Best practices in AI (for environmental data) and manual feature extraction

Artificial Intelligence and Data Mining Methods in Ecology

University of Tuscia – Viterbo, July 21–25, 2025

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 "Education and Research" - Component 2: "From research to business" - Investment 3.1: "Fund for the realisation of an integrated system of research and innovation infrastructures"

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

Where We Left Off

Why AI in Environmental Sciences?

- We face urgent, complex environmental challenges
- We're collecting more data than ever, from sensors, satellites, field work...
- AI/ML can help us:
  - Distill raw data
  - Predict future outcomes
  - Optimize decisions
  - Accelerating scientific discovery
  - Approximate simulations

BUT...

- AI is not a silver bullet
- It needs good data, critical thinking, and clear goals

Where We’re Going Today

How to use AI responsibly and effectively

- 📊 **Best practices** for setting up ML workflows
  → train/test splits, reproducibility, tuning, etc.
- 🧪 How to split data, tune models, and avoid common pitfalls
- 🧼 How to **spot clean (or sketchy) datasets**
- ✅ How to **share your own experiments** with others effectively
- 🧠 Intro to **hand-crafted feature engineering**

📌 We're moving from "Why AI?" → "How to use it properly"

By the end of today, we should be able to…

✅ Understand why reproducibility and good data splits matter

✅ Know how to make your experiments reproducible

✅ Use tools like grid search and random search for tuning

✅ Spot problems in datasets and know how to fix or avoid them

✅ Appreciate (or fear) the chaos of real-world data

✅ Understand what feature engineering is, and why it matters

✅ Describe common feature engineering methods

Day 1

Why AI for Environment?

Day 2

Best Practices

Best practices in AI

Why should we care about “best practices”?

Making your ML workflow reproducible and trustworthy

- ML isn’t just about training a model
  - Getting **reproducible** results
  - Avoiding **accidental cheating** (e.g., data leakage 😱)
  - Making models that **generalize**

- In science, your model is just another experiment → **treat it like one**!

Splitting your data

It is best practice to always split your data in three sets:

- **Training** set
- **Validation** set
- **Test** set

Splitting your data

It is best practice to always split your data in three sets:

- **Training** set
  - used to train and make the model learn the hidden features/patterns in the data
  - should have a diversified set of inputs that reflects past and future scenarios
  - should have a good distribution of examples for each “output”

- **Validation** set
- **Test** set

Splitting your data

It is best practice to always split your data in three sets:

- **Training** set
- **Validation** set
  - separate from the training set
  - used to validate our model performance during training
  - used to tune the model hyperparameters
  - used like a critic telling us whether the training is moving in the right direction or not
  - after every training “step” or “epoch”, use validation set to evaluate the model

- **Test** set

Splitting your data

It is best practice to always split your data in three sets:

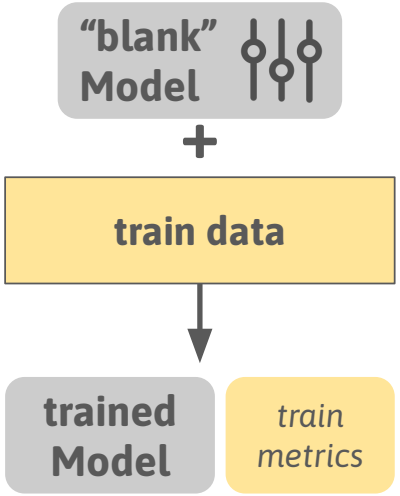
- **Training** set
- **Validation** set
- **Test** set
  - completely separate from the other two sets
  - used to test the model after **completing** the training
  - used to test the model on **new unseen data**

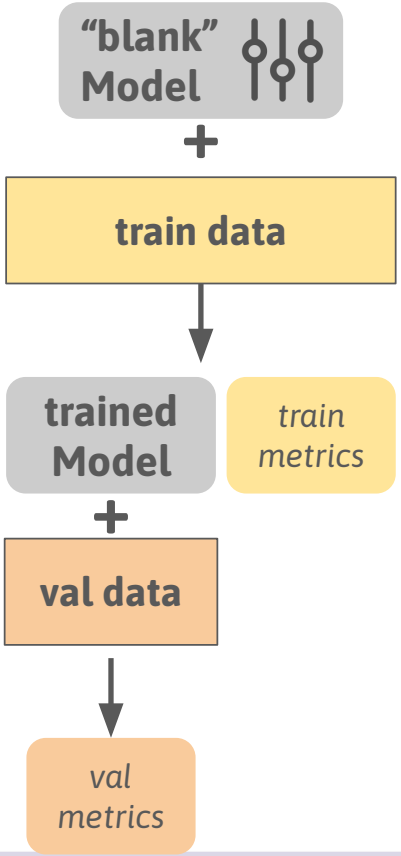
Splitting your data

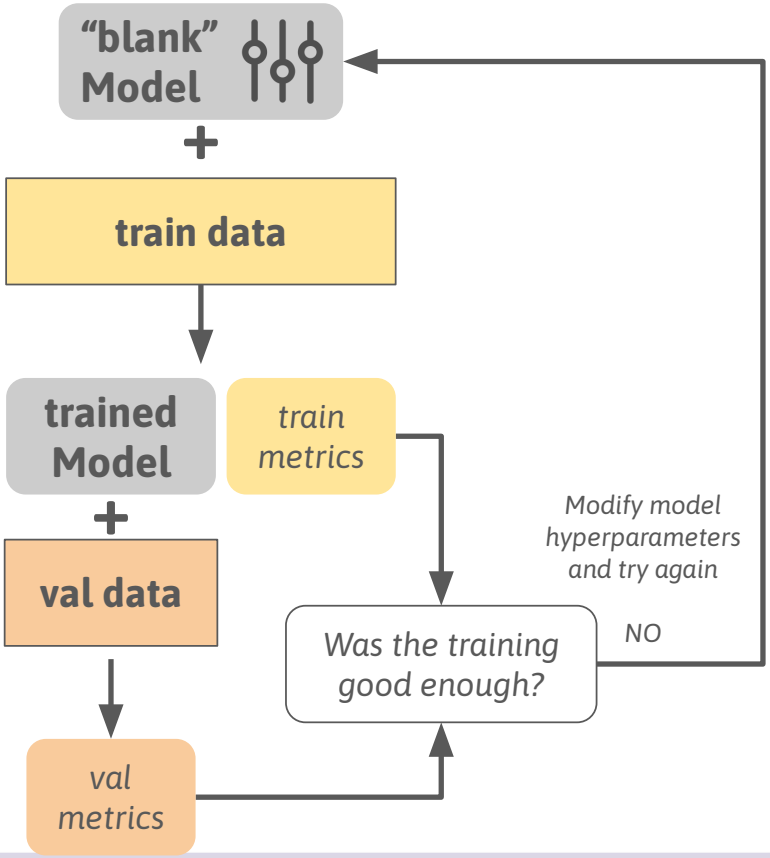
It is best practice to always split your data in three sets:

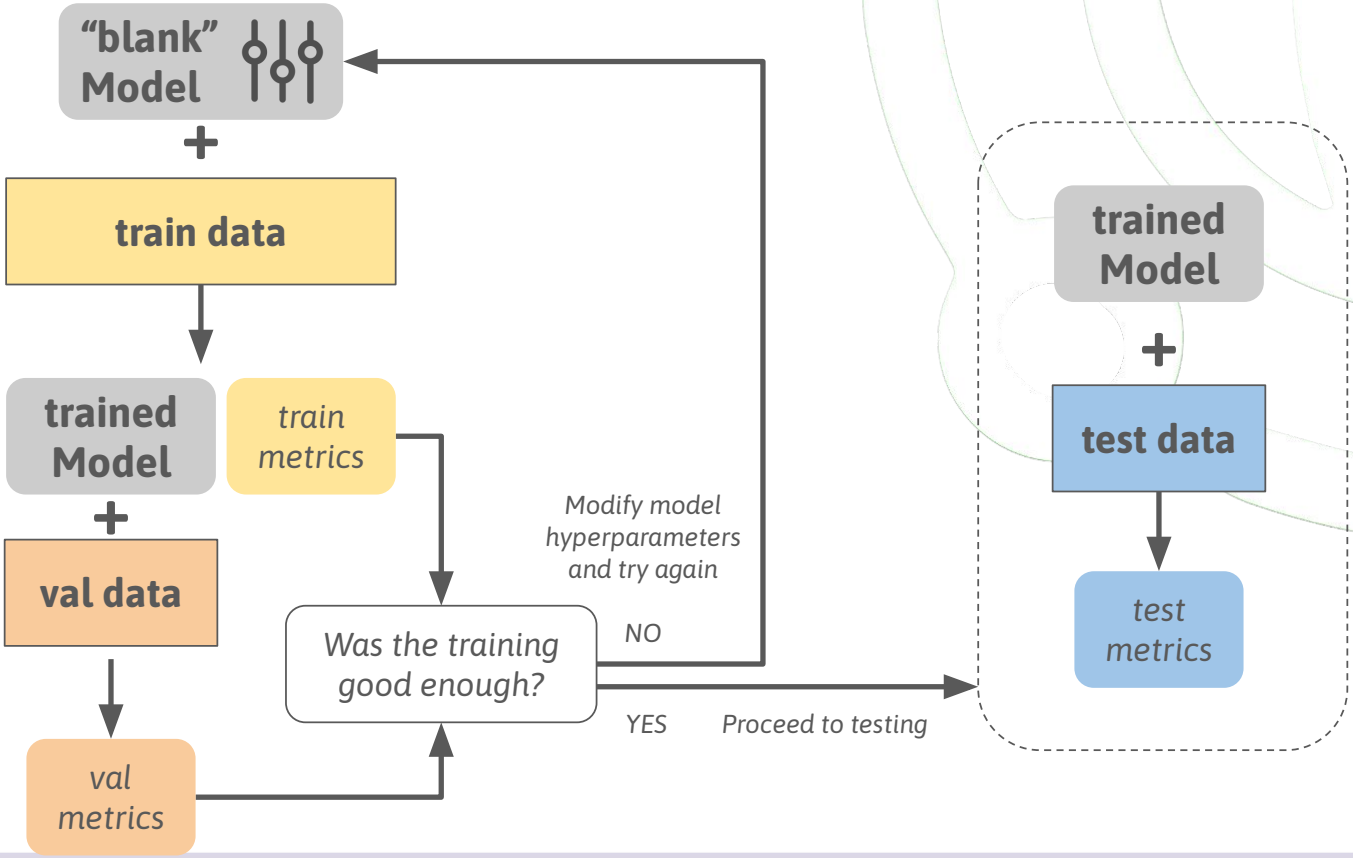
- **Training** set: used to fit the model
- **Validation** set: used to tune hyperparameters
- **Test** set: used only once at the end to evaluate final performance

❌ No peeking at the test set during training!


