

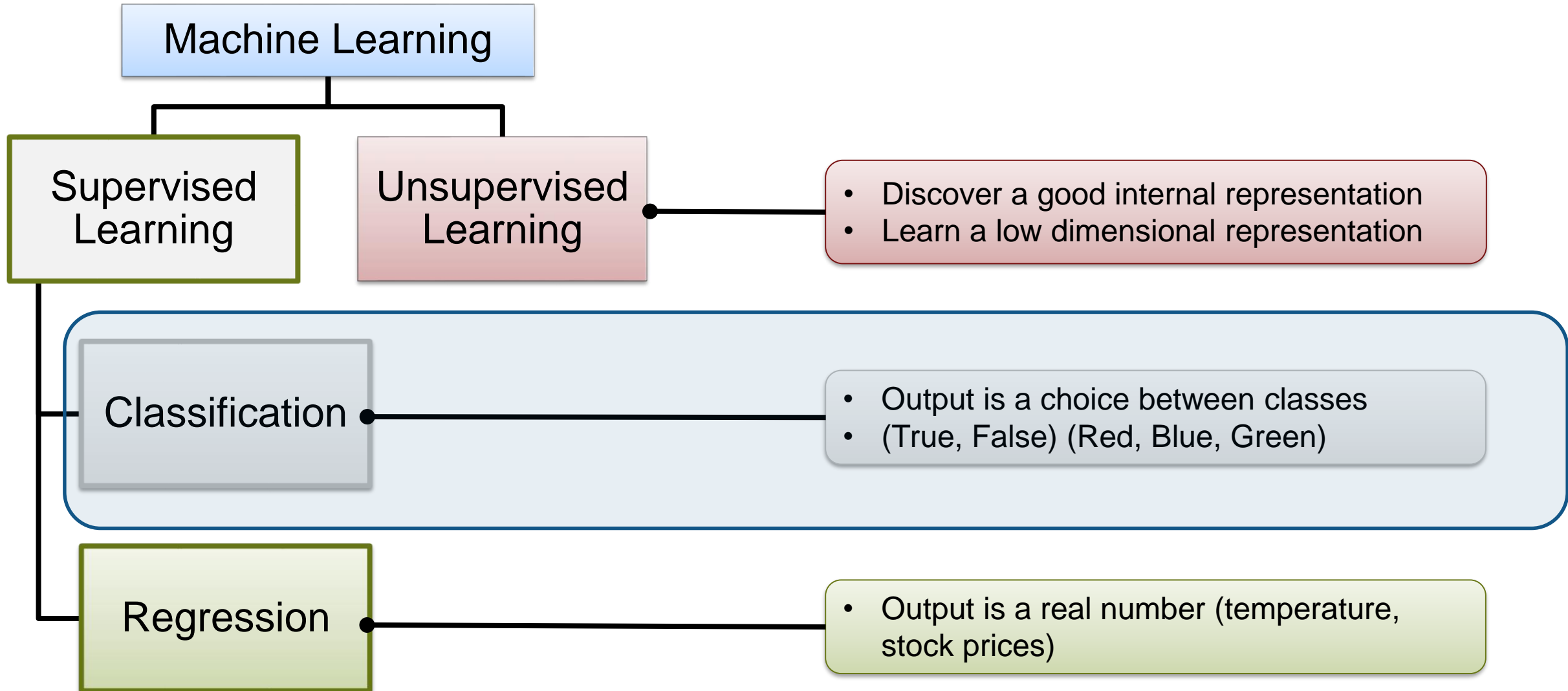
# MATLAB EXPO 2016

## Develop Predictive Maintenance Algorithms using MATLAB

Dr. Sarah Drewes, MathWorks Consulting Services



# Different Types of Learning



# Classification in Predictive Maintenance

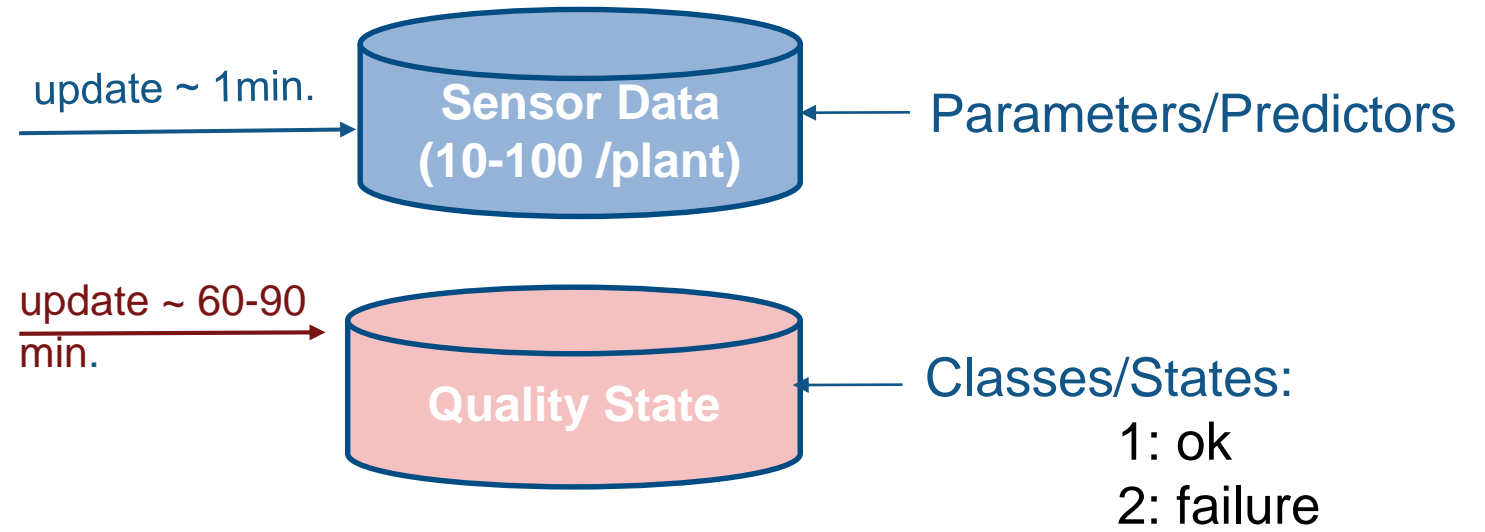
- **Parameters/Predictors:** Sensor data, control settings
- **Classes/States:** Failure states, time horizon until failure/ material fatigue

