SVILUPPO RISORSE PER L'AZIENDA
CISITA
PARMA

Unione Parmense degli Industriali

GIA Gruppo Imprese
Gruppo Imprese Artigiane

An introduction to Machine learning

Artificial Intelligence and Machine Learning

Gianfranco Lombardo MEng, Ph.D Candidate in ICT
gianfranco.lombardo@unipr.it

https://github.com/gfl-datascience/Lessons-material

What is Machine learning (ML) ?

- ML is the field of study that gives computers the ability to learn from data without being explicitly programmed
- A machine-learning system is **trained** rather than explicitly programmed
- It is a subset of the Artificial Intelligence
- Also known as statistical learning

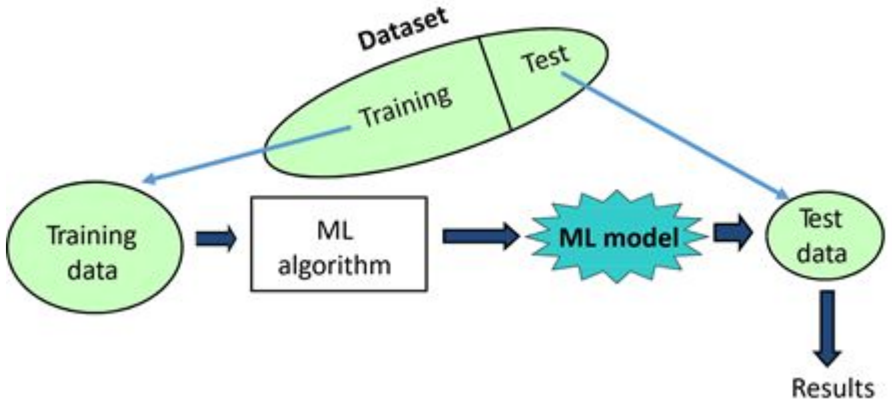
Learning by experience

- We are never certain something will happen, but we usually know (or can estimate rather well) how likely it is to happen or, at least, what is most likely to happen, based on the experience we have acquired throughout our life.
- Experience in Machine learning means: Data in the form of **examples**
- Explore data to find patterns to understand the hidden laws that regulates the domain

How to use data

- **Training Set**: the dataset from which we learn
- **Test Set**: a dataset which must include patterns describing the same problem which do not belong to the training set
    - The test set is used to verify whether what has been learnt in the training phase can be correctly **generalized** on new data

- We want to avoid **overfitting** (the learning algorithm memorizes the training set instead of learning general rules, the model performs well on the training data, but it does not generalize well)



Two ways of learning from examples

- **Supervised learning**: Each example includes an input pattern and the desired output (teaching input) the model is expected to produce when the pattern is input. Learning modifies the model such that actual model outputs become more and more similar to the teaching inputs.

- **Unsupervised learning**: Each example is represented only by the input pattern. Learning modifies the model such that it reflects some regularities and similarities that characterize the data (e.g., by grouping similar data or identifying regions in the pattern space where they are most likely to be located)

A model in supervised learning

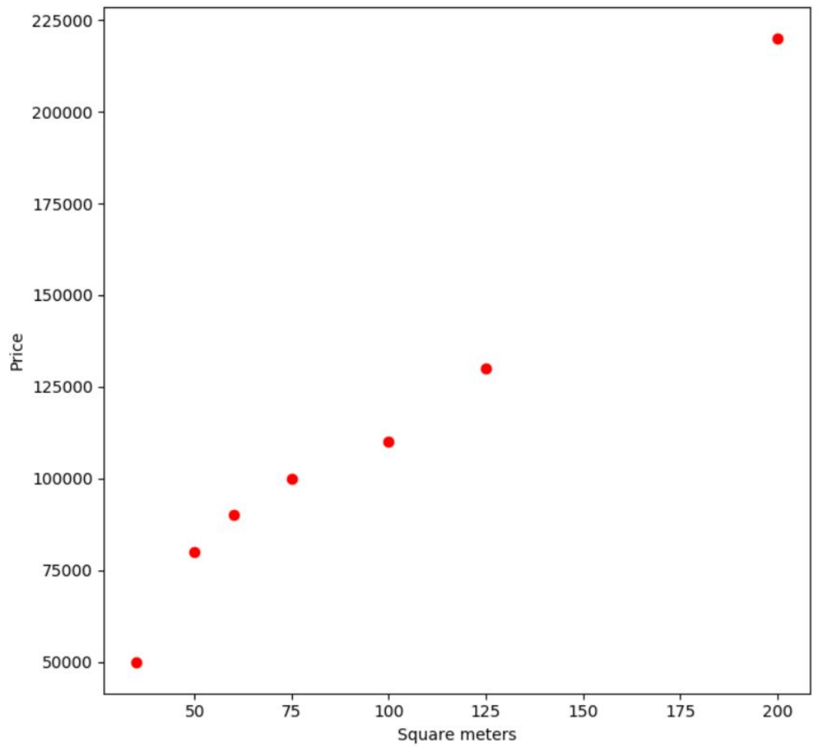
- A model is mathematical tool that maps our examples (observations) **X** to the desired output **y** in the form y ~ F(X) where F() is our model
- F() is a parametric function and we call its parameters **W**
- So, a model is y=F(X,W) and the goal of learning is to estimate these parameters W
- Machine learning algorithms provides different ways to get these parameters
    - Most of them are based on statistics and differential calculus