"blank" Model
+
train data
trained Model
train metrics


"blank" Model
train data
trained Model
train metrics
val data
val metrics


"blank" Model
+
train data
trained Model
train metrics
+
val data
Was the training good enough?
Modify model hyperparameters and try again
NO
val metrics


"blank" Model
train data
trained Model
train metrics
val data
val metrics
Was the training good enough?
Modify model hyperparameters and try again
NO
YES
Proceed to testing
trained Model
test data
test metrics

Splitting your data - how much data?

- How much data should we “sacrifice” to create the validation and test sets?

Splitting your data - how much data?

- How much data should we “sacrifice” to create the validation and test sets?

- Is 60% of the data enough to train the model?

Splitting your data - how much data?

- How much data should we “sacrifice” to create the validation and test sets?

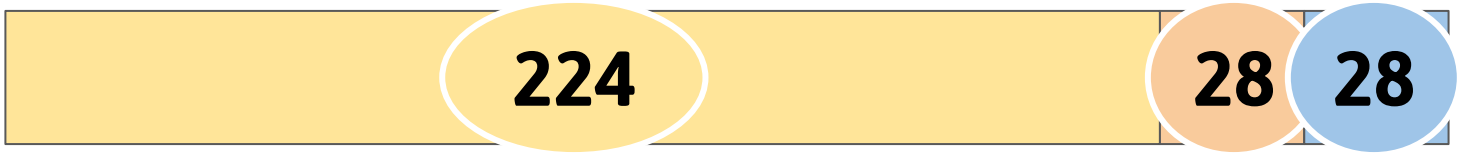
- Is 60% of the data enough to train the model?

- What about 80% then? But maybe 10% of the data is not significant enough to test the model…

Splitting your data - how much data?

- How much data should we “sacrifice” to create the validation and test sets?

- Is 60% of the data enough to train the model?



- What about 80% then? But maybe 10% of the data is not significant enough to test the model…

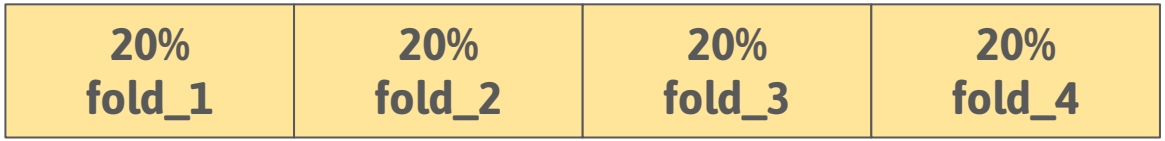
Splitting your data - Cross Validation!

- When there is not enough data to create 3 “regular” splits
- Set aside a large-enough test set

Splitting your data - Cross Validation!

- When there is not enough data to split “regularly”
- Set aside a large-enough test set

- Split the training set into several “folds” or “splits”



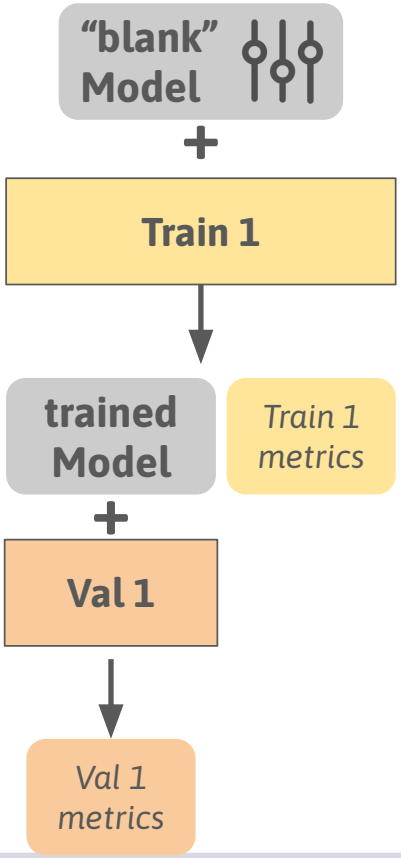
Splitting your data - Cross Validation!

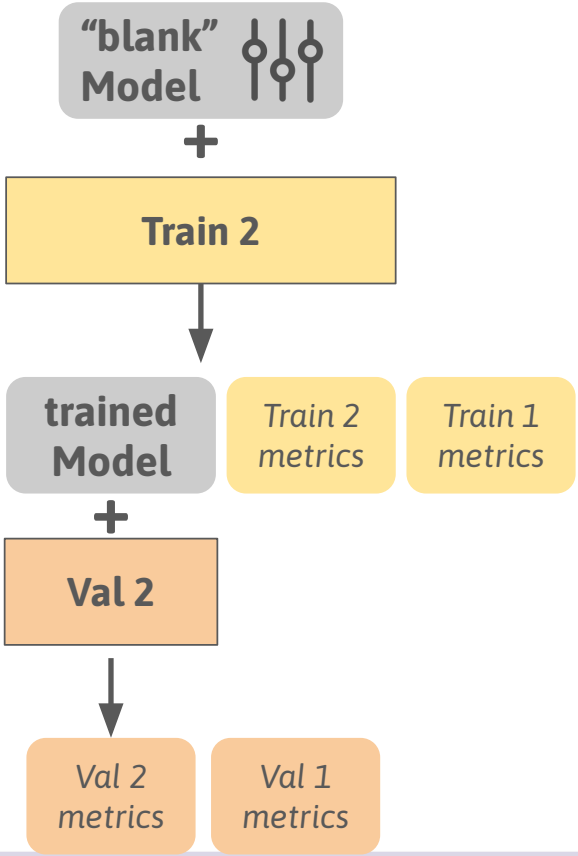
- When there is not enough data to split “regularly”
- Set aside a large-enough test set
- Split the training set into several “folds” or “splits”

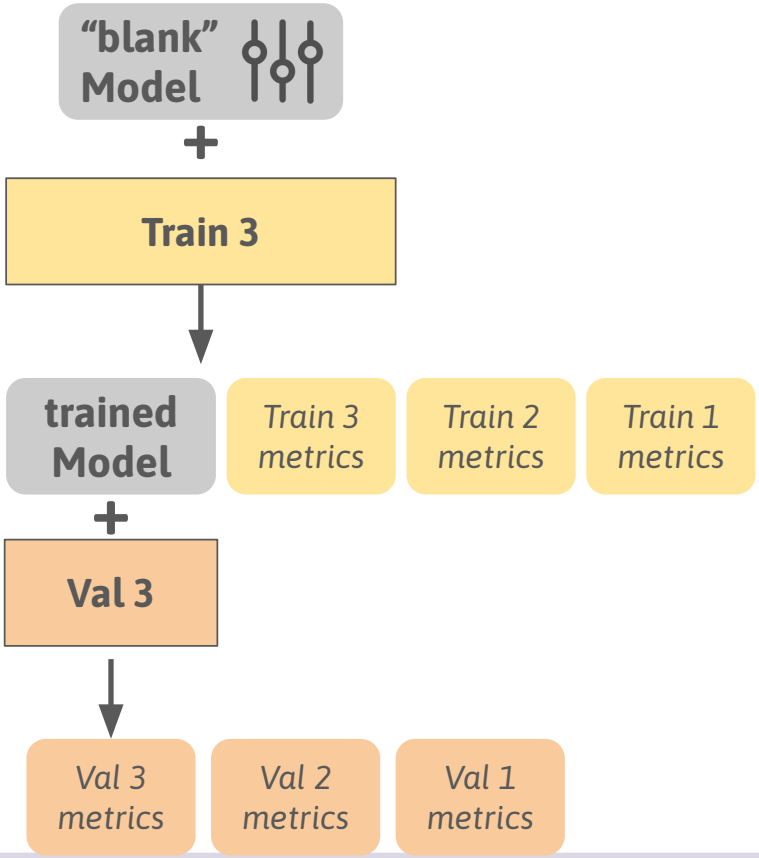
20% fold_1	20% fold_2	20% fold_3	20% fold_4

