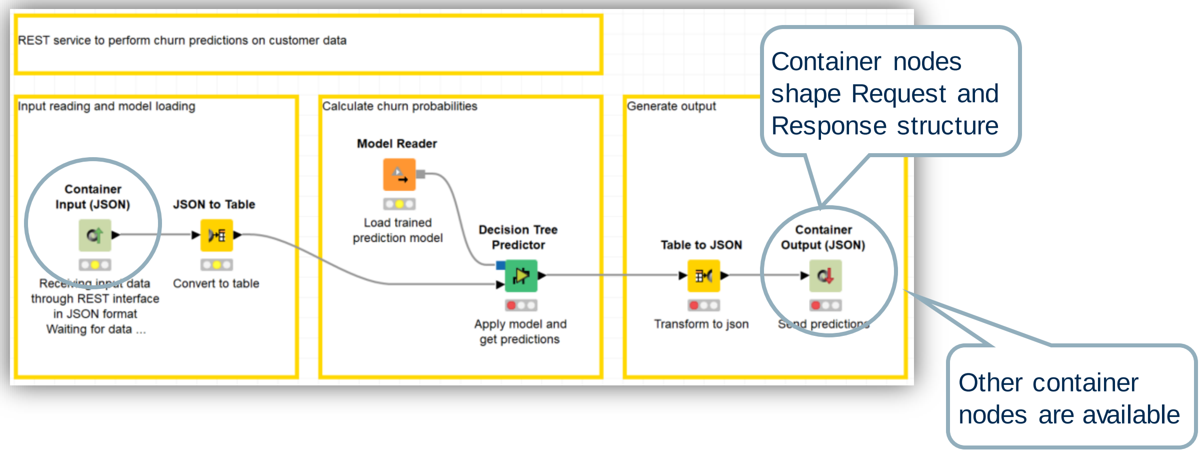
Deployment in a web service

- Operations in a REST web service (over HTTP)
  - GET
  - HEAD
  - POST
  - PUT
  - PATCH
  - DELETE
  - CONNECT
  - OPTIONS
  - TRACE

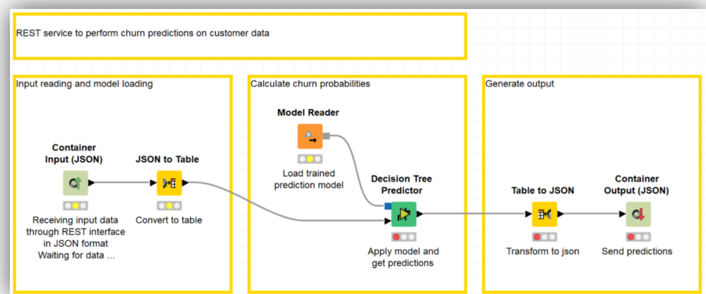


- The Request and Response objects
  - Data is exchanged via a Request object and a Response object
  - The **Request object** sends data to the REST service, together with the required operation
  - The **Response object** passes the result back to the calling system

- Building a REST service requires:
  - To shape the structures of the Request and Response objects
  - To enable the REST API
- Solutions:
  - Container nodes shape the Request and Response objects
  - All workflows uploaded on the KNIME Server are available as REST services



1



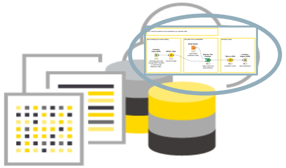
Upload

KNIME Server

REST Service

2

REST Service



KNIME Server

External Application

Deployment as a file in standard format

- Standard formats allow for external applications to consume the network/model
- **PMML**
  - **Predictive Model Markup Language (PMML)** is based on XML
  - Embeds a wide range of predictive models along with aspects of the required pre-processing
  - Can be directly loaded into database systems and applied to data tables
  - PMML works well with standard ML models (decision tree, logistic regression)
  - Representation of new complex models (ensamble, deep learning…) is problematic, either because a standard representation has not been defined or because the size of the resulting file is too large
  - Less and less used
- **ONNX**
  - ONNX = Open Neural Network for eXchange
  - Open **standard** dedicated to represent **neural networks** and **deep learning networks**
  - ONNX represented networks can then be stored into files
  - Standard ensures the portability of the represented network across systems

Note: Data processing (transformation/integration) must be part of the deployed model in production

Frequent Causes for Deployment Failures

- Data Science projects often fail in deployment. Why?
- Common reasons:
  - **Bad project design**: consequences can appear only in deployment phase. For example, a feature, transformation, or a data source that is not available in production.
  - **Data leakage**: data in the test set mixes up with data in the training set. Model scores do not reflect the performances in the real-world.
  - **Dynamic domains**: Features and target variable end up having different domains in the training data vs. the real-world. New values are not handled properly.
  - **Change in Business Objectives**: During or after deployment the business objectives of the project have changed for some reason. For example, the business strategy of the company has changed.
  - **Invalidated assumptions.** What we thought it was true about the data, it is not. Maybe we did not extract a representative sample from the world data.
  - **Shift from inter- to extrapolation**: atypical data (i.e. data not used during training). What to do? Shall we stop everything?
  - **The world changes:** e.g. if new products offered or customers change habits, the data used to build and optimize the model are no longer representative of the reality

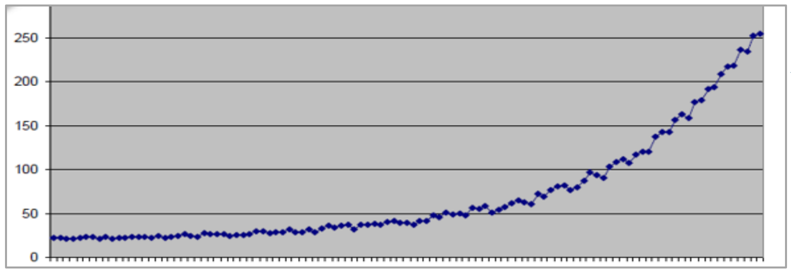
Model Management

- The world change, the business requirement change

- **Model Management** puts in place some mechanisms to ensure that the model keeps performing as expected

