# Artificial Intelligence

## Algorithms and Applications with Python

## Lectorial 04

Dr. Dominik Jung

*dominik.jung42@gmail.com*

</> course material: ↗ [https://github.com/dominikjung42](https://github.com/dominikjung42)

Image source: ↗ Pixabay (2019) / ↗ CC0

# Agenda

4.1 Basics and Repetition

4.2 Predictive Maintenance for Cars

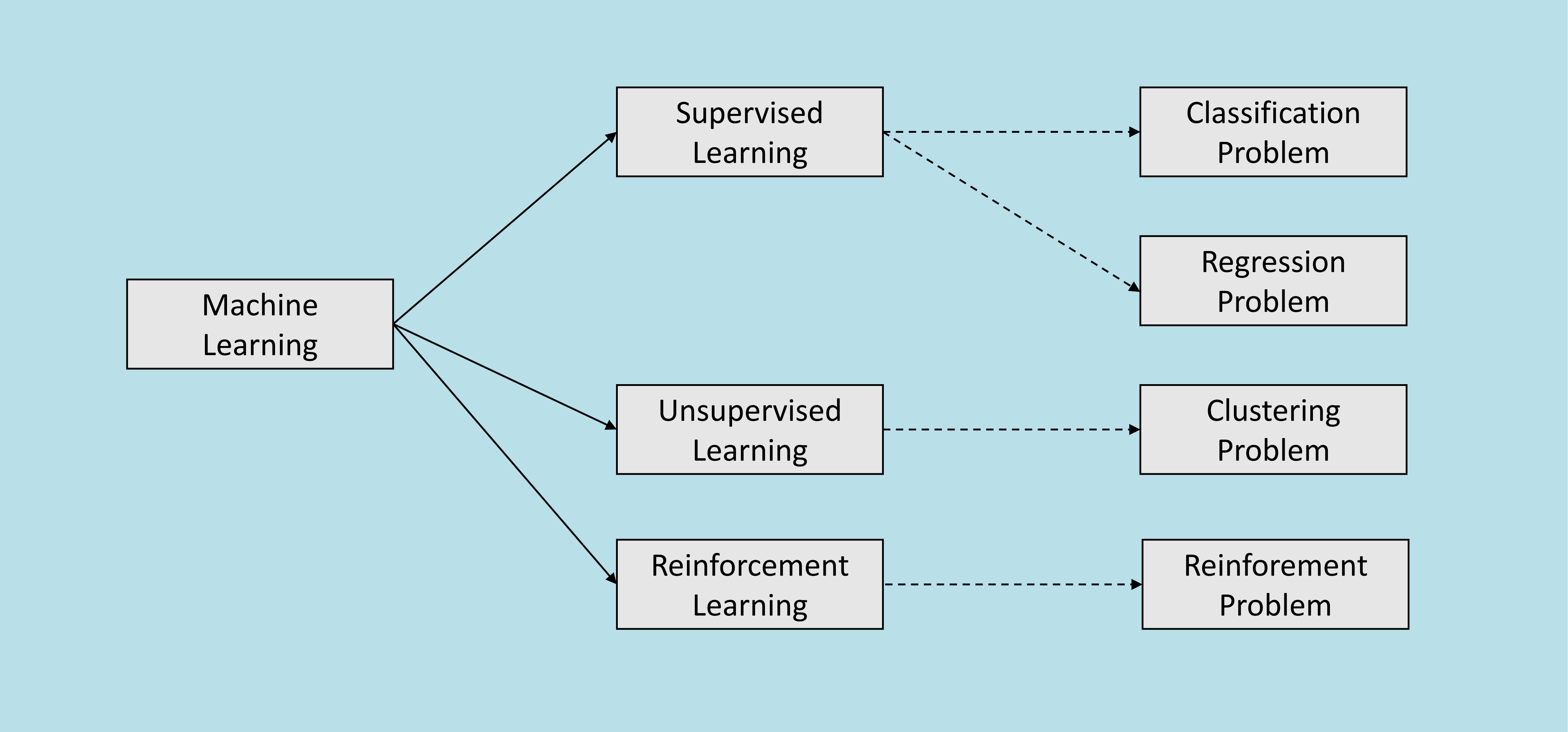
# Note

- This is a lectorial: I will explain/repeat the most important concepts and then you try to solve the programming task by your own

- You are explicitly encouraged to solve this task in groups. And I will help you and give suggestions. However, there is no perfect solution, you will get a possible solution.

- If the task is too hard for you at the moment – relaxe ☺. Just look at the task again at a later point in the course.