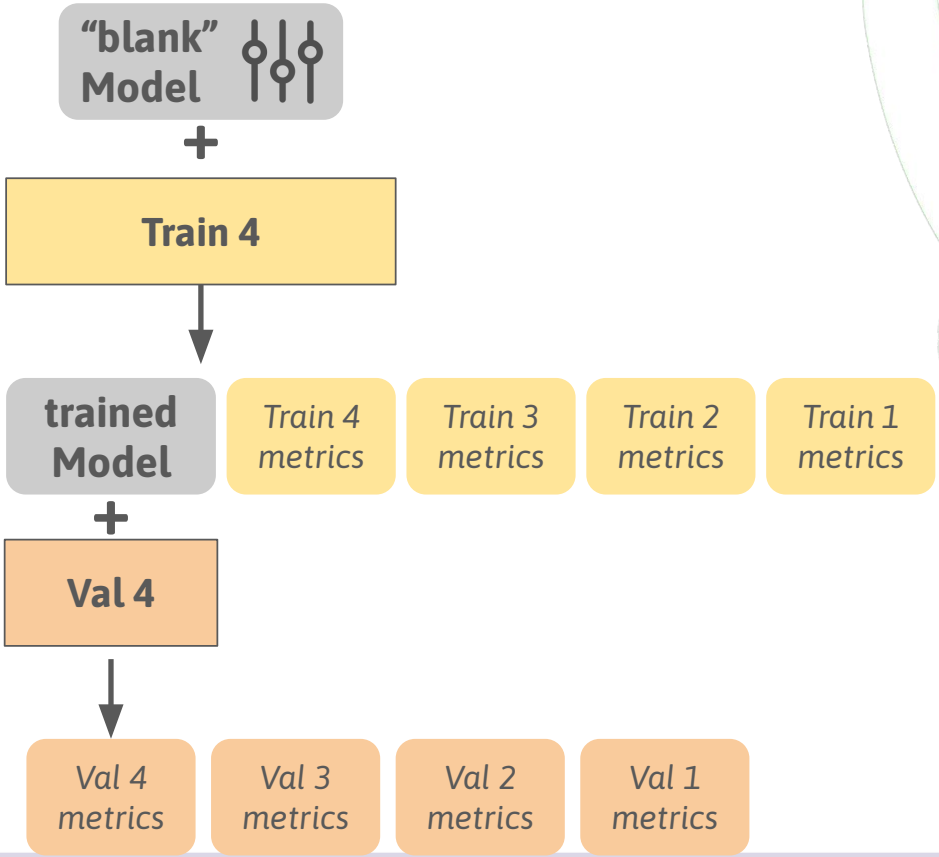
- Make them “take turns” being the validation set

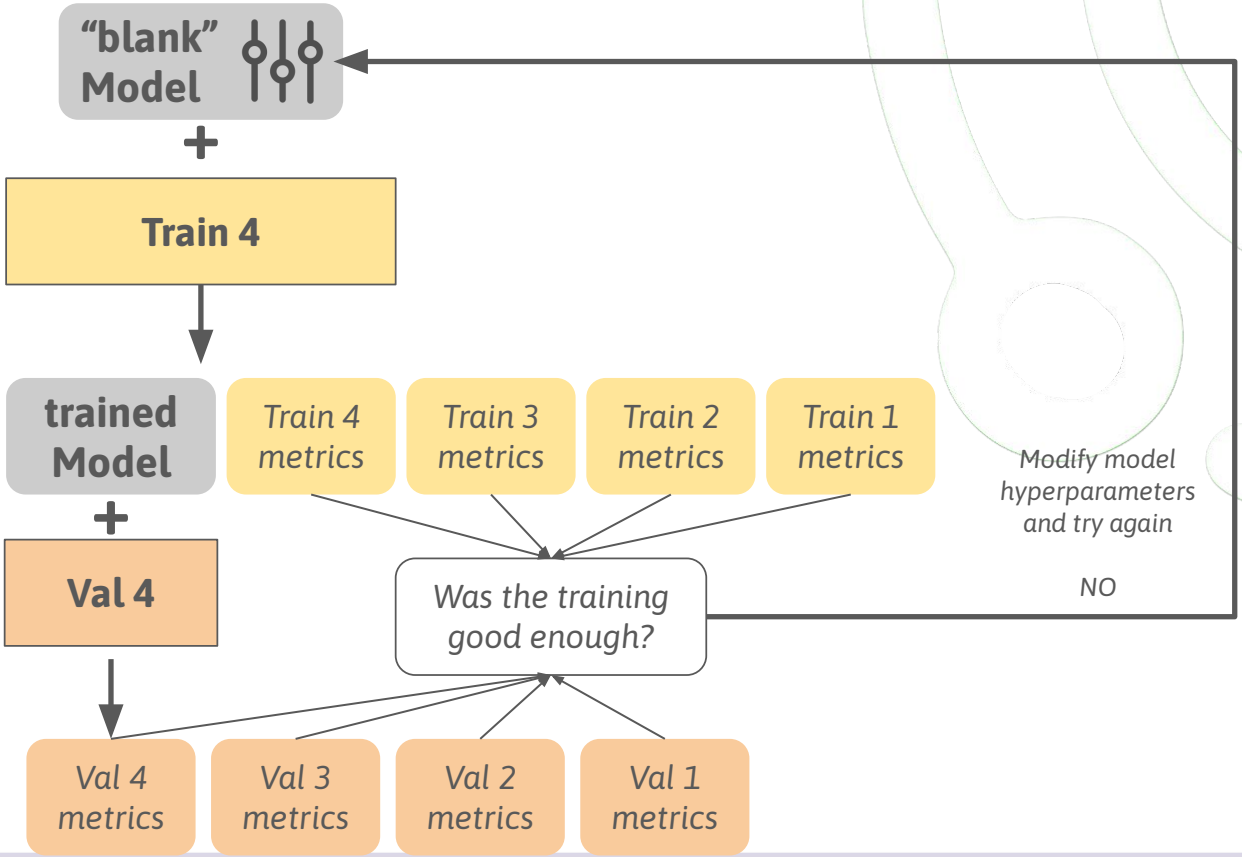
Train 1	Train 1	Train 1	Val 1
Train 2	Train 2	Val 2	Train 2
Train 3	Val 3	Train 3	Train 3
Val 4	Train 4	Train 4	Train 4

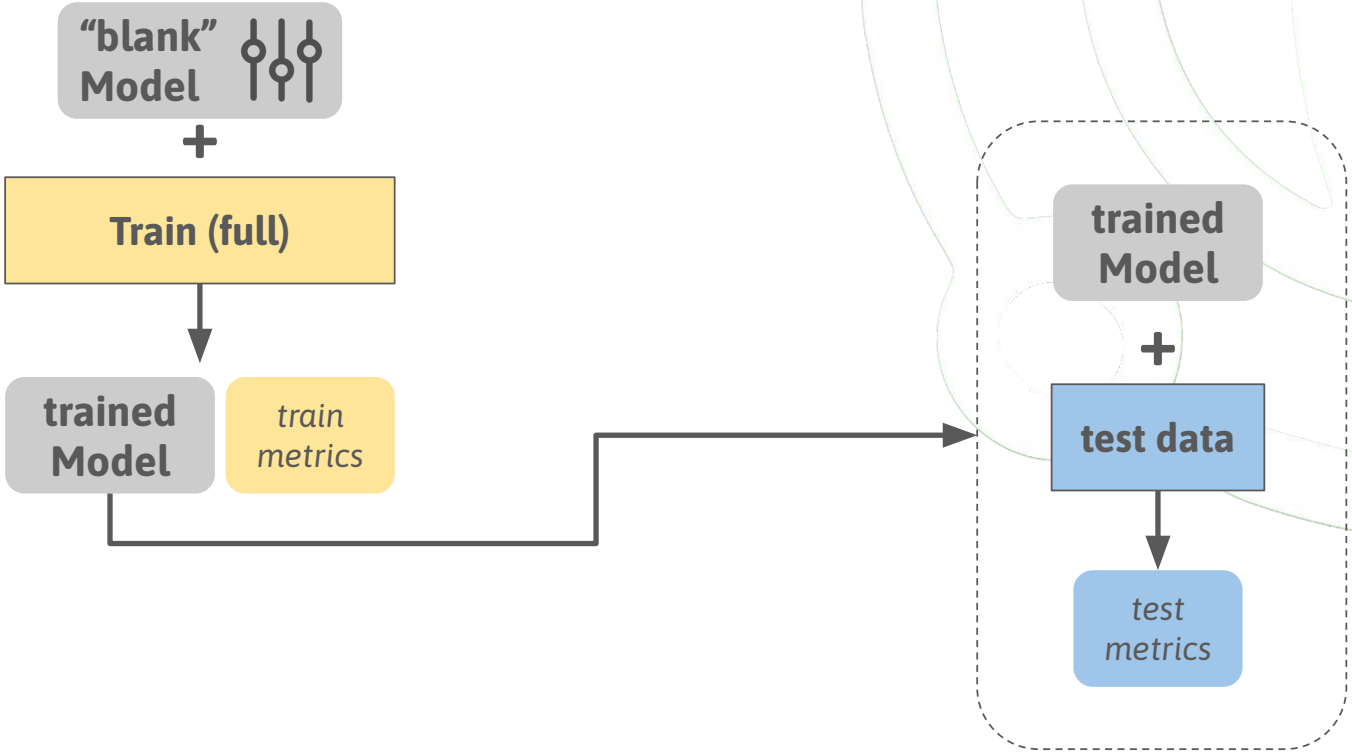

"blank" Model
Train 1
trained Model
Train 1 metrics
Val 1
Val 1 metrics


"blank" Model
Train 2
trained Model
Train 2 metrics
Train 1 metrics
Val 2
Val 2 metrics
Val 1 metrics


"blank" Model
Train 3
trained Model
Train 3 metrics
Train 2 metrics
Train 1 metrics
Val 3
Val 3 metrics
Val 2 metrics
Val 1 metrics


"blank" Model
Train 4
trained Model
Train 4 metrics
Train 3 metrics
Train 2 metrics
Train 1 metrics
Val 4
Val 4 metrics
Val 3 metrics
Val 2 metrics
Val 1 metrics


"blank" Model
Train 4
trained Model
Val 4
Train 4 metrics
Train 3 metrics
Train 2 metrics
Train 1 metrics
Was the training good enough?
Modify model hyperparameters and try again
NO
Val 4 metrics
Val 3 metrics
Val 2 metrics
Val 1 metrics


"blank" Model
+
Train (full)
trained Model
train metrics
trained Model
+
test data
test metrics

Splitting your data - Cross Validation!

