MEDICAL TECHNOLOGY
& INTELLIGENT
INFORMATION SYSTEMS

Python για Machine Learning I: *από τη γλώσσα στα εργαλεία ανάλυσης δεδομένων*



- Τμήμα Μηχανικών Επιστήμης Υλικών
- Website: https://medlab.cc.uoi.gr/
- Ιωάννινα 2026 - 2027

Διαδρομή του μαθήματος και μαθησιακοί στόχοι

Εισαγωγή

Από τη γλώσσα στα εργαλεία: τα ίδια δεδομένα υλικών, λιγότερος κώδικας, περισσότερη ανάλυση

NumPy
πίνακες, πράξεις χωρίς βρόχο

→

Pandas
CSV, επιλογή, καθαρισμός

→

Γραφήματα & EDA
Matplotlib, συσχετίσεις

→

Machine Learning
fit, predict, MAE

→

Εργαστήριο
από το CSV στο μοντέλο

Από το Μάθημα 2 στο Μάθημα 5: ίδια λογική, λιγότερος κώδικας

Μάθημα 2 (Python)	Σήμερα (βιβλιοθήκες)
`λίστα [380, 420, 510]`	`np.array([380, 420, 510])`
`for + if + append (φίλτρο)`	`ys[ys > 450]`
`λίστα από dictionaries`	`DataFrame`
`συσσωρευτής: total / count`	`df.describe()`
`def yield_strength_MPa(...)`	`model.fit(...) · model.predict(...)`

Στο τέλος θα μπορείτε να…

- φορτώνετε και καθαρίζετε ένα dataset υλικών
- επιλέγετε και φιλτράρετε δείγματα με Pandas
- εξερευνάτε τα δεδομένα με γραφήματα και συσχετίσεις
- εκπαιδεύετε και αξιολογείτε ένα πρώτο μοντέλο

Οι έννοιες τις διαλέξεις γίνονται σήμερα κώδικας: πίνακας X και στόχος y, σύνολο εκπαίδευσης και ελέγχου, γενίκευση.

Από το Μάθημα 2

Τι μάθαμε στο Μάθημα 2 — και γιατί χρειαζόμαστε ένα νέο εργαλείο

- print() και σχόλια
- Μεταβλητές και ανάθεση
- int, float, strings, f-strings
- Λίστες, indexing, slicing
- Tuples και dictionaries
- Booleans και συγκρίσεις
- if / elif / else
- for, range(), while
- Συναρτήσεις (def, return)
- Πρώτο πρόγραμμα

Το πρόβλημα: οι λίστες δεν κάνουν μαθηματικά

Python

```
values = [10, 20, 30]
print(values * 2)
```

Αποτέλεσμα (output)

```
[10, 20, 30, 10, 20, 30]
```

Περιμέναμε [20, 40, 60] — πήραμε την ίδια λίστα δύο φορές! Για 10 000 μετρήσεις θα χρειαζόμασταν βρόχο για κάθε πράξη.

Λύση: NumPy arrays →

Βιβλιοθήκες: εισαγωγή πακέτων με import

Εισαγωγή

Χρησιμοποιούμε έτοιμες συναρτήσεις αντί να «ξαναεφεύρουμε τον τροχό»

matplotlib
πακέτο (package) ≈ φάκελος

pyplot
υποσύστημα (module) ≈ αρχείο κώδικα

plot()
συνάρτηση (function) ≈ εντολή μέσα στο αρχείο

plt.plot(...) = matplotlib → pyplot → plot

ΣΥΜΒΟΥΛΗ
ModuleNotFoundError → η βιβλιοθήκη δεν έχει εγκατασταθεί (pip install ...).

Python

```python
import math
print(math.sqrt(16))

import numpy as np     # ψευδώνυμο np
import pandas as pd
import matplotlib.pyplot as plt
```

Αποτέλεσμα (output)

```
4.0
```

ΣΥΧΝΟ ΛΑΘΟΣ
from numpy import * εισάγει τα πάντα και μπορεί να «σκεπάσει» δικές σας μεταβλητές. Προτιμάμε import numpy as np.



NumPy
Αριθμητικοί πίνακες

Python List και NumPy Array: δύο διαφορετικά εργαλεία

1 · NumPy

Γενικός περιέκτης έναντι αριθμητικού πίνακα σχεδιασμένου για επιστημονικούς υπολογισμούς

Python List

γενικού σκοπού περιέκτης (container)

Python

```
values = [10, 20, 30]
doubled = []
for v in values:
    doubled.append(v * 2)
print(doubled)
```

Αποτέλεσμα (output)

```
[20, 40, 60]
```

βρόχος: ένα στοιχείο κάθε φορά

NumPy Array

αριθμητικός πίνακας για scientific computing

Python

```
import numpy as np

values = np.array([10, 20, 30])
print(values * 2)
```

Αποτέλεσμα (output)

```
[20 40 60]
```

διανυσματικοποίηση (vectorization): όλα μαζί

ΚΥΡΙΟ ΜΗΝΥΜΑ

List: μικτοί τύποι, το * 2 επαναλαμβάνει τη λίστα. **Array:** ένας τύπος, το * 2 πολλαπλασιάζει κάθε τιμή, πολύ γρήγορο σε μεγάλα δεδομένα.

Δημιουργία NumPy arrays

1 · NumPy

Από λίστες τιμών ή με έτοιμες συναρτήσεις για ακολουθίες αριθμών

Python

```python
import numpy as np

ys = np.array([380, 420, 510, 465])   # από λίστα
T = np.arange(700, 901, 50)           # αρχή, τέλος, βήμα
x = np.linspace(0, 1, 5)              # 5 τιμές από 0 έως 1
z = np.zeros(3)                       # τρία μηδενικά

print(T)
print(x)
print(z)
```

Αποτέλεσμα (output)

```
[700 750 800 850 900]
[0.   0.25 0.5  0.75 1.  ]
[0. 0. 0.]
```

`np.array(list)`
μετατρέπει λίστα σε array
π.χ. μετρήσεις από το εργαστήριο

`np.arange(a, b, step)`
τιμές με σταθερό βήμα (το b εξαιρείται)
π.χ. θερμοκρασίες πειράματος

`np.linspace(a, b, n)`
n τιμές από a έως b (και τα δύο μέσα)
π.χ. εύρος σύστασης για γράφημα

`np.zeros(n)`
array με μηδενικά
π.χ. «κενό» για αποτελέσματα

ΠΡΟΣΟΧΗ
np.arange(700, 900, 50) σταματά στο 850. Για να συμπεριληφθεί το 900 γράφουμε 901.

Διαστάσεις ενός array: shape, ndim, size, dtype

1 · NumPy

Ένας 2D πίνακας: γραμμές = δείγματα, στήλες = χαρακτηριστικά

	C (wt%)	Mn (wt%)	T (°C)
δείγμα 1	0.20	1.1	750
δείγμα 2	0.30	1.3	800
δείγμα 3	0.45	0.9	850

axis 0 δείγματα (samples)

axis 1: χαρακτηριστικά (features)

```python
data = np.array([[0.20, 1.1, 750],
                 [0.30, 1.3, 800],
                 [0.45, 0.9, 850]])
print(data.shape)
print(data.ndim, data.size)
print(data.dtype)
```

Αποτέλεσμα (output)

```
(3, 3)
2 9
float64
```

shape (γραμμές, στήλες)

ndim πλήθος διαστάσεων

size συνολικά στοιχεία

dtype τύπος των στοιχείων

ΠΡΟΣΟΧΗ

Όλα τα στοιχεία έχουν τον ίδιο τύπο: το 750 αποθηκεύεται ως 750.0 (float64). Ένα 1D array έχει shape (4,) — διαφορετικό από έναν 2D (4, 1).

Επιλογή γραμμών και στηλών σε 2D arrays

1 · NumPy

data[γραμμή, στήλη] — το «:» σημαίνει «όλα»

`data[0]`

1η γραμμή = 1ο δείγμα

0.20	1.1	750
0.30	1.3	800
0.45	0.9	850

`data[:, 2]`

3η στήλη = θερμοκρασία όλων

0.20	1.1	750
0.30	1.3	800
0.45	0.9	850

`data[1, 0]`

δείγμα 2, περιεκτικότητα C

0.20	1.1	750
0.30	1.3	800
0.45	0.9	850

Αποτέλεσμα (output)

```
[2.0e-01 1.1e+00 7.5e+02]
[750. 800. 850.]
0.3
```

ΠΡΟΣΟΧΗ

Επιστημονική σημειογραφία: 7.5e+02 = 7.5 × 10^2 = 750. Το NumPy τη χρησιμοποιεί όταν οι τιμές μιας γραμμής έχουν πολύ διαφορετικά μεγέθη (0.2 και 750).

Διανυσματικές πράξεις (vectorized operations)

1 · NumPy

Μία εντολή εφαρμόζεται σε κάθε στοιχείο του array — χωρίς βρόχο

Python

```
T_C = np.array([750, 800, 850, 900])
T_K = T_C + 273.15                  # °C → K

ys_MPa = np.array([380, 420, 510, 465])
ys_ksi = ys_MPa * 0.145038          # MPa → ksi

print(T_K)
print(ys_ksi.round(1))
```

Αποτέλεσμα (output)

```
[1023.15 1073.15 1123.15 1173.15]
[55.1 60.9 74.  67.4]
```

Πράξη ανά στοιχείο (element-wise):

750	+ 273.15 →	1023.15
800	+ 273.15 →	1073.15
850	+ 273.15 →	1123.15
900	+ 273.15 →	1173.15

Python

```
a = np.array([1, 2, 3])
b = np.array([10, 20, 30])
print(a * b)        # [10 40 90]
```