Example: House price prediction

Mq	Price
50	80,000 $
75	100,000 $
125	130,000 $
35	50,000 $
200	220,000 $
60	90,000 $
100	110,000 $

- We want to build an intelligent system that helps to predict prices of houses in a city
- We only have two information: square meters and the price
- Square meters is what we call a **feature**
- Price represents our **target output**

Example: House price prediction



- X in this case is represented by square meters column
- y in this case is the price column
- Our goal is to identify a model that on the basis of these examples can understand the underlying rule of this domain

What we need ?

- A split of data in training and test set to evaluate if the model is able to generalize
- A ML algorithm to build the model
- A metric for this evaluation

Splitting data

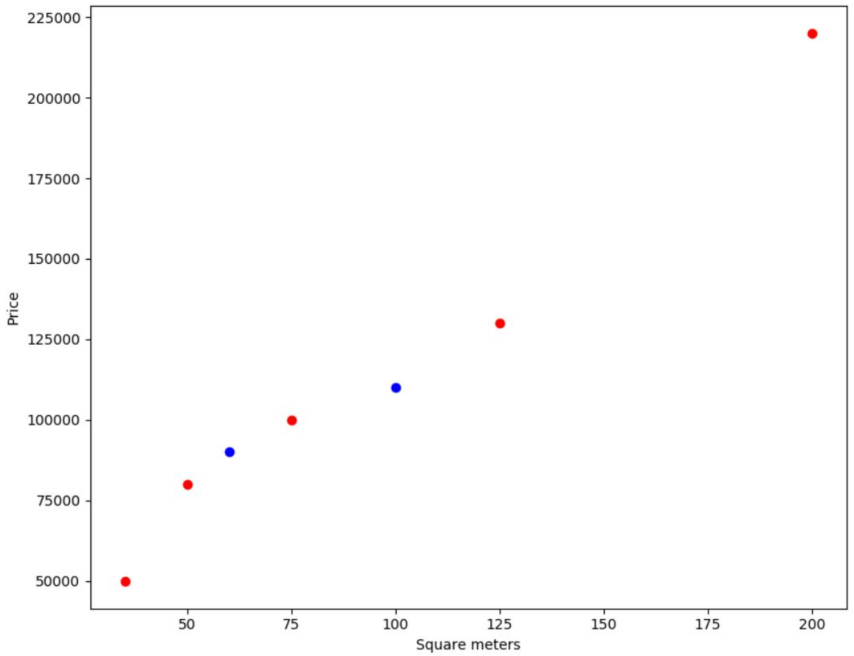
Mq	Price
50	80,000 $
75	100,000 $
125	130,000 $
35	50,000 $
200	220,000 $
60	90,000 $
100	110,000 $

Training set

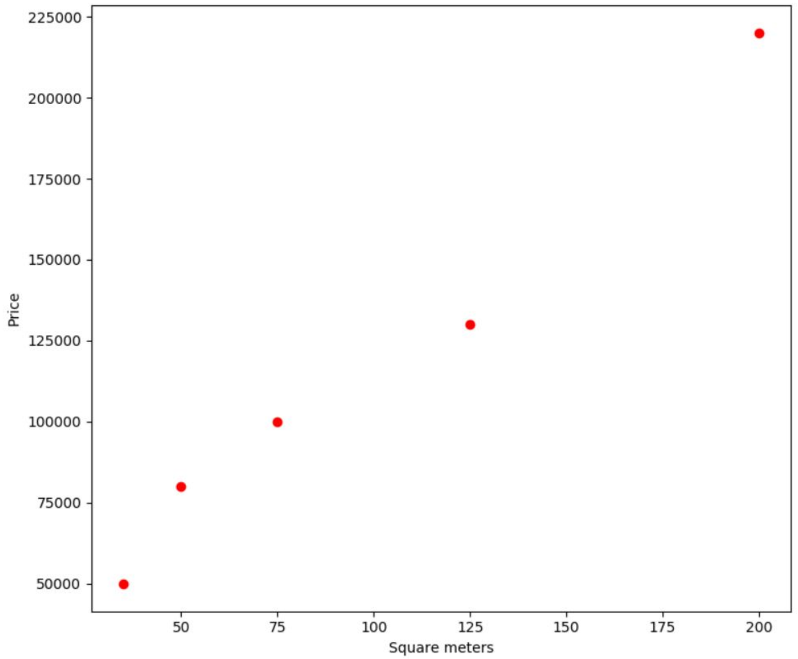
Test set

- A suggested division is to use the 70 % of data in training and 30 % for the test