It is best practice to always split your data in three sets:

- **Training** set: used to fit the model
- **Validation** set: used to tune hyperparameters
- **Test** set: used only once at the end to evaluate final performance

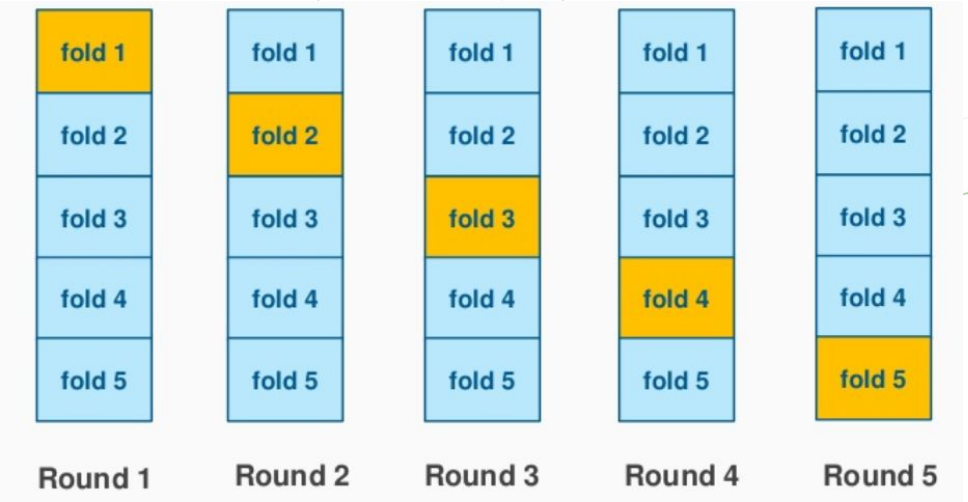
❌ No peeking at the test set during training!

✅ Use cross-validation if your dataset is small

Splitting your data - Cross Validation!

More about cross validation

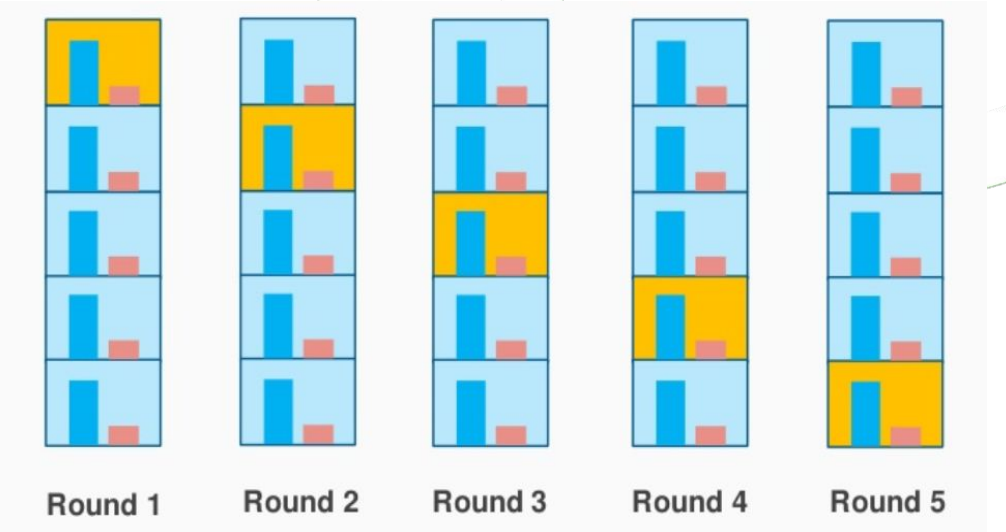
- Random cross validation (KFold)
  - The usual go-to when you don't have specific requirements
  - Usually 5 or 10 folds are used



Splitting your data - Cross Validation!

More about cross validation

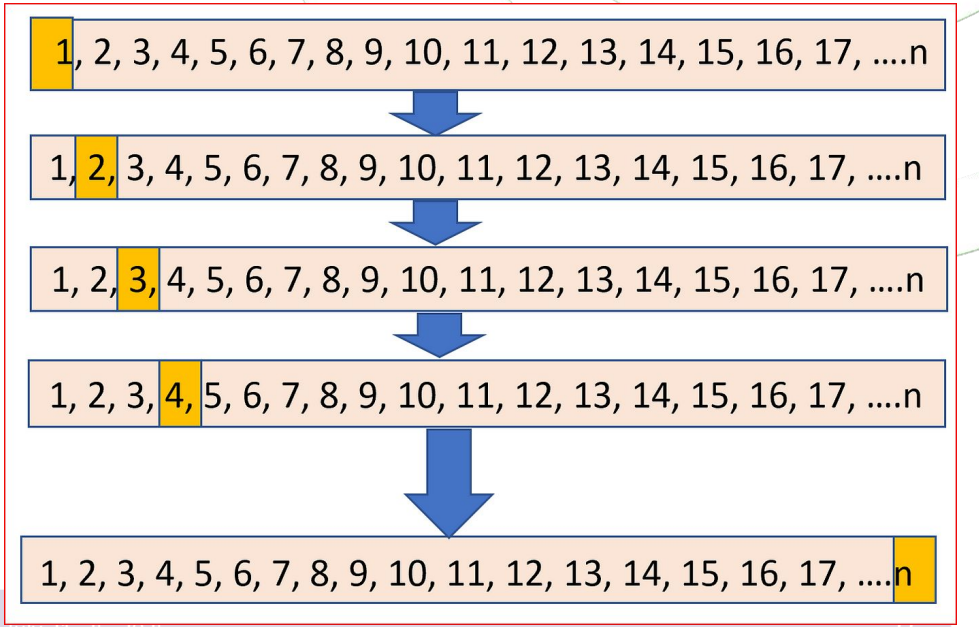
- Random cross validation (KFold)
- Stratified cross validation (StratifiedKFold)
  - Used when dataset has **strong** class imbalance
  - Ensures that each validation set has the same class distribution as the whole dataset



Splitting your data - Cross Validation!

More about cross validation

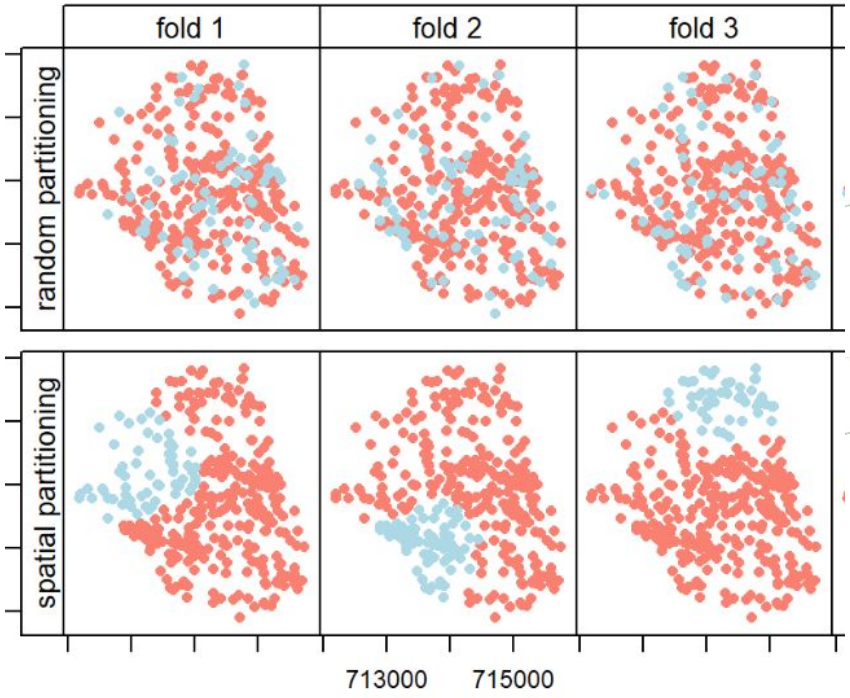
- Random cross validation (KFold)
- Stratified cross validation (StratifiedKFold)
- Edge case: Leave-one-out (LOO)
  - If you really *really* ***really*** are low on data
  - If it makes sense in the "real scenario" that the model will have a tons of historical data
  - Training/validation time is extremely long
  - Usually very noisy



Splitting your data - Cross Validation!

More about cross validation

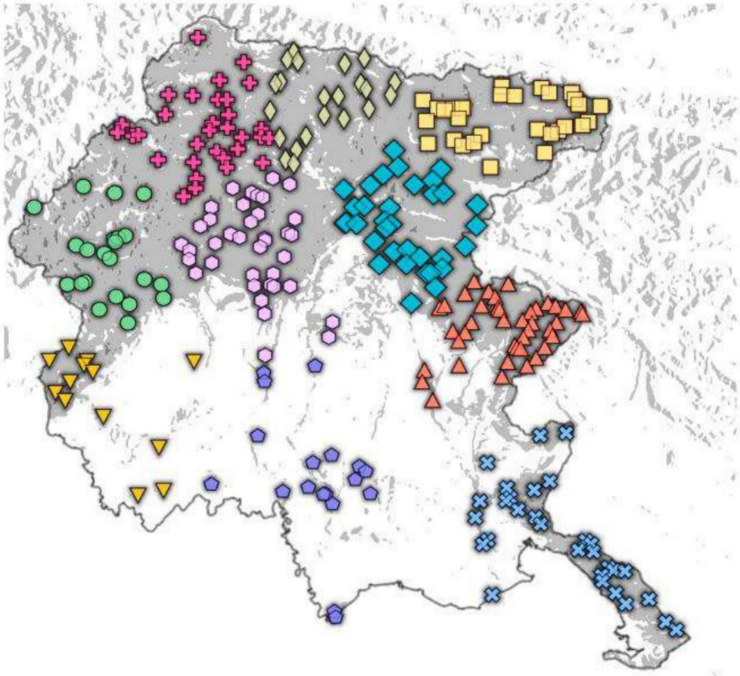
- Random cross validation (KFold)
- Stratified cross validation (StratifiedKFold)
- Edge case: Leave-one-out (LOO)
- Spatial cross-validation
  - When using geospatial data
  - To check if the model generalized on **new areas in space**



Splitting your data - Cross Validation!

More about cross validation

- Random cross validation (KFold)
- Stratified cross validation (StratifiedKFold)
- Edge case: Leave-one-out (LOO)
- Spatial cross-validation
  - When using geospatial data
  - To check if the model generalized on **new areas in space**



Splitting your data - Cross Validation!

