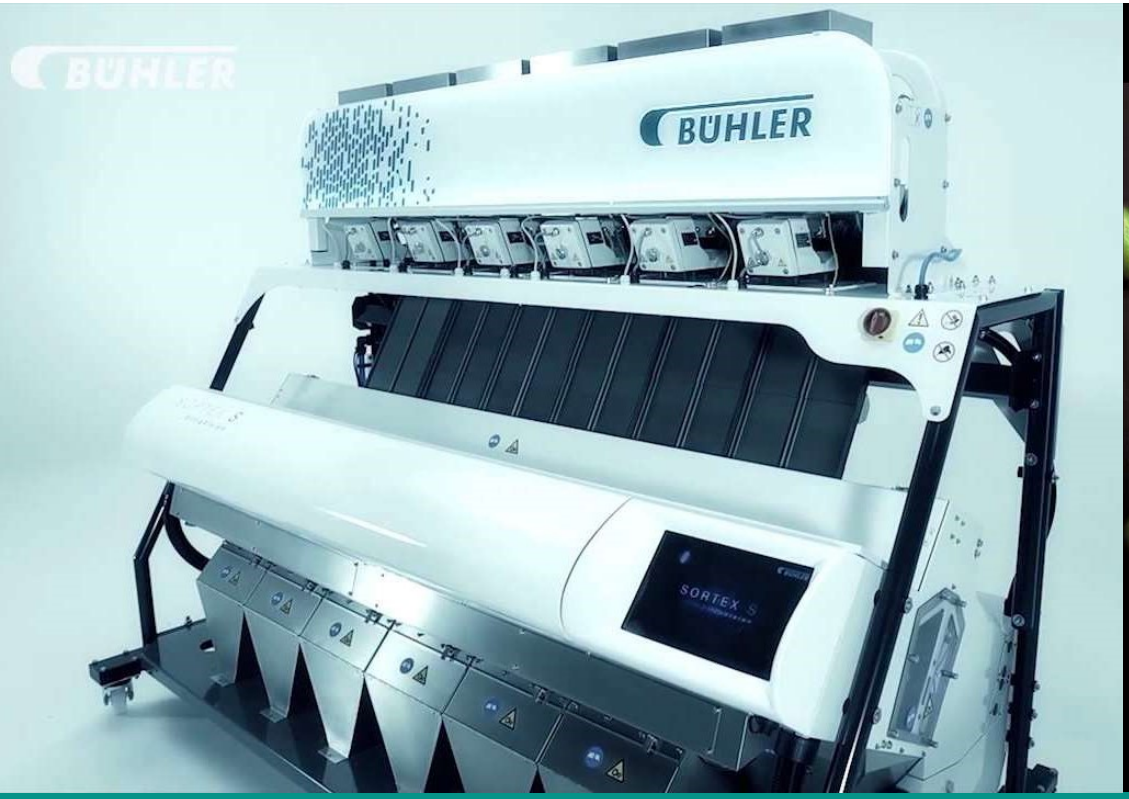
Industrialization of rice grain recognition