Dataset VS training set and test set



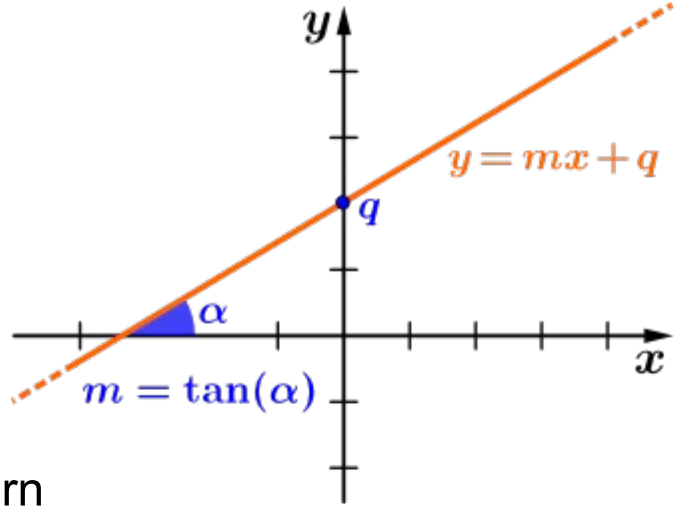
DATASET

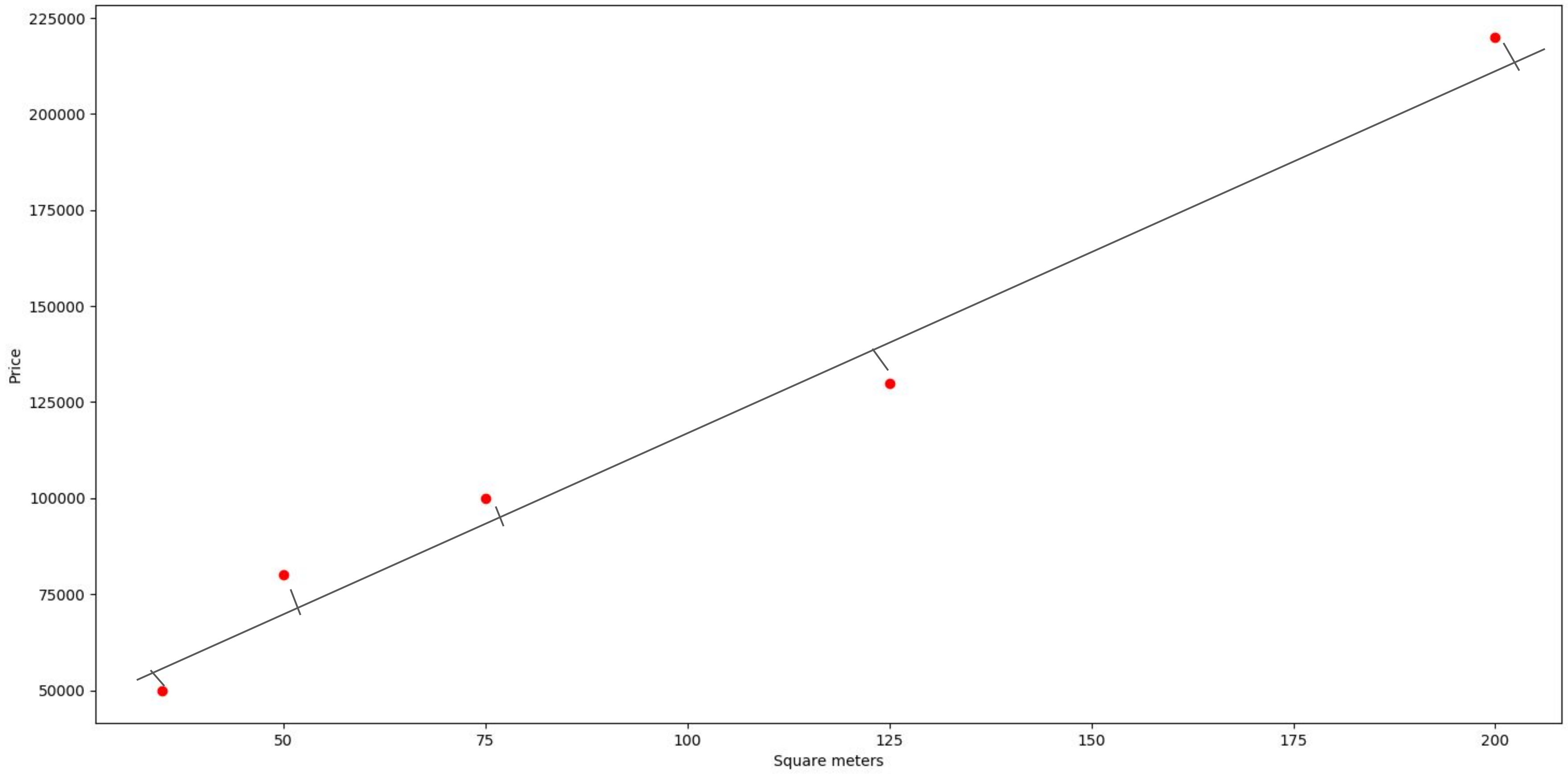


TRAINING SET

Linear model

- We want to find the most easy linear model that fits our data to get predictions on new data
- In our case let's consider the equation of a line
    - $y = mx + q$
        - Where m is the slope
        - q is the intercept
- If we write that equation as $y = w_1 X + w_0$
    - y is our target output
    - X is our feature
    - W are the two parameters that we have to learn