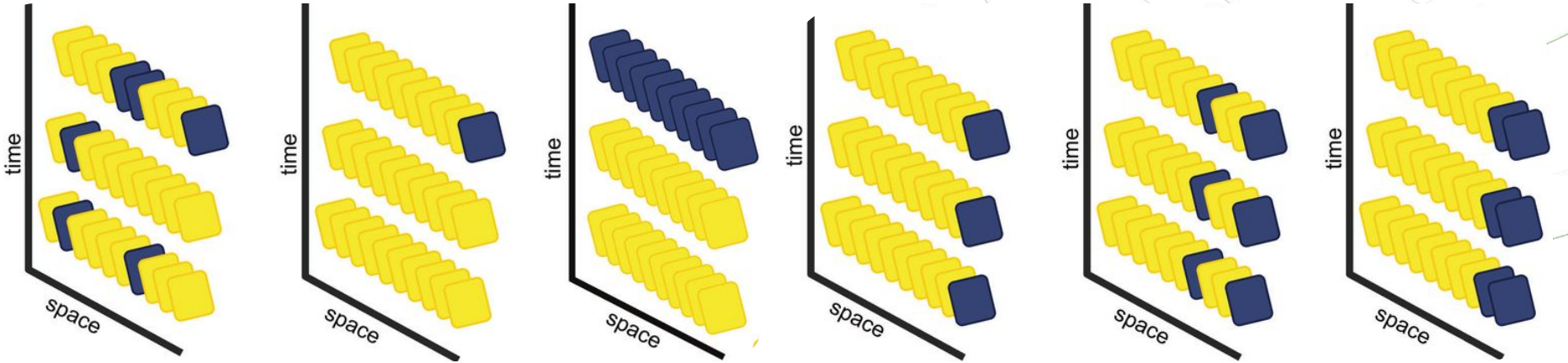
More about cross validation

- Random cross validation (KFold)
- Stratified cross validation (StratifiedKFold)
- Edge case: Leave-one-out (LOO)
- Spatial cross-validation
- Spatio-temporal cross-validation

(image on next slide)

Splitting your data - Cross Validation!

More about cross validation



Splitting your data - Recap

It is best practice to always split your data in three sets:

- **Training** set: used to fit the model
- **Validation** set: used to tune hyperparameters
- **Test** set: used only once at the end to evaluate final performance

❌ No peeking at the test set during training!

✅ Use cross-validation if your dataset is small

⚖️ Pay attention on *how* you split and partition your data

Splitting your data - Validation set

- Why all the fuss about the validation set?
- Isn’t the test set enough to check if we generalize on new data?

Yes... But actually no.

Splitting your data - Validation set

Models have knobs/settings (hyperparameters)

- How many input features?
- How big is the model
  (e.g., number of trees in a random forest)
- How long is the model going to try to analyze the data?
  (e.g., training epochs, decision tree depth, optimization iterations)
- …

and we need to “tune” them! (find the best ones)

Splitting your data - Validation set

Models have knobs/settings (hyperparameters)
and we need to “tune” them! (find the best ones)

If we do that on the **test** set

we are “overfitting” the model for the test set…
even without explicitly *training* on it!

Validation and cross-validation gives us
multiple mock test sets to “truly” assess generalizability

Splitting your data - Validation set

Common strategies:

- Grid search: try all combinations (slow but exhaustive)
- Random search: surprisingly good for many cases
- Bayesian optimization: smart but more advanced

Use **validation set** to evaluate each configuration and find the best one

Reproducibility

Reproducibility

- Choice of dataset / new dataset creation
- Dataset preprocessing steps
- Feature extraction
- Modeling choices
- Hyperparameter choice
- Training method
- Model architecture
- Which metrics to use to evaluate performance
- …

Reproducibility

- Choice of dataset / new dataset creation
- Dataset preprocessing steps
- Feature extraction
- Modeling choices
- Hyperparameter choice
- Training method
- Model architecture
- Which metrics to use to evaluate performance
- …

So many choices to be made to get to the final model!

Reproducibility

- Programming language
- Libraries/packages (and versions)
- Type of hardware
  e.g., windows or linux,
  CPU or GPU, … and which kind?

So many choices to be made to get to the final model!

- How “random” is the code?
- How “lucky” were you when you run it?

Reproducibility

Set your seeds, and log your steps!

- Set random seeds (NumPy, scikit-learn, PyTorch...)
- Repeat model training with different seeds
  (and report the average metrics) to ensure the stability of the results
- Document all preprocessing and pipeline steps
- Save model config + hyperparameters for future reference
- Consider logging tools like WandB if you're fancy 😎

Reproducibility - Sharing is caring

- In many fields it is easy to get datasets and working code to kickstart research
- Some fields in environmental sciences share less (for understandable reasons)
  - Working with satellite images, but cannot share exact coordinates of field experiments
  - Collect a new datasets of fishfarm fish health but cannot share it because of legal issues
  - Use finegrained climatic data from climate agencies, but they are only available “on request” (so cannot be shared “openly” to the community)
  - Experiments using texts from government documents (cannot share full texts)

Reproducibility - Sharing is caring

- In many fields it is easy to get datasets and working code to kickstart research
- Some fields in environmental sciences share less (for understandable reasons)
- But sometimes it may be just because of habit (no one is sharing, so why should I bother?)

Reproducibility - Sharing is caring

- In many fields it is easy to get datasets and working code to kickstart research
- Some fields in environmental sciences share less (for understandable reasons)
- But sometimes it may be just because of habit (no one is sharing, so why should I bother?)
- Sharing resources helps research move forward!

Reproducibility - Sharing is caring

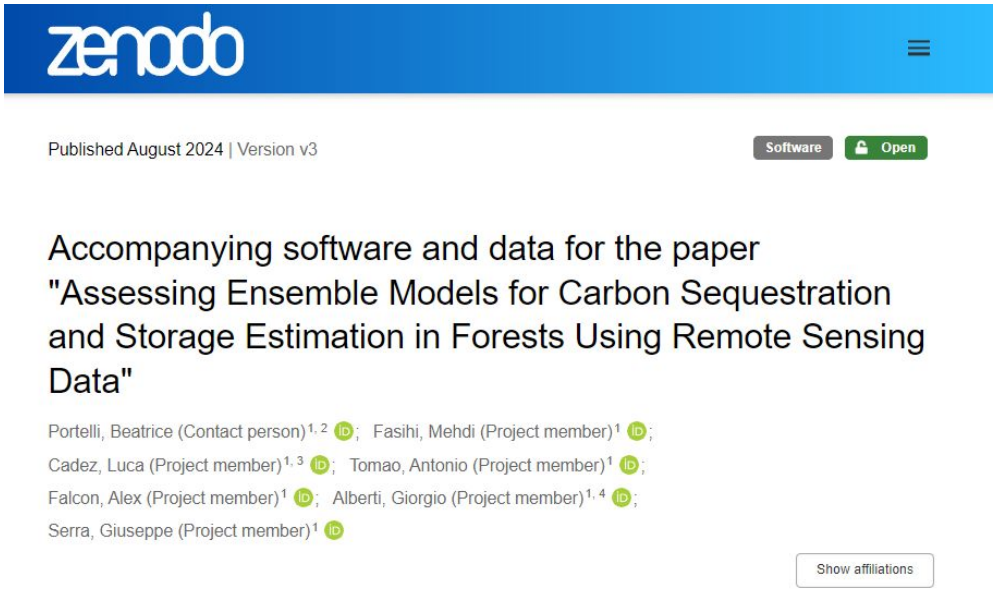
- Sharing resources helps research move forward!

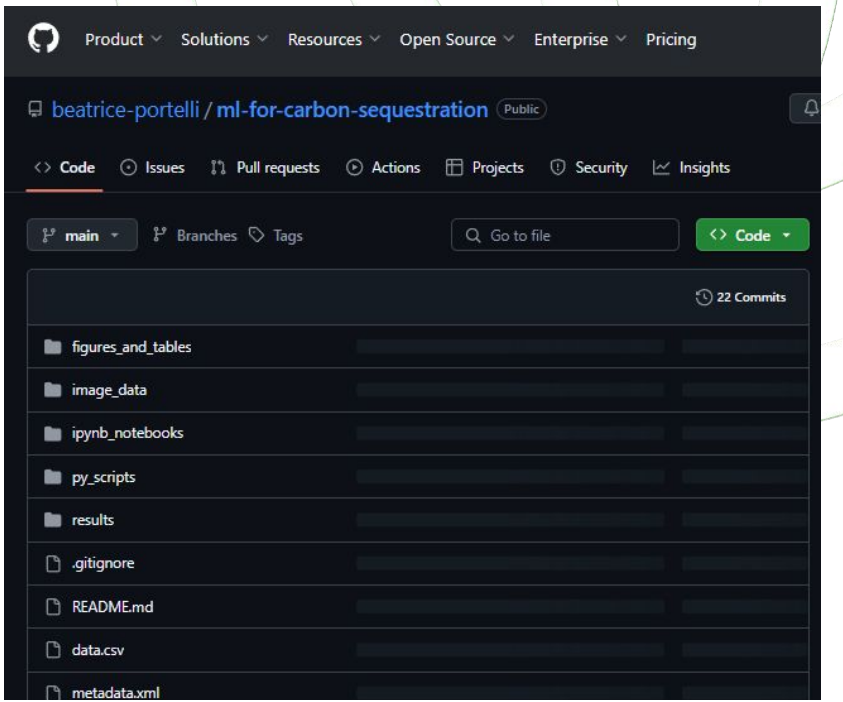
Otherwise everyone who wants to create a new model would also need to collect and annotate a large amounts of data!