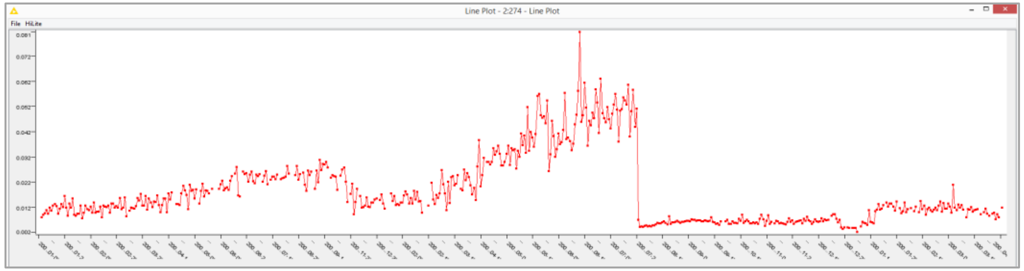
- Model Management includes:
  - Model Monitoring
  - Model Update & Retraining
  - Model Factories

- The world changes, the data change
- Data Drift (data changing slowly over time)



The state of a mechanical piece, the temperature going towards winter, the price of items due to inflation, etc ...

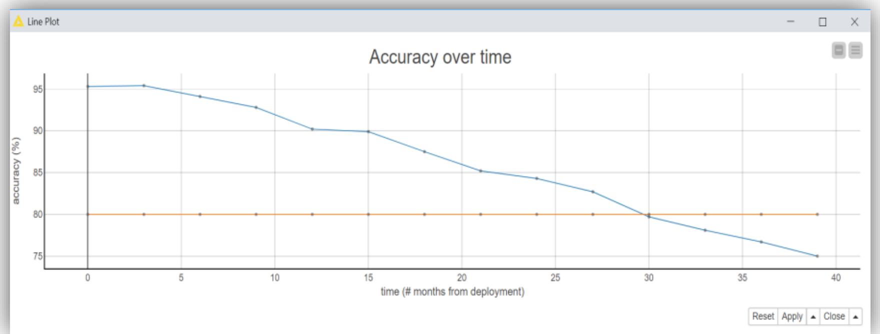
- Data Jump (Data changing suddenly at some point)



The breaking of a mechanical piece, the crash of the stock market, etc ...

Model Drift

- A model with an accuracy of 90% in the past can slowly (or suddenly) degrade to a much lower accuracy over time.
- This is called **Model Drift**



- Periodically check model performance
  - On which data?
  - How often is periodically?
- If model performance below threshold, retrain
  - What threshold value?

Model Performance Evaluation: On which Data?

- To spot the Model Drift (due to an outdated model), you should use **recent data**
- It is of course useless to test the model on data acquired at the time when the training data were collected.
- At every run, production data are stored for monitoring purposes, till a sufficiently large dataset is collected.
- Manually annotated data are also added to test **border cases**
- The model is then tested again on this newly collected dataset.
- No action is taken if performance drops within an acceptable interval. Contrarily, actions for model retraining must be taken, if performance goes below the acceptance threshold.

Model Performance Evaluation: How often?

- What does “periodically” mean?

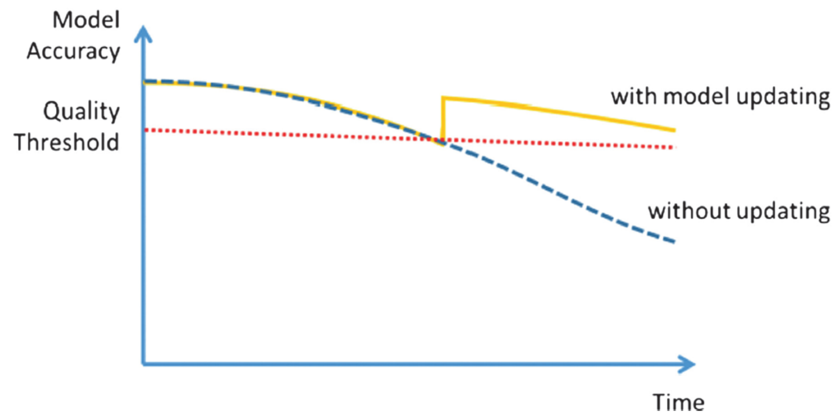
- Shall I test my model performance once a week, once a month, or once a year?

- It depends on the data and on the business case:
  - Stock prices change every minute ➔ model re-evaluation every few days
  - The taste of a customer segment will be the same for a few weeks ➔ model re-evaluation every few months

- Same for the evaluation threshold: the value depends on the data and on the business case

Model Updating and Retraining: Training Set

- **(Automatic) Model Updating**
  - Feed new data points to be incorporated into the model
  - In this way old data are less important (are forgotten)



- **Retraining**
  - Use sampling to provide the right mix of past and more recent data

- Caveats:
  - Seasonality can be a problem. Specialized models or season knowledge manually injected
  - Pre-existing knowledge (e.g. border case handling) better incorporated using a separate rule model instead of manual knowledge injection