225000
200000
175000
150000
125000
100000
75000
50000
Price
50
75
100
125
150
175
200
Square meters

Mean square error

- We want w_0 and w_1 to have a line that has the minimum distance from our point
- Let's call the output of our linear model $y^* = F(x_i, W)$
- We want the best W that minimize the following:
    - $\sum (y_i - y_i^*)^2$: Loss function $\mathcal{L}$
        - The Sum is over all of our training examples i
- To minimize this function we should compute the derivative of $\mathcal{L}$ with respect to W and find where this derivative is equal to 0
    - $\nabla_W \mathcal{L} = 0$

Linear Regression

- A traditional statistics tool that fits a straight line as a model
- It is considered the first ML algorithm with the idea of learning something from data
- It computes the derivative of the loss and estimates the most performing line
- It is available in the python module scikit-learn (sklearn)

Example

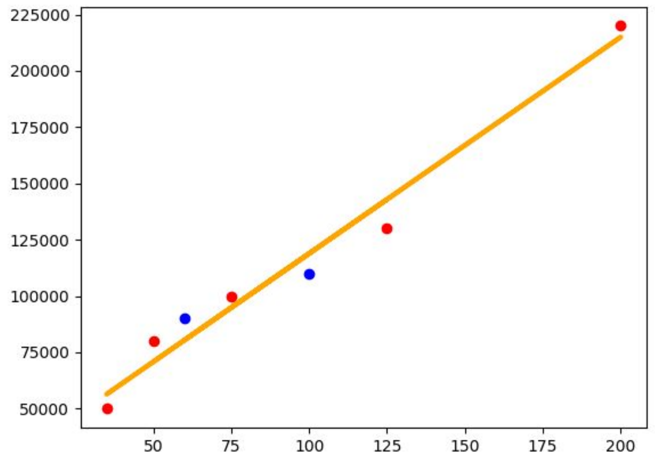
```
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import numpy as np

X_train = [50,75,125,35,200]
X_test = [60,100]

y_train=[80000,100000,130000,50000,220000]
y_test=[90000,110000]
```

Example

```
# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(np.array(X_train).reshape(-1, 1), y_train)

# Make predictions using the testing set
y_pred = regr.predict(np.array(X_test).reshape(-1, 1))

# The coefficients
print('Coefficients: \n', regr.coef_)
print('Coefficients: \n', regr.intercept_)

# The mean squared error
print('Mean squared error: %.2f'% mean_squared_error(y_test, y_pred))
```

Plot

```
plt.plot(X_train,y_train,"ro")
plt.plot(X_test,y_test,"bo")
X =np.concatenate([X_train,X_test])
y = [ x_i*regr.coef_ + regr.intercept_ for x_i in X]

plt.plot(X,y, color='orange', linewidth=3)
plt.show()
```



How good is our model ?

- We are computing the mean square error on the test set trying to understand if our model is good or not
- What is the answer ?
  - MSE = 85318750.59
  - However MSE is not in the same unit of our samples
  - An idea: **Root mean square error** : 9236.814958911717
    - Is it good or not ?