If you publish, consider:

- Sharing your raw and processed data
- Including metadata (how it was collected, cleaned, etc.)
- Respect privacy, licenses, and local context
  if you cannot share data, do not make them public
  but be available to “dataset requests” from future researchers

Reproducibility - Sharing is caring





Reproducibility - Sharing is caring

README

Requirements

For the libraries needed to run the code check `requirements.txt`

You can install the needed libraries using `pip install -r requirements.txt`

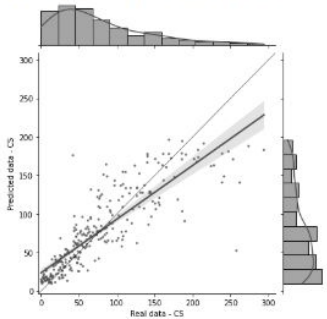
Files overview

- **`results`**
  This folder contains the outputs of the models described in the paper.
  Be careful, running some of the code in this repository might overwrite the results folder. Please rename the folder to keep the original results.
  Also note that the training features (X_train and X_test, as well as y_train) present in the `predictions--*` files have been removed and only the target values/predictions (y_test and y_pred) needed to reproduce the figures in the paper have been kept. Training data will be made available upon request.
- **`py_scripts`**
  This folder contains the *.py version of all the *.ipynb notebooks described below.
- **`configs.py`**
  Contains constants such as:
  - names of the target variables (denoted as `TARGET_*`)
  - definitions of the input configurations (denoted as `CONFIG_*`)

In [2]:

```
MODEL = "+ Ensemble"
CONF = "Conf3"
for TARGET in TARGETS:

    data = df[
        (df.model==MODEL) &\
        (df.config==CONF) &\
        (df.target==TARGET)
    ]

    test = np.concatenate(data.y_test.values.tolist())
    pred = np.concatenate(data.y_pred.values.tolist())

    data_df = pd.DataFrame({"test":test, "pred":pred})

    axs = sns.jointplot(
        data=data_df,
        x='test',
        y='pred',
        color=".3",
        kind="reg",
        joint_kws=dict(scatter_kws=dict(s=10,ec=None,)),
    )

    ax = axs.ax_joint

    ax.set(
        xlabel='Real data - '+TARGET,
        ylabel='Predicted data - '+TARGET,
    )

    xlims = ax.get_xlim()
    ylims = ax.get_ylim()
    maxval = max(xlims[1], ylims[1])

    ax.plot([-maxval/100,maxval], [-maxval/100,maxval], color="k", lw=.5)

    ax.set_xlim((-maxval/100, maxval))
    ax.set_ylim((-maxval/100, maxval))

    ax.set_xlabel("Real data - "+TARGET)
    ax.set_ylabel("Predicted data - "+TARGET)

    plt.savefig(f"../results/trends_with_margins_{MODEL}_{TARGET}_{CONF}.png", facecolor='white', bbox_inches='tight', dpi=
    plt.savefig(f"../results/trends_with_margins_{MODEL}_{TARGET}_{CONF}.pdf", bbox_inches='tight', dpi=300)
    print("figure exported to", f"../results/trends_with_margins_{MODEL}_{TARGET}_{CONF}.[png/pdf]")

    plt.savefig(f"../figures_and_tables/figure_comparison_real_predicted_data_{MODEL}_{TARGET}_{CONF}.png", facecolor='whit
    print("figure exported to", f"../figures_and_tables/figure_comparison_real_predicted_data_{MODEL}_{TARGET}_{CONF}.png")

    plt.show()
```

```
figure exported to ../results/trends_with_margins_+ Ensemble_CS_Conf3.[png/pdf]
figure exported to ../figures_and_tables/figure_comparison_real_predicted_data_+ Ensemble_CS_Conf3.png
```



Reproducibility - Sharing is caring

Sharing helps if the shared data is “good”…

Data - How to spot a low-quality dataset

Unfortunately, not all public datasets can be trusted!

Red flags you may spot in datasets:

- 🚩 No metadata, unclear data provenance or unclear meaning of the features ❗
- 🚩 Too small or imbalanced to be useful
- 🚩 Missing values or inconsistent formatting
- 🚩 Split between train/test not (clearly) defined
- 🚩 Unrealistic values (e.g., NDVI = 99, rainfall = -300 mm) ❗
- 🚩 Data augmentation built into the dataset ‼️

Data - How to spot a low-quality dataset

- “We labeled 10,000 images” yes, but half the “sheep” are goats.
- “The timestamps are all in local time” … okay, but which *local*?
- The paper says the dataset has 10,000 samples with 7 features…
  The shared data has 1000 samples with 20 features
- You spend hours mapping the 20 features to the 7 features to find out…
  Three features have the exact same value, they’re copies.

Even the best model can’t fix a broken dataset!

Data - Looking for a dataset in a niche field

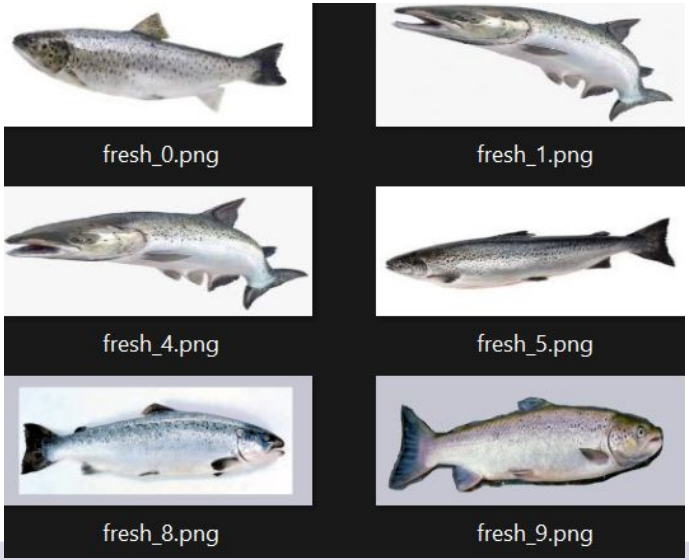
Samples	Train/Test split?	Train/Test separated?	Already Augmented
2450 (350 per class, balanced)	yes (66–33%, balanced)	no	yes

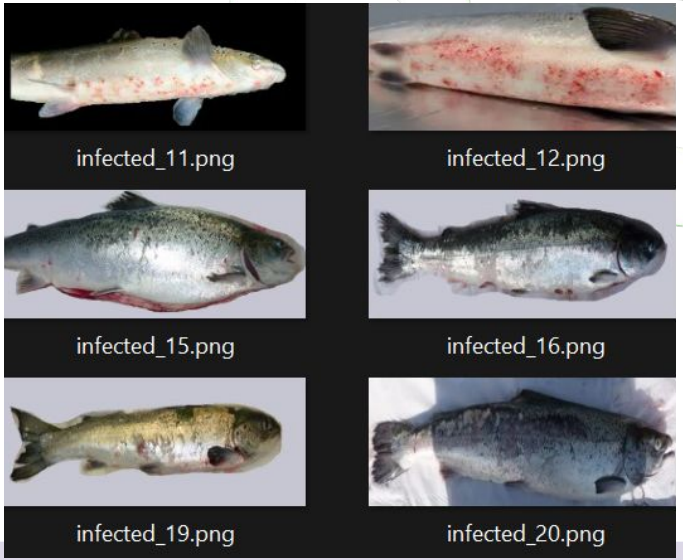
Train

Test

Data - Looking for a dataset in a niche field

Samples	Train/Test split?	Train/Test separated?	Already Augmented
1208 (from 115 unique fishes)	no	–	yes

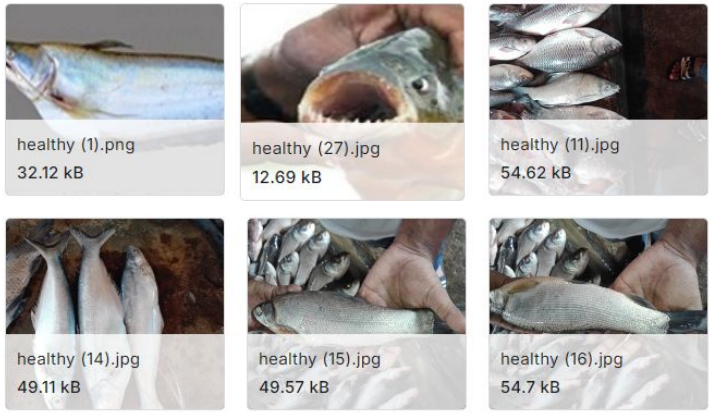




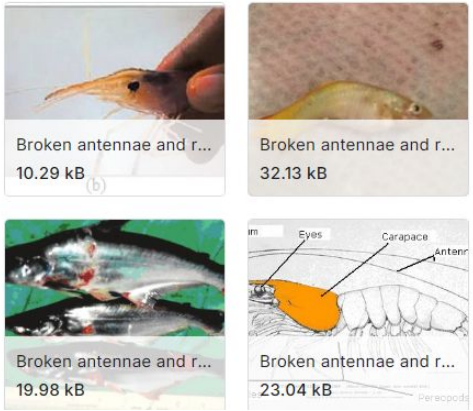
Data - Looking for a dataset in a niche field

Samples	Train/Test split?	Repeated samples?	Already Augmented
113	no	yes	no

Healthy



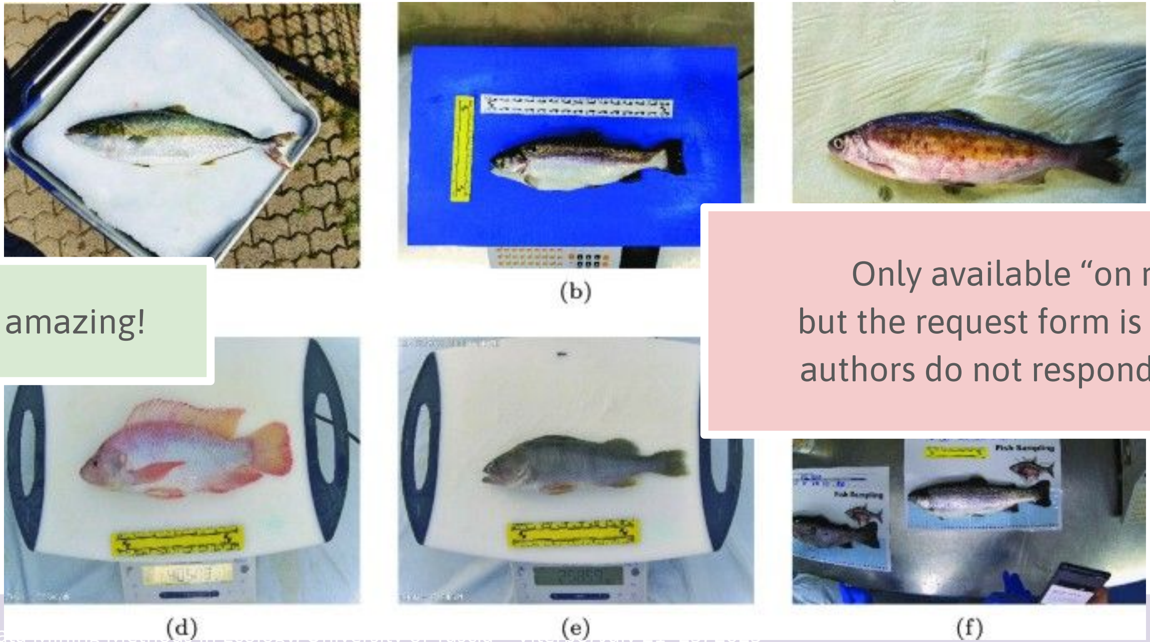
Broken antennae and rostrum



Argulus

Data - Looking for a dataset in a niche field

Samples	Train/Test split?	Train/Test separated?	Already Augmented
2157	?	?	no



Looks amazing!

