Artificial Intelligence and Data Analytics for Engineers (AIDAE)

Lecture 6
June, 12th

Andrés Posada

Anas Abdelrazeq

Marco Kemmerling

Vladimir Samsonov

Today’s Lecturer supported by Aymen Gannouni

Artificial Intelligence and Data Analytics for Engineers
Overview Lectures 1 – 4

1 **Introduction to Data Analytics and Artificial Intelligence in Engineering**: Organizational matters (e.g. exam, exercises, dates). Goals, Challenges, Obstacles, and Processes.

2 **Introduction into the primary programming language of the lecture, Python**: Syntax, libraries, IDEs etc. Why is Python the *lingua franca* of the Data Scientist?

3 **Data Preparation**: Cleansing and Transformation. How do real world data sets look like and why is cleaning and transformation an integral part of a Data Scientist's workflow?

4 **Data Integration**: Architectures, Challenges, and Approaches. How can you integrate various data sources into an overarching consolidating schema and why is this important?

Artificial Intelligence and Data Analytics for Engineers
Overview Lectures 5 – 8

5 **Data Representation**: Feature Extraction and Selection. How to pick relevant features for the task at hand. Manual vs automatic methods. What is the curse of dimensionality?

6 **Data-Driven Learning**: Supervised (Classification, Regression) methods and algorithms. What is an artificial neural net? What methods are there for evaluation of your model?

Today's Lecture

Supervised Learning

What is it?	Methods	Applying

Learning Objectives

Learning Objective
w.r.t. Knowledge/Understanding.

After successfully completing this lecture, the students will have achieved the following learning outcomes:

- Have an understanding of what supervised learning is.
- Know the different families of supervised learning algorithms.
- Learn how to train and evaluate supervised learning algorithms.

Motivation and Introduction

Supervised Learning in Different Industries

In the service industry, **sentiment analysis** is quite common, from both text and voice.

Some of the ways it can be used are:

- Automate systems to run sentiment analysis on all incoming customer support queries.
- Rapidly detect disgruntled customers and surface those tickets to the top.
- Route queries to specific team members best suited to respond.
- Use analytics to gain deep insight into what's happening across your customer support.

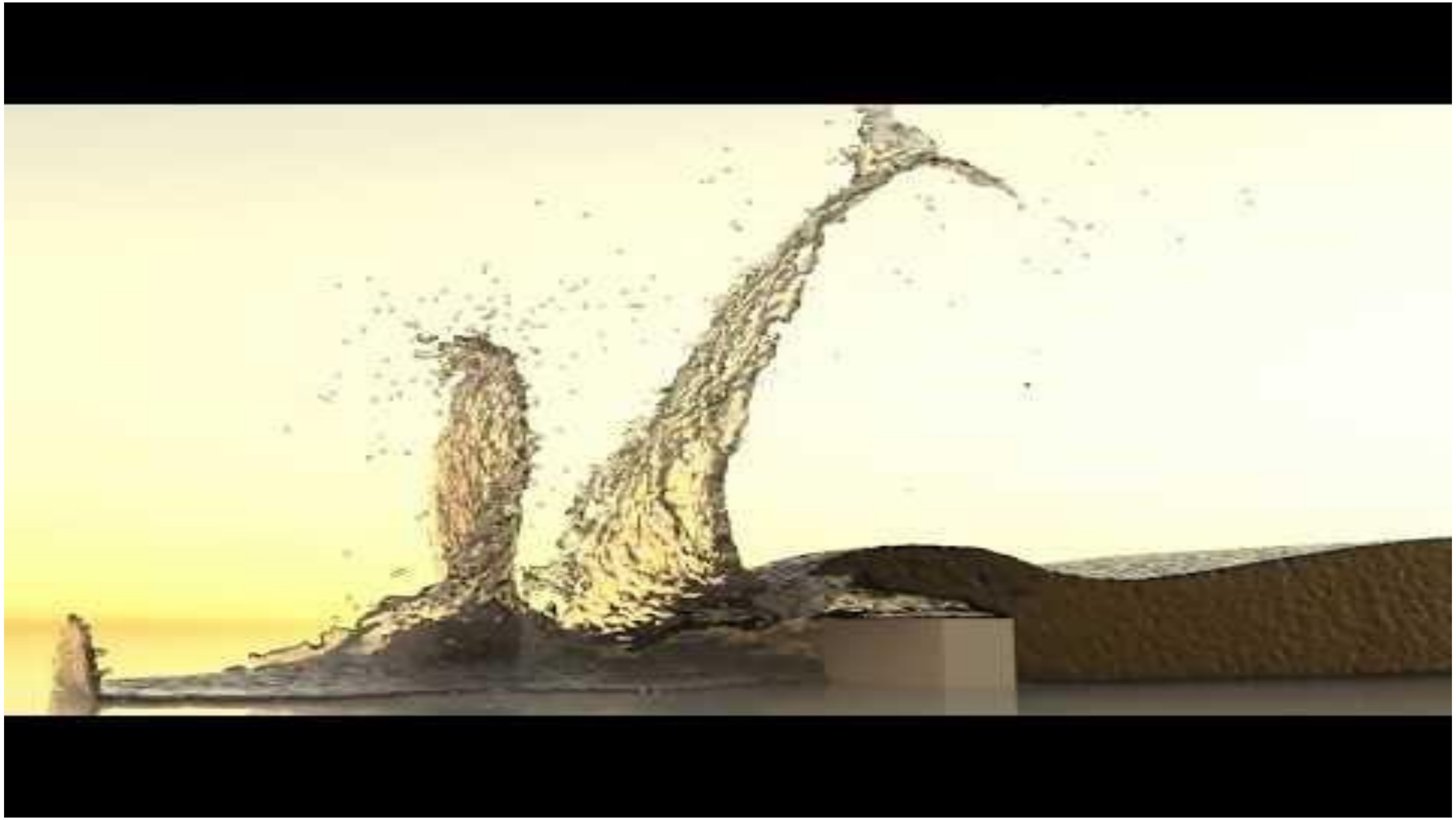
Supervised Learning in Different Industries

- In the protein production industry, supervised learning models are used to monitor and estimate the weight and health of animals.
- Some companies use **image processing**, **neural networks** and **regression algorithms** to calculate weight of living pigs using only images.

Source: Asimetrix. Inc

Supervised Learning in Different Industries

- "…model learns to generate small-scale splash detail for the fluid-implicit-particle method using training data acquired from physically parameterized, high resolution simulations."



[https://ge.in.tum.de/publications/2018-mlflip-um]

Introduction

Introduction

- Brief view into Artificial intelligence and Machine Learning

Artificial Intelligence:
Any technique which enables a computer to mimic human behaviour.

General AI (GAI)
Transfer knowledge across domains.