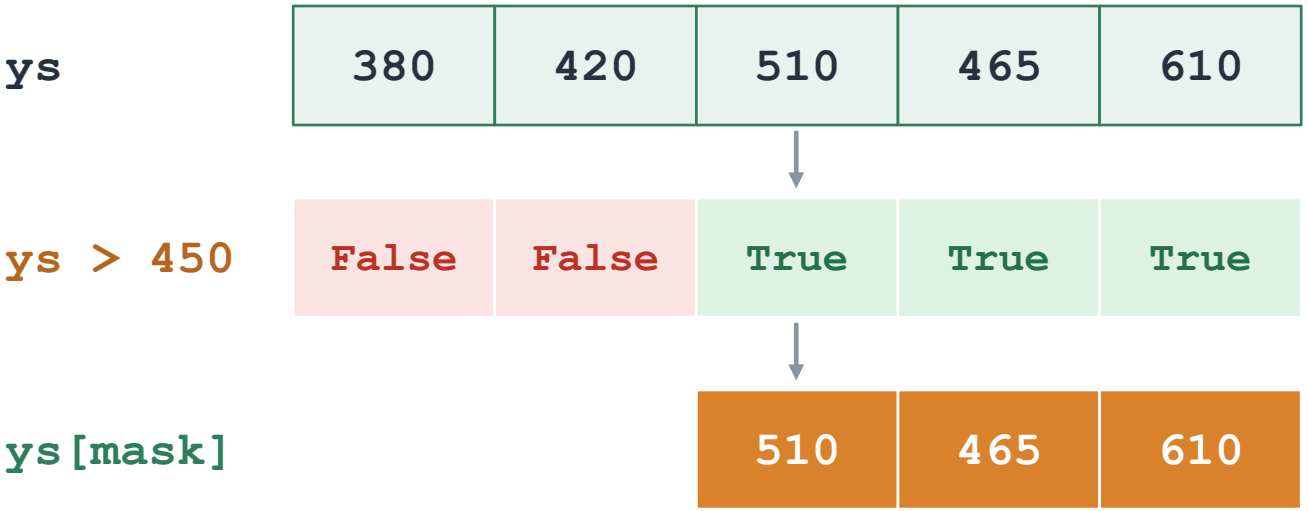
ΠΡΟΣΟΧΗ (MATLAB)

Στο NumPy, a * b είναι γινόμενο ανά στοιχείο (στο MATLAB: a .* b). Το γινόμενο πινάκων γράφεται a @ b.

Λογικές μάσκες (Boolean masks) και φιλτράρισμα

1 · NumPy

Επιλέγουμε μόνο τα δείγματα που ικανοποιούν ένα κριτήριο




```python
ys = np.array([380, 420, 510, 465, 610])
mask = ys > 450
print(mask)
print(ys[mask])
print(mask.sum())        # πόσα True
```

Αποτέλεσμα (output)

```
[False False  True  True  True]
[510 465 610]
3
```

ΕΦΑΡΜΟΓΗ ΣΤΑ ΥΛΙΚΑ

«Ποια δείγματα ξεπερνούν τα 450 MPa;» — η ίδια λογική θα φιλτράρει γραμμές ενός DataFrame στο Pandas.

ΣΥΜΒΟΥΛΗ

Το True μετρά ως 1 και το False ως 0: γι' αυτό το mask.sum() μετρά τα δείγματα που περνούν.

1 · NumPy

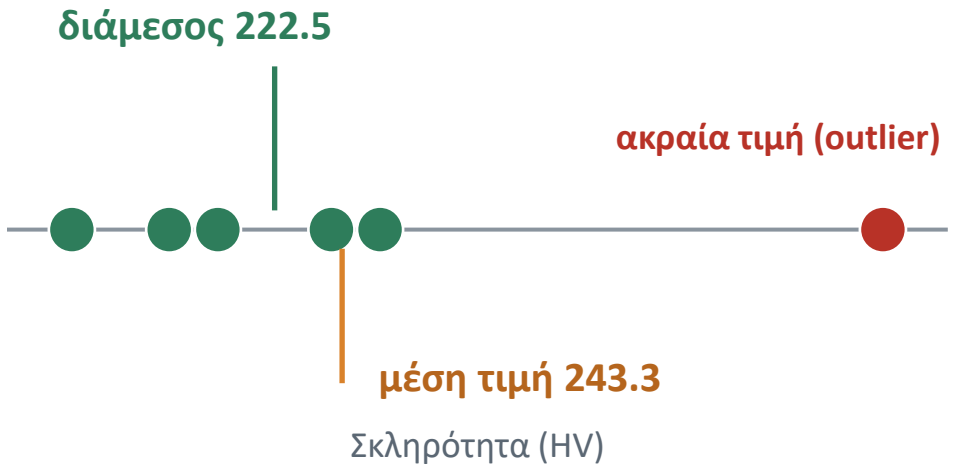
mean, median, std, min, max — και γιατί η διάμεσος είναι πιο «ανθεκτική»

```python
hv = np.array([160, 190, 205, 240,
255, 410])

print(hv.mean(), np.median(hv))
print(hv.std().round(1))
print(hv.min(), hv.max())
```

Αποτέλεσμα (output)

```
243.33333333333334 222.5
80.8
160 410
```



`mean()`	μέση τιμή	`median()`	μεσαία τιμή
`std()`	τυπική απόκλιση	`min()` / `max()`	ελάχιστο / μέγιστο

ΚΥΡΙΟ ΜΗΝΥΜΑ

Μία ακραία μέτρηση (410 HV) τραβά τη μέση τιμή προς τα πάνω, ενώ η διάμεσος σχεδόν δεν επηρεάζεται. Θα το χρησιμοποιήσουμε για τις ελλείπουσες τιμές.

ΣΥΜΒΟΥΛΗ

Σε 2D: data.mean(axis=0) δίνει τη μέση τιμή κάθε στήλης (κάθε feature).

NumPy στην πράξη: περιγραφείς ενός κράματος υψηλής εντροπίας

1 · NumPy

Δύο τύποι από τη βιβλιογραφία, μία γραμμή κώδικα ο καθένας · CoCrFeMnNi (κράμα Cantor)

Python

```python
import numpy as np

elements = ["Co", "Cr", "Fe", "Mn", "Ni"]
c = np.array([0.2, 0.2, 0.2, 0.2, 0.2])     # mol κλάσματα
r = np.array([1.251, 1.249, 1.241, 1.350, 1.246])  # Å
R = 8.314                                    # J/(mol·K)

S_conf = -R * np.sum(c * np.log(c))
r_mean = np.sum(c * r)
delta = np.sqrt(np.sum(c * (1 - r / r_mean) ** 2))

print(round(S_conf, 2), round(r_mean, 3), round(100 *
delta, 2))
```

Αποτέλεσμα (output)

```
13.38 1.267 3.27
```

Οι τύποι

$$\Delta S_{conf} = -R \cdot \Sigma\, c_i \ln c_i$$

εντροπία ανάμειξης (configurational)

$$\delta = \sqrt{\left(\Sigma\, c_i \left(1 - r_i / \bar{r}\right)^2 \right)}$$

διαφορά ατομικού μεγέθους, $\bar{r} = \Sigma\, c_i\, r_i$

ΚΥΡΙΟ ΜΗΝΥΜΑ

Το Σ γίνεται np.sum και η πράξη εφαρμόζεται σε όλα τα στοιχεία μαζί, χωρίς βρόχο for.

ΕΦΑΡΜΟΓΗ ΣΤΑ ΥΛΙΚΑ

ΔS = R·ln5 ≈ 13.4 J/(mol·K), το μέγιστο για 5 στοιχεία. δ ≈ 3.3% < 6%: ευνοείται στερεό διάλυμα. Τέτοιοι περιγραφείς γίνονται τα features μοντέλων για την πρόβλεψη φάσεων.

Mini Exercise NumPy και μετάβαση στο Pandas

1 · NumPy

Ελέγξτε την κατανόησή σας — και εντοπίστε τι λείπει ακόμη

```python
T = np.array([650, 700, 750, 800])
ok = T >= 700
print(ok.sum(), T[ok].mean())
```

Απάντηση: `3 750.0`

Τι μάθαμε: arrays, shape, indexing, διανυσματικές πράξεις, μάσκες, στατιστικά.

ΚΥΡΙΟ ΜΗΝΥΜΑ

Το NumPy είναι ο «αριθμητικός κινητήρας». Όμως τα πραγματικά δεδομένα υλικών έχουν ονόματα, κείμενο και κενά.

Τι ΔΕΝ κάνει καλά ένα NumPy array

Ονόματα στηλών
θυμόμαστε ότι «στήλη 2 = θερμοκρασία»

Μικτούς τύπους
αριθμοί και κείμενο (processing_route) μαζί

Ελλείπουσες τιμές
κενά κελιά από μετρήσεις που δεν έγιναν

Αρχεία CSV / Excel
άμεση ανάγνωση πινάκων δεδομένων

Λύση: Pandas DataFrame →



Pandas
Πίνακες δεδομένων υλικών

Από το NumPy στο Pandas: η δομή Series

2 · Pandas

Series = μονοδιάστατα δεδομένα με ετικέτα (index) για κάθε τιμή

NumPy array
μόνο τιμές

→

Series
τιμές + index

→

DataFrame
πολλές Series = πίνακας

Series: μία «στήλη» με ονόματα γραμμών

index	values
S01	160
S02	190
S03	205

`hv["S02"]` → `190`

Python

```
import pandas as pd

hv = pd.Series([160, 190, 205],
        index=["S01", "S02",
"S03"])
print(hv)
print(hv["S02"])
```

Αποτέλεσμα (output)

```
S01    160
S02    190
S03    205
dtype: int64
190
```

ΣΥΜΒΟΥΛΗ

Αντί για «θέση 1» γράφουμε «δείγμα S02»: ο κώδικας μιλά τη γλώσσα του εργαστηρίου.

Ανατομία ενός DataFrame: γραμμές, στήλες, index

2 · Pandas

Ο πίνακας δεδομένων (DataFrame) οργανώνει τα δεδομένα όπως ένα φύλλο Excel

index	C_wt_pct	Mn_wt_pct	heat_treatment_C	hardness_HV
S01	0.20	1.1	750	160
S02	0.30	1.3	800	190
S03	0.45	0.9	850	230

Γραμμή (row) = ένα δείγμα υλικού
π.χ. το κράμα S02 με όλες τις μετρήσεις του

Στήλη (column) = μία πληροφορία
feature (σύσταση, κατεργασία) ή property (σκληρότητα)

Index = η ετικέτα κάθε γραμμής
μοναδική· εδώ ο κωδικός δείγματος

Python

```
data = {"C_wt_pct": [0.20, 0.30, 0.45],
        "Mn_wt_pct": [1.1, 1.3, 0.9],
        "heat_treatment_C": [750, 800, 850],
        "hardness_HV": [160, 190, 230]}
df = pd.DataFrame(data, index=["S01", "S02", "S03"])
print(df)          # δίνει τον πίνακα που βλέπετε επάνω
```

ΚΥΡΙΟ ΜΗΝΥΜΑ

Κάθε γραμμή ενός DataFrame συνήθως αντιπροσωπεύει ένα δείγμα (sample) και κάθε στήλη ένα χαρακτηριστικό (feature) ή μια ιδιότητα (property).