Only available “on request”
but the request form is broken and
authors do not respond to emails :(

Data - Looking for a dataset in a niche field

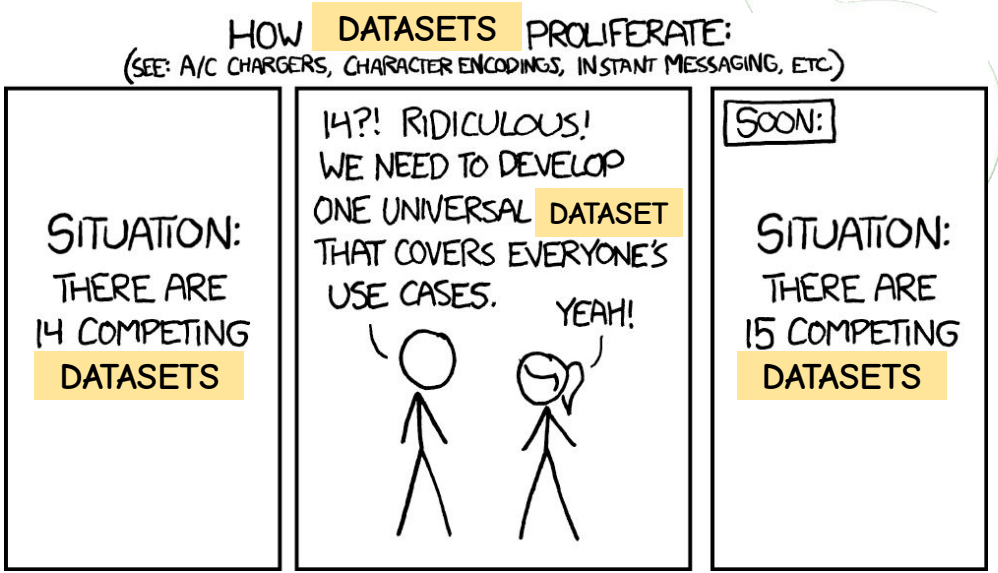
After downloading and analyzing ~20 datasets
After trying to contact the authors of the non-available datasets

...

We started creating our own 🥲

Data - Looking for a dataset in a niche field

We started creating our own 🥲



Best Practices = Better Science

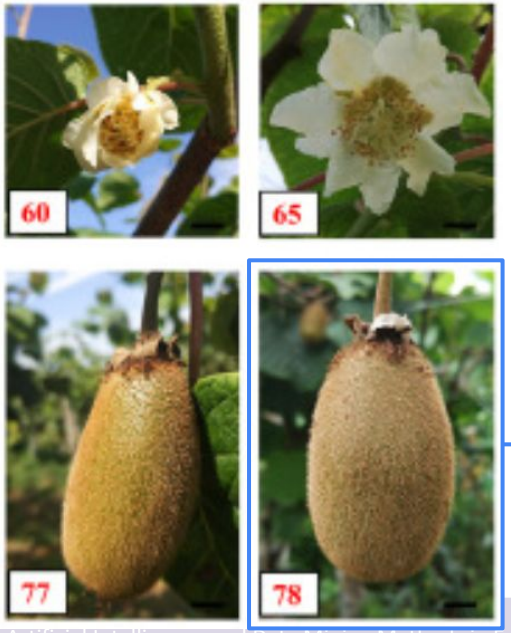
• Data splitting	Prevents cheating (leakage)
• Reproducibility (seeds etc.)	Ensures results can be trusted
• Hyperparameter tuning	Boosts model performance
• Dataset/code sharing	Enables collaboration & reuse
• Dataset sanity checks	Avoids wasted time on bad data

Brief intro to hand-crafted feature engineering

From raw data to (possibly) useful information

There are no “intelligent models” unless we provide them with good information.

↳ *GIGO: Garbage in, garbage out*



How ripe is this kiwi?

How to answer?
Based on size? Shape? Hair density? Color? …?

All of these are ways to translate a complex problem into useful (numerical) information for a model. → **features**

Zhang, Q., Zhang, H., Gui, Y., Geng, S., Liu, Z., Liu, Y., & Liu, Z. (2024). Phenological growth stages of a new kiwifruit cultivar (Actinidia deliciosa ‘Jinfu’). Scientia Horticulturae, 327, 112795.

Feature engineering on tabular data

Tabular data → tables containing numbers

- Columns → variables, features (e.g. crop type, yield)
- Rows → **observations**, examples (e.g. a specific crop type, a specific yield amount)
- Columns describe what quantities will be shown; rows precisely quantify those quantities for a specific instance

Area Country	Item Crop	Year Year	Value
Italy	Maize	1965	32278
Italy	Maize	1966	35532
Italy	Maize	1967	37961
Italy	Maize	1968	41261
Italy	Maize	1969	45235
Italy	Maize	1970	46344
Italy	Maize	1971	48484
Italy	Maize	1972	53204
Italy	Maize	2006	87285
Italy	Maize	2007	93120
Italy	Maize	2008	95726
Italy	Maize	2009	86048
Italy	Maize	2010	91672
Italy	Maize	2025	???

Feature engineering on tabular data

Tabular data → tables containing numbers

How to predict maize yield for 2025?

What kind of info to extract?

Area Country	Item Crop	Year Year	Value
Italy	Maize	1965	32278
Italy	Maize	1966	35532
Italy	Maize	1967	37961
Italy	Maize	1968	41261
Italy	Maize	1969	45235
Italy	Maize	1970	46344
Italy	Maize	1971	48484
Italy	Maize	1972	53204
Italy	Maize	2006	87285
Italy	Maize	2007	93120
Italy	Maize	2008	95726
Italy	Maize	2009	86048
Italy	Maize	2010	91672
Italy	Maize	**2025**	**???**

Feature engineering on tabular data

Tabular data → tables containing numbers

How to predict maize yield for 2025?

What kind of info to extract?

- Statistical descriptors (possibly **temporal**)

  ➔ Avg/median yield
  ➔ 5y/10y avg, deviation from 5y/10y (trend-related)
  ➔ Frequency of extreme events (cold waves, floods, droughts, …) in previous years

Area Country	Item Crop	Year Year	Value
Italy	Maize	1965	32278
Italy	Maize	1966	35532
Italy	Maize	1967	37961
Italy	Maize	1968	41261
Italy	Maize	1969	45235
Italy	Maize	1970	46344
Italy	Maize	1971	48484
Italy	Maize	1972	53204
Italy	Maize	2006	87285
Italy	Maize	2007	93120
Italy	Maize	2008	95726
Italy	Maize	2009	86048
Italy	Maize	2010	91672
Italy	Maize	2025	???

Feature engineering on tabular data

Tabular data → tables containing numbers

How to predict maize yield for 2025?

What kind of info to extract?

- Statistical descriptors (possibly **temporal**)
- Domain-specific derived/interaction features

➔ Water use efficiency (yield/water used)
➔ Fertilizer efficiency (yield/N uses)
➔ Rain during flowering / yearly avg

Area Country	Item Crop	Year Year	Value
Italy	Maize	1965	32278
Italy	Maize	1966	35532
Italy	Maize	1967	37961
Italy	Maize	1968	41261
Italy	Maize	1969	45235
Italy	Maize	1970	46344
Italy	Maize	1971	48484
Italy	Maize	1972	53204
Italy	Maize	2006	87285
Italy	Maize	2007	93120
Italy	Maize	2008	95726
Italy	Maize	2009	86048
Italy	Maize	2010	91672
Italy	Maize	**2025**	**???**

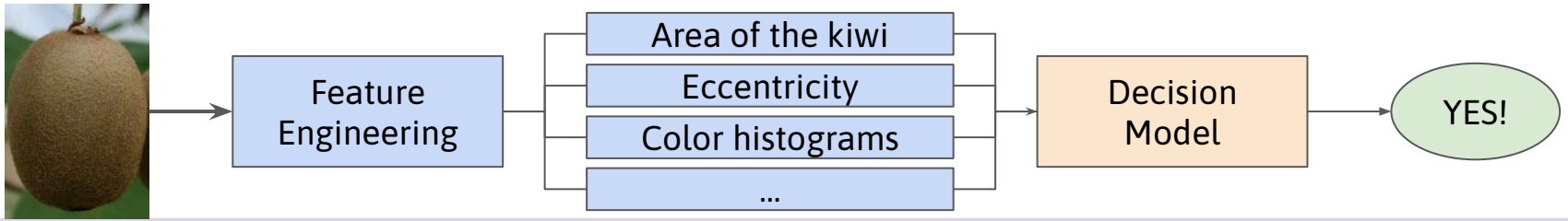
Feature engineering on visual data

Is this kiwi ripe?

The model answers based on *input numerical* information.

How to describe the visual contents in these terms?