**Goal:** Predict failure from sensor data

## Prerequisites:

- Machine-readable data format
- Sufficient historical data containing meaningful information

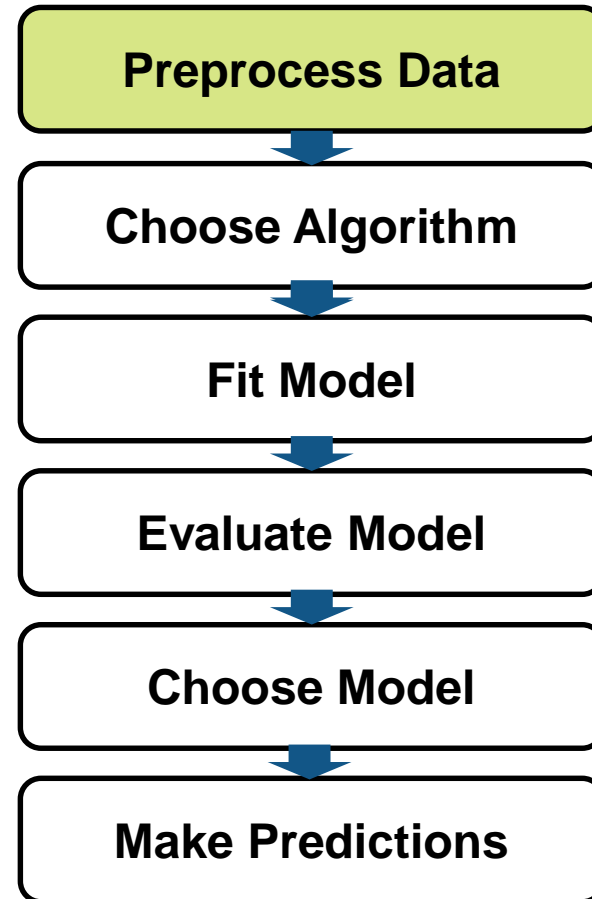
# Classification model generation @MONDI Gronau



Which sensor measurements indicate machine failure?