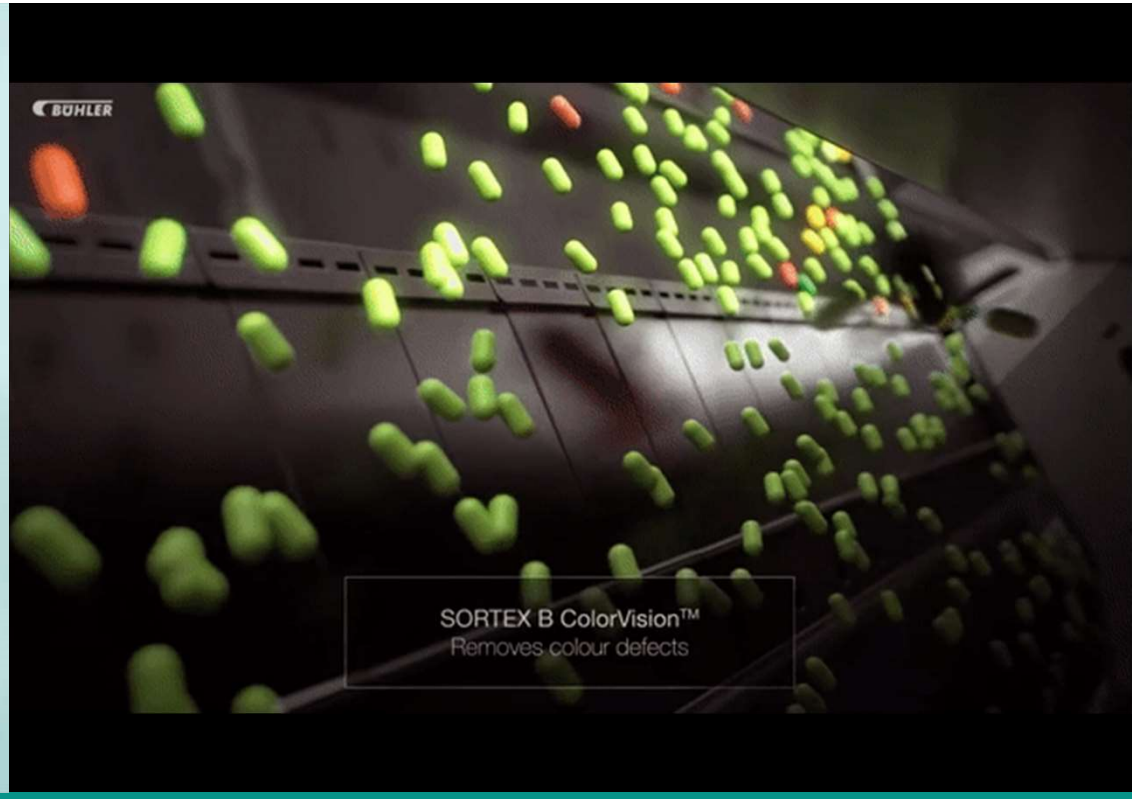
KNIME Development and industrialization

Berlin, March 20th 2019

HES-SO Valais - Switzerland

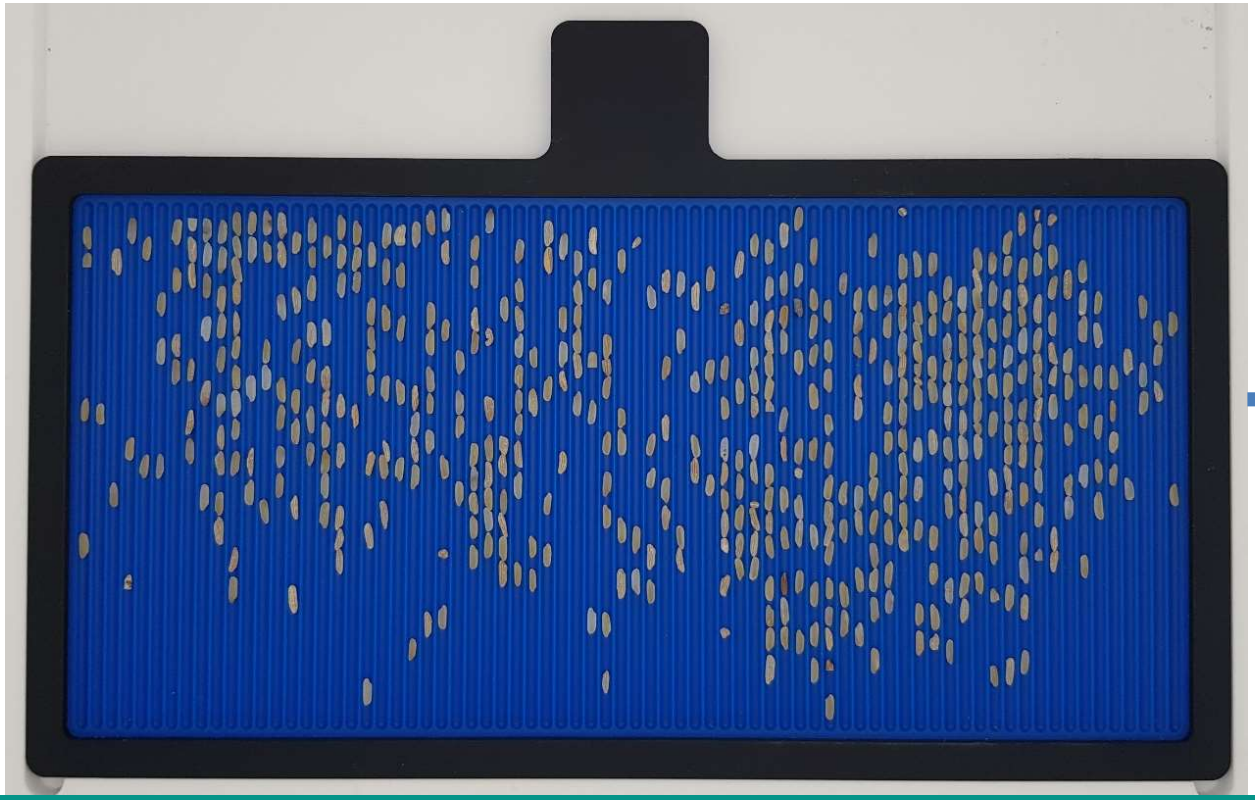
Dominique Genoud – dominique.genoud@hevs.ch

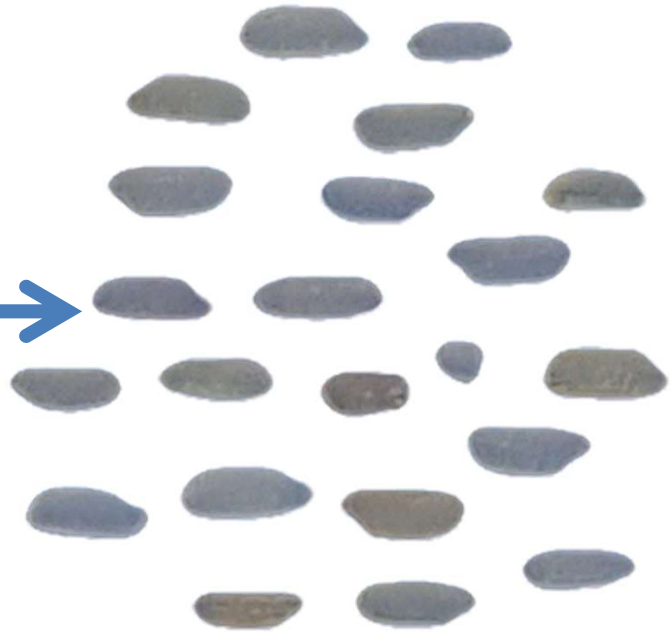




Sorts up to 20 tons per hour → up to 40% of good grains spared
Rice supplies half of the world needs for food

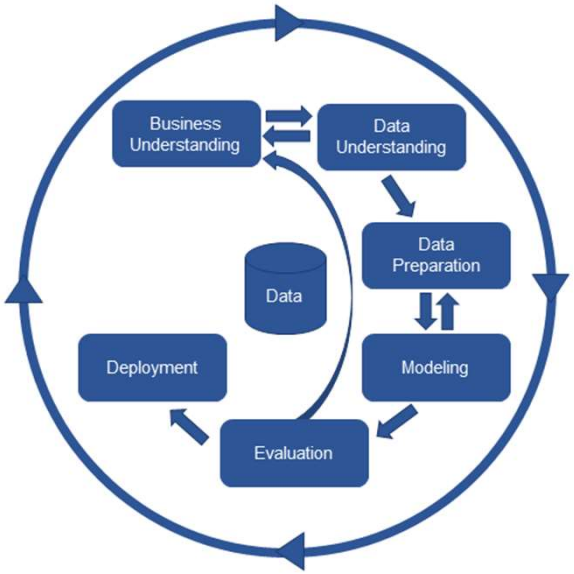
120’000 rice varieties



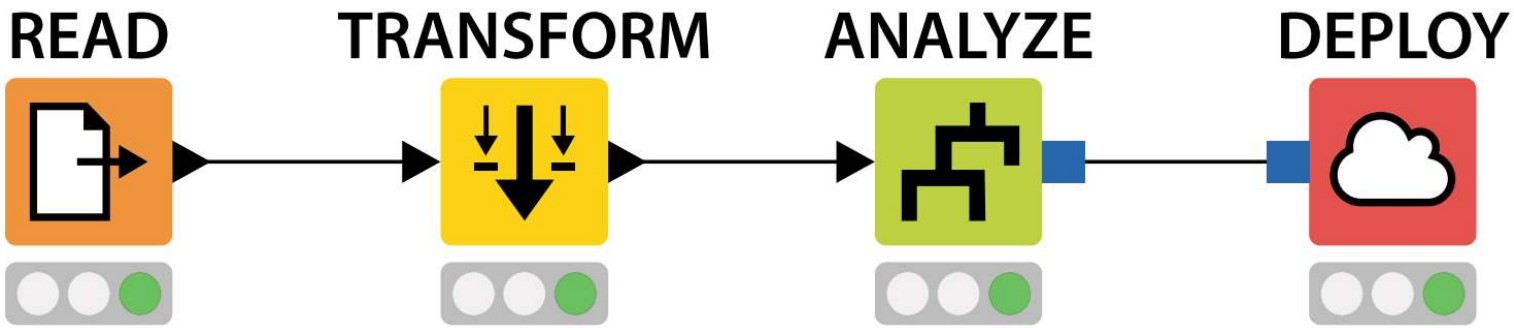


Conventional image processing: Color discrimination | Size of the grain | A lot of manual setup