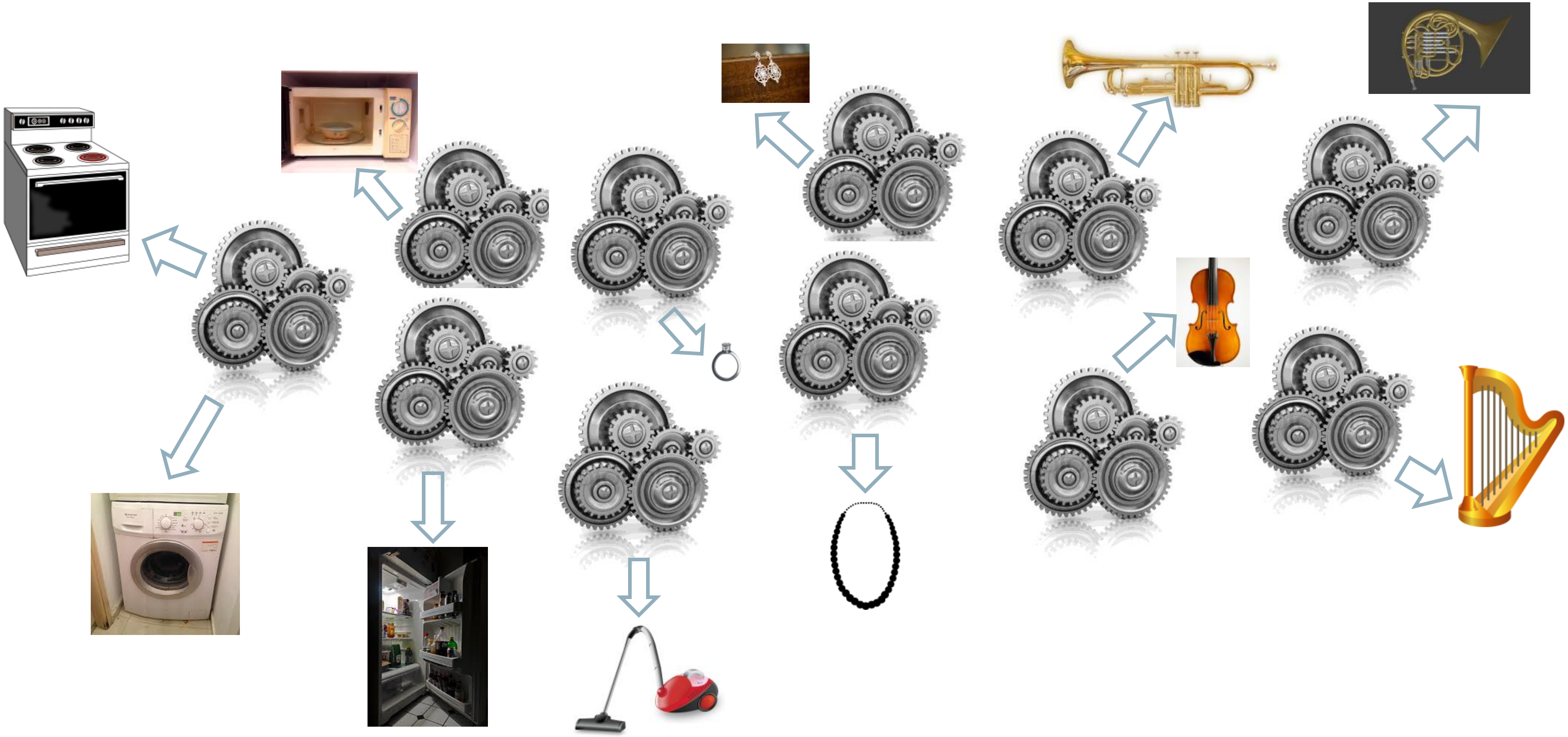
- **Model Replacement**
  - We have retrained a new model. Are we sure it is better than the previous one?
  - New model is the ***challenger***
  - Former model is the ***champion***

IF challenger's performance > champion's performance THEN replace
OTHERWISE keep champion model

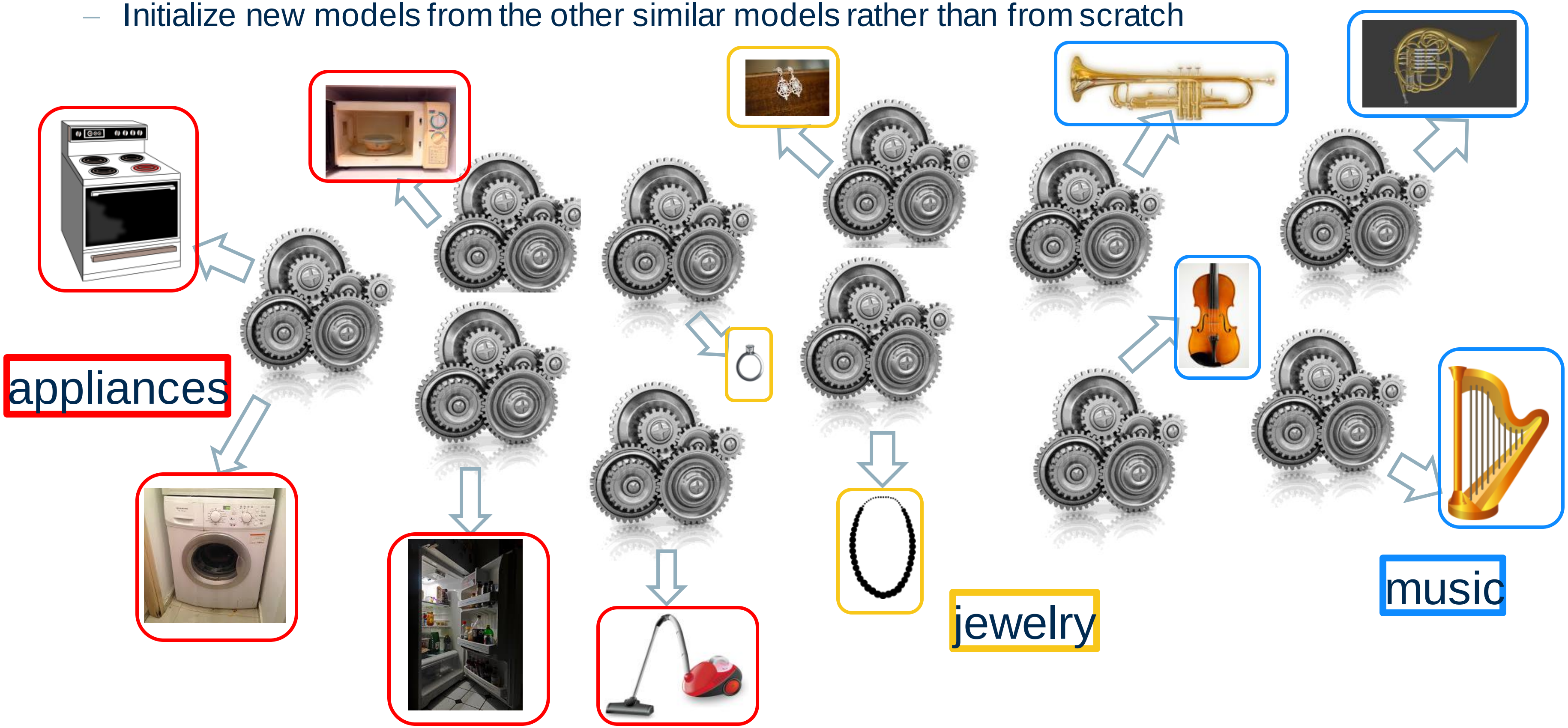
- Caveats:
  - Resources and time demanded