Φόρτωση δεδομένων με pd.read_csv() και πρώτη ματιά με head()

2 · Pandas

Μία γραμμή κώδικα μετατρέπει το αρχείο σε DataFrame

```
df = pd.read_csv("materials_workshop.csv")
```

1 2 3 4

1 `df` όνομα μεταβλητής που διαλέγουμε (data frame)

2 `pd` το ψευδώνυμο της βιβλιοθήκης pandas

3 `read_csv()` συνάρτηση που διαβάζει αρχεία CSV

4 `"…csv"` το όνομα (διαδρομή) του αρχείου, ως string

Επιστρέφει:
ένα DataFrame με 100 γραμμές × 10 στήλες

Python

```
df.head()     # 5 πρώτες
df.tail(3)    # 3 τελευταίες
```

ΣΥΧΝΟ ΛΑΘΟΣ

FileNotFoundError: το αρχείο δεν είναι στον ίδιο φάκελο με το notebook (ή δεν ανέβηκε στο Colab).

Αποτέλεσμα του df.head() (επιλεγμένες στήλες):

	sample_id	C_wt_pct	Mn_wt_pct	…	processing_route	yield_strength_MPa
0	S001	0.58	1.36	…	normalized	585
1	S002	0.41	1.16	…	quenched_tempered	661
2	S003	0.29	1.09	…	quenched_tempered	638
3	S004	0.34	0.55	…	hot_rolled	395
4	S005	0.53	1.05	…	hot_rolled	569

Επιθεώρηση δομής: shape, columns, info(), dtypes

2 · Pandas

Πόσα δεδομένα έχουμε, ποιες μεταβλητές, ποιοι τύποι και πόσα κενά

Python

```python
print(df.shape)
print(df.columns[:4])    # 4 πρώτα ονόματα
df.info()
```

Αποτέλεσμα (output)

```
(100, 10)
Index(['sample_id', 'C_wt_pct',
       'Mn_wt_pct', 'Cr_wt_pct'], dtype=...)
```

Αποτέλεσμα του df.info() (RangeIndex: 100 entries)

#	Column	Non-Null Count	Dtype
0	sample_id	100 non-null	object
1	C_wt_pct	100 non-null	float64
2	**Mn_wt_pct**	**97 non-null**	**float64**
3	Cr_wt_pct	100 non-null	float64
4	Ni_wt_pct	100 non-null	float64
5	heat_treatment_C	100 non-null	int64
6	**cooling_rate_C_s**	**96 non-null**	**float64**
7	processing_route	100 non-null	object
8	**hardness_HV**	**97 non-null**	**float64**
9	yield_strength_MPa	100 non-null	int64

float64 δεκαδικοί αριθμοί

int64 ακέραιοι αριθμοί

object / str κείμενο (κατηγορίες)

ΚΥΡΙΟ ΜΗΝΥΜΑ

Non-Null Count < 100 σημαίνει ελλείπουσες τιμές: Mn (3), ρυθμός ψύξης (4), σκληρότητα (3).

Επιλογή στηλών

2 · Pandas

Μία στήλη → Series · λίστα στηλών → DataFrame

```
df["hardness_HV"]
```

όνομα στήλης, σε εισαγωγικά, μέσα σε [ ]

Python

```python
hv = df["hardness_HV"]            # Series
print(hv.head(3))
print(df["yield_strength_MPa"].mean())

sub = df[["C_wt_pct", "yield_strength_MPa"]]
print(sub.head(3))
```

Αποτέλεσμα (output)

```
0    232.0
1    255.0
2    253.0
Name: hardness_HV, dtype: float64
491.39
```

Αποτέλεσμα (output)

```
   C_wt_pct  yield_strength_MPa
0      0.58                 585
1      0.41                 661
2      0.29                 638
```

Εναλλακτικά: df.hardness_HV

ΣΥΧΝΟ ΛΑΘΟΣ

`df["C_wt_pct", "yield_strength_MPa"]` → KeyError. Για πολλές στήλες χρειάζονται διπλές αγκύλες [[ ]]: η εσωτερική είναι λίστα.

Επιλογή με ετικέτες και θέσεις: loc και iloc

2 · Pandas

loc → με ονόματα (labels) · iloc → με αριθμούς θέσης (integer location)

df.loc[,] · με index και ονόματα στηλών

```python
df.loc[0, "yield_strength_MPa"]
df.loc[0:2, ["sample_id",
"yield_strength_MPa"]]
```

Αποτέλεσμα (output)

```
585

  sample_id  yield_strength_MPa
0      S001                 585
1      S002                 661
2      S003                 638
```

df.iloc[,] · με αριθμούς θέσης, όπως στο NumPy

```python
df.iloc[0:3, 0:3]
```

	sample_id	C_wt_pct	Mn_wt_pct	Cr_wt_pct
0	S001	0.58	1.36	0.62
1	S002	0.41	1.16	0.09
2	S003	0.29	1.09	1.03
3	S004	0.34	0.55	0.67

ΠΡΟΣΟΧΗ

loc[0:2] περιλαμβάνει και το 2 (3 γραμμές)· iloc[0:3] εξαιρεί το 3 (επίσης 3 γραμμές). Επισημασμένα δεξιά: το αποτέλεσμα του iloc[0:3, 0:3].

Φιλτράρισμα δειγμάτων με λογικές συνθήκες

2 · Pandas

Από τον πλήρη πίνακα στα δείγματα που μας ενδιαφέρουν

1. Αρχικός πίνακας

	sample_id	processing_route	yield_strength_MPa
0	S001	normalized	585
1	S002	quenched_tempered	661
2	S003	quenched_tempered	638
3	S004	hot_rolled	395
4	S005	hot_rolled	569

2. Μάσκα: > 600

mask
False
True
True
False
False

3. Αποτέλεσμα

	sample_id	processing_route	yield_strength_MPa
1	S002	quenched_tempered	661
2	S003	quenched_tempered	638

Python

```
strong = df[df["yield_strength_MPa"] > 600]
print(strong.shape)

qt = df[(df["processing_route"] == "quenched_tempered")
        & (df["yield_strength_MPa"] > 600)]
print(len(qt))
```

Αποτέλεσμα (output)

```
(24, 10)
21
```

& = και · | = ή
παρενθέσεις
υποχρεωτικές

ΠΡΩΤΗ ΠΑΡΑΤΗΡΗΣΗ

21 από τα 24 δείγματα με όριο διαρροής > 600 MPa είναι βαφής-επαναφοράς.

Ελλείπουσες τιμές (missing values): εντοπισμός

2 · Pandas

NaN (Not a Number): το Pandas σημειώνει έτσι κάθε κενό κελί

	sample_id	Mn_wt_pct	cooling_rate_C_s	processing_route	hardness_HV
2	S003	1.09	56.0	quenched_tempered	253.0
3	S004	0.55	**NaN**	hot_rolled	169.0
4	S005	1.05	1.2	hot_rolled	229.0
5	S006	0.35	61.0	quenched_tempered	201.0
6	S007	**NaN**	6.1	hot_rolled	215.0

Τα κόκκινα κελιά είναι NaN: η τιμή δεν υπάρχει στο αρχείο.

Γιατί υπάρχουν κενά σε δεδομένα υλικών;

Μέτρηση που δεν έγινε, αστοχία δοκιμίου, όργανο εκτός λειτουργίας, στοιχείο που δεν καταγράφηκε στο πρωτόκολλο.

Python

```
df.isna().sum()
```

Αποτέλεσμα (output)

```
sample_id            0
C_wt_pct             0
Mn_wt_pct            3
Cr_wt_pct            0
Ni_wt_pct            0
heat_treatment_C     0
cooling_rate_C_s     4
processing_route     0
hardness_HV          3
yield_strength_MPa   0
```

ΠΡΟΣΟΧΗ

Τα περισσότερα μοντέλα ML δεν δέχονται NaN.

Αντιμετώπιση ελλειπουσών τιμών: dropna() και fillna()

2 · Pandas

Διαγραφή γραμμών ή συμπλήρωση με τη διάμεσο (median imputation)

Αρχικά

	sample_id	Mn_wt_pct	cooling_rate_C_s
3	S004	0.55	NaN
4	S005	1.05	1.2
5	S006	0.35	61.0
6	S007	NaN	6.1

dropna(): διαγραφή

	sample_id	Mn_wt_pct	cooling_rate_C_s
~~3~~	~~S004~~	~~0.55~~	~~NaN~~
4	S005	1.05	1.2
5	S006	0.35	61.0
~~6~~	~~S007~~	~~NaN~~	~~6.1~~

fillna(median)

	sample_id	Mn_wt_pct	cooling_rate_C_s
3	S004	0.55	3.2
4	S005	1.05	1.2
5	S006	0.35	61.0
6	S007	1.03	6.1

```python
df_drop = df.dropna()
print(df_drop.shape)

df =
df.fillna(df.median(numeric_only=True))
print(df.isna().sum().sum())
```

Αποτέλεσμα (output)

```
(90, 10)
0
```

ΓΙΑΤΙ ΔΙΑΜΕΣΟΣ

Η διάμεσος δεν επηρεάζεται από ακραίες τιμές (διαφάνεια 12).

ΠΡΟΣΟΧΗ

Το dropna() εδώ πετά 10% των δειγμάτων. Η συμπλήρωση «εφευρίσκει» τιμές: καταγράφουμε πάντα τι κάναμε.