Methodology



CRISP-DM

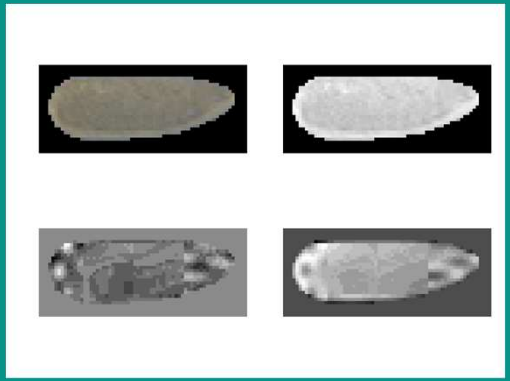


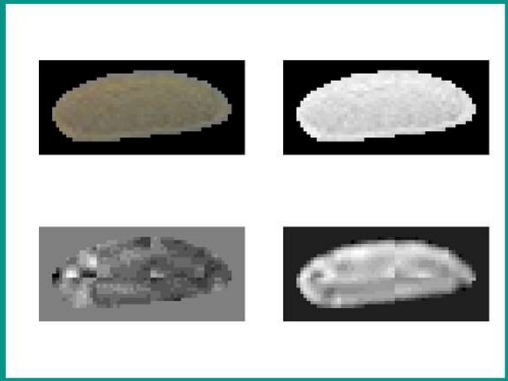
KNIME

Feature Extraction

S filename	D Column...	D Column...	I Column...	D Column...	D Column...	D Column...	D Column...	D Column...
training_segmented\CA_23_1_GalaxyS8\CA_23_1_GalaxyS8_1.png	0.738	0.834	67	3.68	0.769	0.365	147.864	9.39
training_segmented\CA_23_1_GalaxyS8\CA_23_1_GalaxyS8_10....	0.896	0.884	40	3.222	0.783	0.448	136.311	9.417
training_segmented\CA_23_1_GalaxyS8\CA_23_1_GalaxyS8_2.png	1.071	0.866	45	3.25	0.732	0.414	135.466	9.346
training_segmented\CA_23_1_GalaxyS8\CA_23_1_GalaxyS8_3.png	1.067	0.917	34	2.7	0.766	0.574	122.255	10.948
training_segmented\CA_23_1_GalaxyS8\CA_23_1_GalaxyS8_4.png	0.834	0.928	28	2.273	0.82	0.82	145.127	9.317
training_segmented\CA_23_1_GalaxyS8\CA_23_1_GalaxyS8_5.png	0.781	0.863	65	4.25	0.762	0.299	146.002	8.49
training_segmented\CA_23_1_GalaxyS8\CA_23_1_GalaxyS8_6.png	1.064	0.816	41	2.864	0.699	0.478	148.126	10.653

- External python script
- ~100 features
  - Color
  - Spatial
  - Geometric
  - 6 Different phone models





Pre-processing

International Organization for Standardization

USDA

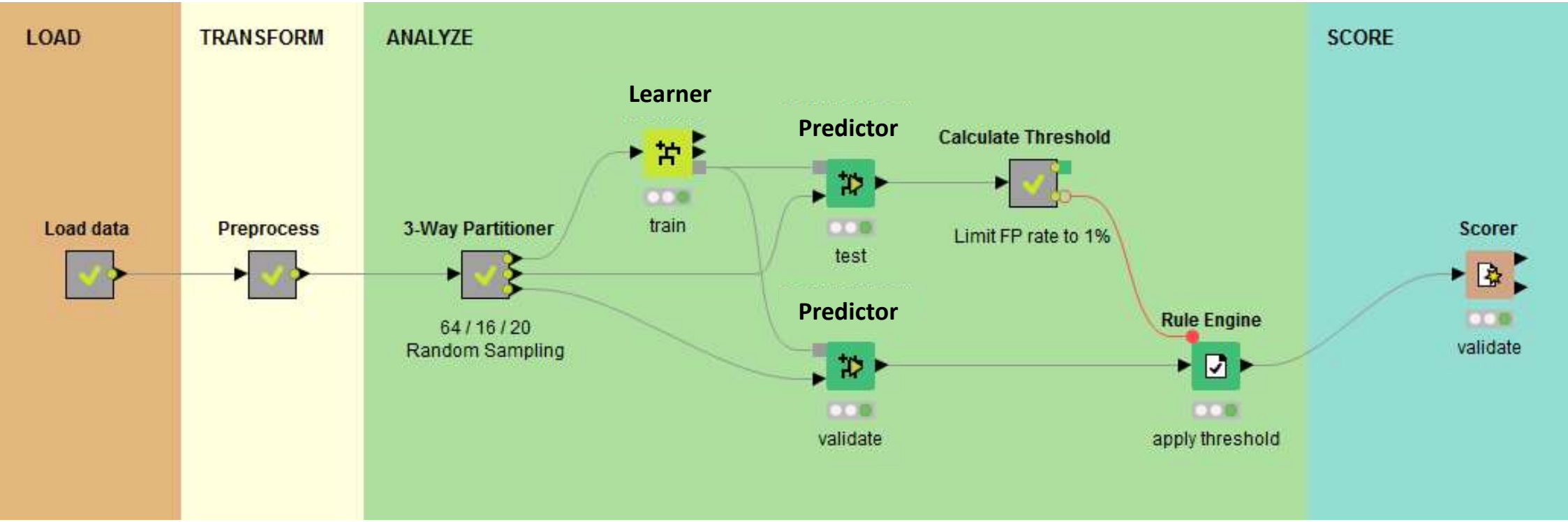
BÜHLER

```
$Sample$ = 1 => "sample"
$Sample$ = 2 => "sortex rejects"
$Sample$ = 3 => "damage (other)"
$Sample$ = 4 => "red"
$Sample$ = 5 => "damage (pecky"
$Sample$ = 6 => "damage (other)"
$Sample$ = 7 => "damage (other)"
$Sample$ = 8 => "heat damage"
$Sample$ = 9 => "damage (stain)"
$Sample$ >= 10 AND $Sample$ <= 12 => "good"
```

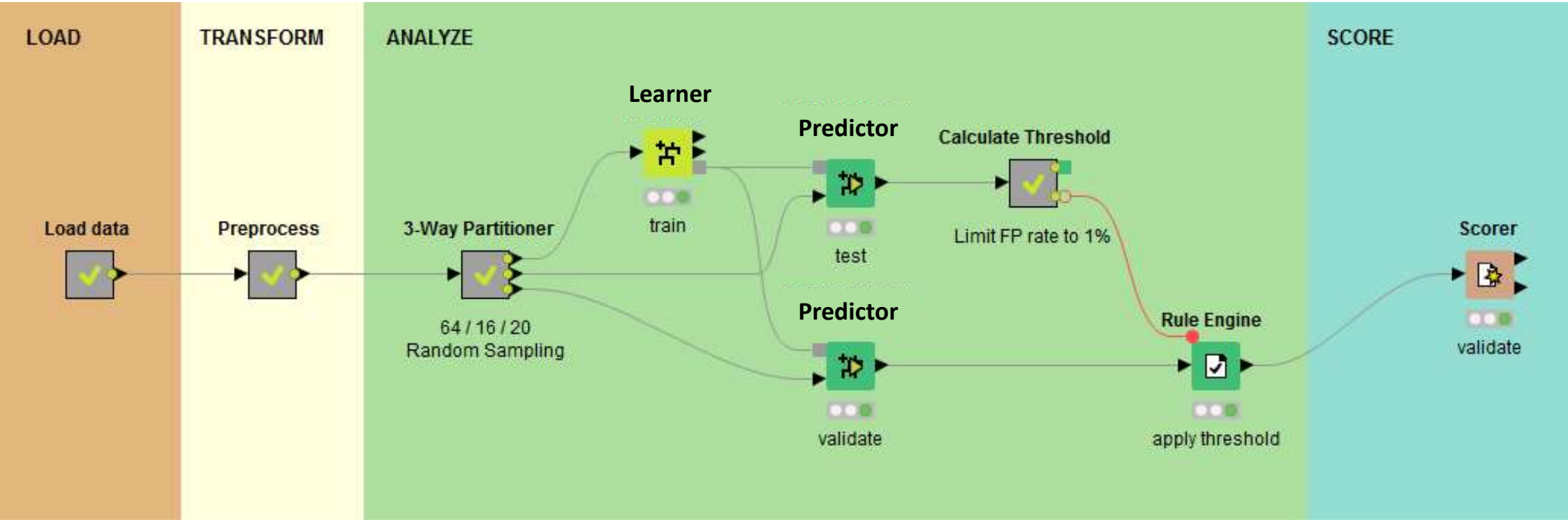
	S samplename	S class
S8_147.png	good	accept
S8_316.png	good	accept
S8_15.png	damage (stain)	reject
S8_183.png	heat damage	reject
S8_3.png	damage (other)	reject
S8_75.png	good	accept
_1.png	heat damage	reject
S8_68.png	good	accept