Narrow AI
Performs a single task extremely well

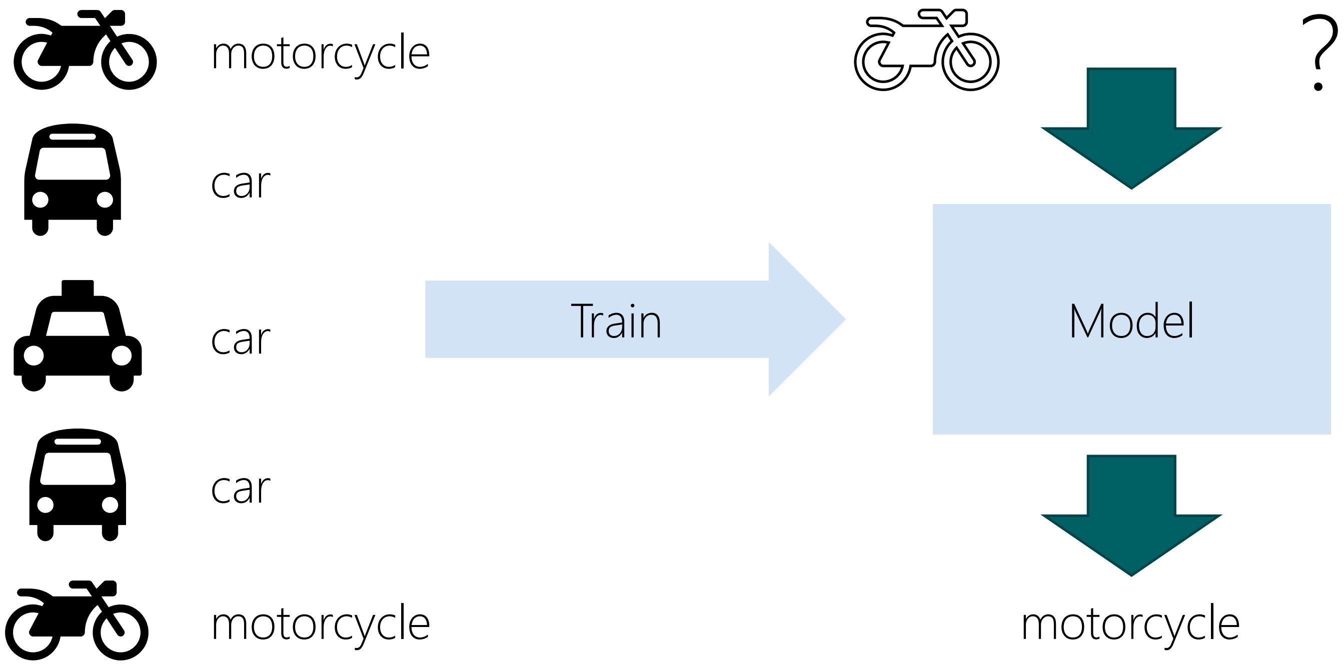
Machine Learning
Algorithms whose performance improve as they are exposed to more data.

- Supervised Learning (Regression, Classification, etc.)
- Unsupervised Learning (Clustering, Dimensionality, Reduction.Z, Rule discovery, etc.)
- Reinforced Learning

What is supervised learning?

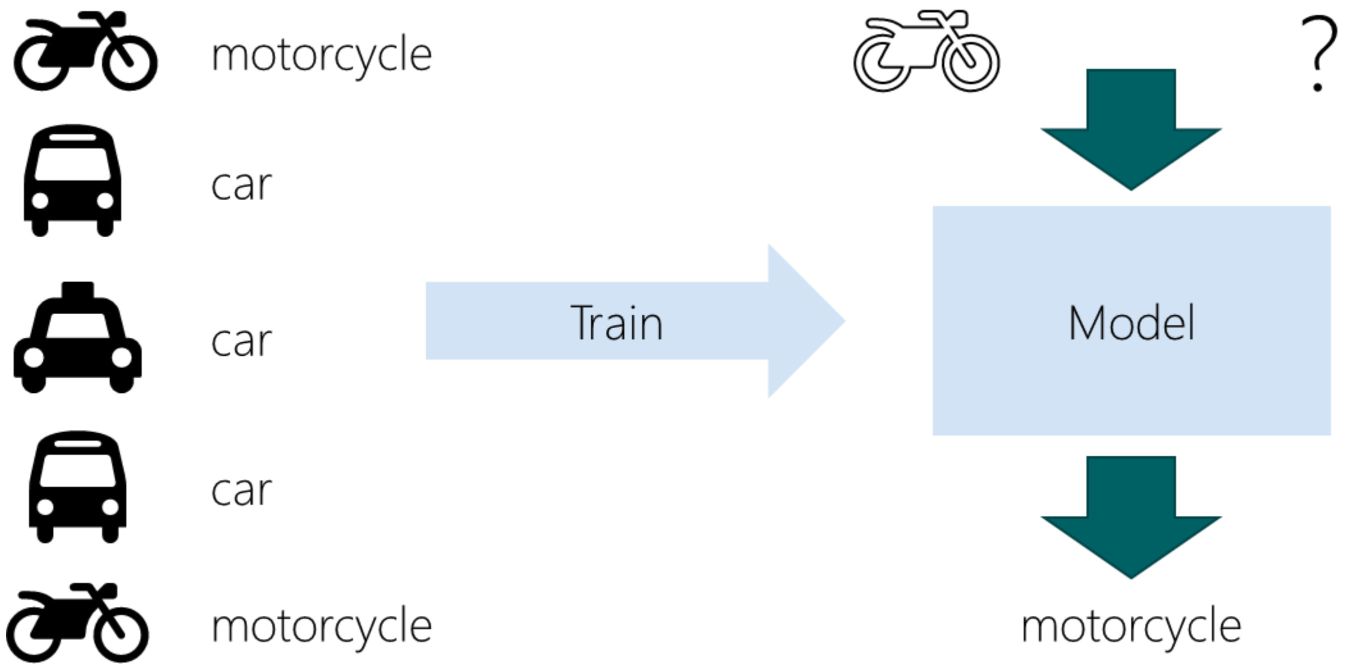
Working Definition

Supervised learning is a task that consists in training an algorithm over example labeled data (input-output pairs), to be able to map an input space into an output space.



What is supervised learning?

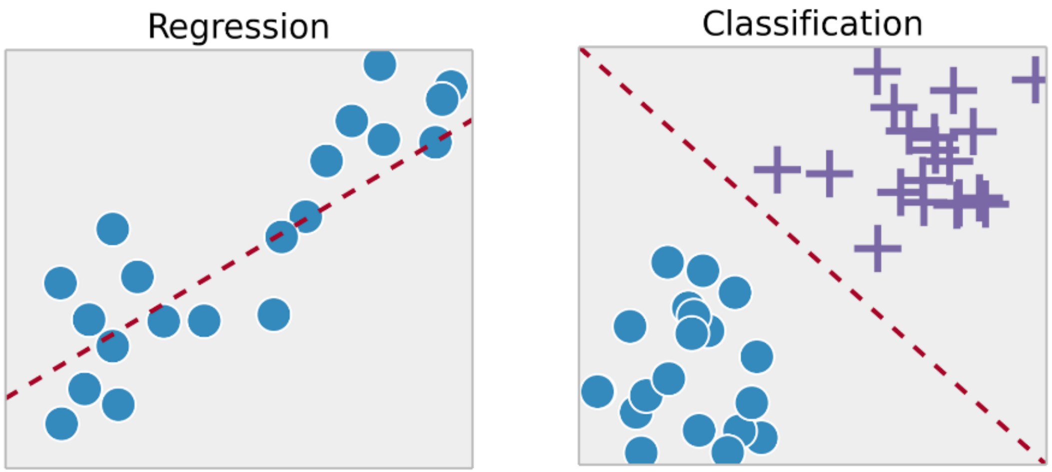
- Having:
  - The input variables x
  - And the output variables Y
  - As well as “n” training examples {(x1, Y1), (x1, Y1), …. (xn, Yn)}
  - Find an appropriate mapping function “f” where Y = f(x) , where x is the input space and Y is the output space.
  - Usually, the function is parametrized with a set of parameters θ.
  - Learning θ from data is what’s called “supervised learning” (in a process called “training”). The training process yields a “model”, i.e. the mapping function.