Model Factories

– Orchestration of a set of models – e.g. predicting prices



- How to manage a set of models?
  - Exploit grouping (families of similar models rather than single ones)
  - Initialize new models from the other similar models rather than from scratch



Model Factories

- How to communicate to the user the status of thousands of models?
  - An application for the frontend

- Who controls the process and the dependencies?
  - A separate program that handles the management process in the correct order

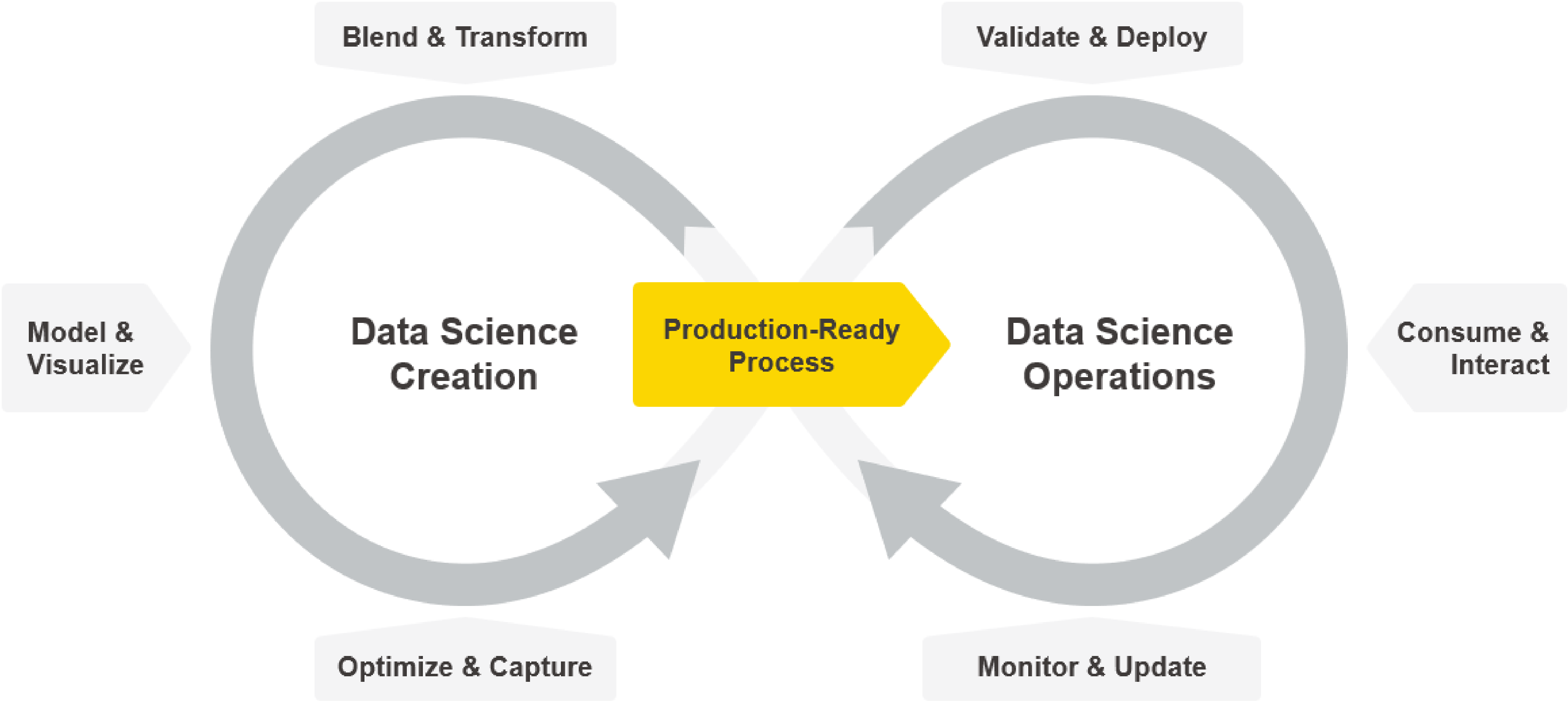
category	music				jewelry			appliances				
item	horn	trumpet	violin	harp	ring	Ear-rings	Neck-lace	fridge	wash. mach.	micro wave	stove	Vacuum cleaner
Threshold on accuracy	0.75	0.90	0.85	0.85	0.9	0.9	0.85	0.7	0.8	0.75	0.75	0.8
retrain	If 3 out of 4 perform below threshold				If all perform below threshold			If one performs below threshold				

- The term **MLOps (**or **DataOps** or **DSOps)** refers to all those operations required to deploy, monitor, update/retrain a model and comply with the general company rules for auditing and data protection.