First loop → separate good grains from defect grains

Model creation

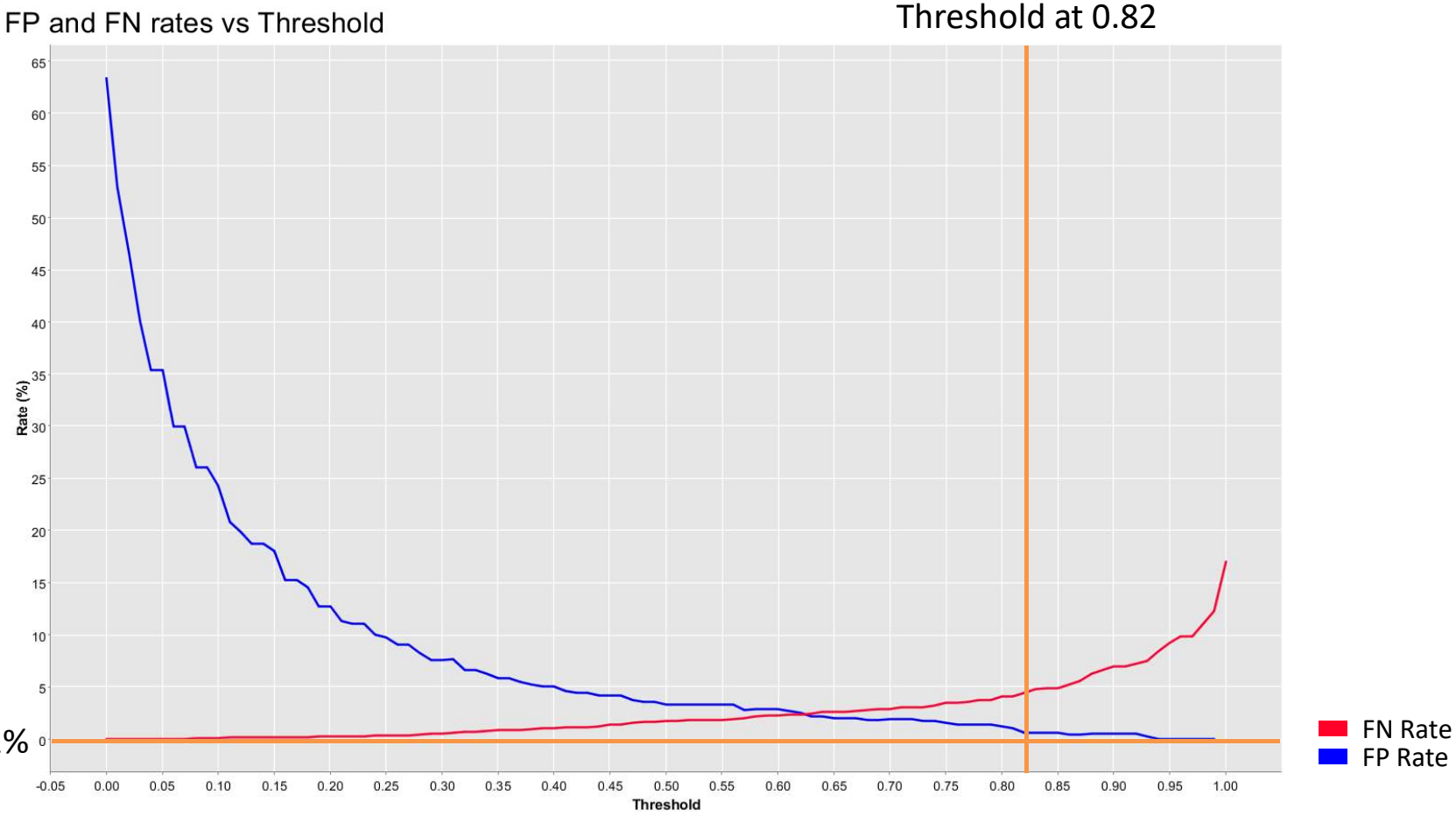


Model creation → learn / predict / compute threshold



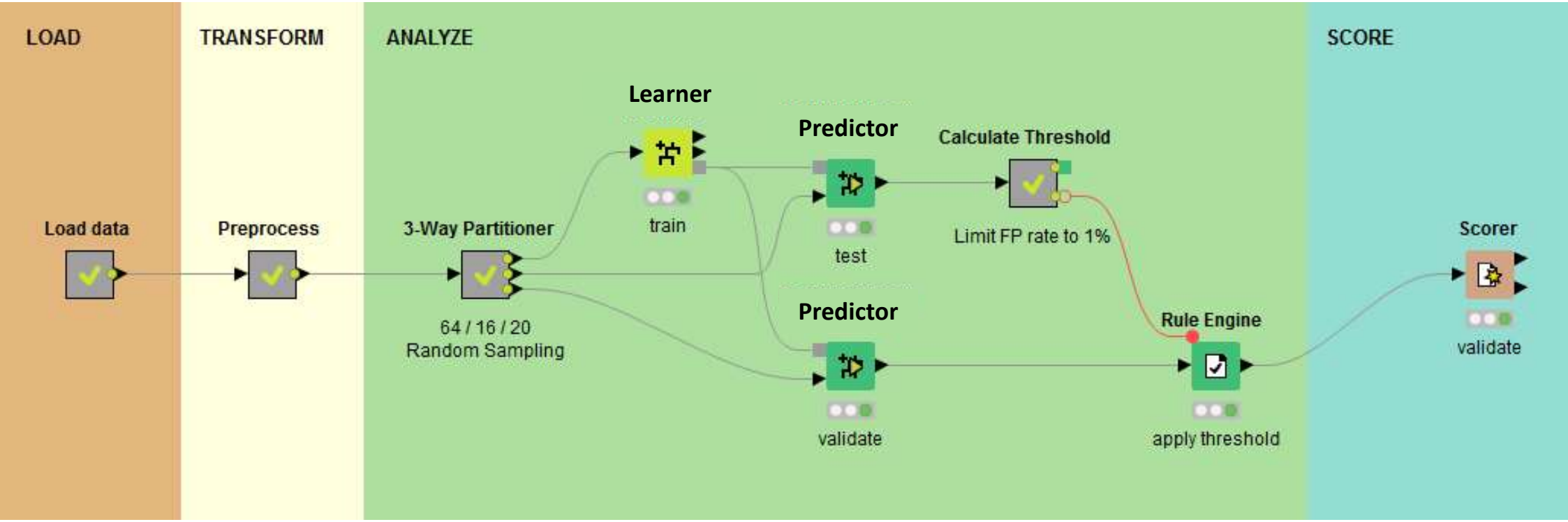
Training

Threshold at 0.82

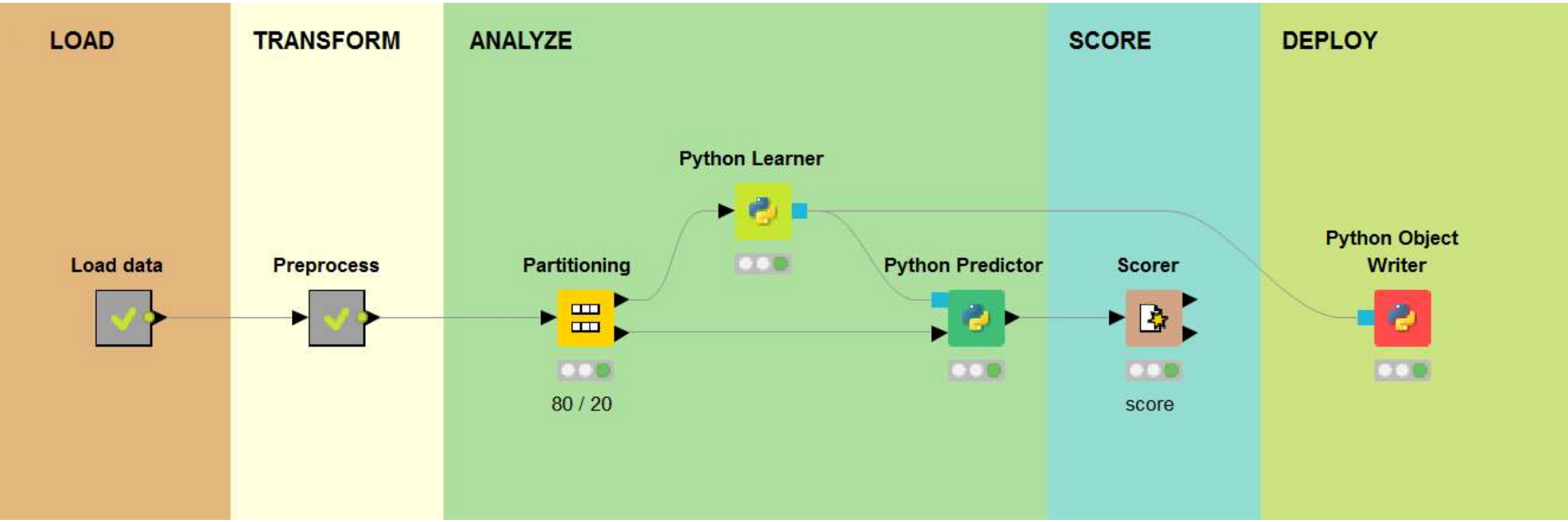


FP rate at 1%

Model creation → validate performance with pre-computed threshold



Industrialisation → customer constraints (python and Azure server)



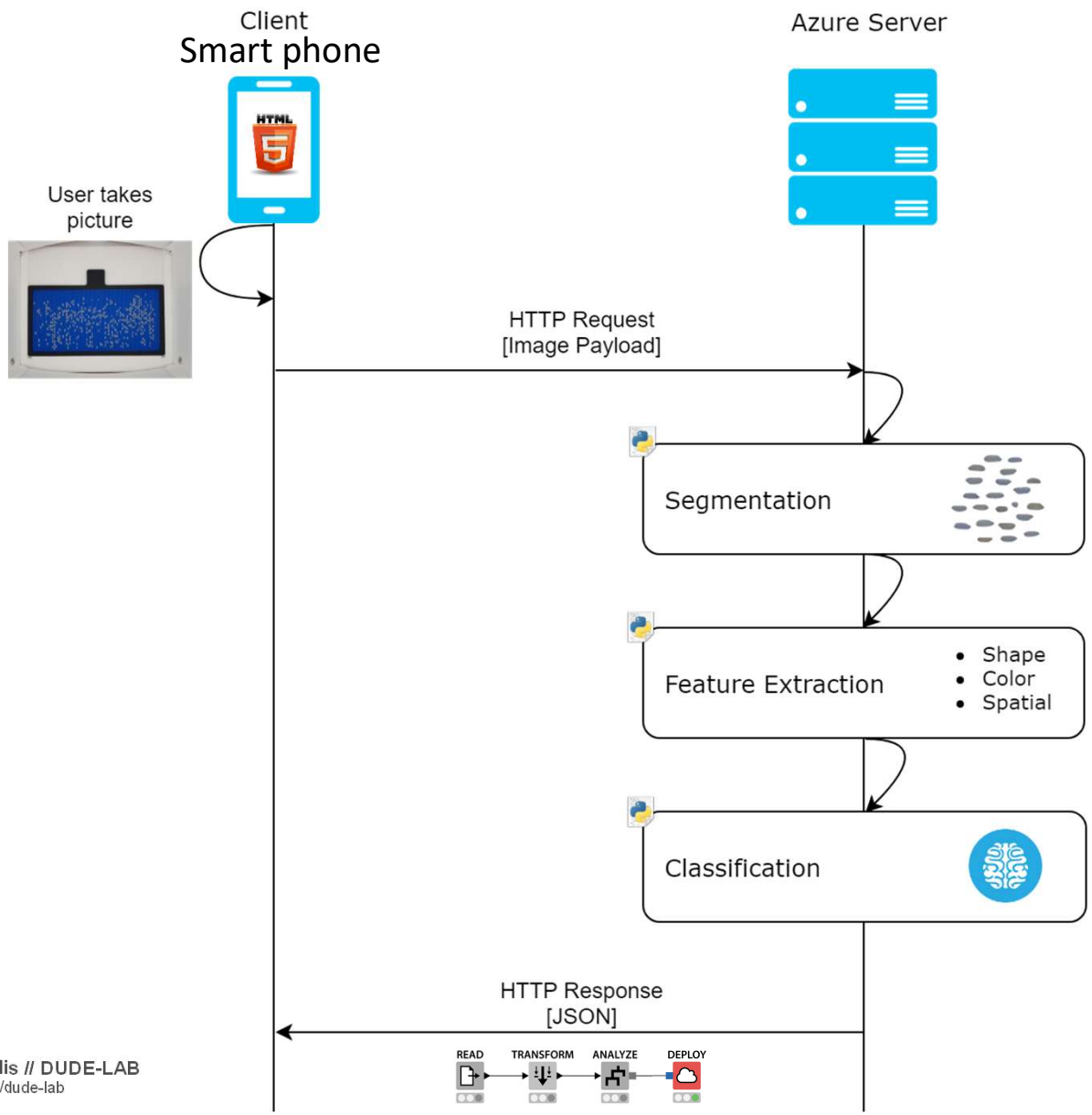
Industrialisation

Python Learner

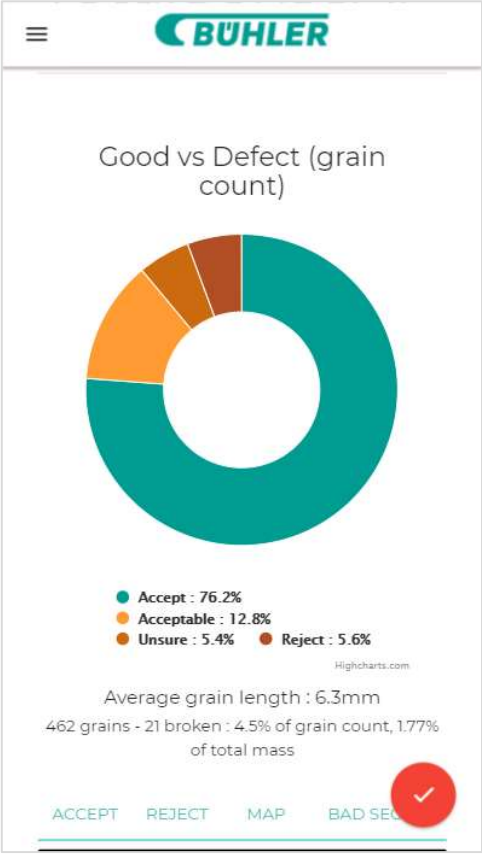
```python
from sklearn.ensemble import [illegible]Classifier
from sklearn import preprocessing



#set columns
columns = sorted(input_table.columns.drop(['filename', 'samplename', 'class']))

#label encoder

# Designate features to learn from
features = input_table[columns]


# Create a [illegible] classifier
output_model = {"columns" : columns, "labels" : sorted(input_table['class'].unique()), "threshold" : 0.82}

# Choose class labels column
label_encoder = preprocessing.LabelEncoder()
label_encoder.fit(output_model['labels'])
labels = label_encoder.transform(input_table['class'])

output_model['model'] = [illegible]Classifier([illegible])

# Fit the model using the features and labels
output_model['model'].fit(features, labels)
```