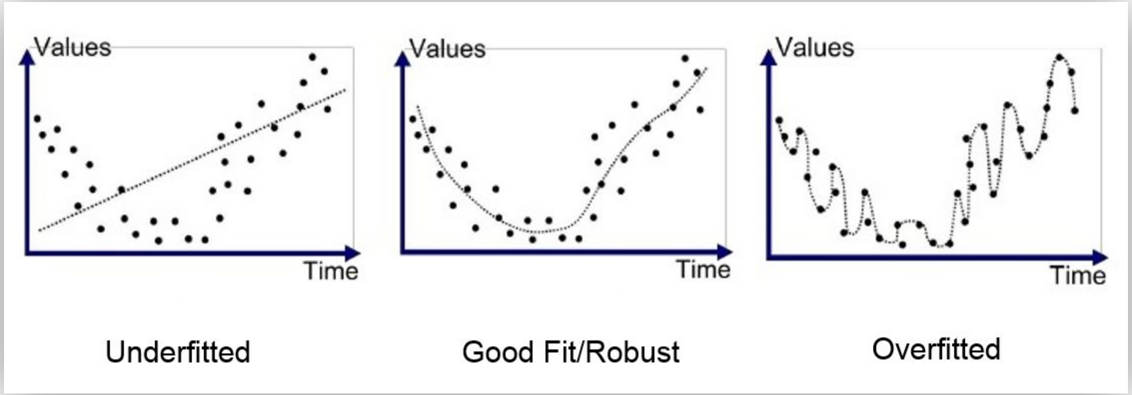
Some problems

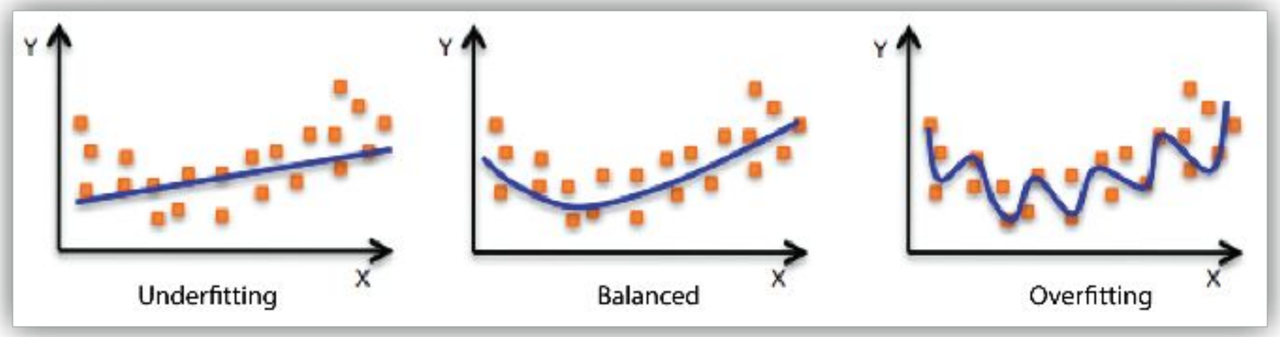
- Data are purely invented so we cannot expect great results
    - Data are important, AI is not magic

- Maybe a polynomial or a non linear model could be better choices

- Dataset is too small
    - Data are important and we need the bigger dataset of observation that is possible
    - Overfitting and underfitting can be relevant with a small dataset