Παράγωγα χαρακτηριστικά (derived features) με φυσική σημασία

2 · Pandas

Νέες στήλες από υπάρχουσες: το ισοδύναμο άνθρακα (carbon equivalent)

$$CE_{IIW} = C + Mn/6 + (Cr + Mo + V)/5 + (Ni + Cu)/15$$ *(σε wt%· εδώ Mo, V, Cu = 0)*

Python

```
df["CE_IIW"] = (df["C_wt_pct"] + df["Mn_wt_pct"] / 6
                + df["Cr_wt_pct"] / 5 + df["Ni_wt_pct"] / 15)

print(df[["sample_id", "C_wt_pct", "Mn_wt_pct",
"CE_IIW"]].head(3))
```

Αποτέλεσμα (output)

	sample_id	C_wt_pct	Mn_wt_pct	CE_IIW
0	S001	0.58	1.36	1.064
1	S002	0.41	1.16	0.659
2	S003	0.29	1.09	0.682

Γιατί παράγωγο feature;

- Συμπυκνώνει φυσική γνώση σε μία στήλη
- Σχετίζεται με εμβαπτότητα και συγκολλησιμότητα
- Μπορεί να βοηθήσει ένα απλό μοντέλο

ΣΥΜΒΟΥΛΗ

Πράξεις μεταξύ στηλών γίνονται ανά γραμμή — όπως οι διανυσματικές πράξεις του NumPy.

Περιγραφική στατιστική όλων των στηλών με describe()

2 · Pandas

Μία εντολή: πλήθος, μέση τιμή, διασπορά και κατανομή κάθε αριθμητικής στήλης

```python
cols = ["C_wt_pct", "cooling_rate_C_s",
        "yield_strength_MPa"]
df[cols].describe().round(2)
```

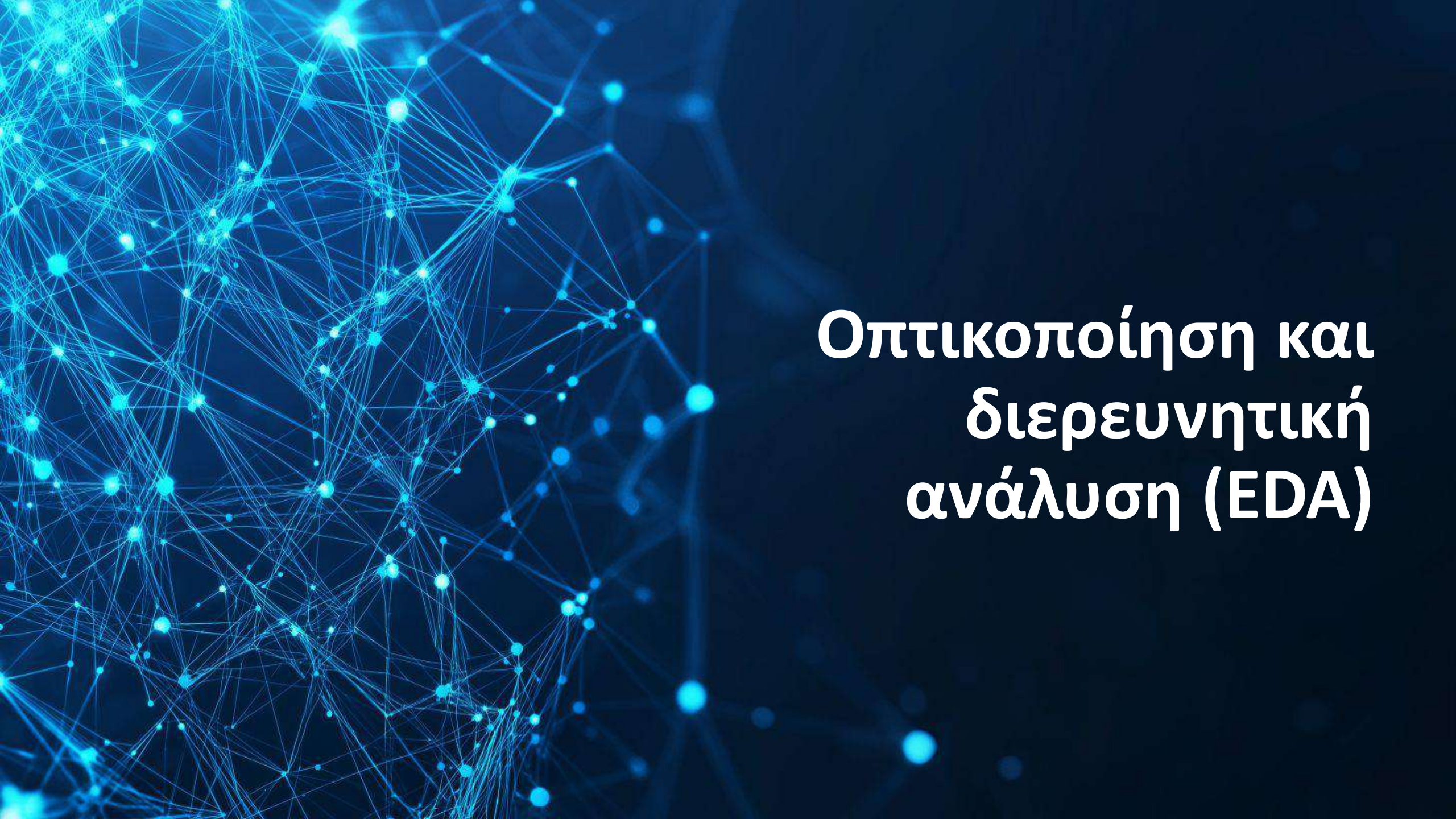
		C_wt_pct	cooling_rate_C_s	yield_strength_MPa
count	πλήθος	100.00	100.00	100.00
mean	μέση τιμή	0.34	**17.13**	491.39
std	τυπ. απόκλιση	0.16	27.66	148.14
min	ελάχιστο	0.05	0.60	201.00
25%	1ο τεταρτημόριο	0.20	1.50	365.00
50%	διάμεσος	0.34	**3.20**	495.00
75%	3ο τεταρτημόριο	0.48	21.50	591.25
max	μέγιστο	0.59	99.10	793.00

count = 100 παντού
δεν απέμειναν κενά μετά τον καθαρισμό

Ρυθμός ψύξης: μέση τιμή 17.1 ≫ διάμεσος 3.2
ασύμμετρη κατανομή: λίγα δείγματα βαφής με πολύ υψηλούς ρυθμούς

Όριο διαρροής 201–793 MPa
εύρος τυπικό για χάλυβες· το target έχει αρκετή διακύμανση

ΚΥΡΙΟ ΜΗΝΥΜΑ
Το describe() μας δείχνει ΤΙ να κοιτάξουμε στα γραφήματα: ασυμμετρίες, ακραίες τιμές, εύρος.

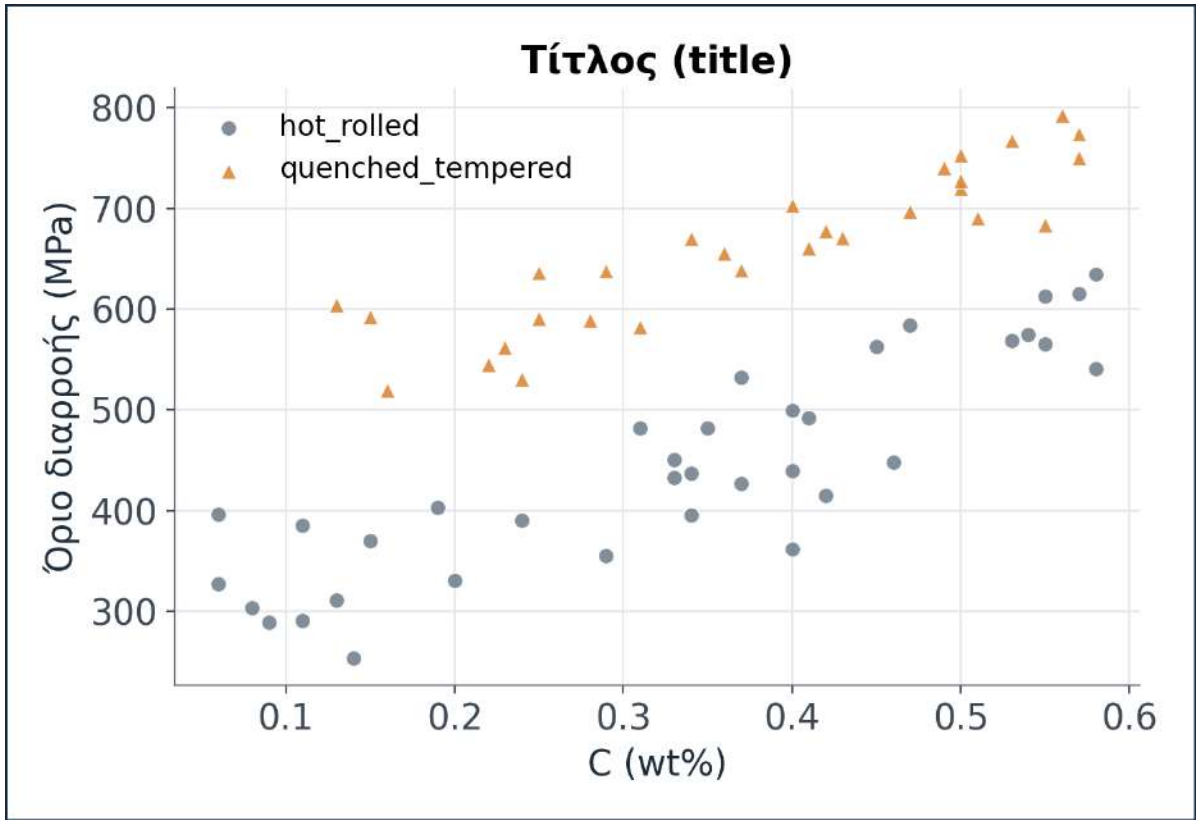


Οπτικοποίηση και διερευνητική ανάλυση (EDA)

Matplotlib: η δομή ενός επιστημονικού γραφήματος

3 · Οπτικοποίηση & EDA

Figure (το «χαρτί») και Axes (το γράφημα με τους άξονές του) —



Ίδια μορφή σε όλα τα γραφήματα του μαθήματος: fig, ax = plt.subplots()

`Figure`	ολόκληρη η εικόνα: μέγεθος, αποθήκευση
`Axes`	το γράφημα: δεδομένα, άξονες, πλέγμα
`xlabel / ylabel`	όνομα μεγέθους και μονάδα
`title`	τι δείχνει το γράφημα
`legend`	ποια σειρά είναι ποια