Regression & Classification

What is supervised learning?

- Having:
  - The input variables x
  - And the output variables Y
  - As well as "n" training examples {(x1, Y1), (x1, Y1), ... (xn, Yn)}
  - Find an appropriate mapping function "f" where Y = f(x) , where x is the input space and Y is the output space.
  - Usually, the function is parametrized with a set of parameters θ.
  - Learning θ from data is what's called "supervised learning" (in a process called "training"). The training process yields a "model", i.e. the mapping function.

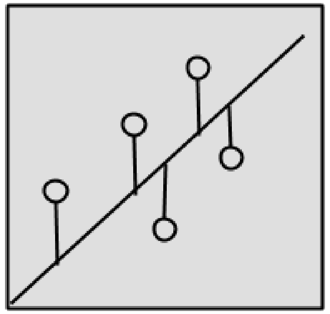


- If Y (output variable) is a real value, the problem is called a Regression.
- If Y (output variable) is a category, the problem is called a Classification.

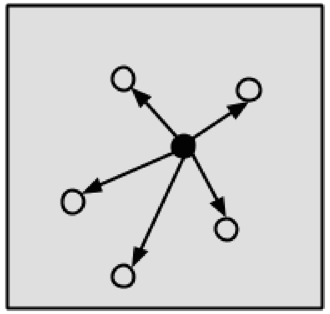
Regression & Classification

Families of algorithms

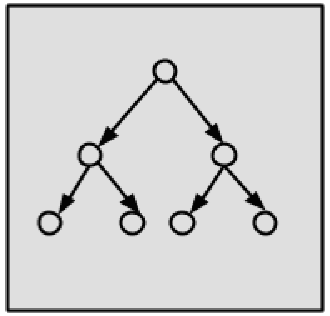
Can all of them be used for regression and classification?



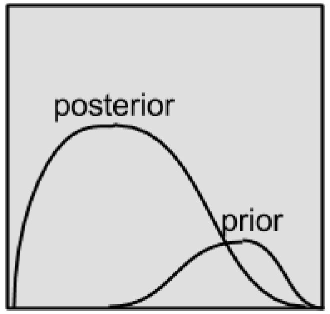
Regression Algorithms



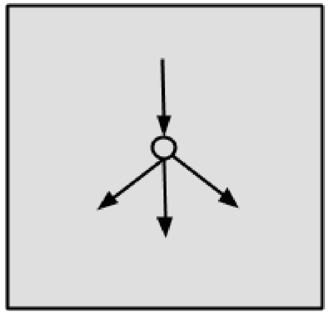
Instance-based Algorithms



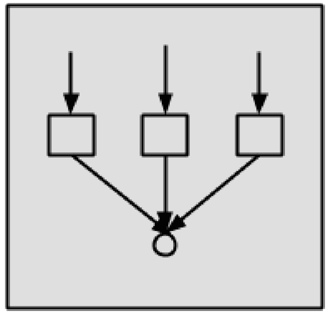
Decision Tree Algorithms



Bayesian Algorithms



Artificial Neural Network Algorithms

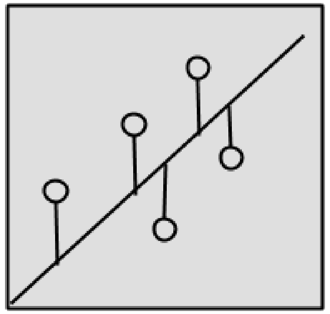


Ensemble Algorithms

Regression & Classification

Families of algorithms

Can all of them be used for regression and classification?



Regression Algorithms

- Algorithms based set of statistical processes for estimating the relationships among variables. Normally based on estimating parameters of fixed classical equations such as least squares.

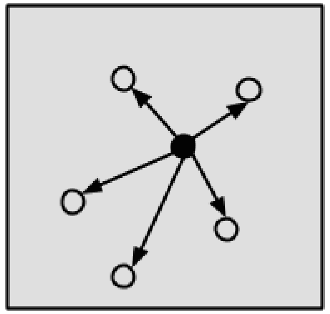
- Regression: Linear Regression
- Classification: Logistic Regression

```
from sklearn import linear_model
x = [[0, 0], [1, 1], [2, 2]]
y = [0, 1, 2]
model = linear_model.LinearRegression()
model.fit(x, y)
new_x = [[1.5, 1.5]]
predicted_y = model.predict(new_x)
```

Regression & Classification

Families of algorithms

Can all of them be used for regression and classification?



Instance-based Algorithms

- Methods that build up a database of example data and compare new data to the database.

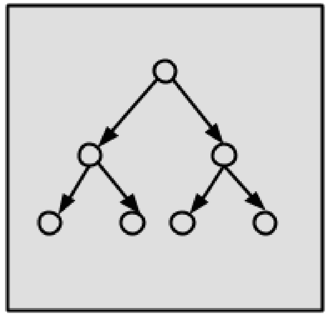
- Regression: K Neighbors Regressor, Support Vector Regressor
- Classification: K Nearest Neighbors, Support Vector Machine

```
from sklearn.neighbors import KNeighborsClassifier
x = [[0], [1], [2], [3]]
y = [0, 0, 1, 1]
model = KNeighborsClassifier(n_neighbors=2)
model.fit(x, y)
new_x = [[1.5]]
predicted_y = model.predict(new_x)
```

Regression & Classification

Families of algorithms

Can all of them be used for regression and classification?



Decision Tree Algorithms

- Methods that construct a model of decisions made based on actual values of attributes in the data.

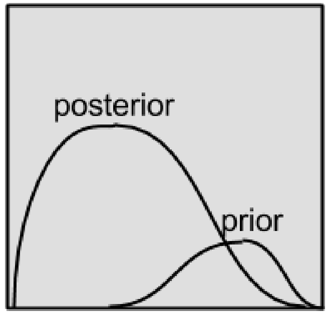
- Regression: Decision Tree Regressor
- Classification: Classification Tree

```
from sklearn import tree
x = [[0], [1], [2], [3]]
y = [0, 0, 1, 1]
model = tree.DecisionTreeClassifier()
model.fit(x, y)
new_x = [[1.5]]
predicted_y = model.predict(new_x)
```

Regression & Classification

Families of algorithms

Can all of them be used for regression and classification?