# Problem Types in Machine Learning (High-Level)



# Supervised Algorithms

- Decision tree
- KNN
- Regression
- Support Vector Machine
- Ensemble-Techniques

# DecisionTreeClassifier in Python

DecisionTreeClassifier()

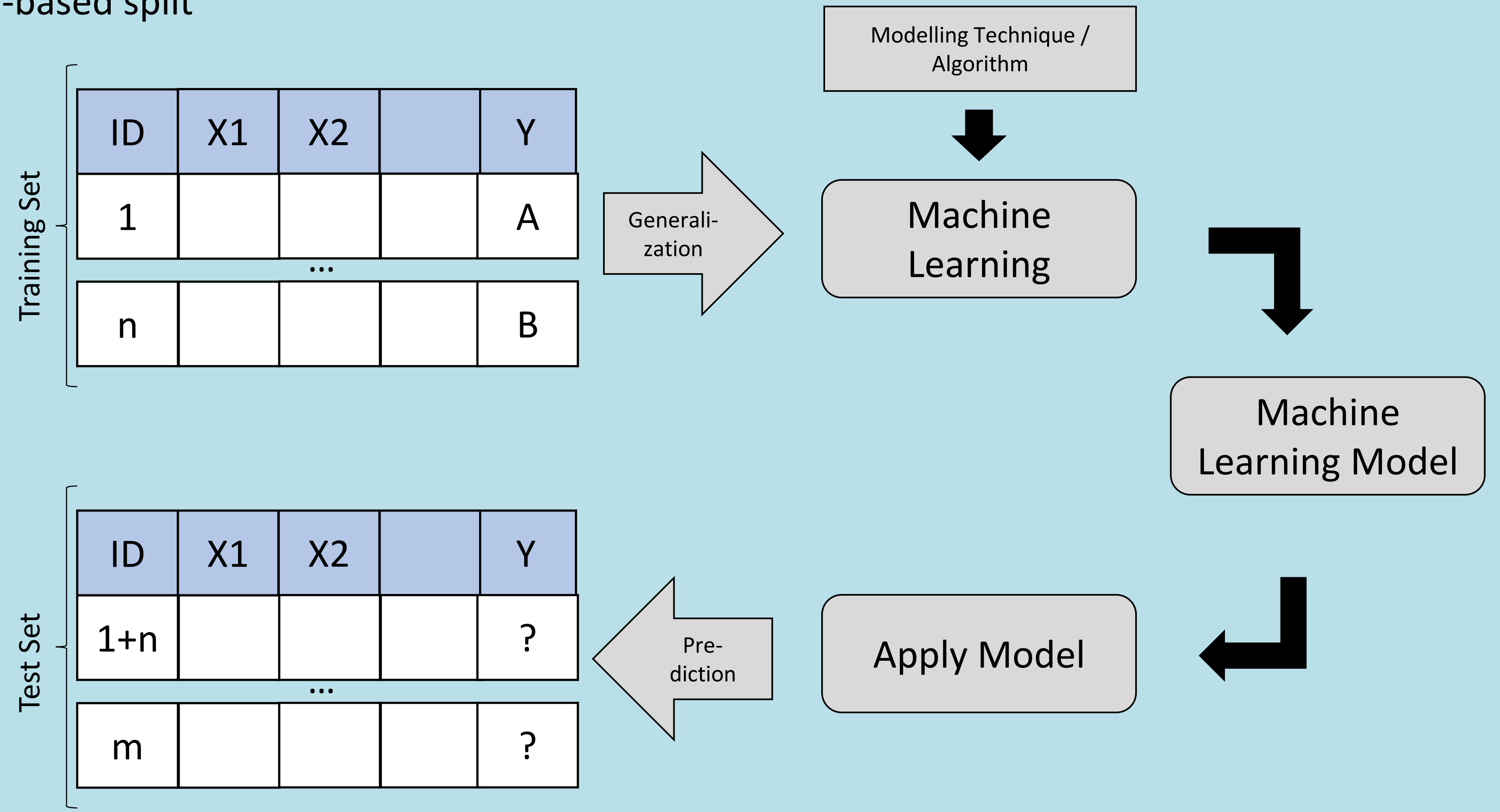
```
DecisionTreeClassifier(criterion, max_depth)
```

**Parameters**

| | |
|---|---|
| criterion | The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain. |
| max_depth | You can set an individual type of index for the row labels, the default index if no index is passed is np.arrange(n). |

# Split Data into Test and Train to Measure Performance

Simple ID-based split



Training Set

| ID | X1 | X2 | | Y |
|---|---|---|---|---|
| 1 | | | | A |
| | | … | | |
| n | | | | B |

Generali-zation

Modelling Technique / Algorithm

Machine Learning

Machine Learning Model

Apply Model

Pre-diction

Test Set

| ID | X1 | X2 | | Y |
|---|---|---|---|---|
| 1+n | | | | ? |
| | | … | | |
| m | | | | ? |

Adapted from Scitkit-learn (www.scikit-learn.org); Géron, A. (2017)

# Classroom Case

**Case**

Your next assignment is in the *After-Sales Department*. At the moment your colleagues have to handle many unexpected field problems and the managers asks you to develop an AI tool to predict car failures depending on the status and behavior of cars during the yearly checkups. This will help to plan capacities and avoid unhappy customers.

Please implement and evaluate different machine learning algorithms to detect possible candidates.