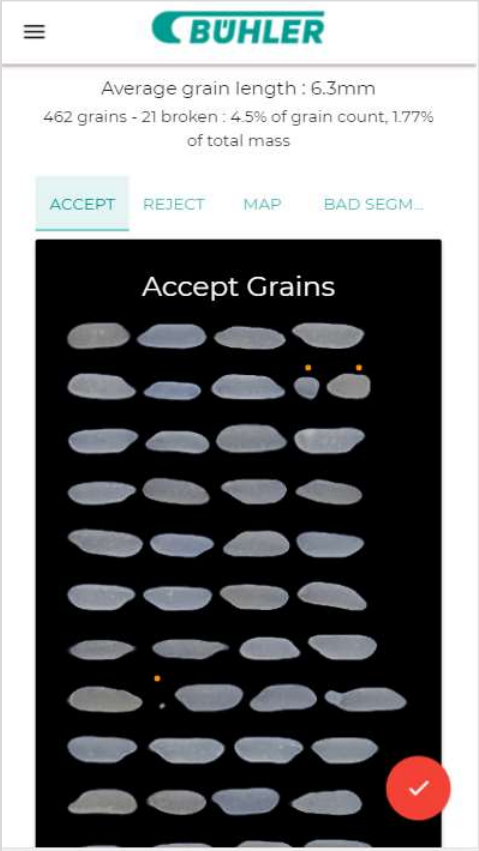
Python Predictor

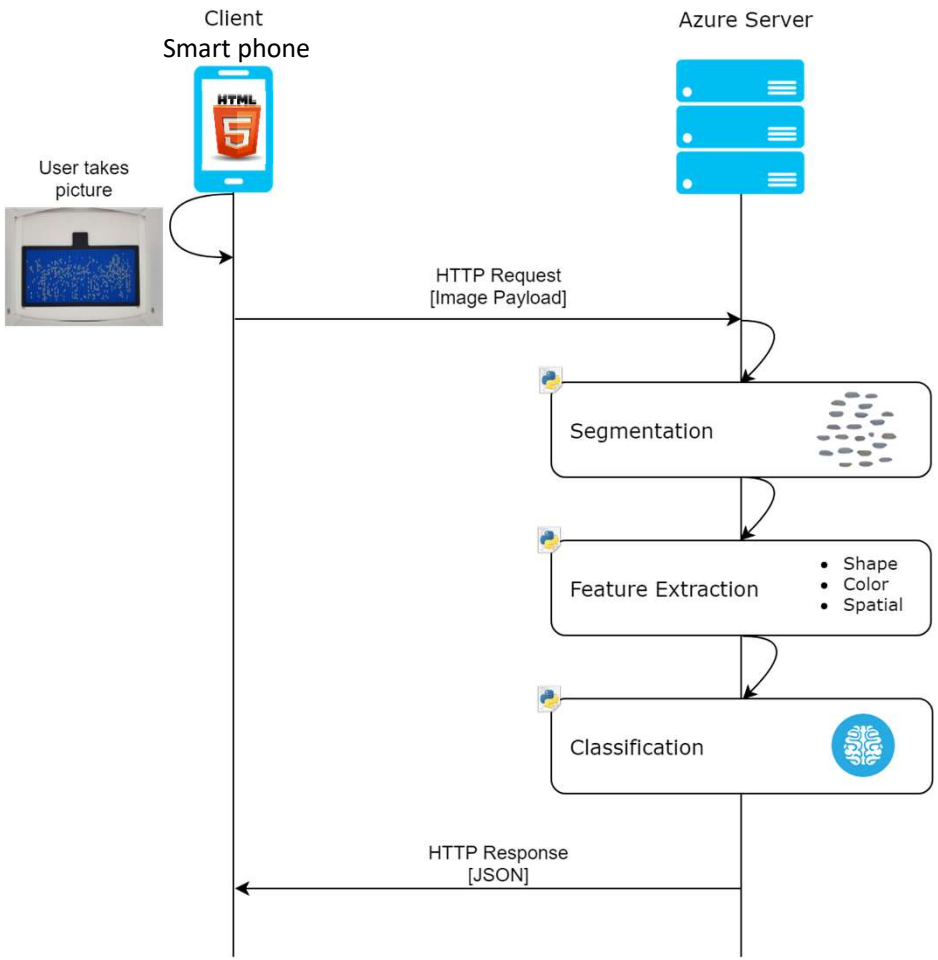
```python
mport preprocessing
s np

res from the model
ut_table[input_model['columns']]

able to output
 input_table.copy()

g existing model + new data
= input_model['model'].predict_proba(new_data)

al probabilities for classes

_model['labels']:
le[input_model['labels'][c] + "_p"] = probabilities[:,c]


 (taking threshold into account)
[0 if k >= input_model['threshold'] else 1 for k in probabilities[:,0]]

#output class as string
le = preprocessing.LabelEncoder()
le.fit(input_model['labels'])
output_table['prediction'] = le.inverse_transform(predictions)
```



Client
Smart phone
Azure Server
User takes picture
HTTP Request
[Image Payload]
Segmentation
Feature Extraction
Shape
Color
Spatial
Classification
HTTP Response
[JSON]
READ
TRANSFORM
ANALYZE
DEPLOY