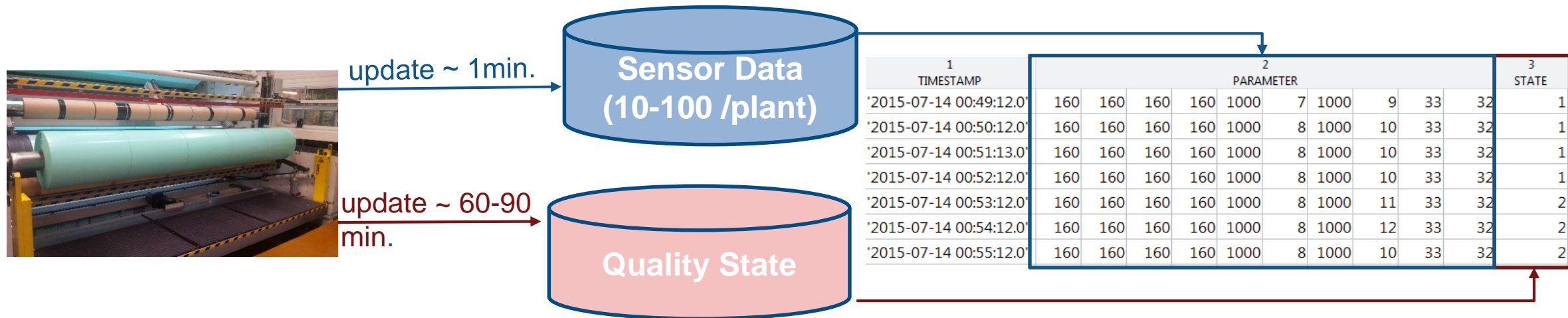
# Classification model generation

## Basic Workflow



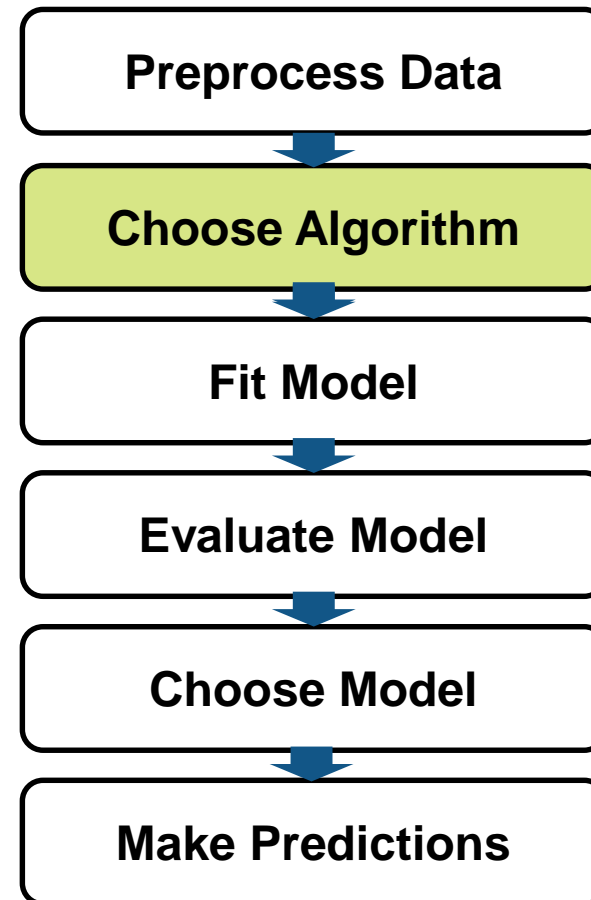
# Classification model generation- Prepare data

- Preprocess sensor data: clean invalid data, disregard constant values, identify data types
- Aggregate per time stamp



# Classification model generation

## Basic Workflow

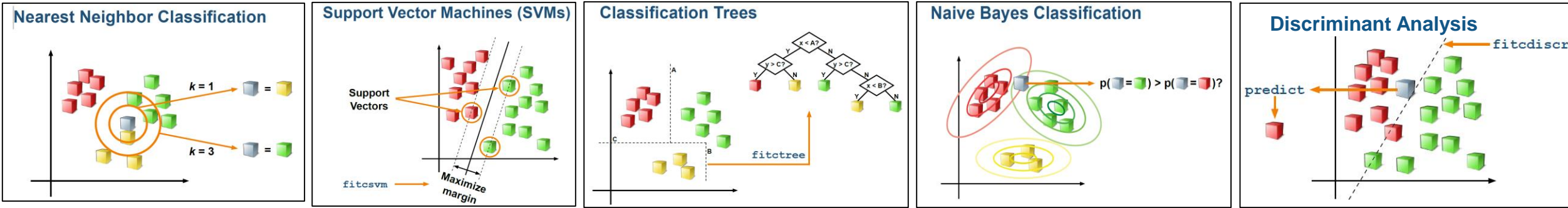


# Classification model generation

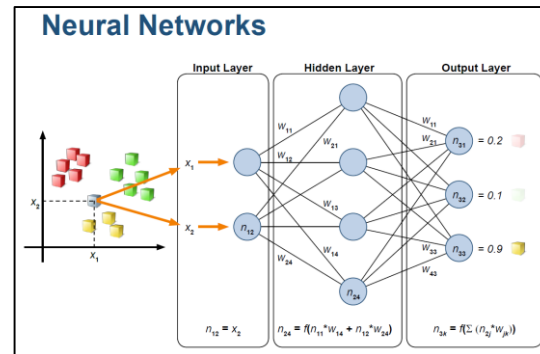
## Choose algorithms

### Possible Classification Methods

#### Statistics and Machine Learning



#### Neural Network





# Classification model generation

## Choose an algorithm

- Distinguish 'categorical' (= discrete) and other (= continuous) predictors
- A priori analysis of data, e.g., test for normal distribution
- Reduce dimension of predictor variables, e.g., principal component analysis (PCA)
- Use ensemble learning to reduce sensitivity of learning algorithms, e.g. TreeBagger for classification trees