Overfitting VS underfitting





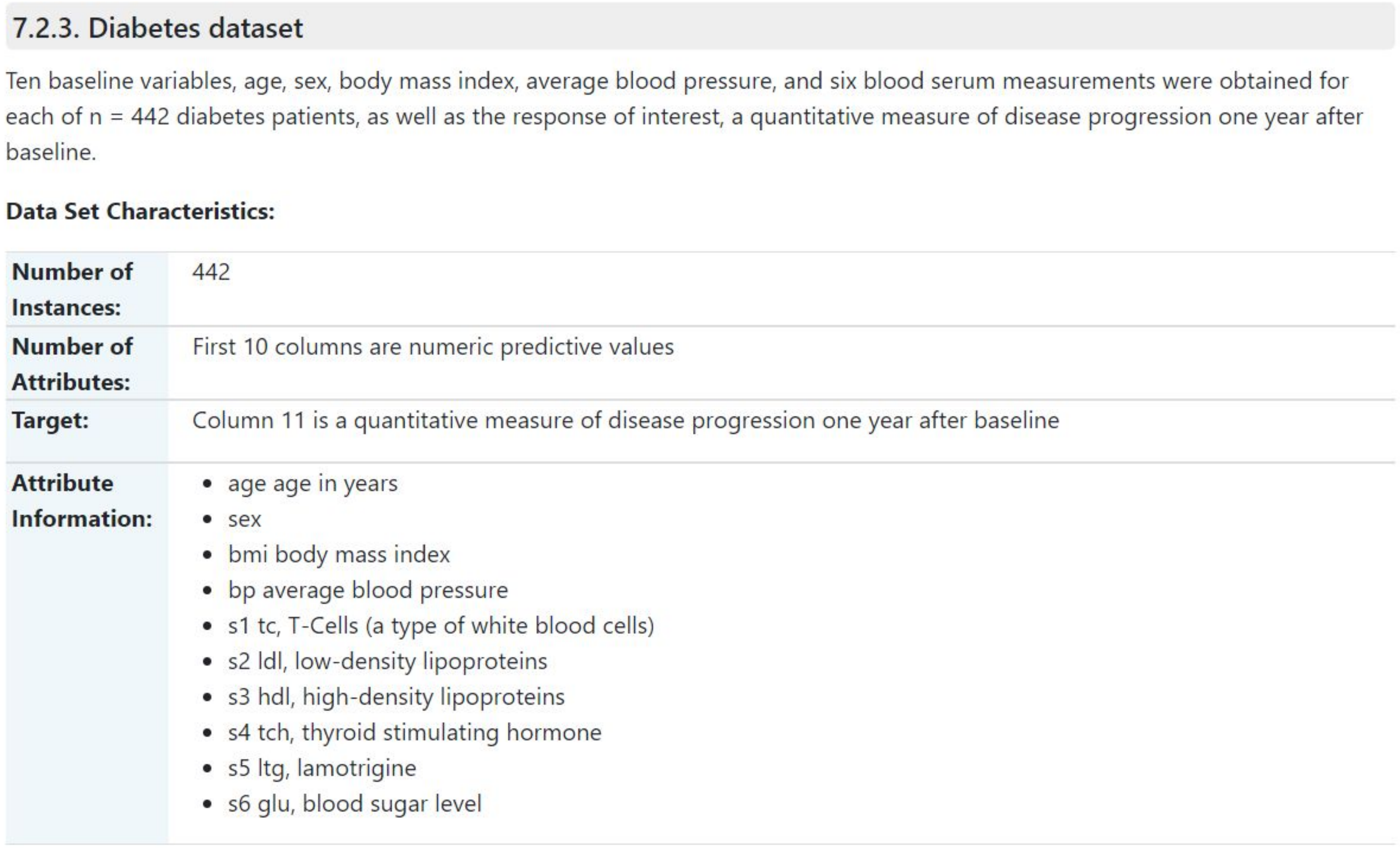
Good datasets for good models

- A model ‘learnt from data’ is generally as good as the data from which it has been derived
- A good dataset should be
  - Large
  - Correct (affected by noise as little as possible)
  - Consistent (examples must not be contradictory and a pattern should exist)
  - Well balanced (all classes adequately represented)

Scikit-learn datasets

- Scikit learn provides several public datasets to develop intelligent system but in particular for teaching reasons
  - `from sklearn import datasets`
  - `Complete list:` https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets
  - `https://scikit-learn.org/stable/datasets/`

Examples with several features



7.2.3. Diabetes dataset

Ten baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements were obtained for each of n = 442 diabetes patients, as well as the response of interest, a quantitative measure of disease progression one year after baseline.

Data Set Characteristics:

Number of Instances:	442
Number of Attributes:	First 10 columns are numeric predictive values
Target:	Column 11 is a quantitative measure of disease progression one year after baseline
Attribute Information:	• age age in years • sex • bmi body mass index • bp average blood pressure • s1 tc, T-Cells (a type of white blood cells) • s2 ldl, low-density lipoproteins • s3 hdl, high-density lipoproteins • s4 tch, thyroid stimulating hormone • s5 ltg, lamotrigine • s6 glu, blood sugar level

Diabete

```
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

# Load the diabetes dataset
X, y = datasets.load_diabetes(return_X_y=True)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33)
```

Diabete

```
# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(X_train, y_train)

# Make predictions using the testing set
y_pred = regr.predict(X_test)

# The coefficients
print('Coefficients: \n', regr.coef_)
# The mean squared error
print('Mean squared error: %.2f'
      % mean_squared_error(y_test, y_pred))
from math import sqrt
rmse = sqrt(mean_squared_error(y_test, y_pred))
print(rmse)
```

BREAK

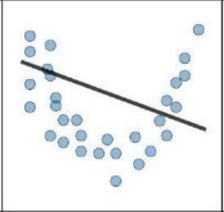
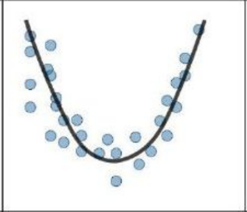
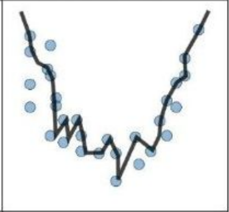
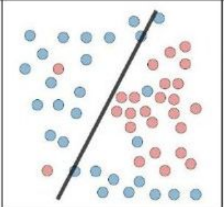
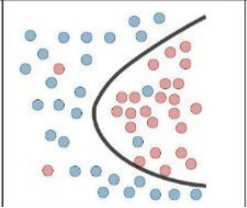
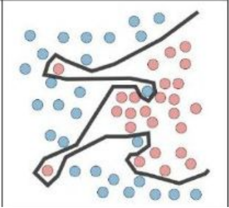
Regression and classification

- When the target value we aim to predict is a numerical real value, we are performing a **regression task**
- When the target value to be predicted is a class among two or more choices but limited, we are performing a **classification task**
  - Examples:
    - Price prediction is regression
    - Detect a cat or a dog in a image is classification
    - Detect a cancer (yes or not) is a binary classification

- In machine learning there algorithms only for regression tasks (e.g linear regression) and some only for classification. Finally some others for both

Model for classification task

- We want to estimate a parametric function F(X,W) that maps our input features X to one or more of our target labels in y, where y is a limited set and is called class.
- The goal is to find the decision boundaries in the features space

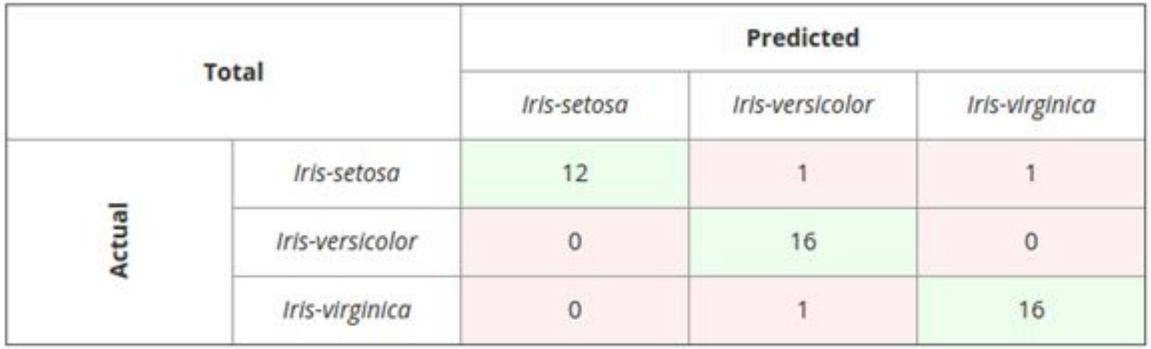
	Underfitting	Just right	Overfitting
Symptoms	- High training error - Training error close to test error - High bias	- Training error slightly lower than test error	- Low training error - Training error much lower than test error - High variance
Regression			
Classification			