- In a sense they are similar to DevOps for software applications in a production environment, only that they deal with Machine Learning models and data science operations in general.
- MLOps Examples
  - Deployment and moving into production
  - Monitoring of Model Performance
  - Triggering of Retraining[s]
  - Storage of Information for Auditing Purposes

Model Deployment and Management in Practice with KAP and KNIME Server

Creating and Productionizing Data Science



KNIME Software – One Ecosystem

Create

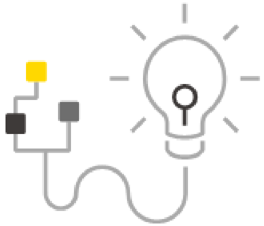
Productionize

Blend & Transform

Model & Visualize

Deploy & Manage

Consume & Interact



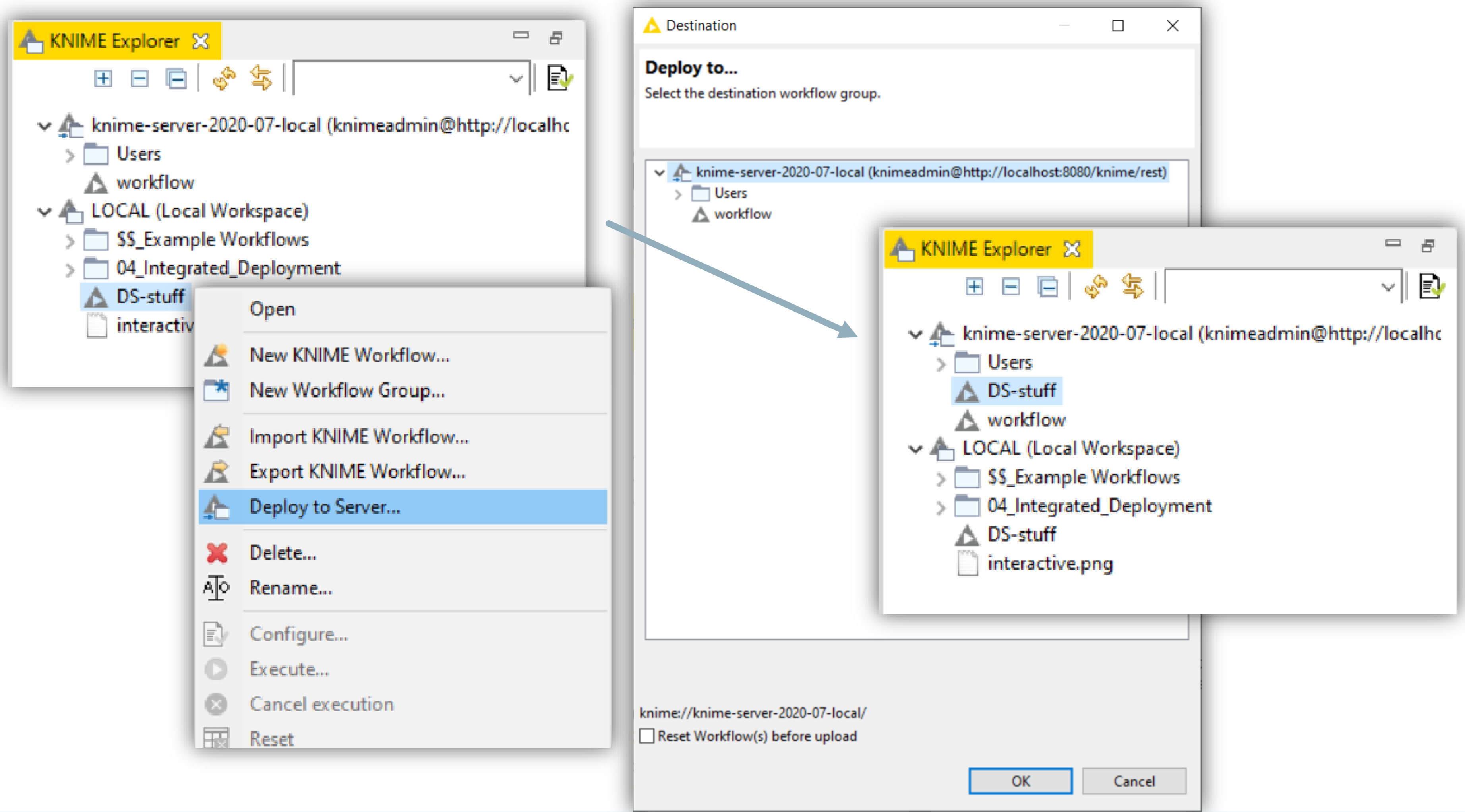
KNIME Analytics Platform

- KNIME Extensions
- KNIME Integrations
- Community Extensions
- Partner Extensions