Python

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(6, 4))
ax.scatter(x_values, y_values)
ax.set_xlabel("C (wt%)")
plt.show()
```

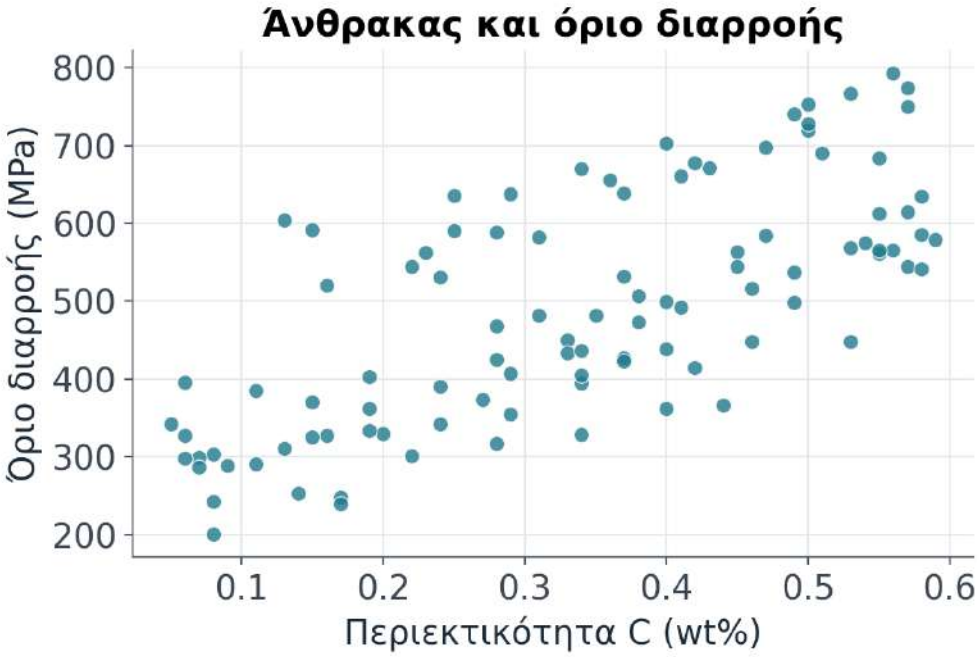
Διάγραμμα διασποράς (scatter plot): σχέση δύο μεταβλητών

3 · Οπτικοποίηση & EDA

Κάθε σημείο είναι ένα δείγμα: x = περιεκτικότητα C, y = όριο διαρροής

Python

```
fig, ax = plt.subplots(figsize=(6, 4))
ax.scatter(df["C_wt_pct"], df["yield_strength_MPa"])
ax.set_xlabel("Περιεκτικότητα C (wt%)")
ax.set_ylabel("Όριο διαρροής (MPa)")
ax.set_title("Άνθρακας και όριο διαρροής")
plt.show()
```



- **Τάση:** περισσότερος άνθρακας → υψηλότερο όριο διαρροής (θετική σχέση).
- **Διασπορά:** για ίδιο C οι τιμές διαφέρουν κατά ~200 MPa — και άλλες μεταβλητές παίζουν ρόλο.
- **Κάθε σημείο** = μία γραμμή του DataFrame.

ΣΥΜΒΟΥΛΗ

Πάντα μονάδες στους άξονες: «(wt%)», «(MPa)».

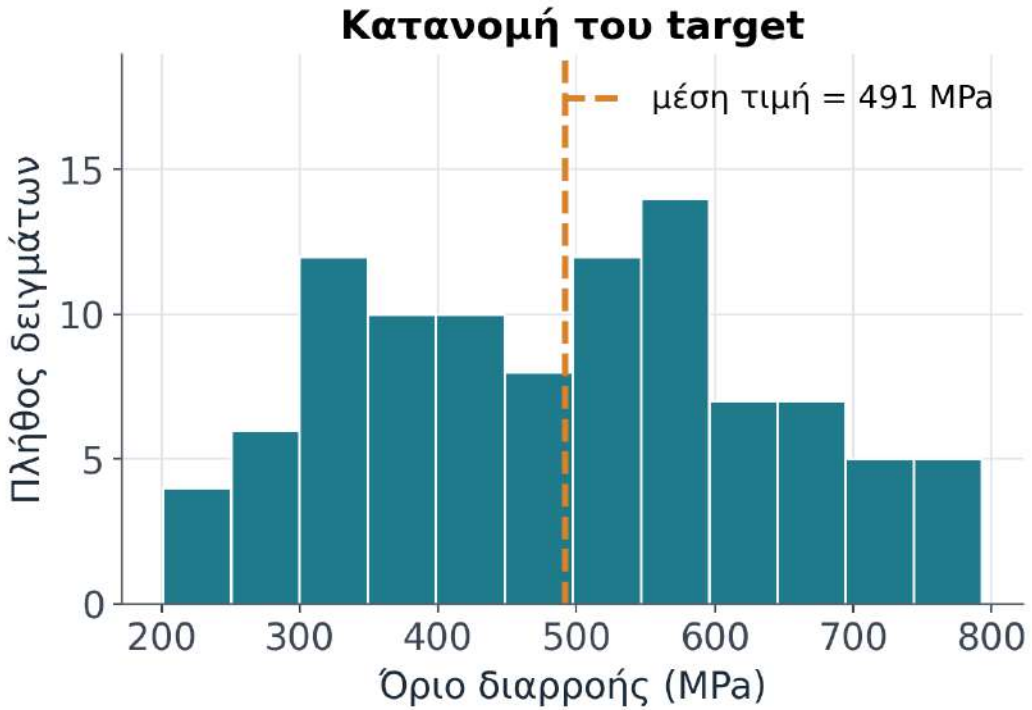
ΕΦΑΡΜΟΓΗ ΣΤΑ ΥΛΙΚΑ

Ο άνθρακας αυξάνει την αντοχή των χαλύβων (στερεό διάλυμα, καρβίδια, μαρτενσίτης). Το γράφημα συμφωνεί — αλλά δεν αποδεικνύει μηχανισμό.

Ιστόγραμμα (histogram): η κατανομή μιας μεταβλητής

3 · Οπτικοποίηση & EDA

Πόσα δείγματα βρίσκονται σε κάθε διάστημα τιμών (bin)




```python
ys = df["yield_strength_MPa"]

fig, ax = plt.subplots(figsize=(6, 4))
ax.hist(ys, bins=12)
ax.axvline(ys.mean(), linestyle="--")
ax.set_xlabel("Όριο διαρροής (MPa)")
ax.set_ylabel("Πλήθος δειγμάτων")
plt.show()
```

- **bins=12:** το εύρος τιμών χωρίζεται σε 12 ίσα διαστήματα.
- **Ύψος στήλης:** πόσα δείγματα πέφτουν σε κάθε διάστημα.
- **axvline:** κατακόρυφη γραμμή — εδώ η μέση τιμή.

ΚΥΡΙΟ ΜΗΝΥΜΑ

Το ιστόγραμμα του target (201–793 MPa) δείχνει εύρος και σχήμα: τι θα προσπαθήσει να προβλέψει το μοντέλο.

Ομάδες σε ένα γράφημα: χρώματα, legend, λογαριθμικός άξονας

3 · Οπτικοποίηση & EDA

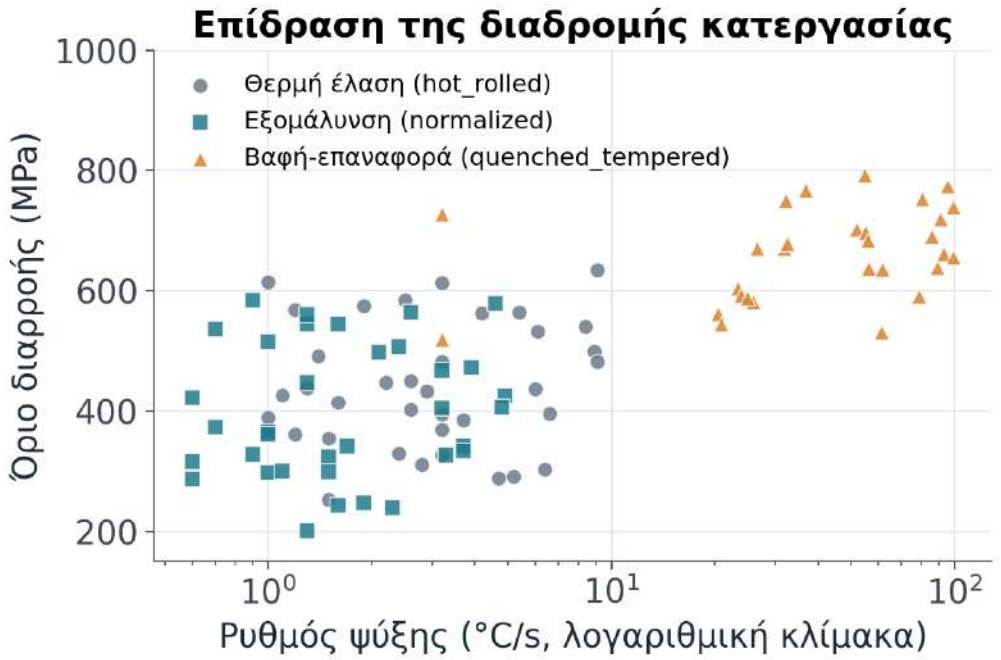
Ρυθμός ψύξης και όριο διαρροής ανά διαδρομή κατεργασίας

Python