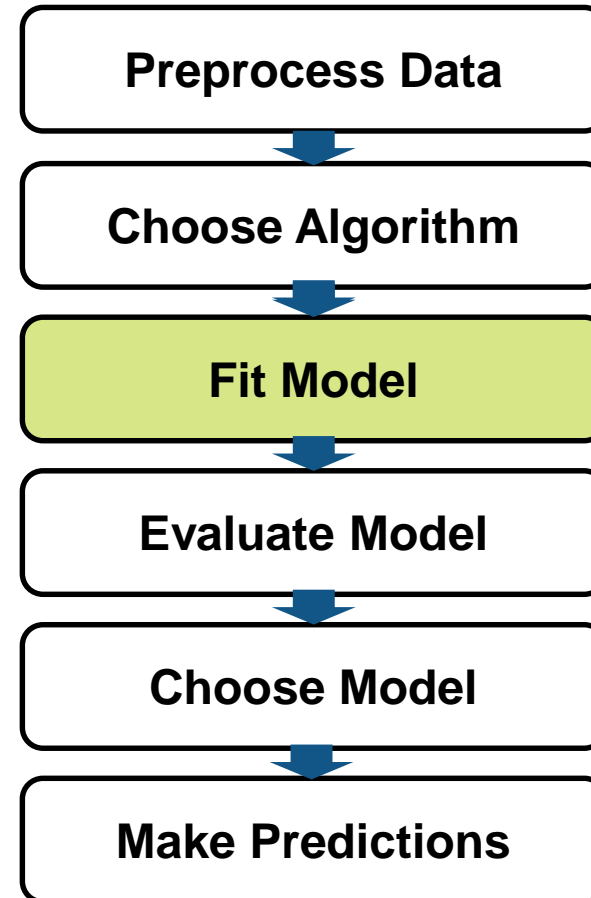
# Classification model generation

## Choose an algorithm

	Algorithm	Function	Categorical Predictors?	Data	Functions to Examine Data	Notes
→	Nearest Neighbor	<code>fitcknn</code>	Y (but not both)	Normalize (distance-based calculation).	<code>pdist</code> <code>pdist2</code>	Better results in lower dimensions. High memory usage.
- - →	Naive Bayes	<code>fitcnb</code>	Y	Assumes normal distributions (can specify kernel for nonnormal).	<code>probplot</code> <code>jbtest</code> <code>ksdensity</code>	Popular for high dimensional problems. Computationally efficient. Widely used for text classification.
	Discriminant Analysis	<code>fitcdiscr</code>	N <b>Y</b>	Multivariate normal distribution by class. <b>N</b>	<code>cov</code> <code>vartestn</code> <code>anova1</code> <code>kruskalwallis</code>	Determines mean and covariance for each class. Can specify linear or quadratic discriminant type.
→	Trees	<code>fitctree</code> <code>fitrtree</code>	Y	Any arrangement. Binary comparisons and structure of tree can be examined/adjusted.	<code>view</code>	Computationally efficient. Highly sensitive to training data.
	SVM	<code>fitcsvm</code> <code>fitcecoc</code>	N <b>Y</b>	Linearly separable hyperplane (can specify nonlinear kernel).	<code>ksdensity</code>	Can specify nonlinear kernel distributions. Can adjust optimization parameters.
- - →	Neural Network	<code>patternnet</code>	N <b>Y</b>	Transpose (columns are observations). All data must be numeric.	<code>dummyvar</code> <code>plotconfusion</code> <code>plotroc</code>	Use <code>dummyvar</code> for categorical classes. Models are available as Simulink® blocks.

# Classification model generation

## Basic Workflow

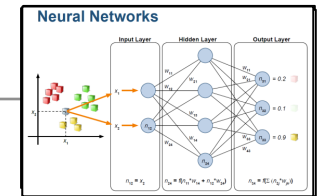


# Classification model generation

## Fit model

Fit model based on historic data

`PredictionModel = fitcoba (PARAMETER, STATE)`



Training Data,  
e.g. 70% of  
historic data

1 TIMESTAMP	2 PARAMETER										3 STATE
'2015-07-14 00:49:12.0'	160	160	160	160	1000	7	1000	9	33	32	1
'2015-07-14 00:50:12.0'	160	160	160	160	1000	8	1000	10	33	32	1
'2015-07-14 00:51:13.0'	160	160	160	160	1000	8	1000	10	33	32	1
'2015-07-14 00:52:12.0'	160	160	160	160	1000	8	1000	10	33	32	1
'2015-07-14 00:53:12.0'	160	160	160	160	1000	8	1000	11	33	32	2
'2015-07-14 00:54:12.0'	160	160	160	160	1000	8	1000	12	33	32	2
'2015-07-14 00:55:12.0'	160	160	160	160	1000	8	1000	10	33	32	2

# Classification model generation

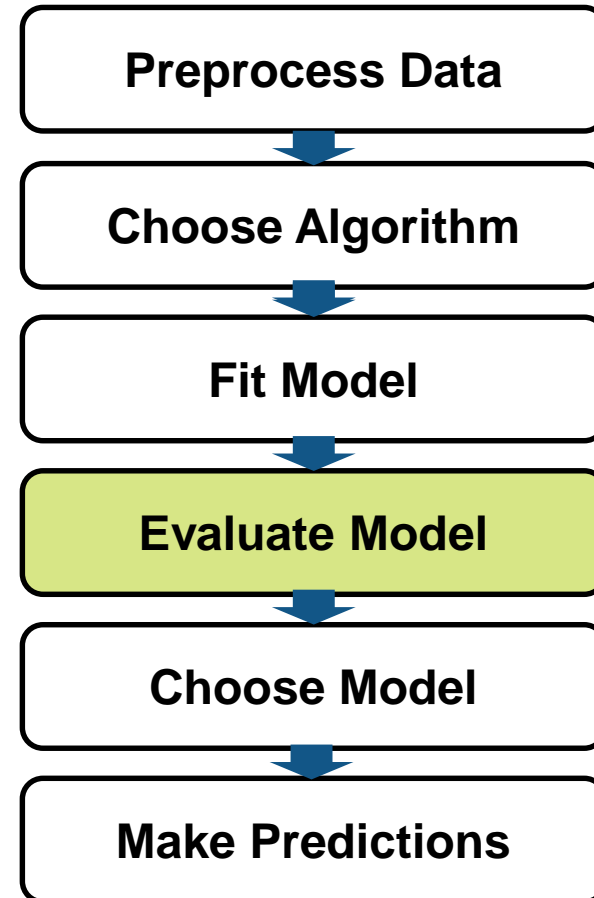
## Fit model

The screenshot displays the 'Classification Learner - Confusion Matrix' window. The interface is divided into several sections:

- CLASSIFICATION LEARNER:** Includes buttons for 'New Session', 'Feature Selection', and 'PCA'.
- Data Browser:** Shows a 'History' list of models:
  - 1 KNN (Last change: Fine KNN)
  - 2 SVM (Last change: Linear SVM)
  - 3 Ensemble (Last change: Bagged Trees)
  - 4 Linear Discriminant (Last change: Linear Discriminant)
  - 5 SVM (Last change: Fine Gaussian SVM)
- Model Selection Panels:**
  - DECISION TREES:** Complex Tree, Medium Tree, Simple Tree, All Trees.
  - DISCRIMINANT ANALYSIS:** Linear Discriminant, Quadratic Discriminant, All Discrimina...
  - LOGISTIC REGRESSION CLASSIFIERS:** Logistic Regression.
  - SUPPORT VECTOR MACHINES:** Linear SVM, Quadratic SVM, Cubic SVM, Fine Gaussian ..., Medium Gaussian ..., Coarse Gaussian ...
  - NEAREST NEIGHBOR CLASSIFIERS:** Fine KNN, Medium KNN, Coarse KNN, Cosine KNN, Cubic KNN, Weighted KNN.
  - ENSEMBLE CLASSIFIERS:** Boosted Trees, Bagged Trees, Subspace Discriminant, Subspace KNN, RUSBoost..., All Ensembles.
- Current model:** Model number 5, Status: Trained, Accuracy: 76.6%, Prediction speed: ~1300 obs/sec, Training Time: 382.26 secs.
- Classifier:** Preset: Fine Gaussian SVM.
- Plot:** A vertical bar chart showing the distribution of response classes. The top bar is red (25%), the middle bar is green (75%), and the bottom bar is red (25%).
- Plot Options:**
  - Number of observations
  - True Positive Rates
  - False Negative Rates
  - Positive Predictive Values
  - False Discovery Rates
- What is the confusion matrix?:** A text area for help.
- Bottom Bar:** Response Classes: 2, Size of Dataset: [blank], Validation: [blank].

# Classification model generation

## Basic Workflow



# Classification model generation

## Evaluate model

predictedState = PredictionModel(**Parameter**)

PredictionModel

Validation Data, e.g. 30% of historic data

1 TIMESTAMP	2 PARAMETER										3 STATE
'2015-07-14 00:49:12.0'	160	160	160	160	1000	7	1000	9	33	32	1
'2015-07-14 00:50:12.0'	160	160	160	160	1000	8	1000	10	33	32	1
'2015-07-14 00:51:13.0'	160	160	160	160	1000	8	1000	10	33	32	1
'2015-07-14 00:52:12.0'	160	160	160	160	1000	8	1000	10	33	32	1
'2015-07-14 00:53:12.0'	160	160	160	160	1000	8	1000	11	33	32	2
'2015-07-14 00:54:12.0'	160	160	160	160	1000	8	1000	12	33	32	2
'2015-07-14 00:55:12.0'	160	160	160	160	1000	8	1000	10	33	32	2