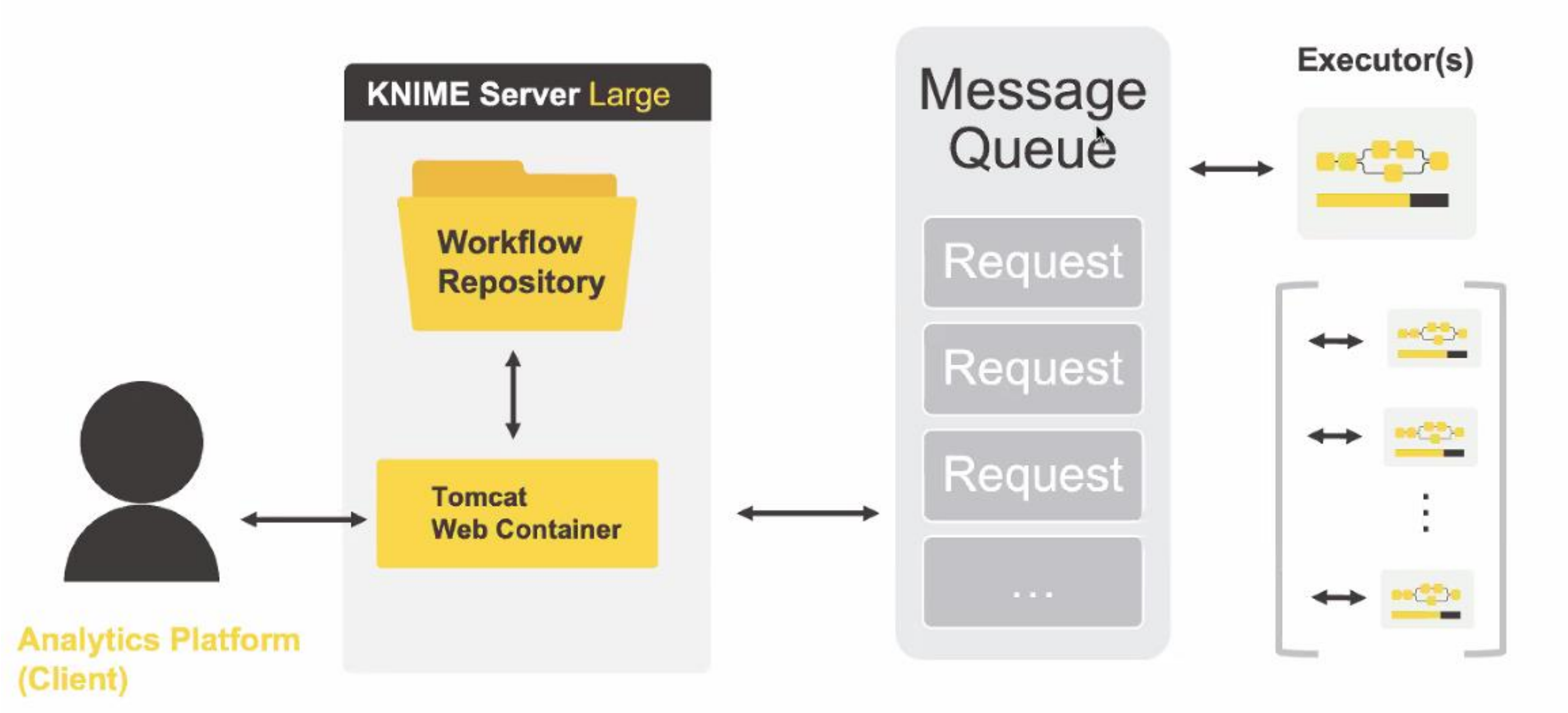
KNIME Server

- Team Collaboration
- End User Applications
- API Services
- Managed Execution

How does deployment actually look like in KNIME?