Bayesian Algorithms

- Methods that explicitly apply Bayes' Theorem. $P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$

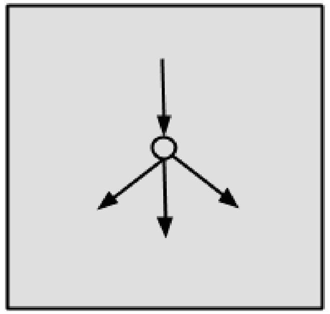
- Regression: Bayesian linear regression
- Classification: Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
x = [[0], [1], [2], [3]]
y = [0, 0, 1, 1]
model = GaussianNB()
model.fit(x, y)
new_x = [[1.5]]
predicted_y = model.predict_proba(new_x)
```

Regression & Classification

Families of algorithms

Can all of them be used for regression and classification?



Artificial Neural Network Algorithms

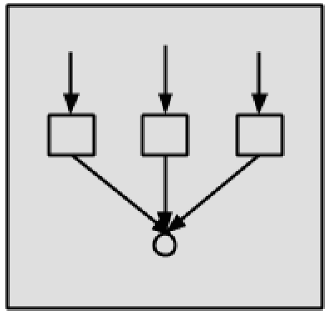
- Methods inspired on biological neurons structure and working mechanisms. Highly dependent on different architectures, learning rates, epochs and batches.
- Regression: Perceptron
- Classification: Convolutional Neural Network