For that purpose

- Use the sample file `car_maintenance.xlsx`, which contains failures rates and some cars checkup-informations
- In a first step, explore the current dataset and do some data cleaning (e.g. remove possible outliers)
- After you have a first understanding of your dataset, start to implement a simple decision tree with `DecisionTreeClassifier()` and Holdout Evaluation. Use `astype("category")` and `cat.codes` from pandas to prepare the data for the decision tree. Evaluate and discuss the results with your colleagues. Imagine that you present your results the After-Sales Manager who no AI-knowledge at all.

| Variable | Values | Description |
|---|---|---|
| PART_1023,<br>PART_99<br>PART_02 | numeric | Number of car warnings in the log file that are related to the part |
| OIL | numeric | Oil volume at check‘s day |
| CHECK_STATUS | char | Result of the last checkup {unacc = unaccepted, acc = accepted, good = good, vgood = very good}.<br>Note: Cars can be decline also due to non-technical reasons. |

# Coding Session

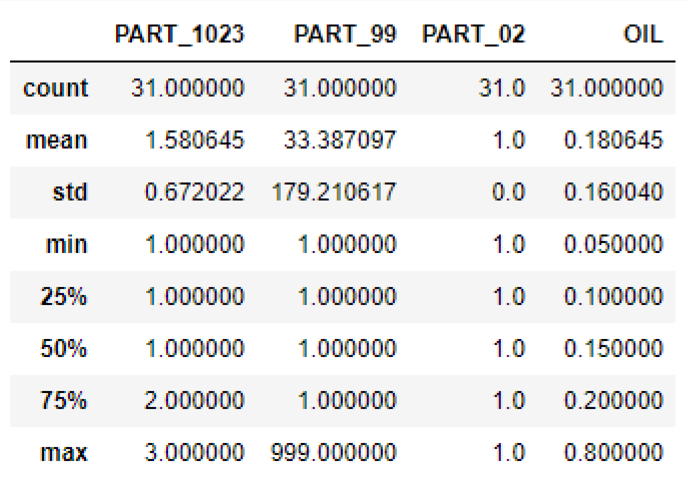
# Classroom Case

- In a first step we load the existing data into our IDE for further exploration

```
dataset = read_excel(open("car_maintenance.xlsx", "rb"))
```

- Our descriptive statistics show that PART_02 has no information, while PART_99 has potential outliers

```
dataset.describe()
```



| | PART_1023 | PART_99 | PART_02 | OIL |
|---|---|---|---|---|
| count | 31.000000 | 31.000000 | 31.0 | 31.000000 |
| mean | 1.580645 | 33.387097 | 1.0 | 0.180645 |
| std | 0.672022 | 179.210617 | 0.0 | 0.160040 |
| min | 1.000000 | 1.000000 | 1.0 | 0.050000 |
| 25% | 1.000000 | 1.000000 | 1.0 | 0.100000 |
| 50% | 1.000000 | 1.000000 | 1.0 | 0.150000 |
| 75% | 2.000000 | 1.000000 | 1.0 | 0.200000 |
| max | 3.000000 | 999.000000 | 1.0 | 0.800000 |

# Classroom Case

- For a first PoC, I decided to drop the useless column

```
dataset = dataset.drop(columns=["PART_02"])
```

- and to remove the outliers from `PART_99`

```
# thr = dataset["PART_99"].quantile(0.999)
thr = 3
dataset = dataset[dataset["PART_99"] < thr]
```

# Coding Session

# Classroom Case

- Before we can run our DecisionTreeClassifier we have to do some preprocesing

```
dataset["CHECK_STATUS"] = dataset["CHECK_STATUS"].astype("category")
dataset["CHECK_STATUS"] = dataset["CHECK_STATUS"].cat.codes
```

- Then we can generate our datasets to train our models

```
X = dataset.loc[:, "PART_1023":"CHECK_STATUS"]
Y = dataset.loc[:, "FOLLOW-UP"]

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
```

# Classroom Case

- Simple decision tree

```
clf = DecisionTreeClassifier()
clf = clf.fit(X_train, Y_train)
prediction = clf.predict(X_test)
```

```
>>> prediction
array(['no', 'no', 'yes', 'yes', 'no', 'no', 'no', 'no', 'no'], dtype=object)
```

- Or random forest

```
rnd_clf = RandomForestClassifier(n_estimators=1000, max_leaf_nodes=20)
rnd_clf = rnd_clf.fit(X_train, Y_train)
prediction = rnd_clf.predict(X_test)
```

# Coding Session

# Classroom Case

- Confusion matrix

```
confusion_matrix(Y_test, prediction)
```

- Accuracy scores

```
accuracy_score(Y_test, prediction)
```

# Classroom Case

**Case**

Please try to understand how the different phases of AI modelling influence the final result:

- Try to implement other classification models and compare them to the existing ones
- Generate some information about your model and prepare a short presentation for your manager, where you explain him if a “predictive maintenance agent” will be useful or not. Explain your decision

{ Just Keep Coding }