Condition Monitoring of Structures, Machines and Processes

Tutorial 08

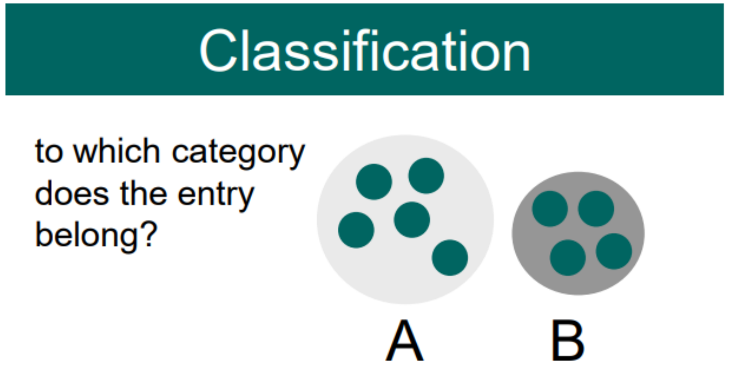
8

MATLAB:

CLASSIFICATION

Classification

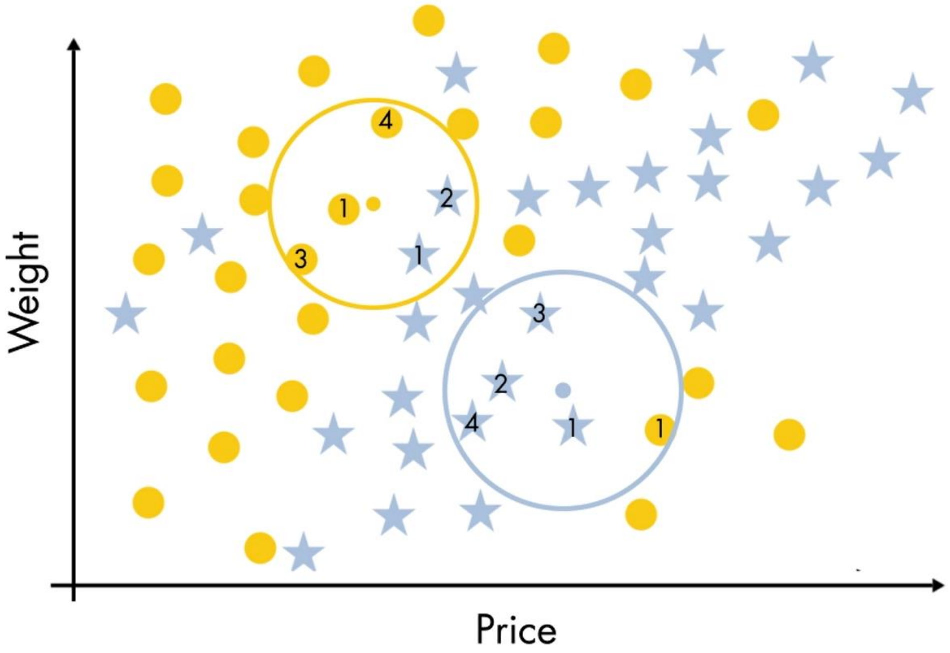
You have learned how to organise your data to set it up for features analysis and a first, simple ML-Model.
In this tutorial you will learn how to evaluate and further optimise the model, to obtain accurate predictions.