```
from sklearn.neural_network import MLPRegressor
x = [[0, 0], [1, 1], [2, 2]]
y = [0, 1, 2]
model = MLPRegressor(hidden_layer_sizes=(2,1,), max_iter=1000)
model.fit(x, y)
new_x = [[1.5, 1.5]]
predicted_y = model.predict(new_x)
```

Regression & Classification

Families of algorithms

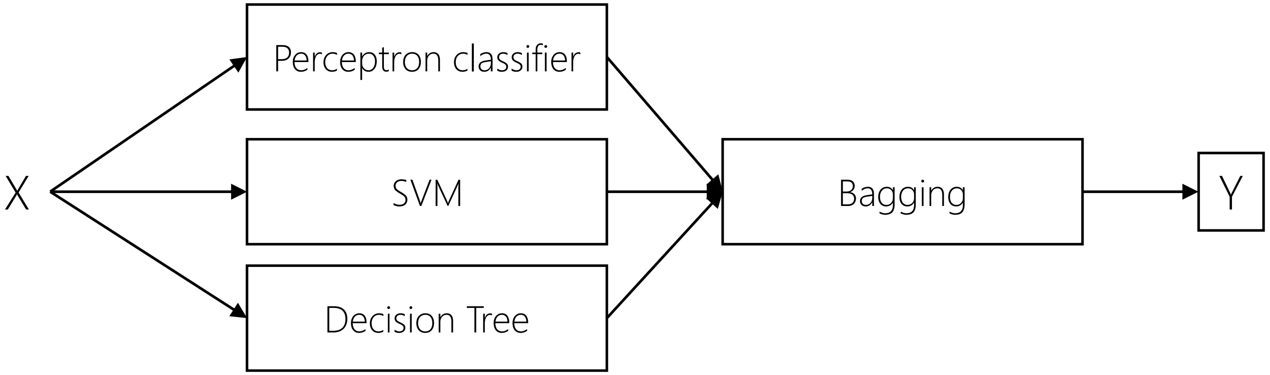
Can all of them be used for regression and classification?



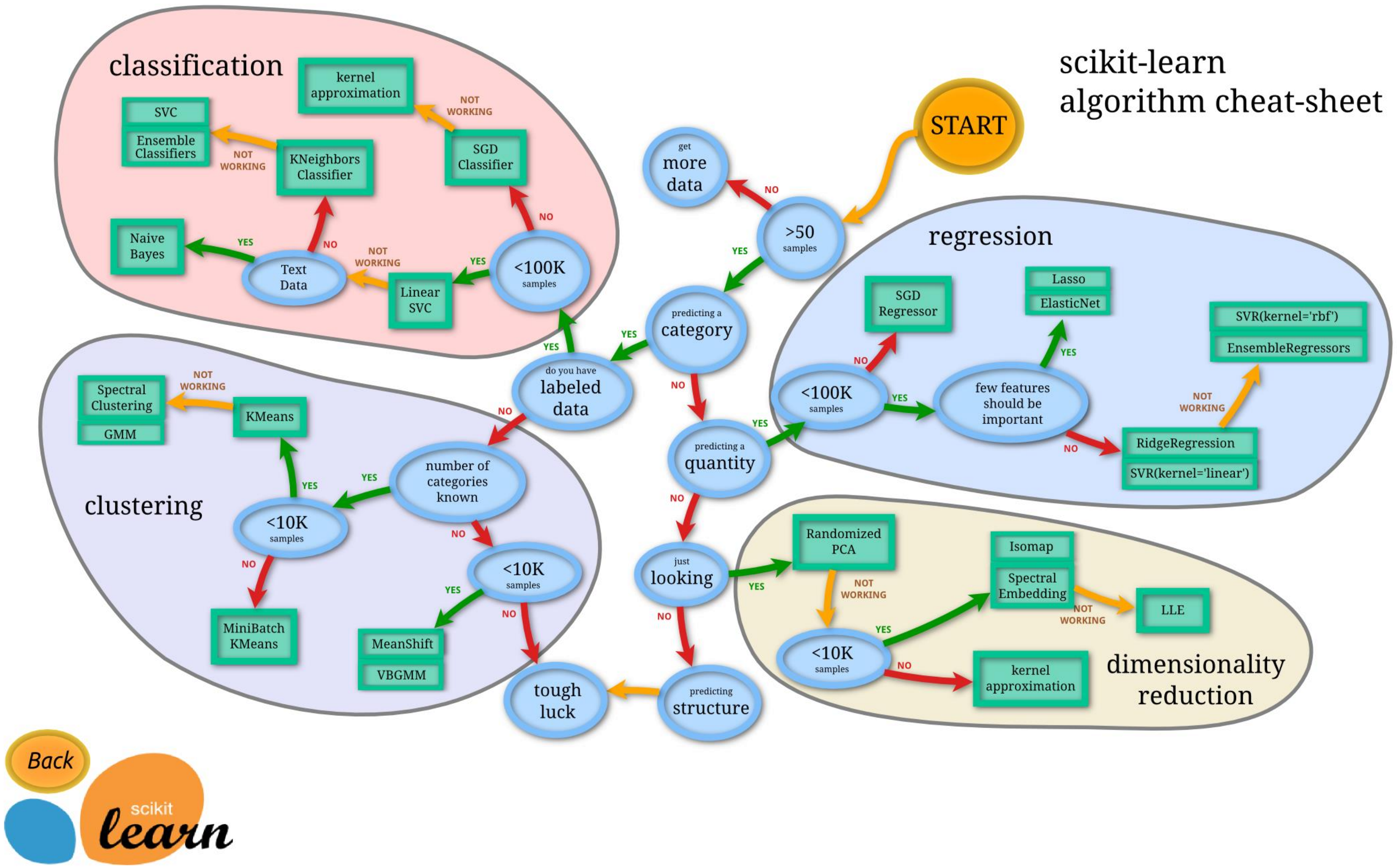
Ensemble Algorithms

- Methods that combine multiple weaker models independently trained to obtain a better result. Typically this algorithms build multiple black box models and aggregates the individual predictions to generate a consensus.

- Regression: Bagging
- Classification: Random Forest, XGBoost, Bagging



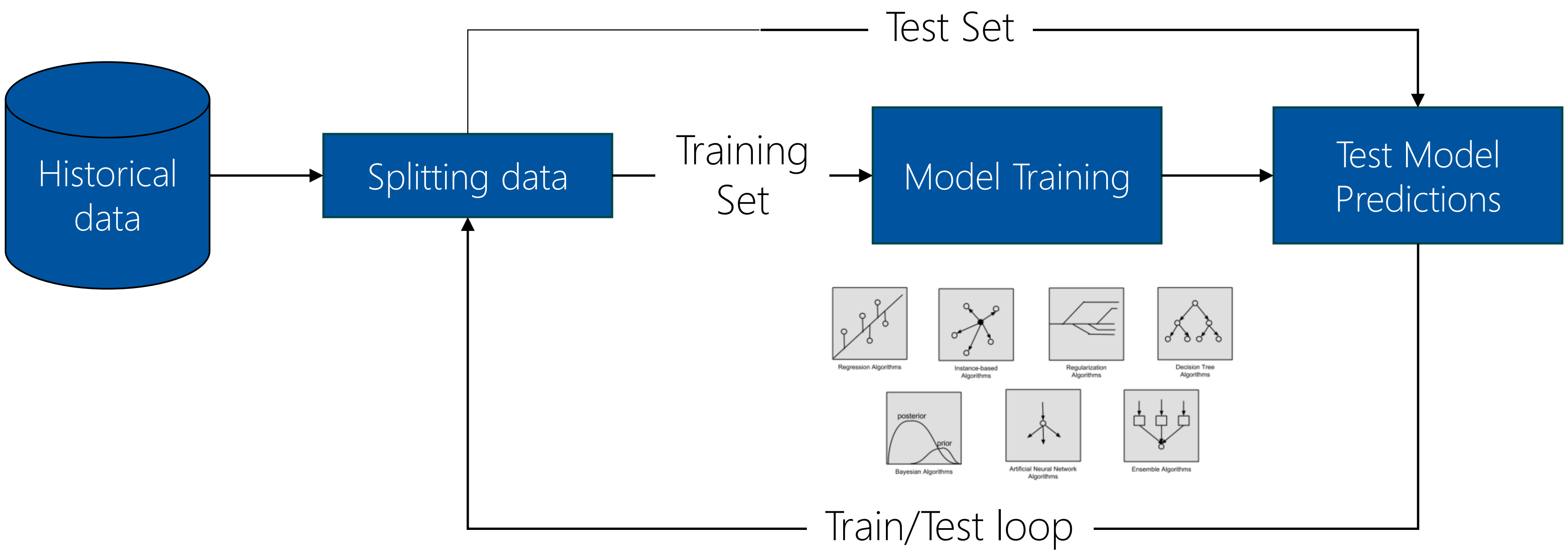
Regression & Classification



Training, Errors and Metrics

Training, errors and metrics

Model Building



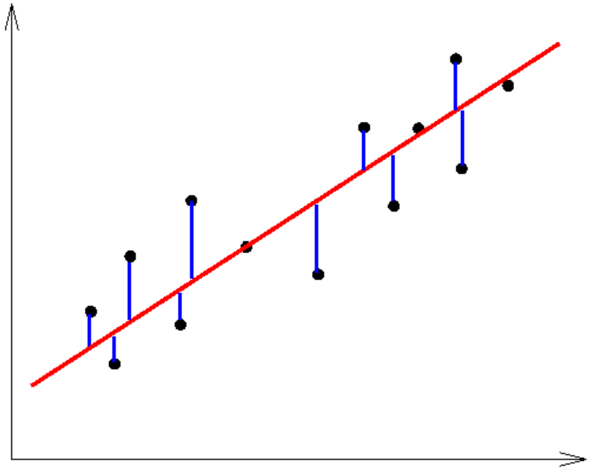
Training, errors and metrics

Regression metrics

Three of the most common metrics for evaluating predictions on regression:

- Mean absolute error

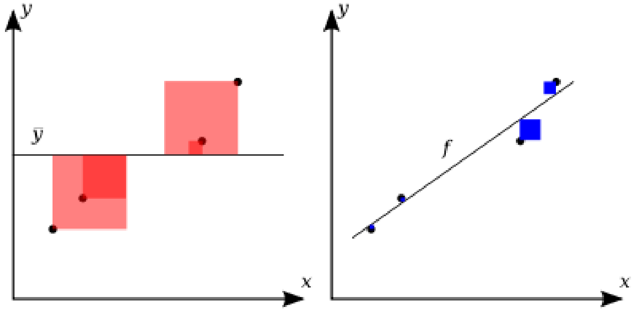
$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$



- Mean square error

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- R2 score: proportion of the variance in the dependent variable that is predictable from the independent variable(s).



$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

Training, errors and metrics

Classification metrics

Confusion matrix

- Multiple metrics can be derived from a confusion matrix, but the most used ones are the accuracy and the F1-score

Predicted class

		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

accuracy (ACC)