```
fig, ax = plt.subplots(figsize=(6, 4))

for route in df["processing_route"].unique():
    part = df[df["processing_route"] == route]
    ax.scatter(part["cooling_rate_C_s"],
               part["yield_strength_MPa"],
label=route)

ax.set_xscale("log")
ax.set_xlabel("Ρυθμός ψύξης (°C/s)")
ax.set_ylabel("Όριο διαρροής (MPa)")
ax.legend()
plt.show()
```



ΣΥΜΒΟΥΛΗ

Επαναχρησιμοποιούμε for + φιλτράρισμα: μία σειρά σημείων ανά κατηγορία, με label για το legend.

ΕΦΑΡΜΟΓΗ ΣΤΑ ΥΛΙΚΑ

Η βαφή-επαναφορά ξεχωρίζει: υψηλός ρυθμός ψύξης και υψηλή αντοχή. Ο log άξονας «απλώνει» τιμές 0.6–100 °C/s.

Η ροή εργασίας της EDA στο dataset μας

3 · Οπτικοποίηση & EDA

Inspect → Clean → Summarize → Visualize → Question → Model

Επιθεώρηση Inspect	Καθαρισμός Clean	Σύνοψη Summarize	Οπτικοποίηση Visualize	Ερωτήματα Question	Μοντέλο Model
Εντολές `head(), shape, info()` **Εύρημα** 100 × 10, μία κατηγορική στήλη	**Εντολές** `isna(), fillna(median)` **Εύρημα** 10 κενά → 0	**Εντολές** `describe()` **Εύρημα** ψύξη: ασύμμετρη κατανομή	**Εντολές** `hist, scatter` **Εύρημα** ο C και η ψύξη σχετίζονται με την αντοχή	**Εντολές** `«Τι επηρεάζει το όριο διαρροής;»` **Εύρημα** επιλογή features	**Εντολές** `LinearRegression` **Εύρημα** επόμενη ενότητα

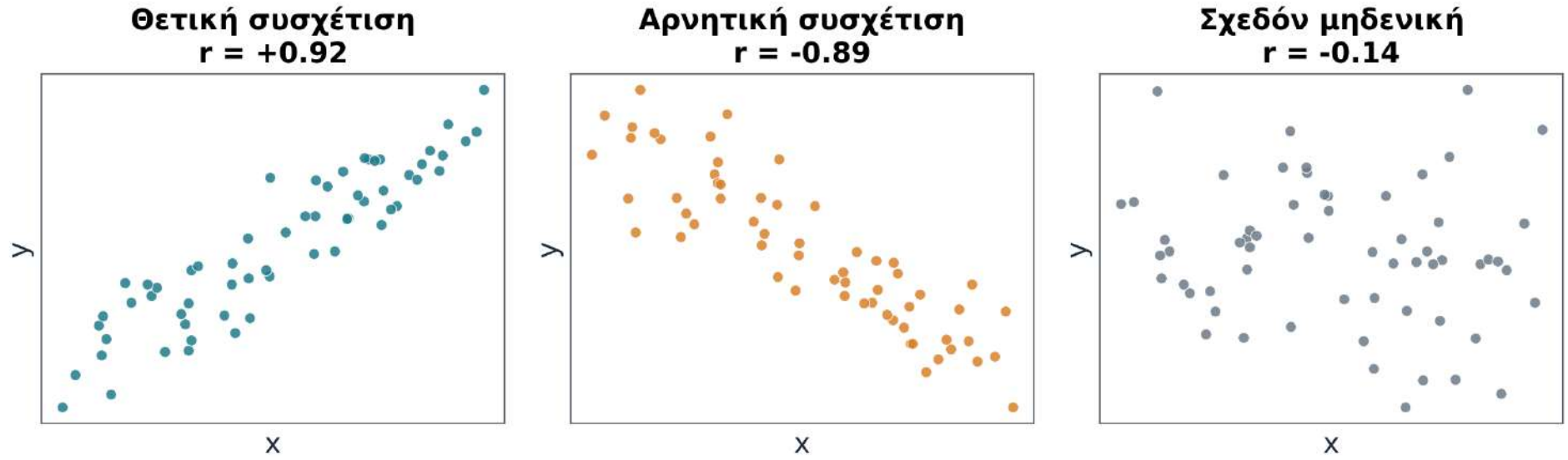
ΣΥΜΒΟΥΛΗ

Η EDA είναι κυκλική: ένα γράφημα γεννά νέο ερώτημα, που μας στέλνει πίσω σε καθαρισμό ή σε νέο γράφημα.

Συσχέτιση (correlation): πόσο «μαζί» μεταβάλλονται δύο μεγέθη

3 · Οπτικοποίηση & EDA

Ο συντελεστής Pearson r παίρνει τιμές από −1 έως +1



r > 0: θετική
όταν το x αυξάνεται, το y τείνει να αυξάνεται.
π.χ. C → όριο διαρροής

r < 0: αρνητική
όταν το x αυξάνεται, το y τείνει να μειώνεται.
π.χ. θερμοκρασία επαναφοράς → σκληρότητα

r ≈ 0: καμία γραμμική
δεν φαίνεται γραμμική σχέση — μπορεί όμως να υπάρχει μη γραμμική

ΠΡΟΣΟΧΗ
Το r μετρά μόνο γραμμικές σχέσεις και επηρεάζεται από ακραίες τιμές. Κοιτάμε πάντα και το scatter plot.

Πίνακας συσχετίσεων (correlation matrix) και ερμηνεία του

3 · Οπτικοποίηση & EDA

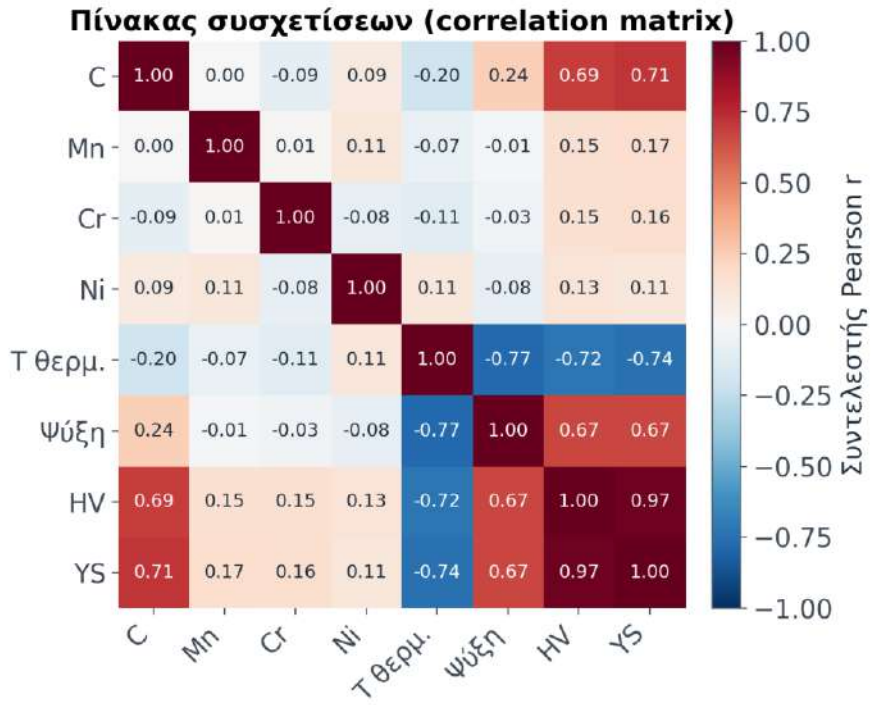
df.corr() υπολογίζει το r για κάθε ζεύγος αριθμητικών στηλών

```python
num = df.select_dtypes("number")    # μόνο αριθμοί
corr = num.corr()
corr["yield_strength_MPa"].sort_values()
```

Αποτέλεσμα (output)

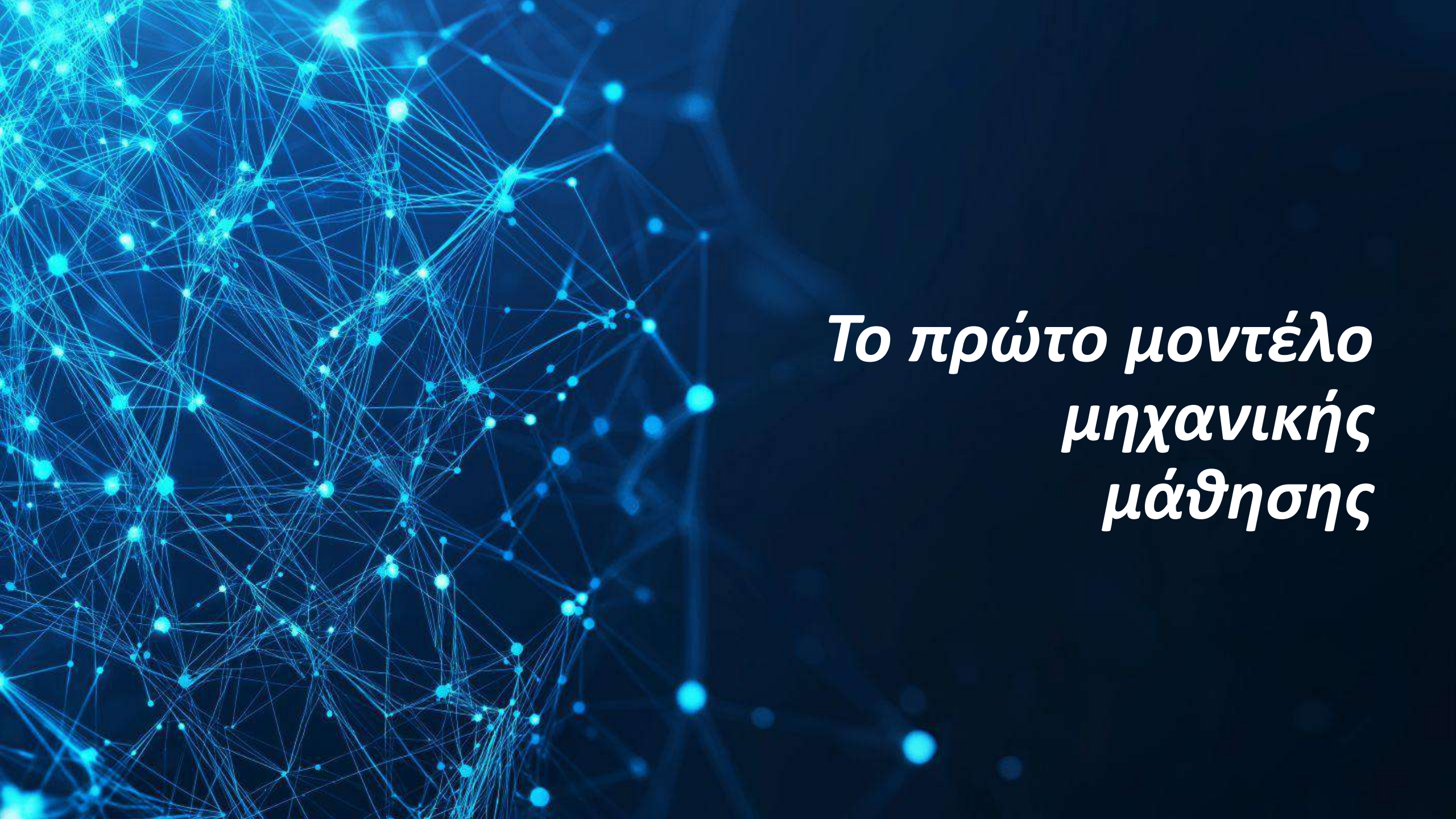
```
heat_treatment_C     -0.74
Ni_wt_pct             0.11
Cr_wt_pct             0.16
Mn_wt_pct             0.17
cooling_rate_C_s      0.67
C_wt_pct              0.71
hardness_HV           0.97
```

Ισχυρότερες σχέσεις με το target:
HV (+0.97), C (+0.71), ψύξη (+0.67), θερμοκρασία (−0.74)



ΠΡΟΣΟΧΗ

Η συσχέτιση δεν συνεπάγεται αιτιότητα (correlation does not imply causation). Π.χ. η θερμοκρασία έχει r = −0.74 κυρίως επειδή τα δείγματα βαφής-επαναφοράς έχουν χαμηλή heat_treatment_C ΚΑΙ υψηλό ρυθμό ψύξης.


Το πρώτο μοντέλο
μηχανικής
μάθησης

Από τα δεδομένα στην πρόβλεψη: features (X) και target (y)

4 · Machine Learning

Εποπτευόμενη μάθηση (supervised learning): μαθαίνουμε από δείγματα με γνωστή απάντηση

Σύσταση
C, Mn, Cr, Ni

Συνθήκες κατεργασίας
ρυθμός ψύξης

Θερμική κατεργασία
heat_treatment_C

Είσοδοι · features · X

Μοντέλο ML

Target · y
`yield_strength_MPa`

Αποτέλεσμα (output)