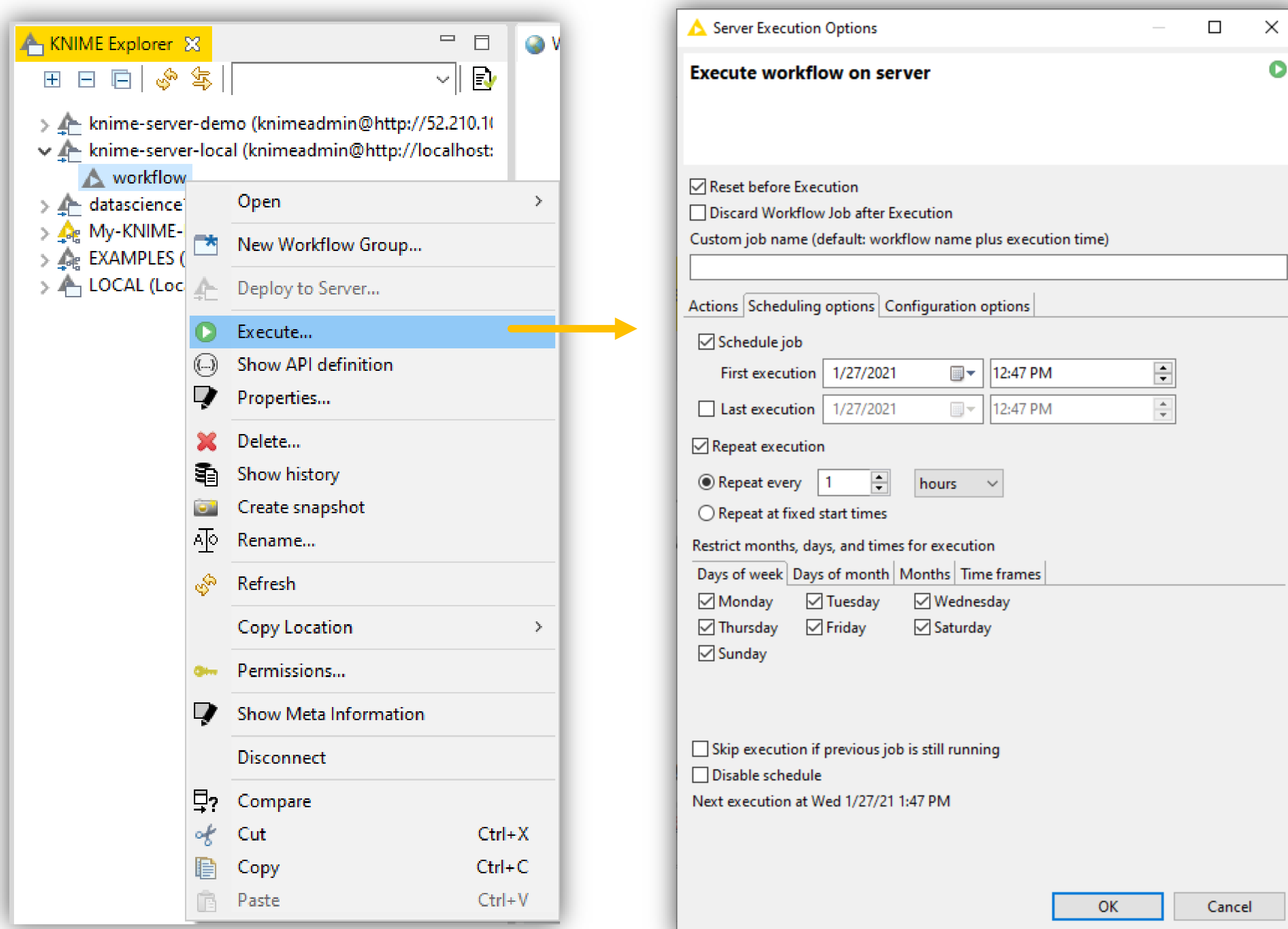
KNIME Server Architecture



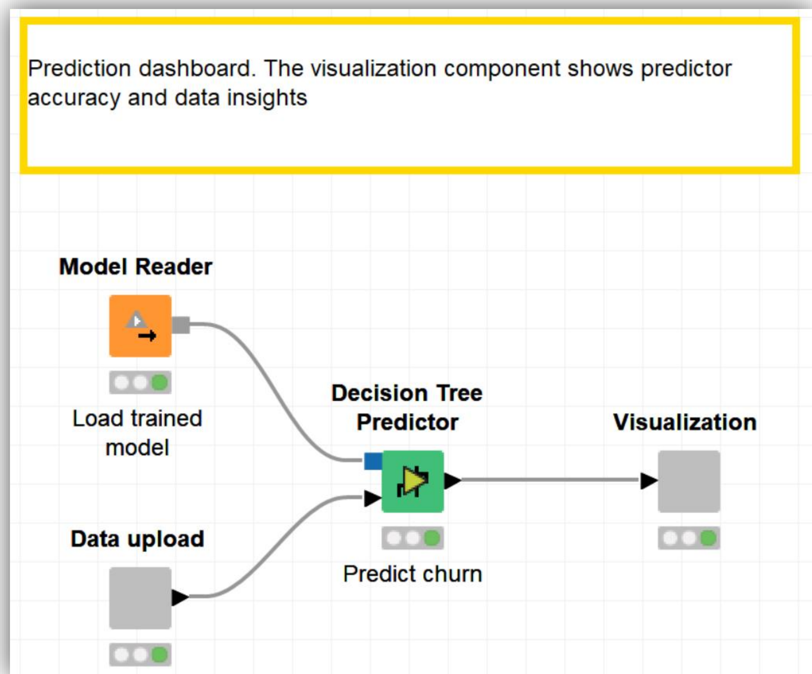
Usual ways to deploy a workflow on KNIME Server

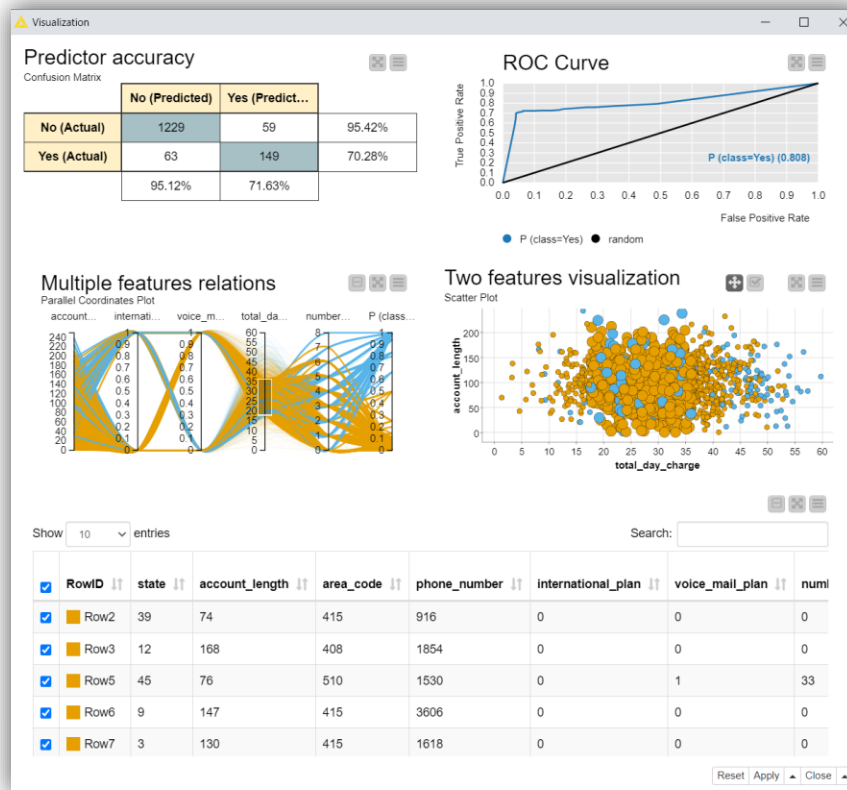
- Deployment by Scheduling Automation
- Deployment to Guided Analytics Application
- Deployment as a REST Service