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{P + N} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

F1 score

is the harmonic mean of precision and sensitivity

$$F_1 = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

sensitivity, recall, hit rate, or true positive rate (TPR)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

specificity or true negative rate (TNR)

$$\mathrm{TNR} = \frac{\mathrm{TN}}{N} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

precision or positive predictive value (PPV)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

negative predictive value (NPV)

$$\mathrm{NPV} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FN}}$$

miss rate or false negative rate (FNR)

$$\mathrm{FNR} = \frac{\mathrm{FN}}{P} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} = 1 - \mathrm{TPR}$$

fall-out or false positive rate (FPR)

$$\mathrm{FPR} = \frac{\mathrm{FP}}{N} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} = 1 - \mathrm{TNR}$$

false discovery rate (FDR)

$$\mathrm{FDR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TP}} = 1 - \mathrm{PPV}$$

false omission rate (FOR)

$$\mathrm{FOR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TN}} = 1 - \mathrm{NPV}$$

Training, errors and metrics

Classification metrics
False Positive vs False Negative

False Positive

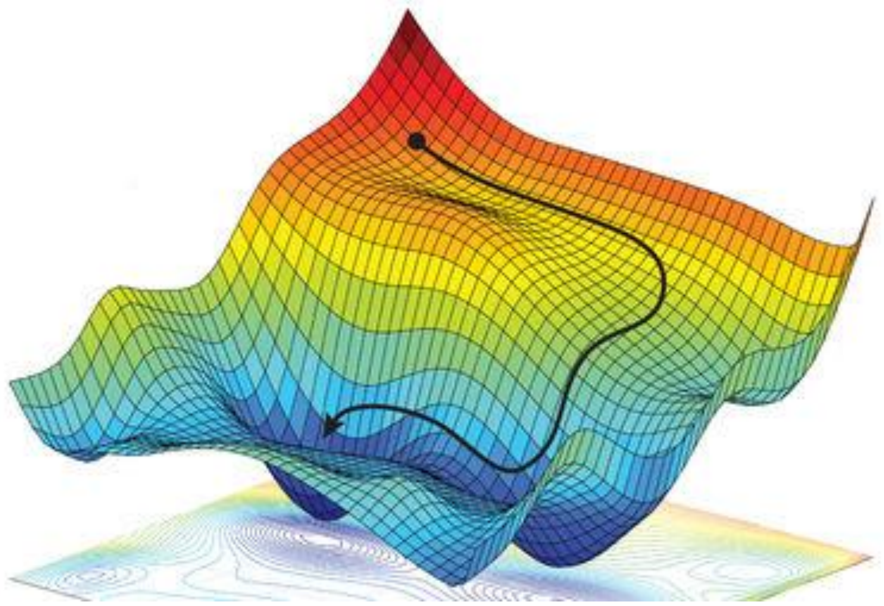
Type I error

False Negative

Type II error

Training, errors and metrics

- Gradient descent: Method for minimizing a real-valued differentiable function f
- Define a loss function L
- Concept:
  - Start with any (random) parameters, e.g. $\theta_0, \dots, \theta_n$
  - Change iterative $\beta_0, \dots, \beta_n$
  - $\theta_j = \theta - \alpha \frac{\partial}{\partial \theta_j} \mathrm{L}(f_\theta(x))$
  - Stop if procedure converges