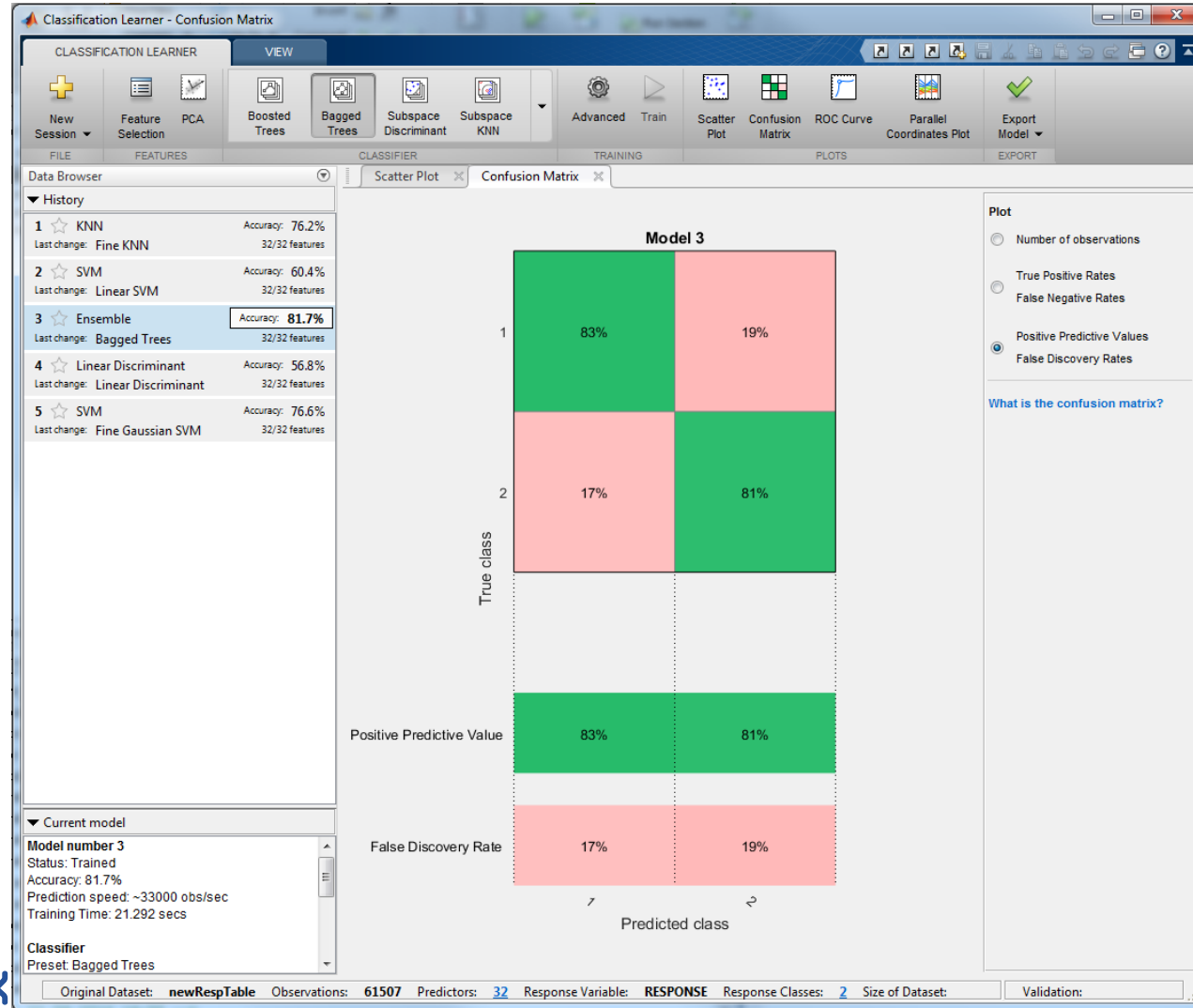
- ✓
- ✓
- ✓
- ✓
- ✓
- ✓
- ✓
- ✗

predictedState
1
1
1
1
2
2
1

Misclassification rate 1 of 7: 14.28 %  
 Accuracy: 85.72 %

# Classification model generation

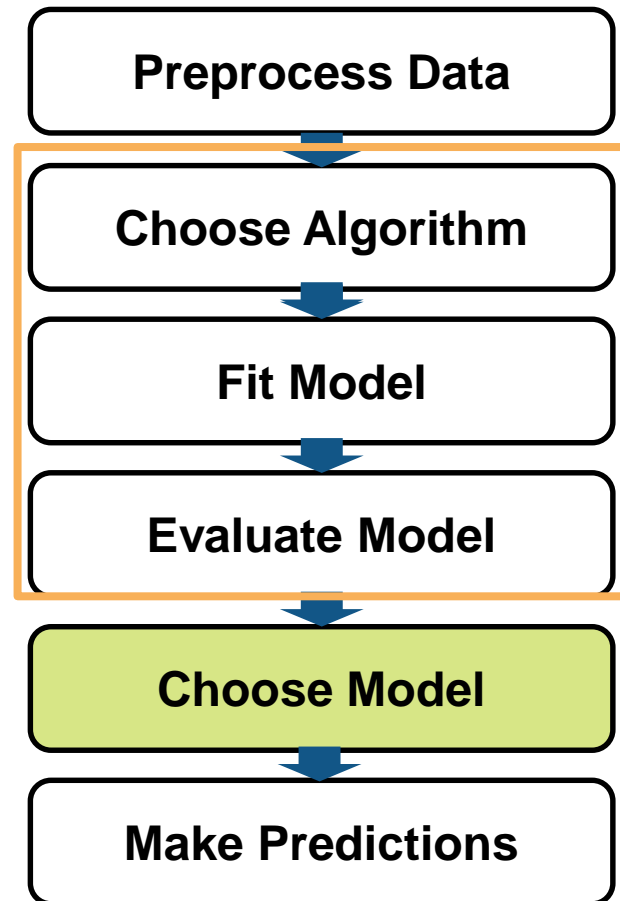
## Evaluate model - using Classification Learner App