```
(100, 6) (100,)
```

Python

```
features = ["C_wt_pct", "Mn_wt_pct", "Cr_wt_pct",
            "Ni_wt_pct", "heat_treatment_C", "cooling_rate_C_s"]
X = df[features]                    # πίνακας εισόδων (DataFrame)
y = df["yield_strength_MPa"]        # στόχος (Series)
print(X.shape, y.shape)
```

ΠΡΟΣΟΧΗ

Γιατί όχι hardness_HV; Είναι κι αυτή μετρημένη ιδιότητα (r = 0.97). Θέλουμε να προβλέψουμε την αντοχή από σύσταση και κατεργασία, πριν από τις δοκιμές. Η processing_route (κείμενο) απαιτεί κωδικοποίηση — μελλοντικό θέμα.

Διαχωρισμός σε σύνολο εκπαίδευσης και σύνολο ελέγχου

4 · Machine Learning

Training set για να μάθει το μοντέλο · test set για να το αξιολογήσουμε σε δείγματα που δεν έχει δει



80 δείγματα: σύνολο εκπαίδευσης (training set)

20: σύνολο ελέγχου (test set)

κάθε κουτί ≈ 5 δείγματα · η επιλογή γίνεται τυχαία

```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

print(X_train.shape, X_test.shape)
```

Αποτέλεσμα (output)

```
(80, 6) (20, 6)
```

`X_train, y_train`
features και target για εκπαίδευση

`X_test, y_test`
κρατιούνται «κρυφά» για τον έλεγχο

`test_size=0.2`
20% των δειγμάτων στο test set

`random_state=42`
ίδιος χωρισμός σε κάθε εκτέλεση

Δημιουργία, εκπαίδευση και πρόβλεψη: fit() και predict()

4 · Machine Learning

Το ίδιο μοτίβο τριών βημάτων ισχύει σχεδόν για κάθε μοντέλο του scikit-learn



Python

```python
from sklearn.linear_model import LinearRegression

model = LinearRegression()          # 1. δημιουργία
model.fit(X_train, y_train)         # 2. εκπαίδευση
y_pred = model.predict(X_test)      # 3. πρόβλεψη

print(y_pred[:3].round(1))
```

Αποτέλεσμα (output)

```
[447.2 454.4 598.7]
```

`model`	ο εκτιμητής (estimator): ένα «κενό» μοντέλο
`fit()`	εκπαίδευση: βρίσκει τους συντελεστές
`X_train`	features των δειγμάτων εκπαίδευσης
`y_train`	οι γνωστές τιμές-στόχοι
`predict()`	χρησιμοποιεί ΜΟΝΟ το X_test

Τι έμαθε το μοντέλο (συντελεστές):

$\hat{y} = b_0 + b_1 \cdot C + b_2 \cdot Mn + \ldots + b_6 \cdot \psi\acute{\upsilon}\xi\eta$

C: +540.96 MPa ανά 1 wt% · θερμ.: -0.52 MPa/°C

Αξιολόγηση με το μέσο απόλυτο σφάλμα (Mean Absolute Error, MAE)

4 · Machine Learning

Πόσο απέχουν κατά μέσο όρο οι προβλέψεις από τις μετρημένες τιμές

Δείγμα test	Μετρημένο (MPa)	Προβλεπόμενο (MPa)	|Σφάλμα| (MPa)
1	437	447.2	**10.2**
2	366	454.4	**88.4**
3	582	598.7	**16.7**
4	793	805.4	**12.4**
5	317	331.8	**14.8**
…	…	…	…

MAE = μέση τιμή της τελευταίας στήλης για τα 20 δείγματα του test set

```python
from sklearn.metrics import mean_absolute_error

mae = mean_absolute_error(y_test, y_pred)
print(f"MAE = {mae:.1f} MPa")
```

29.2 MPa

MAE του μοντέλου · περίπου 6% της μέσης τιμής (491 MPa)

ΚΥΡΙΟ ΜΗΝΥΜΑ (TAKEAWAY)

Αν MAE = 29.2 MPa, οι προβλέψεις απέχουν κατά μέσο όρο περίπου 29.2 MPa από τις πραγματικές τιμές. Το MAE έχει τις ίδιες μονάδες με το target.

$$MAE = (1/n) \cdot \Sigma \, |y_i - \hat{y}_i|$$

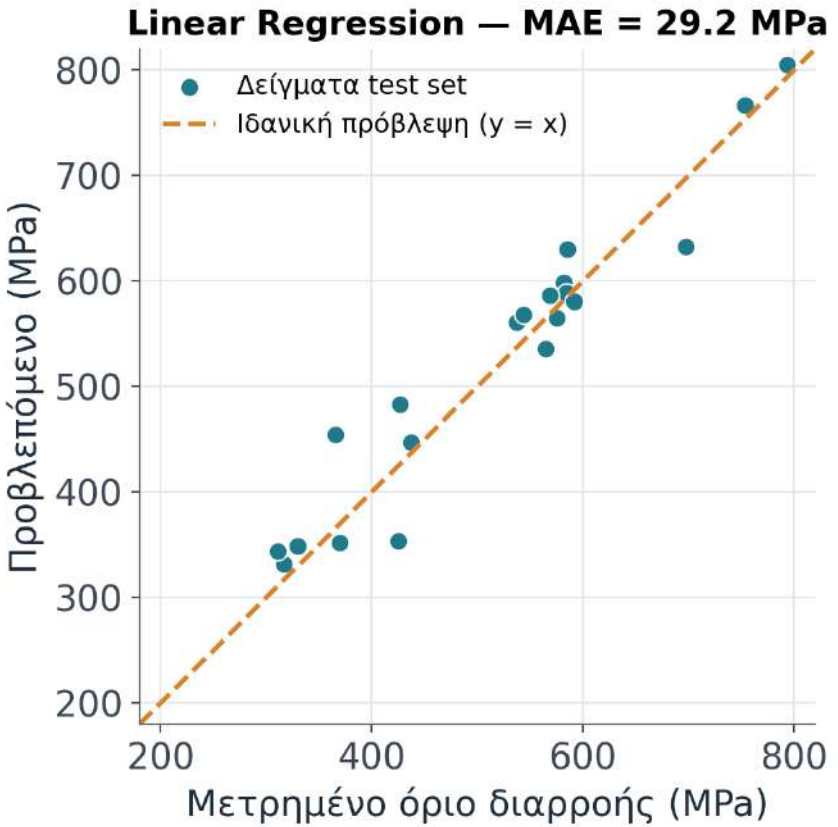
Διάγραμμα προβλεπόμενων έναντι μετρημένων τιμών

4 · Machine Learning

Predicted-vs-measured: η βασική οπτική αξιολόγηση ενός μοντέλου παλινδρόμησης

```python
fig, ax = plt.subplots(figsize=(5, 5))
ax.scatter(y_test, y_pred)
ax.plot([200, 800], [200, 800], "--")   # y = x
ax.set_xlabel("Μετρημένο όριο διαρροής (MPa)")
ax.set_ylabel("Προβλεπόμενο (MPa)")
plt.show()
```

- **Πάνω στη διαγώνιο:** τέλεια πρόβλεψη.
- **Πάνω από τη διαγώνιο:** το μοντέλο υπερεκτιμά· κάτω: υποεκτιμά.
- **Απόσταση από τη διαγώνιο:** το σφάλμα κάθε δείγματος.
- **Συστηματική καμπύλωση:** ένδειξη ότι λείπει μη γραμμική σχέση.



ΠΕΡΙΟΡΙΣΜΟΙ

Μόνο 20 δείγματα ελέγχου, συνθετικά δεδομένα, γραμμικό μοντέλο: το αποτέλεσμα είναι πρώτη εκτίμηση, όχι εργαλείο σχεδιασμού.

Ένα δεύτερο μοντέλο: δέντρο απόφασης

4 · Machine Learning

Αλλάζουμε μία γραμμή κώδικα, και βλέπουμε στην πράξη την υπερπροσαρμογή της Διάλεξης 1

Python

```python
from sklearn.tree import DecisionTreeRegressor

tree = DecisionTreeRegressor(random_state=42)
tree.fit(X_train, y_train)                    # ίδιο API

pred_train = tree.predict(X_train)
pred_test = tree.predict(X_test)
print(round(mean_absolute_error(y_train, pred_train), 1),
      round(mean_absolute_error(y_test, pred_test), 1))
print(tree.get_n_leaves(), "φύλλα")          # για 80 δείγματα
```

Αποτέλεσμα (output)