Deployment by Scheduling Automation

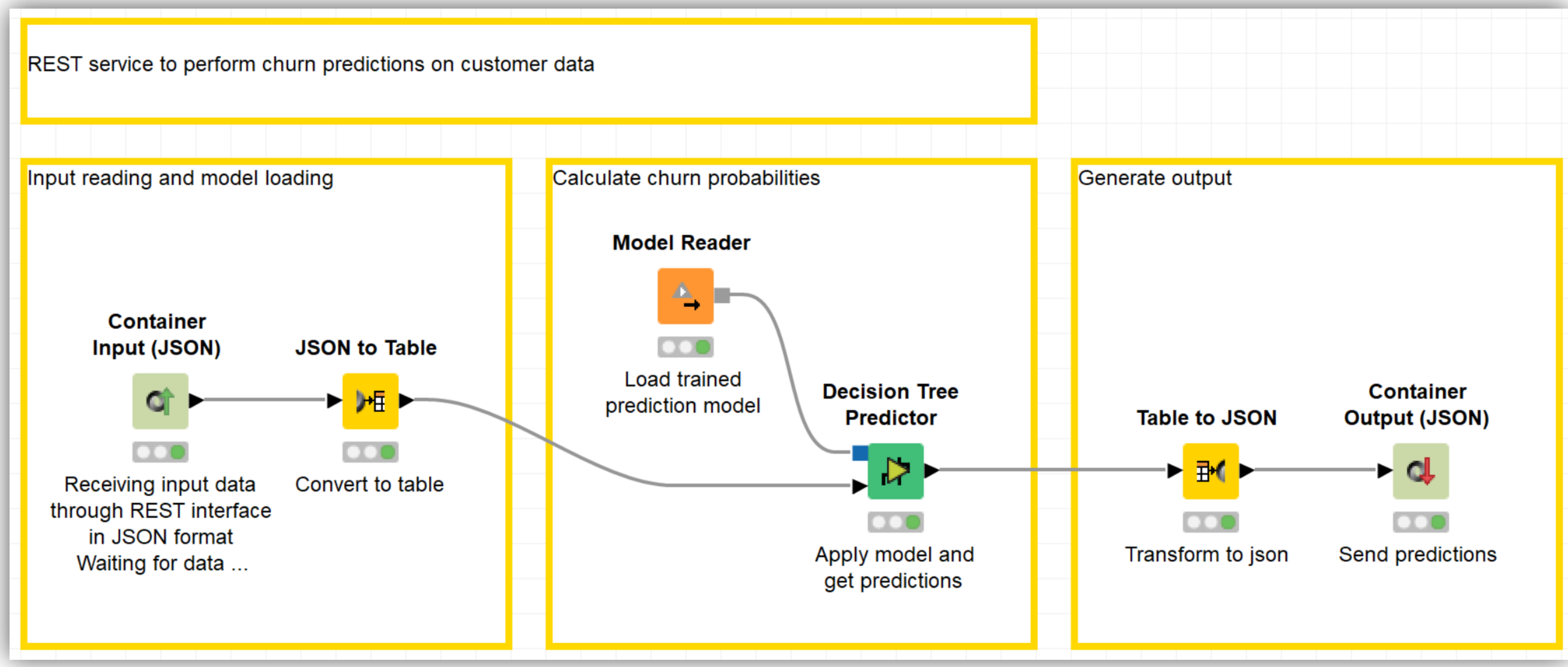


Deployment to Guided Analytics Application





Deployment as a REST Service



Usual ways to deploy a workflow on KNIME Server

manually

- Deployment by Scheduling Automation
- Deployment to Guided Analytics
- Deployment as a REST Service

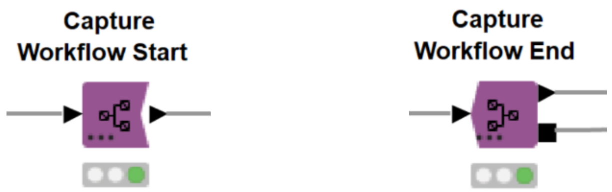
- Integrated Deployment

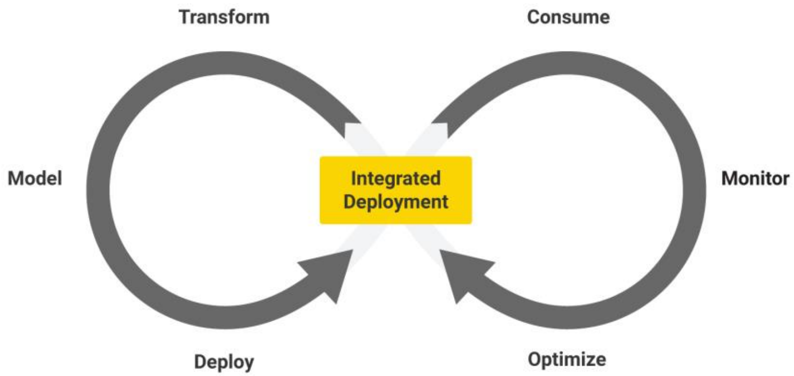
Deployment can be a repetitive task:

- *Monitor & Update* (from cycle)

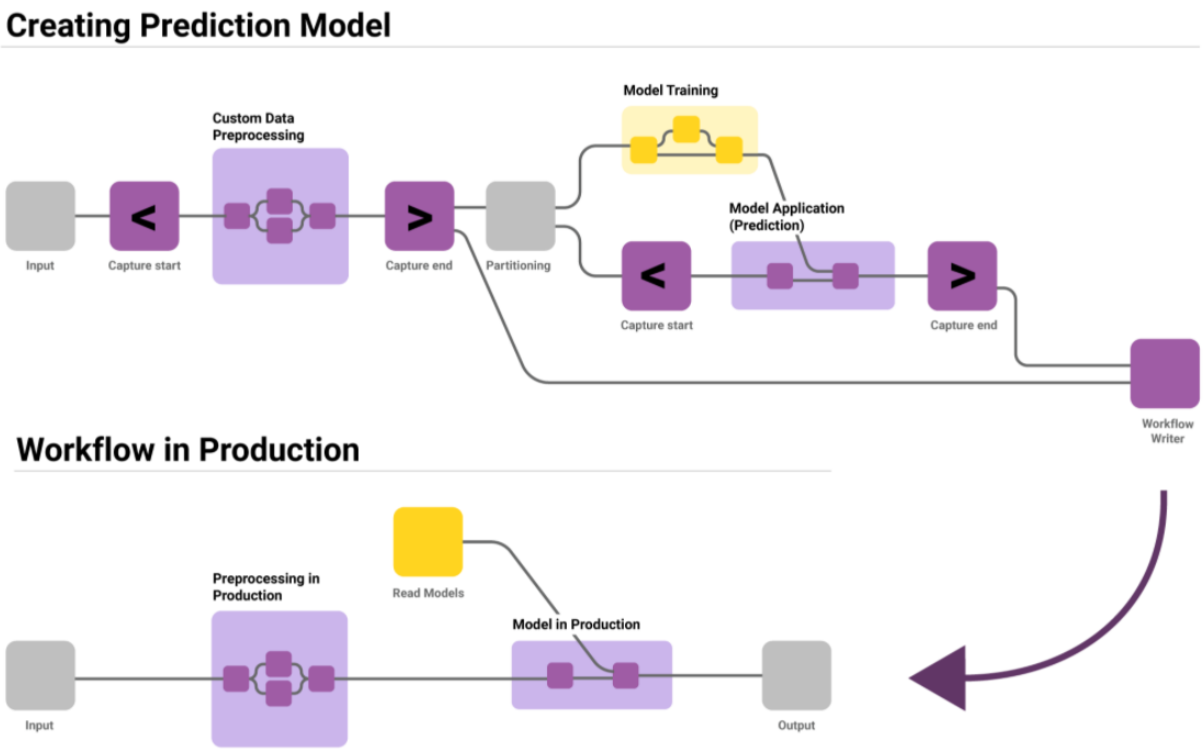
Automating deployment of any of the above, especially **REST Service**

- Build an optimal model
- Isolate core parts of the workflow (preprocessing, model building…) with the special nodes *Capture Workflow Start* and *Capture Workflow End* from the training workflow
- Export the extracted pieces to build the deployment workflow





Integrated Deployment



- Automatically build and deploy deployment workflows
- Mostly used to automatically capture and deploy a model as REST API from the workflow which trains and validates the model

Thank you

For any questions please contact: education@knime.com