Classification

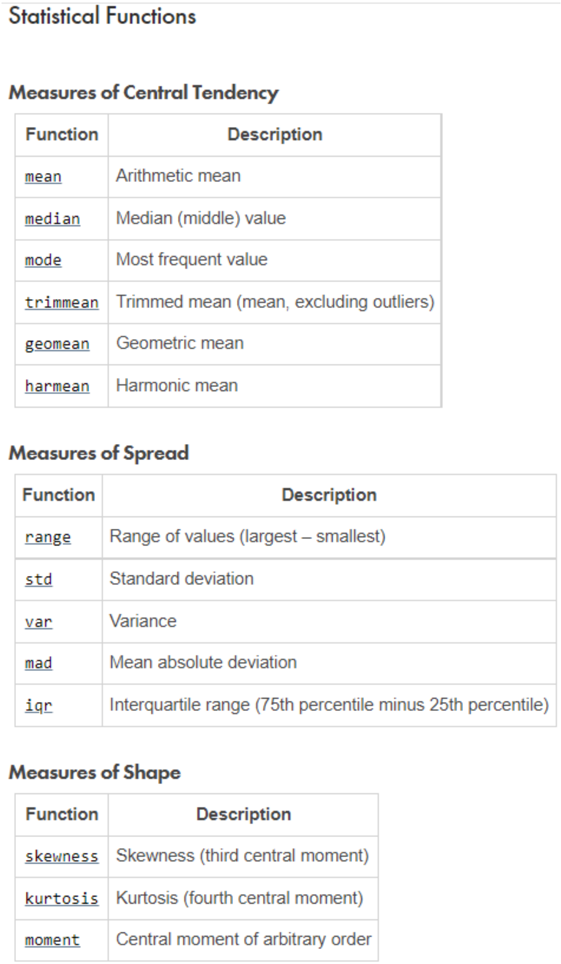
to which category does the entry belong?

Example: k-neairest neighbours, k = 4



Classification

Overview for Statistical functions, predefined in Matlab:



Statistical Functions

Measures of Central Tendency

Function	Description
mean	Arithmetic mean
median	Median (middle) value
mode	Most frequent value
trimmean	Trimmed mean (mean, excluding outliers)
geomean	Geometric mean
harmean	Harmonic mean

Measures of Spread

Function	Description
range	Range of values (largest – smallest)
std	Standard deviation
var	Variance
mad	Mean absolute deviation
iqr	Interquartile range (75th percentile minus 25th percentile)

Measures of Shape

Function	Description
skewness	Skewness (third central moment)
kurtosis	Kurtosis (fourth central moment)
moment	Central moment of arbitrary order

Exercise 16: Classification

- Evaluating the model and investigating features:
  - Based on the knn-model created in the last tutorial, try to play around with the value k. How does this value influences the accuracy of the predictions and the sensitivity to outliers?
  - For further evaluation of the predictions values, consider using the command `confusionchart()`, creating a confusion matrix which allows for a better evaluation between the different labels. Interpret the diagram.
    - Which class is the hardest to identify?
    - For further investigation, extract the observations causing false predictions of this class and store it in a Matrix MissClass.

Exercise 16: Classification

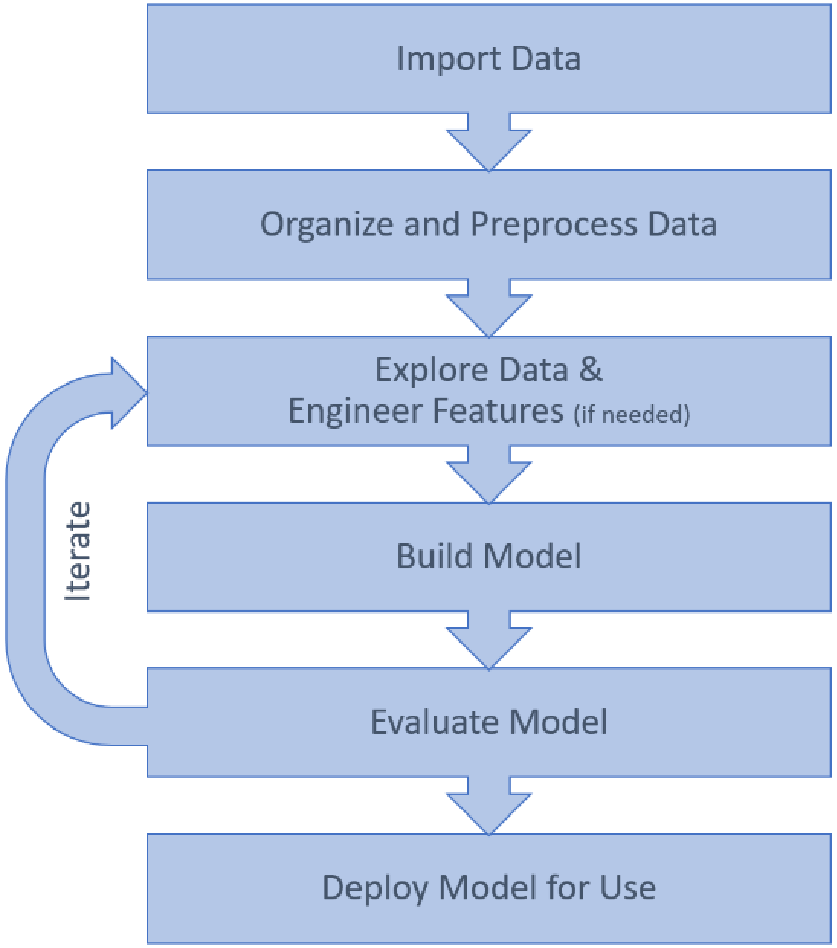
- Use the command `parallelcoords()` to obtain a overview of the value of the features. To compare the feature values of different classes, use the "Group" option as in `parallelcoords(data,"Group",classes).`