# Classification model generation

## Basic Workflow

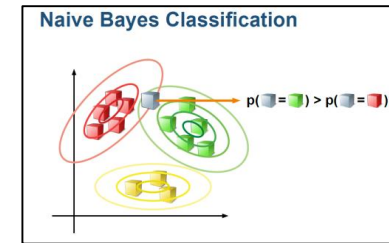
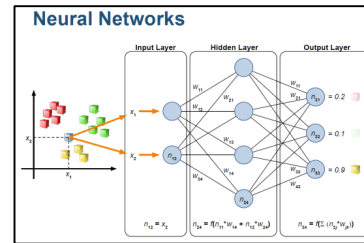
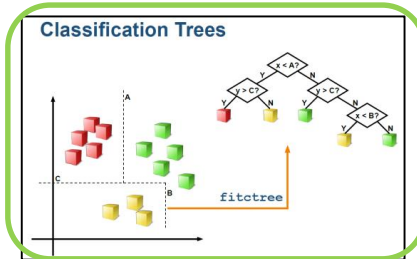
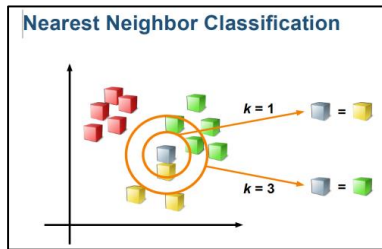


For each  
classification  
method

# Classification model generation

## Choose model

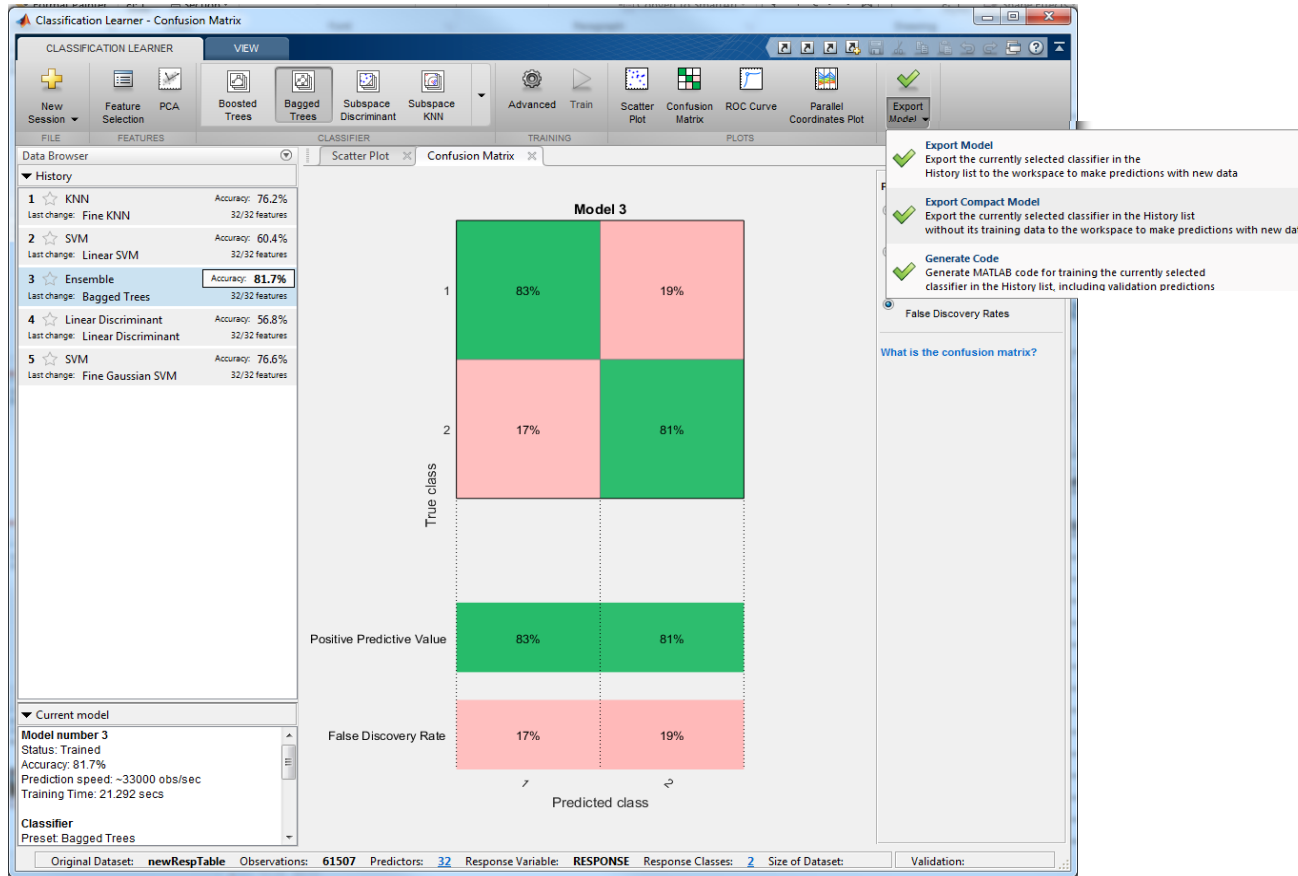
Choose Model with best misclassification rate



	NearestNeighbour	TreeBagger	NeuralNetwork	NaiveBayes
	<i>Misclassification % (Mean)</i>	<i>Misclassification % (Mean)</i>	<i>Misclassification % (Mean)</i>	<i>Misclassification % (Mean)</i>
M151	24%	2%	8%	10%
M152	44%	5%	23%	13%
M153	23%	2%	13%	13%
M156	12%	2%	3%	9%
M157	11%	1%	10%	8%
M158	29%	2%	14%	17%
M159	21%	0%	3%	2%
M181	1%	0%	1%	2%