Accuracy

- To evaluate a classification model we can use the **accuracy**
- The accuracy is defined as the ratio among the number of correct predictions over the total number of predictions
- The accuracy is often expressed as a percentage
- **Pay attention: Accuracy has a meaning only when the dataset is balanced**
- If the dataset is unbalanced we have to check other metrics as precision, recall, F1-score etc..

Confusion matrix

- The diagonal elements represent the number of points for which the predicted label is equal to the true label. The off-diagonal elements are those that are mislabeled by the classifier.
- Perfect classification: diagonal matrix



Total		**Predicted**		
		Iris-setosa	*Iris-versicolor*	*Iris-virginica*
Actual	*Iris-setosa*	12	1	1
	Iris-versicolor	0	16	0
	Iris-virginica	0	1	16

Accuracy = (12+16+16) / (12+1+1+..) =
= 0,93

Example: Iris plants dataset

Number of Instances

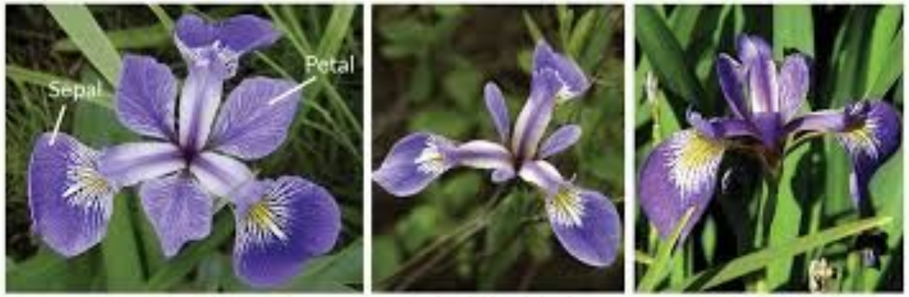
150 (50 in each of three classes)

Number of Attributes

4 numeric, predictive features and the class

Features Information

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- **class:**
  - Iris-Setosa
  - Iris-Versicolour
  - Iris-Virginica



Iris Versicolor Iris Setosa Iris Virginica

Logistic Regression

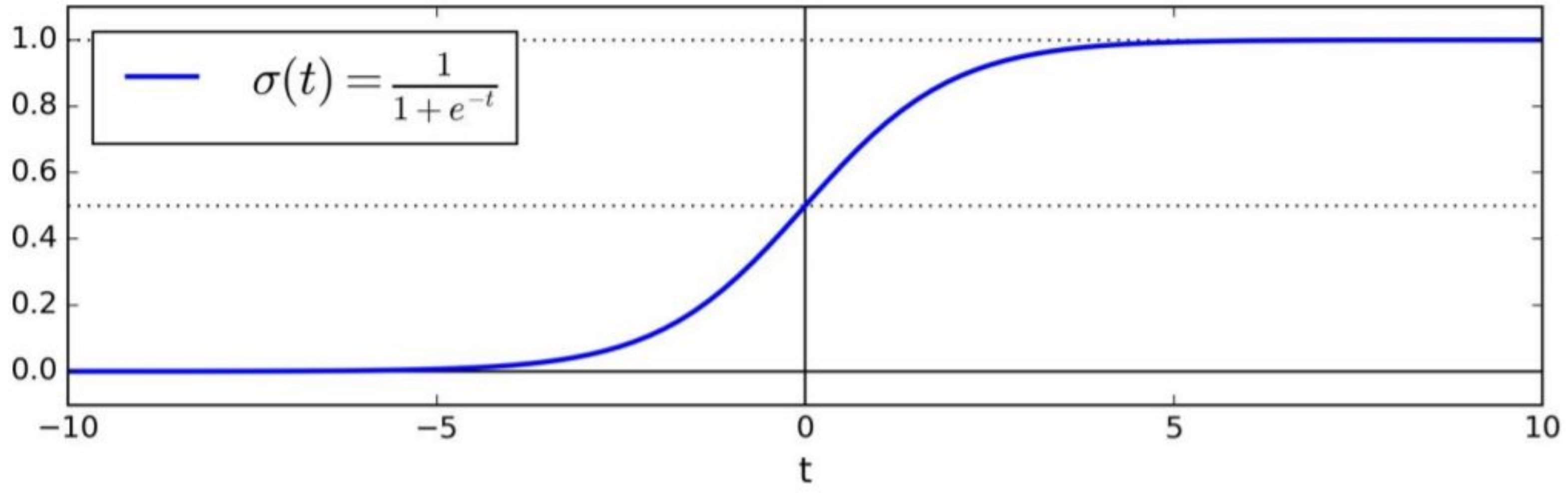
- Logistic regression (despite the name) is a classification algorithm
- The name is due to the past when the logit function was invented and used as a regression algorithm
- As a classification algorithm it estimates the probability that an instance belongs to a class or not (Binary classification).
  - If the estimated probability is greater than 50% then the model predicts that class