  Note : When plotting multiple observations by groups, it can be helpful to view the median and a range for each group, rather than every individual observation. You can use the "Quantile" option to do this.

  - For a better overview in case of huge differences in feature value, you can use the `'Standardize',‘on‘` option.
  - To get a sense of what might cause missclassification, plot a random Features from the MissClass Matrix into the parallel coordinate plot by using the `plot()` and `hold on` command

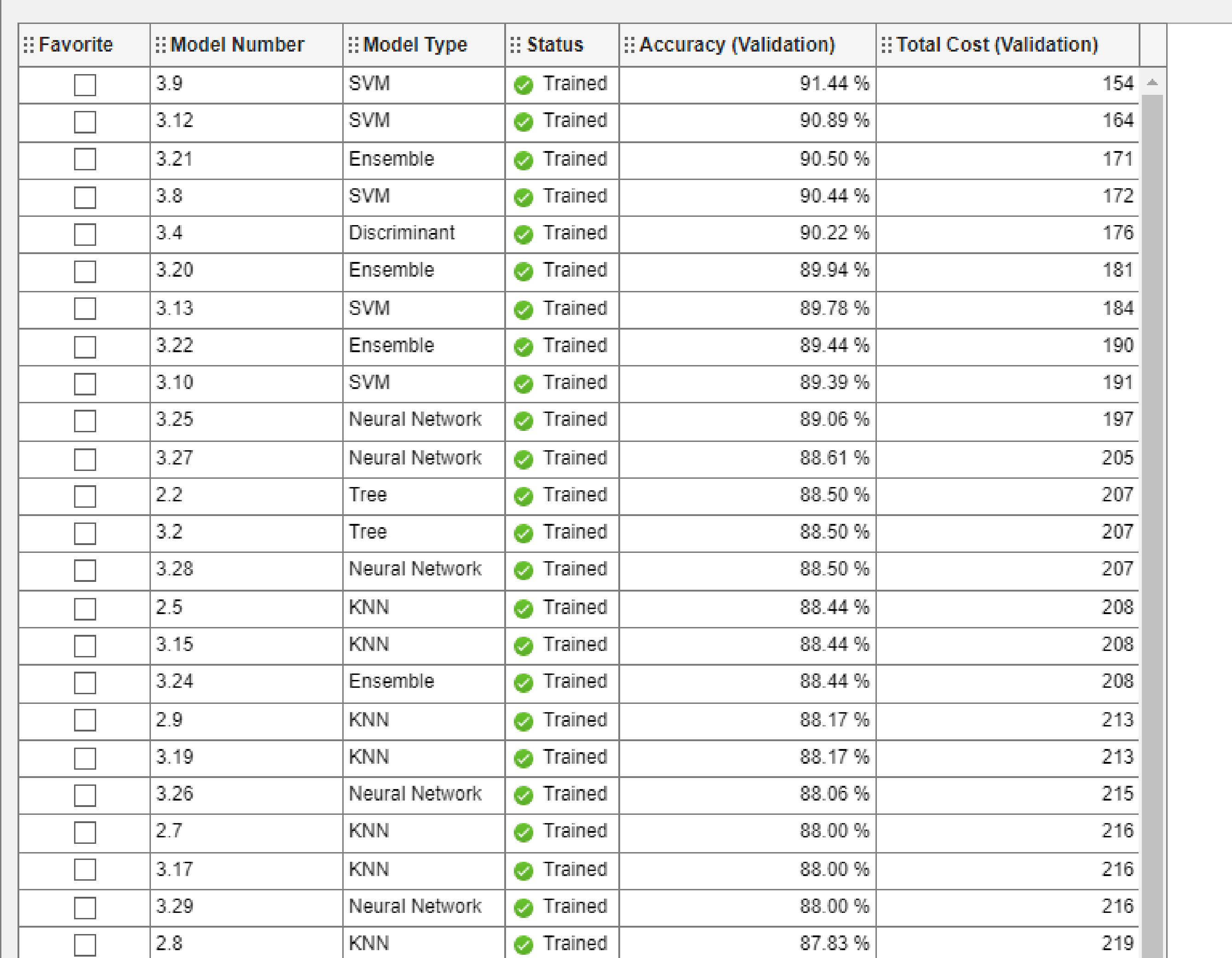
Exercise 16: Classification

- Investigating the Features might help to identify critical feautures, which re-engineering might provide a solid foundation for the ML-Model. The worklflow is summarized in the figure on the right hand side:



Exercise 16: Classification

- Training a Model
- Use the command classificationLearner to open the Classification Learner app.
  - Note: You could use the complete data as trainings data, since the classification learner automatically uses cross validation to validate the data. That is, you could rearrange the trainingsdata in a 1800x24 matrix,