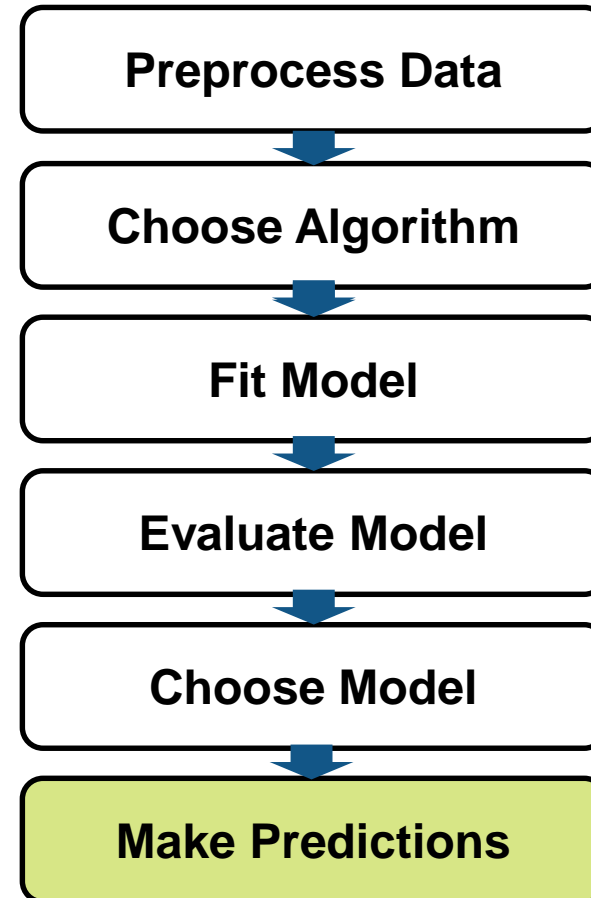
# Classification model generation

## Choose model



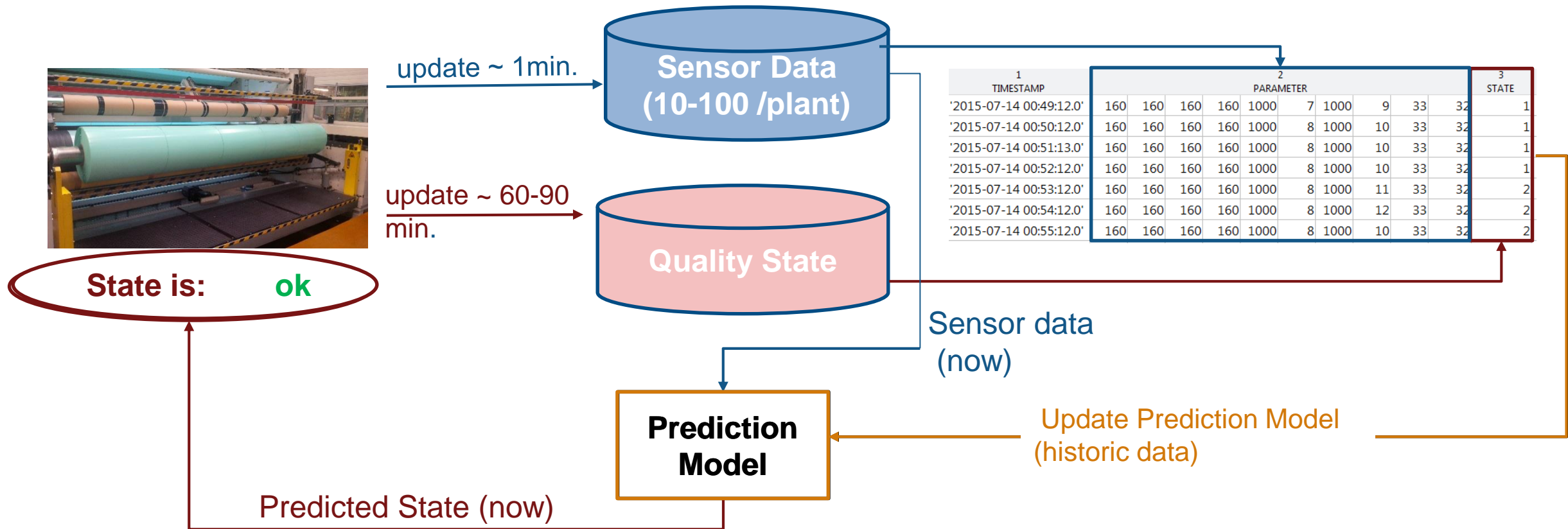
# Classification model generation

## Basic Workflow



# Predictive monitoring at MONDI Gronau - Use the predictive model

Predict current machine states during operation.



# Process monitoring at MONDI Gronau – Domain knowledge and tools

## Tools:

- MATLAB
- Database Toolbox
- Statistics and Machine Learning Toolbox
- Neural Network Toolbox
- MATLAB Compiler