- The logistic function is a sigmoid function (S-shape) that outputs a number between 0 and 1

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

Logistic function

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$



Logistic function

- Just like a linear regression model, a logistic regression computes a weighted sum of the input features plus a bias term
- But instead of outputting the result directly, it outputs the logistic of this result
  - So when the probability that an instance belongs to the positive class is computed, the y^* can be computed easily:
    - Class 0 (or negative) if $p < 0.5$
    - Class 1 (or positive) if $p >= 0.5$
- When possible labels are more than two, several binary classifier are built to identify the correct class

Iris plants

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

# import some data to play with
iris = datasets.load_iris()
X = iris.data  # we only take the first two features.
y = iris.target

logreg = LogisticRegression()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
```

Iris plants

```
# Create an instance of Logistic Regression Classifier and fit the data.
logreg.fit(X_train, y_train)

# Make predictions using the testing set
y_pred = logreg.predict(X_test)
acc= accuracy_score(y_test, y_pred)
print(acc)
```

Pandas

- Library for data structures and data analysis
  - High performance
  - Easy to use
- Useful to import datasets, for data cleaning, data splitting
- Easy import of CSV files and their management
- Base module for several projects in statistical modeling and machine learning
- Install with : `pip install pandas`

- Main concept : DataFrame (can be seen as an Excel worksheet)
- Values in a DataFrame are in the form of numpy arrays with indexes for rows and columns

Creating a DataFrame

Py

```
>>> import pandas as pd
>>> df = pd.DataFrame([[1, 2], [4, 5], [7, 8]],
                      index=['cobra', 'viper',
'sidewinder'],
                      columns=['max_speed', 'shield'])
>>> df
            max_speed  shield
cobra               1       2
viper               4       5
sidewinder          7       8
```

Selection of columns

Py

```
>>> df['max_speed']  # or simply df.max_speed
cobra         1
viper         4
sidewinder    7
Name: max_speed, dtype: int64

>>> df[['max_speed', 'shield']]
            max_speed  shield
cobra               1       2
viper               4       5
sidewinder          7       8
```

Selection of rows

- With brackets, a slice of rows can be selected
  - Provide both start and stop, not a single position or label
  - With labels, *the stop label is included!*

```
>>> df[1:3]
            max_speed  shield
viper               4       5
sidewinder          7       8

>>> df['viper':'sidewinder']
            max_speed  shield
viper               4       5
sidewinder          7       8
```

Py

Selection in both axis

- Slicing by labels (*the stop label is included!*)

```
>>> df.loc['viper':'sidewinder', 'max_speed':'shield']
            max_speed  shield
viper               4       5
sidewinder          7       8
```

- Slicing by positions (*the stop position is not included!*)

```
>>> df.iloc[1:3, 0:2]
            max_speed  shield
viper               4       5
sidewinder          7       8
```

Set a value

- Create a copy of the DataFrame with copy()
- Use assignment operator and loc to set new values

Py

```
>>> df2 = df.copy()
>>> df2.loc[['viper', 'sidewinder'], ['shield']] = 50
>>> df2
            max_speed  shield
cobra               1       2
viper               4      50
sidewinder          7      50
```

Adding a column and isin method

```
>>> df2 = df.copy()
>>> df2['label'] = ['one','two','three']
>>> df2
            max_speed  shield  label
cobra               1       2    one
viper               4       5    two
sidewinder          7       8  three


#isin as another way for slicing on both axis


>>> df2[df2['label'].isin(['one','two'])]
            max_speed  shield  label
cobra               1       2    one
viper               4       5    two
```

Data visualization

- Using matplotlib and the iris dataset.csv:
    - Read the dataset with pandas or with traditional python functions
    - Plot iris data using sepal length (x axis) and sepal width (y axis), representing different classes with different colors.
    - Plot iris data using petal length (x axis) and petal width (y axis), representing different classes with different colors.
- Compare the two representations

Example: Boston house price

- Regression task with 13 features

- from sklearn.datasets import load_boston
- X, y = load_boston(return_X_y=True)

- Divide the dataset in 70/30 train and test

- Train a linear regression model

- Repeat data splitting and training for 10 times and report the average root mean square error

- What happens if you use a data splitting like 50/50 ?

Example: Breast cancer

Number of Instances
569

Number of Attributes
30 numeric, predictive attributes and the class

Attribute Information

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness (perimeter^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension (“coastline approximation” - 1)
- **class:**
  - WDBC-Malignant
  - WDBC-Benign

Breast cancer

- X, y = datasets.load_breast_cancer(return_X_y=True)

- Train a logistic classifier with logistic regression algorithm
- Repeat data splitting and training for 10 times and report the average accuracy
- Repeat the entire exercise with a splitting 50/50%, is it worse ?