```
0.0 46.3
79 φύλλα
```

MAE (MPa): ίδιο split με πριν, test_size=0.2, random_state=42

Μοντέλο	Εκπαίδευση	Έλεγχος
Γραμμική παλινδρόμηση	26.4	**29.2**
Δέντρο, χωρίς όριο βάθους	**0.0**	**46.3**
Δέντρο, max_depth=4	22.5	41.5

ΚΥΡΙΟ ΜΗΝΥΜΑ

Τέλειο στην εκπαίδευση, χειρότερο στον έλεγχο: το δέντρο απομνημόνευσε τα 80 δείγματα. Και σε πραγματικά δεδομένα: random forest για τη διάβρωση χαλύβων, R^2 0.94 στην εκπαίδευση, 0.73 στον έλεγχο.

ΠΡΟΣΟΧΗ

Εδώ κερδίζει το γραμμικό μοντέλο, γιατί το συνθετικό dataset φτιάχτηκε με σχεδόν γραμμικές σχέσεις.

Καθοδηγούμενη εργαστηριακή άσκηση (Guided Laboratory Exercise)

Εργαστήριο & Homework

Ολόκληρη η ροή σε ένα notebook

Checkpoint 1
Data loaded

- Φόρτωση materials_workshop.csv
- head()
- shape
- info()

Checkpoint 2
Data cleaned

- Εντοπισμός missing values
- Συμπλήρωση με διάμεσο
- describe()
- Filtering

Checkpoint 3
EDA completed

- Histogram του target
- Δύο scatter plots
- Correlations
- Correlation matrix

Checkpoint 4
First ML model

- Ορισμός X και y
- train/test split
- LinearRegression, predict
- MAE, predicted-vs-measured

ΣΥΜΒΟΥΛΗ

Σε κάθε checkpoint σηκώνουμε το χέρι. Κολλήσατε; Διαβάστε την τελευταία γραμμή του σφάλματος, ελέγξτε ονόματα στηλών και σειρά κελιών.

Checkpoints 1 και 2 — φόρτωση και καθαρισμός

Εργαστήριο & Homework

Οδηγός κώδικα· γράψτε τον στο δικό σας notebook

Checkpoint 1 · Data loaded

Python

```python
import pandas as pd
import matplotlib.pyplot as plt

df =
pd.read_csv("materials_workshop.csv")
print(df.head())
print(df.shape)
df.info()
```

✓ Αναμενόμενα αποτελέσματα

(100, 10) και 10 στήλες στο info()

97 / 96 / 97 non-null σε Mn, ψύξη, HV

Checkpoint 2 · Data cleaned

Python

```python
print(df.isna().sum())

df = df.fillna(df.median(numeric_only=True))
print(df.isna().sum().sum())

print(df.describe().round(2))

route = df["processing_route"]
qt = df[route == "quenched_tempered"]
print(len(qt))
```

✓ Αναμενόμενα αποτελέσματα

0 ελλείπουσες μετά τη συμπλήρωση

29 δείγματα quenched_tempered

Εργαστήριο: Checkpoints 3 και 4 — EDA και πρώτο μοντέλο

Εργαστήριο & Homework

Τα γραφήματα scatter των διαφανειών 31 και 33 συμπληρώνουν το Checkpoint 3

Checkpoint 3 · EDA completed

Python

```
fig, ax = plt.subplots()
ax.hist(df["yield_strength_MPa"], bins=12)
ax.set_xlabel("Όριο διαρροής (MPa)")
plt.show()

corr = df.select_dtypes("number").corr()
print(corr["yield_strength_MPa"].round(2))

fig, ax = plt.subplots(figsize=(7, 6))
im = ax.imshow(corr, cmap="RdBu_r", vmin=-1,
vmax=1)
ax.set_xticks(range(len(corr)), corr.columns,
              rotation=90)
ax.set_yticks(range(len(corr)), corr.columns)
fig.colorbar(im)
plt.show()
```

Checkpoint 4 · First ML model

Python

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

features = ["C_wt_pct", "Mn_wt_pct", "Cr_wt_pct",
            "Ni_wt_pct", "heat_treatment_C",
            "cooling_rate_C_s"]
X = df[features]
y = df["yield_strength_MPa"]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(mean_absolute_error(y_test, y_pred))
```

✓ MAE ≈ 29.2 MPa + predicted-vs-measured (διαφάνεια 42)

Τι μάθαμε σήμερα και τα πιο συχνά λάθη

Σύνοψη

Στο τέλος του μαθήματος ο/η φοιτητής/τρια μπορεί να…

NumPy

- φτιάχνει arrays και ελέγχει το shape
- κάνει πράξεις σε όλη τη στήλη
- φιλτράρει με μάσκες

Pandas

- φορτώνει CSV σε DataFrame
- επιλέγει με [], loc, iloc
- εντοπίζει και συμπληρώνει κενά

Γραφήματα & EDA

- φτιάχνει histogram και scatter
- βάζει άξονες με μονάδες
- ερμηνεύει συσχετίσεις με κριτική

Machine Learning

- ορίζει X και y
- χωρίζει σε train και test
- εκπαιδεύει, προβλέπει, αξιολογεί με MAE

ΤΑ 6 ΠΙΟ ΣΥΧΝΑ ΛΑΘΗ

df["a", "b"] αντί για df[["a", "b"]] · and αντί για & στις μάσκες · NaN που έμειναν πριν το fit() · αξιολόγηση στο X_train αντί για το X_test · συσχέτιση που ερμηνεύεται ως αιτιότητα · γράφημα χωρίς μονάδες

Homework 2 — Πρόβλεψη σκληρότητας: δύο μοντέλα, μία σύγκριση

Εργαστήριο & Homework

Core: υποχρεωτικές ασκήσεις · παράδοση ως notebook (.ipynb) που τρέχει με Restart & Run All

1. Δεδομένα

- Φόρτωση του materials_workshop.csv
- Πλήθος δειγμάτων και πλήθος features
- Αριθμητικές και κατηγορικές στήλες
- Εντοπισμός και αντιμετώπιση missing values

2. EDA

- Περιγραφική στατιστική (describe)
- Histogram της σκληρότητας (νέο target)
- Τουλάχιστον δύο scatter plots
- Correlation matrix

3. Μοντέλο

- Target: hardness_HV · features χωρίς το όριο διαρροής
- Train/test split (random_state=42)
- Linear Regression και δέντρο απόφασης
- MAE εκπαίδευσης/ελέγχου, predicted-vs-measured

4. ΕΡΜΗΝΕΙΑ

Σύντομη επιστημονική ερμηνεία (150–250 λέξεις): ποιες μεταβλητές σχετίζονται με τη σκληρότητα; Ποιο μοντέλο γενικεύει καλύτερα; Είναι οι σχέσεις φυσικά εύλογες; Τι σημαίνει το MAE σας στην πράξη; Ποιοι είναι οι περιορισμοί;

ΚΡΙΤΗΡΙΑ

Κάθε γράφημα με ετικέτες και μονάδες. Σχόλια (#) στον κώδικα. Τα αποτελέσματα πρέπει να αναπαράγονται.

Εξάσκηση στο σπίτι: 12 ασκήσεις για να συμπληρώσετε

Εργαστήριο & Homework

Στο notebook του μαθήματος, ενότητα 7 · σκανάρετε τον κωδικό για να το ανοίξετε

https://tinyurl.com/bd59mk5b

Ασκήσεις	Θέμα	Πλήθος
7.1–7.3	NumPy: arrays, πράξεις, μάσκες	**3**
7.4–7.7	Pandas: επιλογή, φίλτρα, κενά, νέα στήλη	**4**
7.8–7.9	Γραφήματα: histogram και scatter	**2**
7.10–7.12	Machine Learning: X, y, split, fit, MAE	**3**

Πώς δουλεύετε

1. ανοίγετε το notebook: απευθείας στο Colab ή τοπικά στο JupyterLab (αρχείο .ipynb)
2. αντικαθιστάτε τα ... με τον δικό σας κώδικα (οδηγός: οι γραμμές με ✏️)
3. τρέχετε το κελί και συγκρίνετε με το «Αναμενόμενο αποτέλεσμα»
4. κολλήσατε; οι λύσεις είναι στο τέλος του notebook

ΣΥΜΒΟΥΛΗ

Το dataset είναι ενσωματωμένο στο notebook: τρέχει όπως είναι, χωρίς να ανεβάσετε αρχείο. Κάντε τις ασκήσεις **πριν από το Homework 2**.

Επόμενα βήματα και πηγές

Σύνοψη

Τι δεν κάναμε ακόμη, και πού θα το βρούμε

Cross-validation

ένα μόνο split δίνει «θορυβώδη» εκτίμηση· με k-fold αξιολογούμε πιο αξιόπιστα

Ρύθμιση υπερπαραμέτρων

π.χ. το max_depth του δέντρου, επιλεγμένο συστηματικά και όχι με το μάτι

Κατηγορικές μεταβλητές

η processing_route είναι κείμενο· χρειάζεται κωδικοποίηση για να μπει στο μοντέλο

Ολοκληρωμένη ροή

προεπεξεργασία, μοντέλο και αξιολόγηση σε ένα ενιαίο, αναπαραγώγιμο pipeline

ΚΥΡΙΟ ΜΗΝΥΜΑ

Όλα αυτά τα βλέπουμε στο Μάθημα 11: ολοκληρωμένη ροή μηχανικής μάθησης, από τα δεδομένα στο αξιόπιστο μοντέλο.

Βιβλιογραφία και τεκμηρίωση

- Z. “L.” Liu, Artificial Intelligence for Engineers, Κεφ. 2 (Springer, 2025)
- Y. Cheng, T. Wang, G. Zhang (eds.), Artificial Intelligence for Materials Science, Κεφ. 1–2 (Springer, 2021)
- M. A. Mellal (ed.), Artificial Intelligence in Material Science: Advances, Κεφ. 10–11 (CRC Press, 2024)
- Pavlina & Van Tyne, J. Mater. Eng. Perform. 17 (2008) 888–893: σκληρότητα και όριο διαρροής
- numpy.org · pandas.pydata.org · matplotlib.org · scikit-learn.org

Ερωτήσεις;