https://www.sciencemag.org/news/2018/05/ai-researchers-allege-machine-learning-alchemy

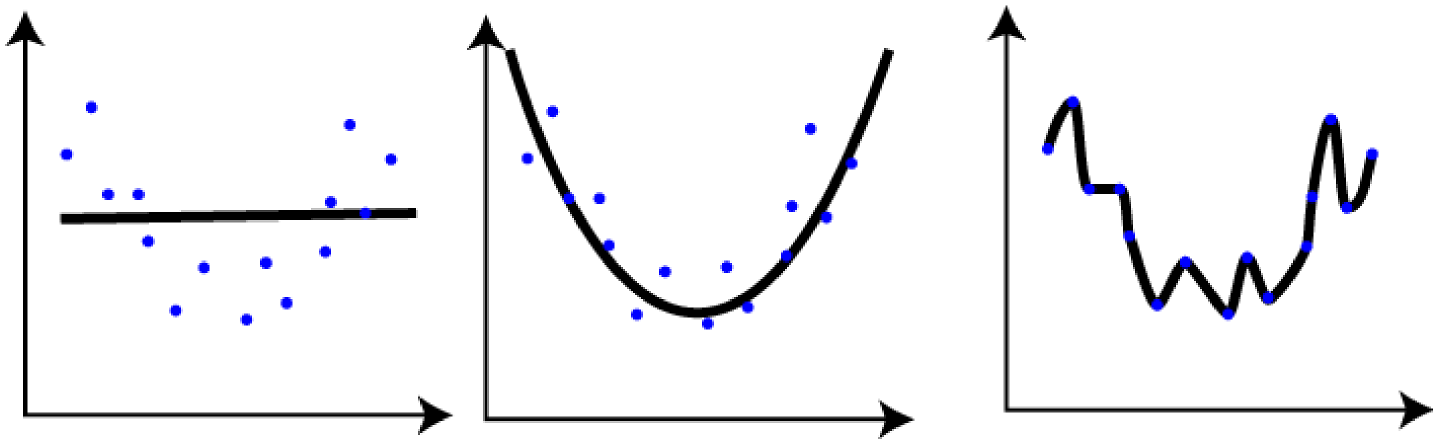
Implemented in sklearn, e.g. `sklearn.linear_model.SGDRegressor`

Training, errors and metrics

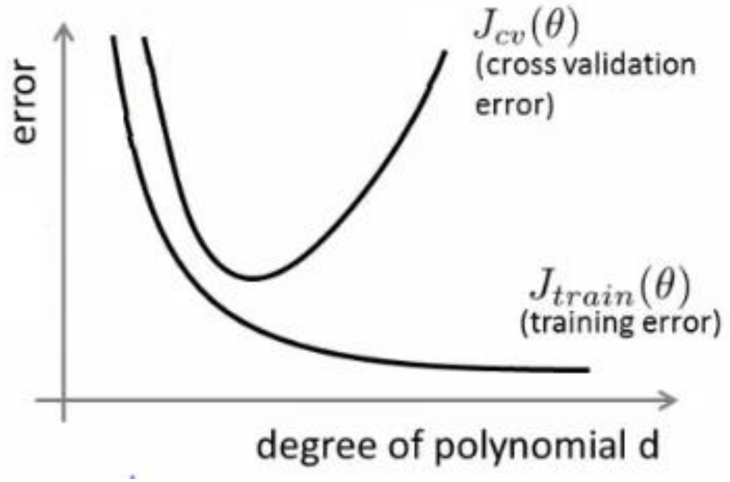
Overfitting

Overfitting refers to a model that “models” the training data too well.

- Error on the Training set diminishes.
- The model loses the ability to generalize.



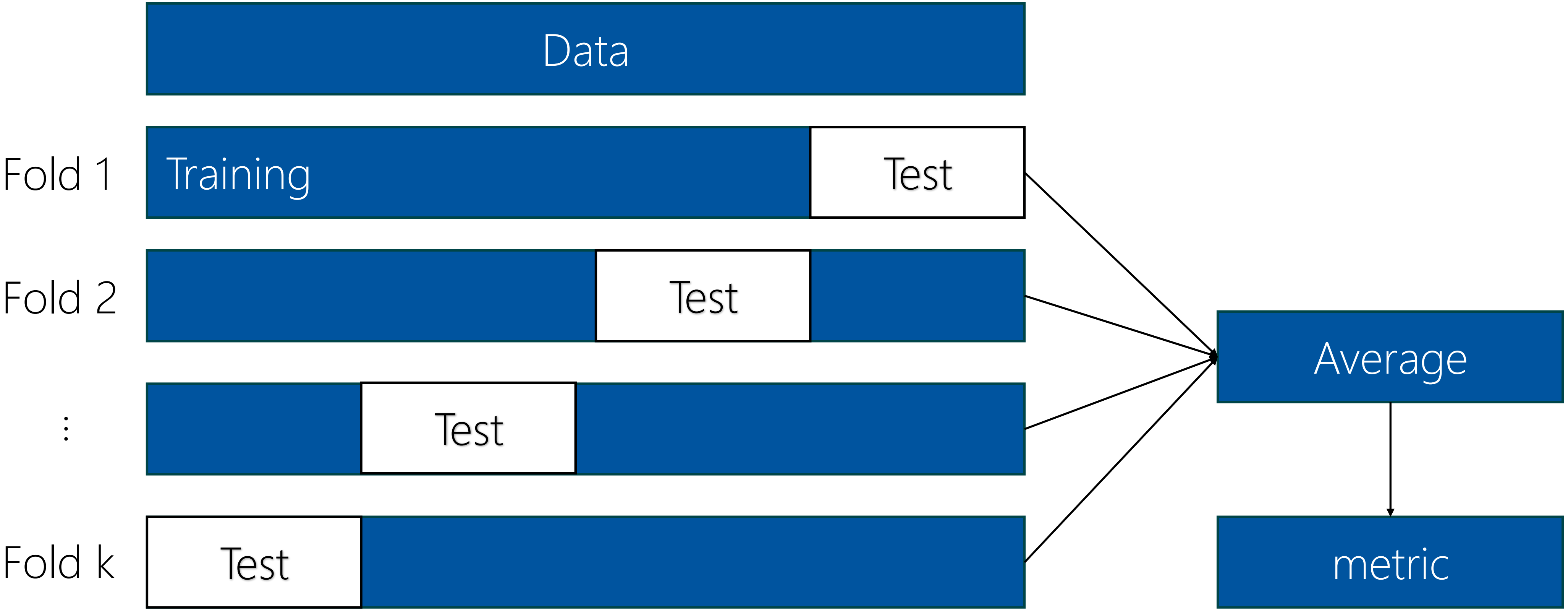
Under-fitting Optimum Overfitting



Training, errors and metrics

Cross-Validation (CV)

Technique to validate models/classifiers



Examples and Implementation

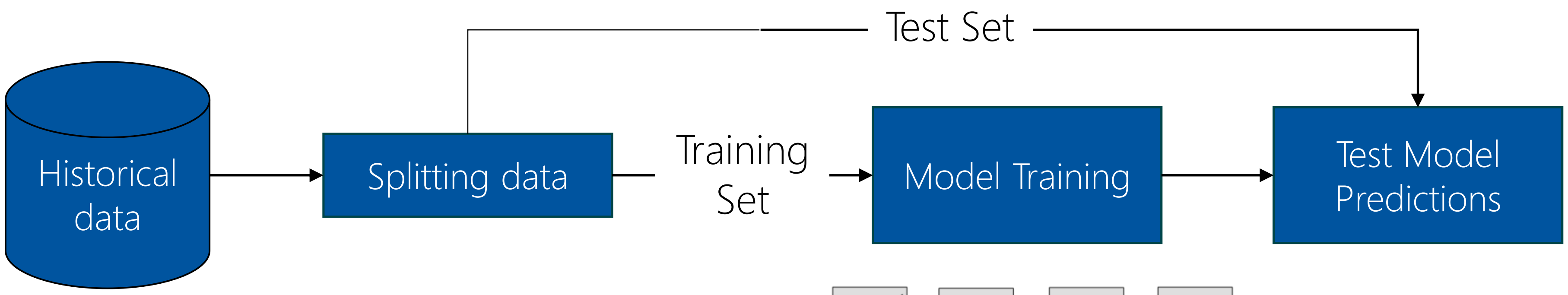
Examples and implementation: Regression

- We want to predict the output power of a combined cycle power plant, based on sensors data.

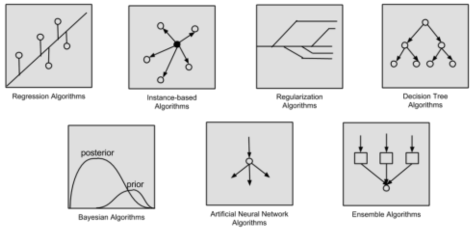
AT	V	AP	RH	PE
8.34	40.77	1010.84	90.01	480.48
23.64	58.49	1011.4	74.2	445.75
29.74	56.9	1007.15	41.91	438.76
19.07	49.69	1007.22	76.79	453.09
11.8	40.66	1017.13	97.2	464.43
13.97	39.16	1016.05	84.6	470.96
22.1	71.29	1008.2	75.38	442.35
14.47	41.76	1021.98	78.41	464
31.25	69.51	1010.25	36.83	428.77
6.77	38.18	1017.8	81.13	484.31
28.28	68.67	1006.36	69.9	435.29

Examples and implementation: Regression

Model Building



AT	V	AP	RH	PE
8.34	40.77	1010.84	90.01	480.48
23.64	58.49	1011.4	74.2	445.75
29.74	56.9	1007.15	41.91	438.76
19.07	49.69	1007.22	76.79	453.09
11.8	40.66	1017.13	97.2	464.43
13.97	39.16	1016.05	84.6	470.96
22.1	71.29	1008.2	75.38	442.35
14.47	41.76	1021.98	78.41	464
31.25	69.51	1010.25	36.83	428.77
6.77	38.18	1017.8	81.13	484.31
28.28	68.67	1006.36	69.9	435.29



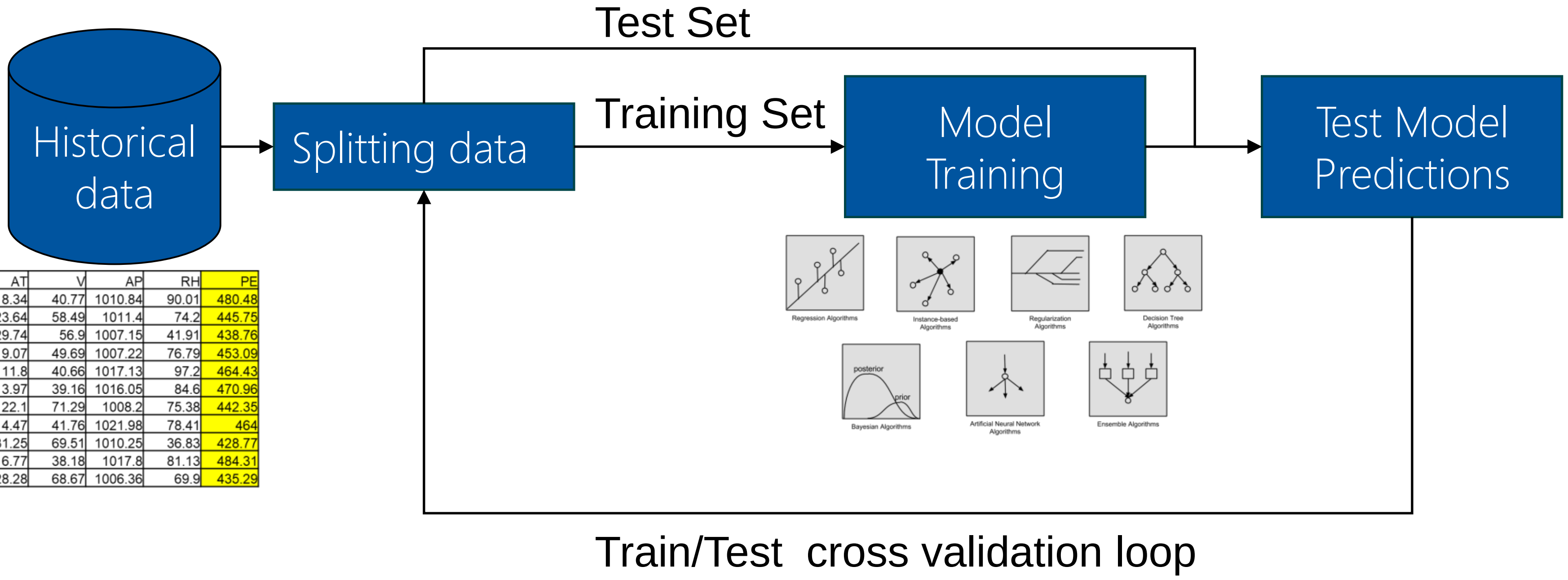
IfU IMA RWTH AACHEN UNIVERSITY

Examples and implementation: Regression