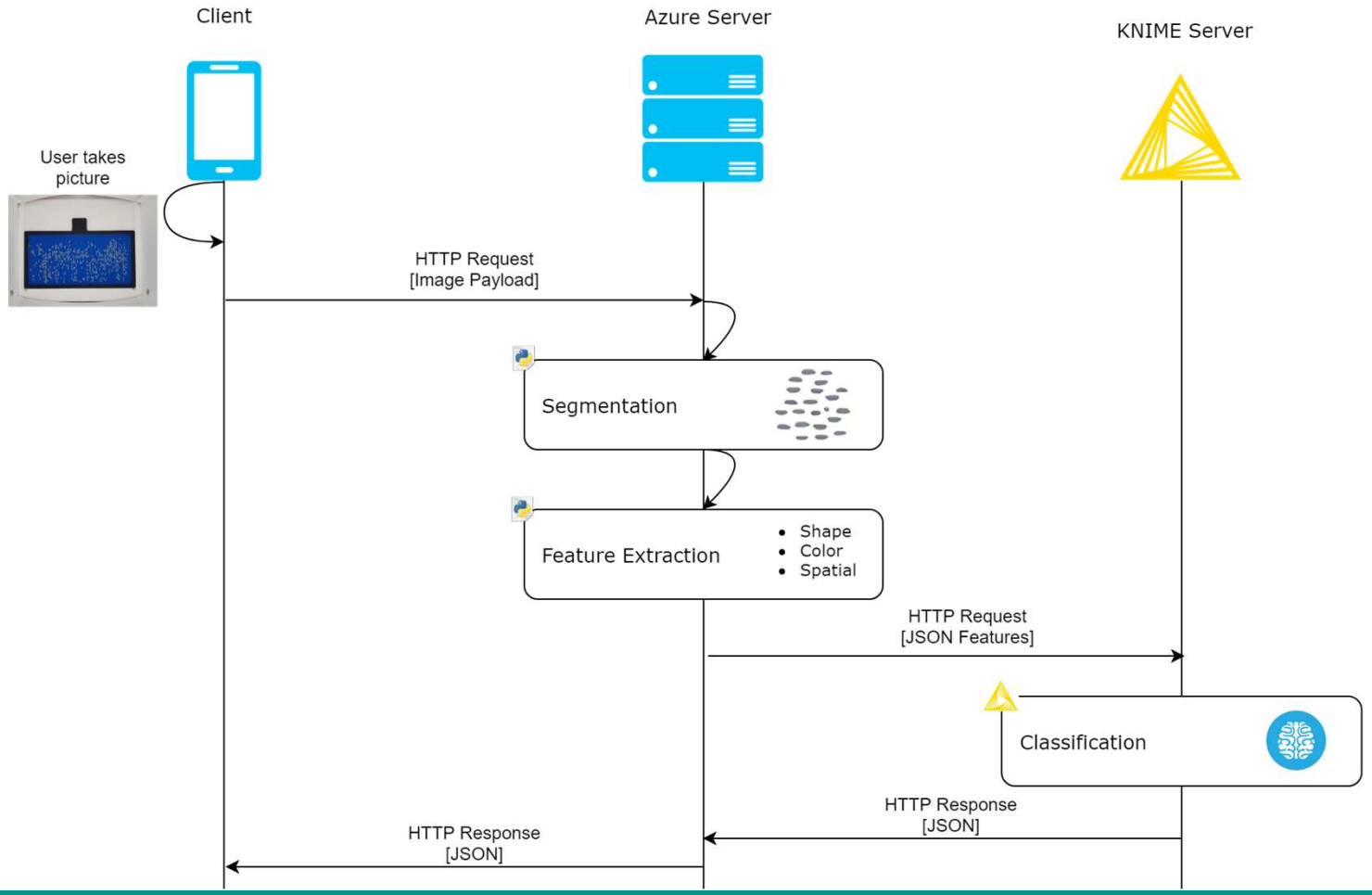
Recognition and statistics



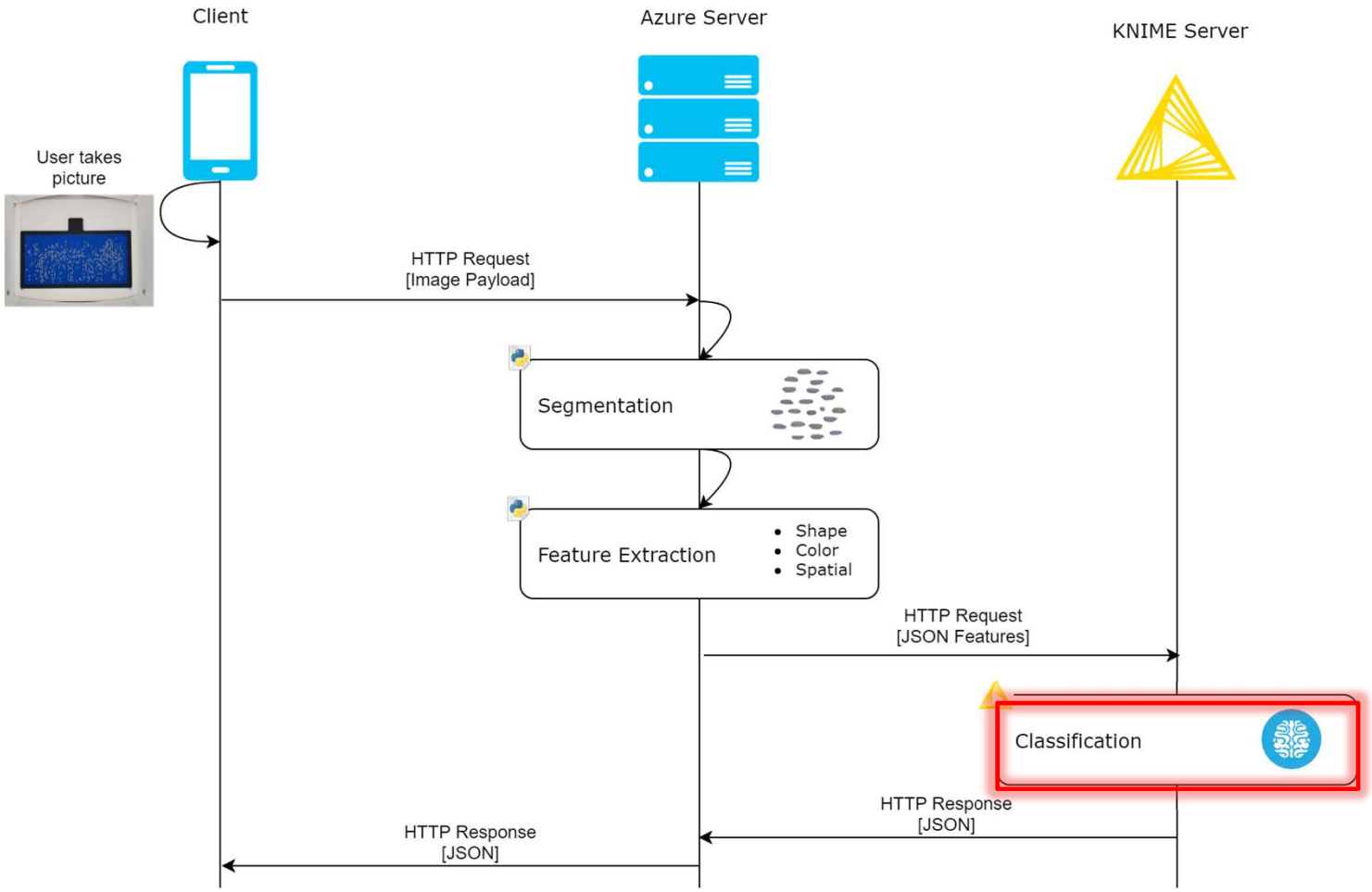




Current implementation, with customer constraints

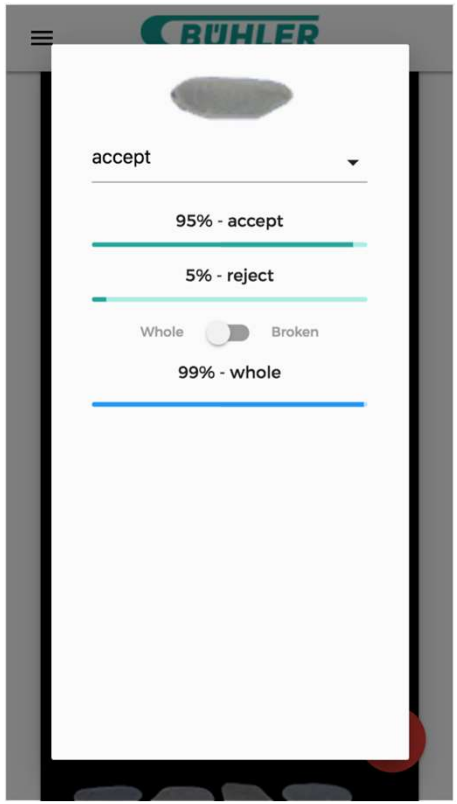


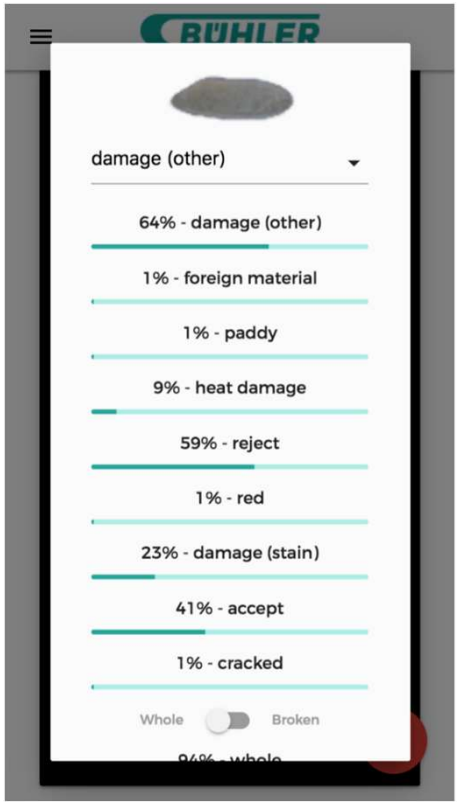
Further performance improvements using KNIME server

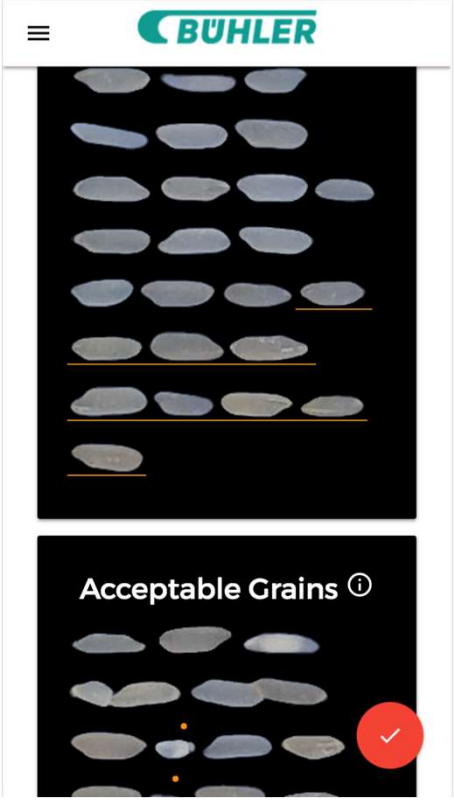


Further performance improvements using KNIME server

TotalSense.ai results







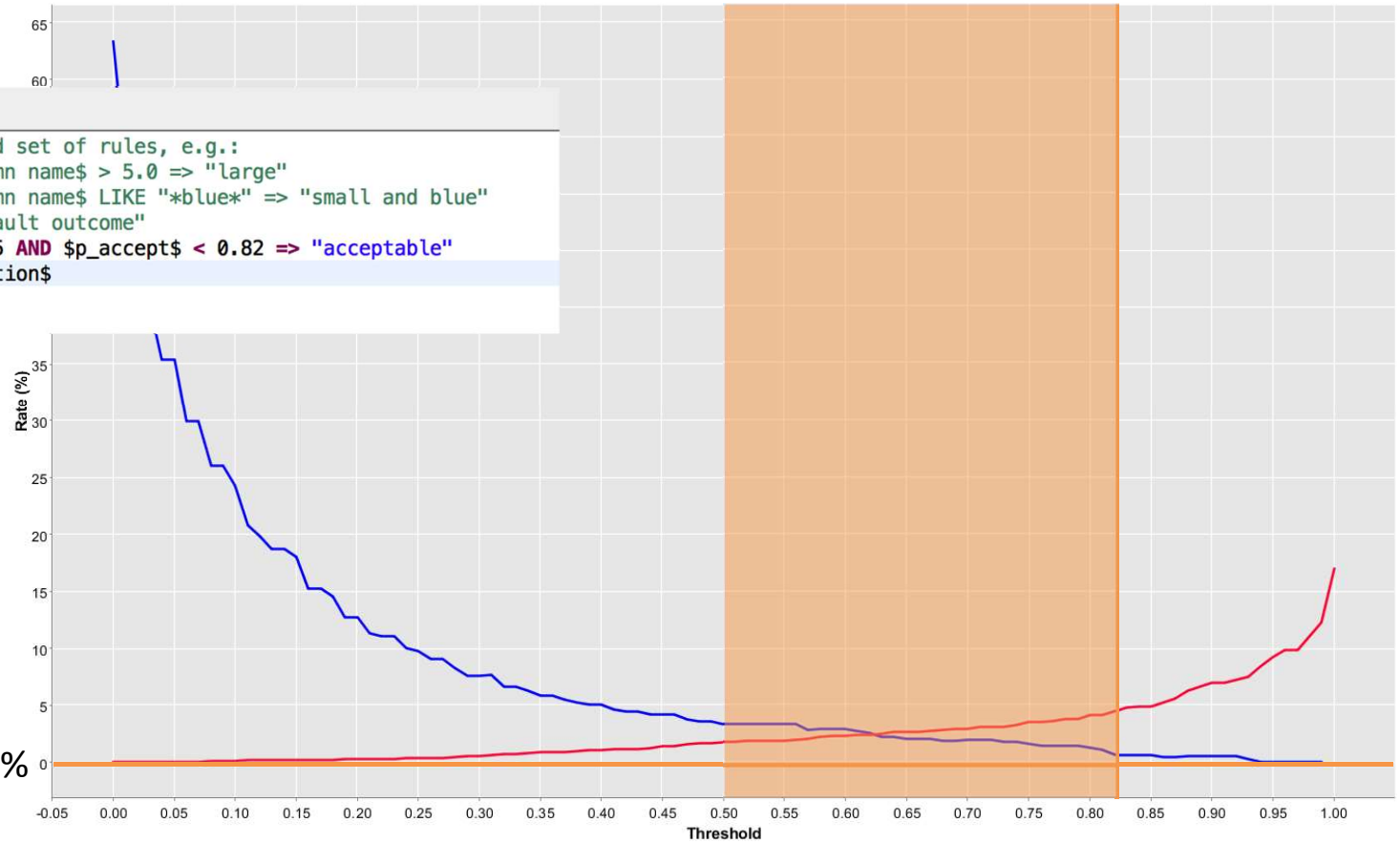
Training

Threshold at 0.82

FP and FN rates vs Threshold

Expression

```
// enter ordered set of rules, e.g.:
// $double column name$ > 5.0 => "large"
// $string column name$ LIKE "*blue*" => "small and blue"
// TRUE => "default outcome"
$p_accept$ > 0.5 AND $p_accept$ < 0.82 => "acceptable"
TRUE => $prediction$
```



FP rate at 1%

FN Rate
FP Rate

READ TRANSFORM ANALYZE DEPLOY

Thank you for your attention.

Any questions?