- Try a few of the standard models with default options. See if you can achieve at least 90% accuracy.
- If you are unsure which model to employ, you could try several models simultaneously.
- Note: the app also provides diagrams as the confusions matrix and parallel coordinates plot.



Favorite	Model Number	Model Type	Status	Accuracy (Validation)	Total Cost (Validation)
□	3.9	SVM	✓ Trained	91.44 %	154
□	3.12	SVM	✓ Trained	90.89 %	164
□	3.21	Ensemble	✓ Trained	90.50 %	171
□	3.8	SVM	✓ Trained	90.44 %	172
□	3.4	Discriminant	✓ Trained	90.22 %	176
□	3.20	Ensemble	✓ Trained	89.94 %	181
□	3.13	SVM	✓ Trained	89.78 %	184
□	3.22	Ensemble	✓ Trained	89.44 %	190
□	3.10	SVM	✓ Trained	89.39 %	191
□	3.25	Neural Network	✓ Trained	89.06 %	197
□	3.27	Neural Network	✓ Trained	88.61 %	205
□	2.2	Tree	✓ Trained	88.50 %	207
□	3.2	Tree	✓ Trained	88.50 %	207
□	3.28	Neural Network	✓ Trained	88.50 %	207
□	2.5	KNN	✓ Trained	88.44 %	208
□	3.15	KNN	✓ Trained	88.44 %	208
□	3.24	Ensemble	✓ Trained	88.44 %	208
□	2.9	KNN	✓ Trained	88.17 %	213
□	3.19	KNN	✓ Trained	88.17 %	213
□	3.26	Neural Network	✓ Trained	88.06 %	215
□	2.7	KNN	✓ Trained	88.00 %	216
□	3.17	KNN	✓ Trained	88.00 %	216
□	3.29	Neural Network	✓ Trained	88.00 %	216
□	2.8	KNN	✓ Trained	87.83 %	219