```python
# %% load and split data
import pandas as pd
from sklearn.model_selection import train_test_split
data = pd.read_excel("dataset.xlsx")
X = data.drop("PE", axis=1)
y = data["PE"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=42)

# %% train model
from sklearn import linear_model
model = linear_model.LinearRegression()
model.fit(X_train, y_train)

# %% test model
from sklearn.metrics import r2_score
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
```

Examples and implementation: Regression

Model Building



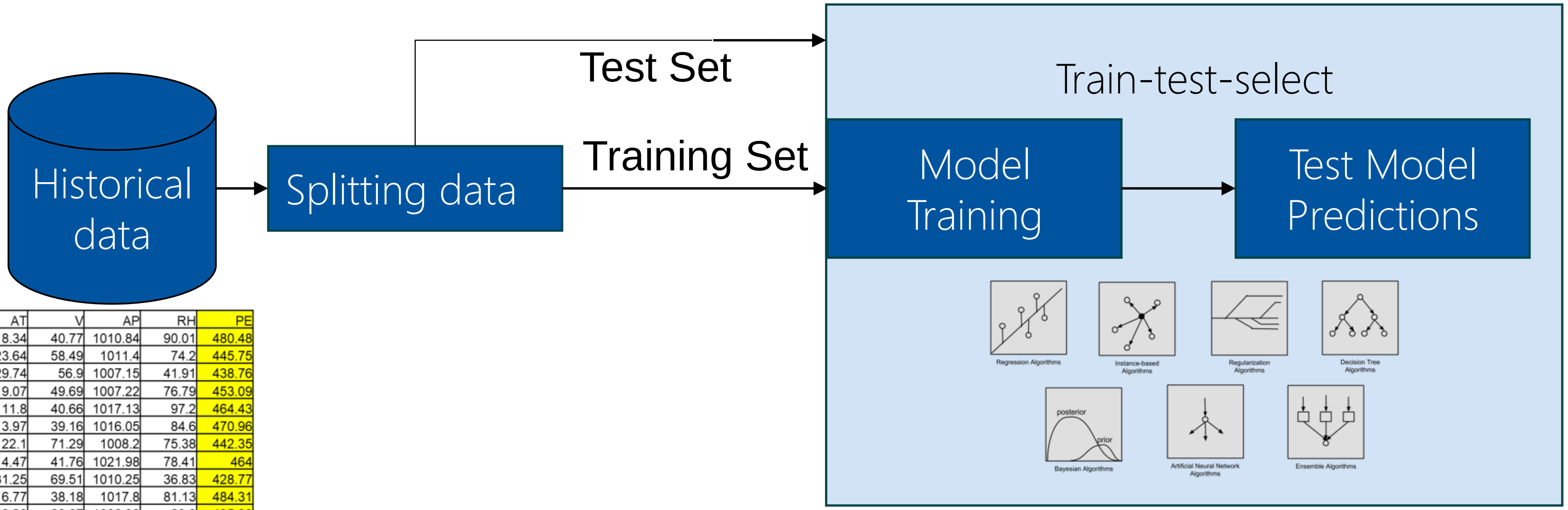
AT	V	AP	RH	PE
8.34	40.77	1010.84	90.01	480.48
23.64	58.49	1011.4	74.2	445.75
29.74	56.9	1007.15	41.91	438.76
19.07	49.69	1007.22	76.79	453.09
11.8	40.66	1017.13	97.2	464.43
13.97	39.16	1016.05	84.6	470.96
22.1	71.29	1008.2	75.38	442.35
14.47	41.76	1021.98	78.41	464
31.25	69.51	1010.25	36.83	428.77
6.77	38.18	1017.8	81.13	484.31
28.28	68.67	1006.36	69.9	435.29

Examples and implementation: Regression

```
# %% load and split data
import pandas as pd
from sklearn.model_selection import train_test_split
data = pd.read_excel("dataset.xlsx")
X = data.drop("PE", axis=1)
y = data["PE"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=42)

# %% train model
from sklearn import linear_model
model = linear_model.LinearRegression()
model.fit(X_train, y_train)

# %% cross validation
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X_train, y_train, cv=5, scoring='r2')
```

Examples and implementation: Regression

Model Building



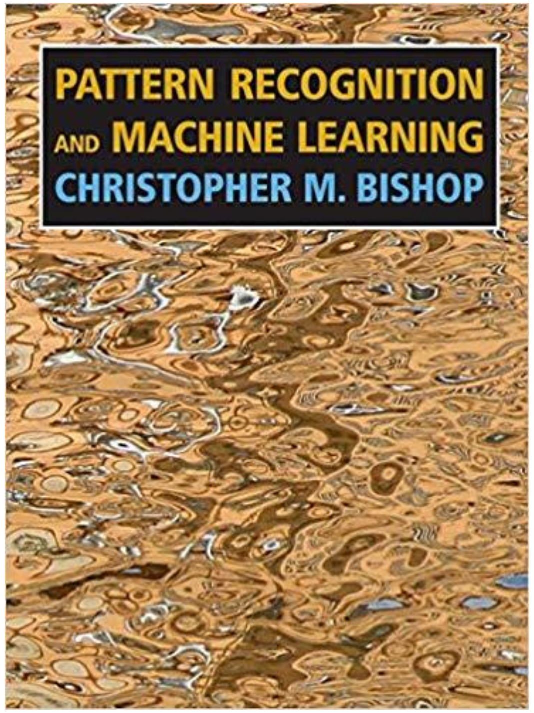
AT	V	AP	RH	PE
8.34	40.77	1010.84	90.01	480.48
23.64	58.49	1011.4	74.2	445.75
29.74	56.9	1007.15	41.91	438.76
19.07	49.69	1007.22	76.79	453.09
11.8	40.66	1017.13	97.2	464.43
13.97	39.16	1016.05	84.6	470.96
22.1	71.29	1008.2	75.38	442.35
14.47	41.76	1021.98	78.41	464
31.25	69.51	1010.25	36.83	428.77
6.77	38.18	1017.8	81.13	484.31
28.28	68.67	1006.36	69.9	435.29

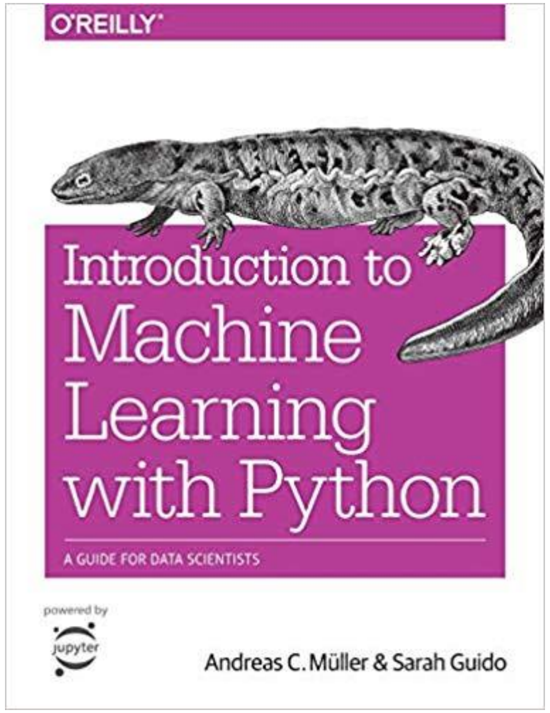
Examples and implementation: Regression

```python
# %% load and split data
import pandas as pd
from sklearn.model_selection import train_test_split
data = pd.read_excel("dataset.xlsx")
X = data.drop("PE", axis=1)
y = data["PE"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=42)

# %% tpot
from tpot import TPOTRegressor

tpot = TPOTRegressor(generations=5, population_size=10, scoring="r2", verbosity=2)
tpot.fit(X_train.values, y_train.values)
print(tpot.score(X_test, y_test))
tpot.export('tpot_model_pipeline.py')
```

Further Reading Material

- https://www.youtube.com/watch?v=bQI5uDxrFfA [Introduction Supervised Learning, Andrew Ng]
- https://scikit-learn.org/stable/supervised_learning.html [Short Explanation and Code Snippets]





Thank you for your attention!

Lecture Team AIDAE

aidae@ima-ifu.rwth-aachen.de