Feature engineering: before Deep Learning, a fundamental pre-processing step to describe visual characteristics of images.

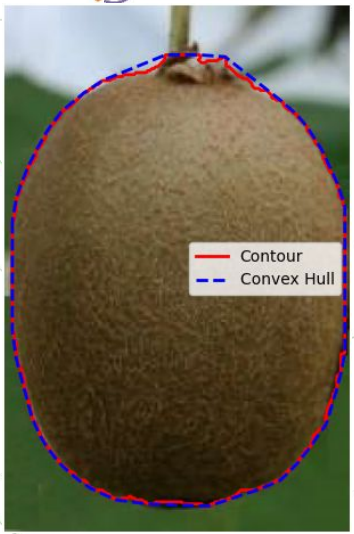


Feature engineering on visual data

- Shape analysis
  - ★ Geometric characteristics of objects within an image
  - ★ Contour-based
    - ➔ Length of contour ➔ 693.4
    - ➔ Circularity (1=circle) ➔ 0.821

Feature engineering on visual data

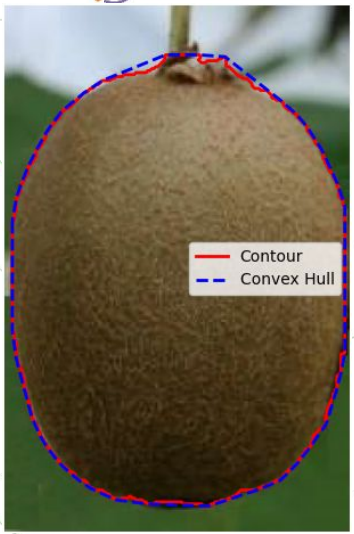
- Shape analysis
  - ★ Geometric characteristics of objects within an image
  - ★ Contour-based
    - ➔ Length of contour ➔ 693.4
    - ➔ Circularity (1=circle) ➔ 0.821



Feature engineering on visual data

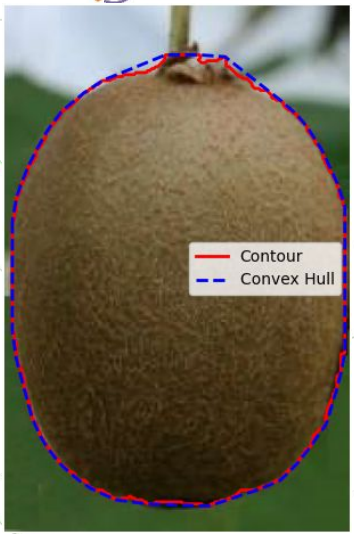
- Shape analysis
  - ★ Geometric characteristics of objects within an image
  - ★ Contour-based

➔ Length of contour	➔ 693.4
➔ Circularity (1=circle)	➔ 0.821
➔ Convexity (1=contour coincides with its hull)	➔ 0.933
➔ Bending energy (quantifies tortuosity of the contour)	➔ 0.099



Feature engineering on visual data

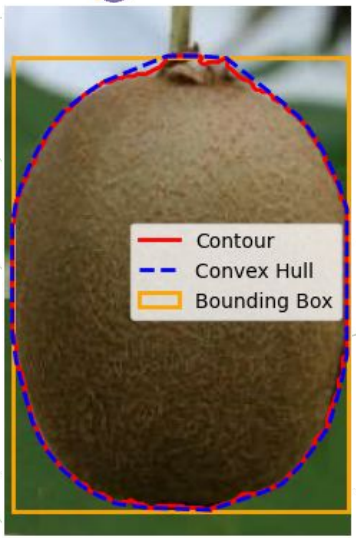
- Shape analysis
  - ★ Geometric characteristics of objects within an image
  - ★ Contour-based
  - ★ Region-based
    - ➔ Area → 31417
    - ➔ Eccentricity (1=line, 0=circle) → 0.650
    - ➔ Solidity (area/hull area) → 0.985



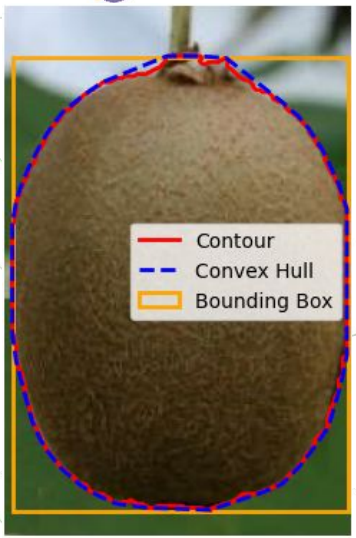
Feature engineering on visual data

- Shape analysis
  - ★ Geometric characteristics of objects within an image
  - ★ Contour-based
  - ★ Region-based

➔ Area	➔ 31417
➔ Eccentricity (1=line, 0=circle)	➔ 0.650
➔ Solidity (area/hull area)	➔ 0.985
➔ Extent (area/bbox area)	➔ 0.820



Feature engineering on visual data



- Shape analysis
  - ★ Geometric characteristics of objects within an image
  - ★ Contour-based
  - ★ Region-based

➔ Area	➔ 31417
➔ Eccentricity (1=line, 0=circle)	➔ 0.650
➔ Solidity (area/hull area)	➔ 0.985
➔ Extent (area/bbox area)	➔ 0.820
➔ Hu moments (7 moments, invariant to translation, rotation, scale)	➔ 1.63e+08, 1.92e+15, ...

Feature engineering on visual data

- Texture analysis
  - ★ Spatial arrangement of pixel intensities; describes surfaces, their smoothness or roughness, etc.

Feature engineering on visual data

- Texture analysis
  - ★ Spatial arrangement of pixel intensities; describes surfaces, their smoothness or roughness, etc.
  - ➔ GLCM - Grey-level co-occurrence matrix
    - How frequently do neighboring pixels have similar grey levels?
    - Matrix → features extracted: contrast, dissimilarity, homogeneity, energy, correlation, ...

GLCM - Example

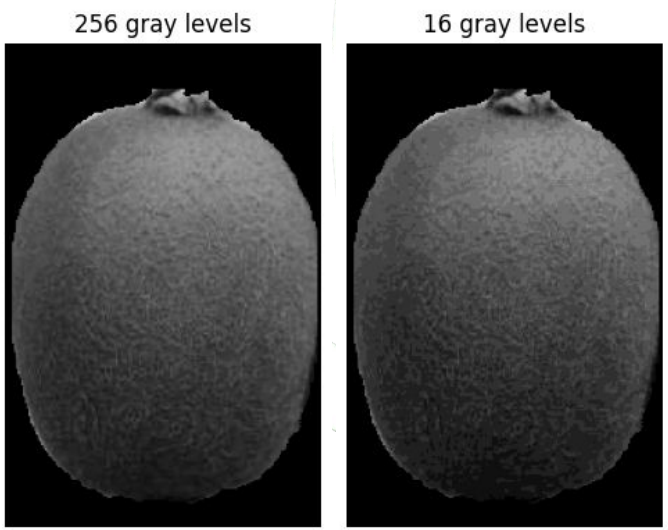
GLCM - Example

Levels	Contrast	Energy	Homogeneity
256	130.5960	0.3125	0.4443

- ➔ Contrast: high → strong variation in neighboring pixels (rough texture)
- ➔ Energy: high → presence of patterns, repetitions
- ➔ Homogeneity: high → neighbors are similar, few borders

GLCM - Example

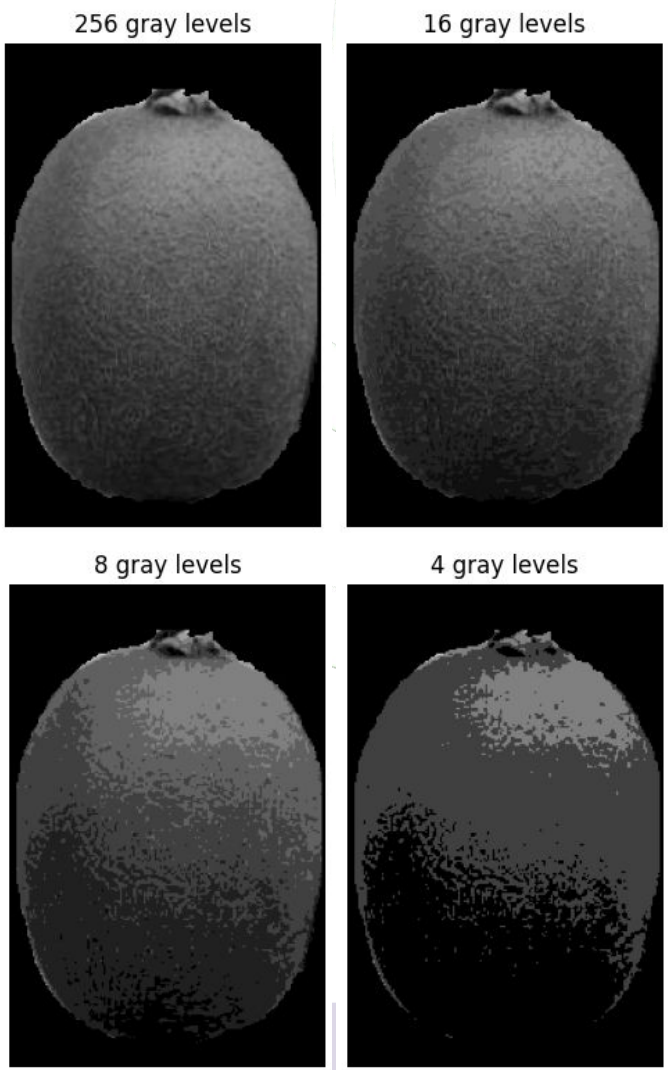
Levels	Contrast	Energy	Homogeneity
256	130.5960	0.3125	0.4443
16	0.5644	0.3587	0.8793



- ➔ Contrast: high → strong variation in neighboring pixels (rough texture)
- ➔ Energy: high → presence of patterns, repetitions
- ➔ Homogeneity: high → neighbors are similar, few borders

GLCM - Example

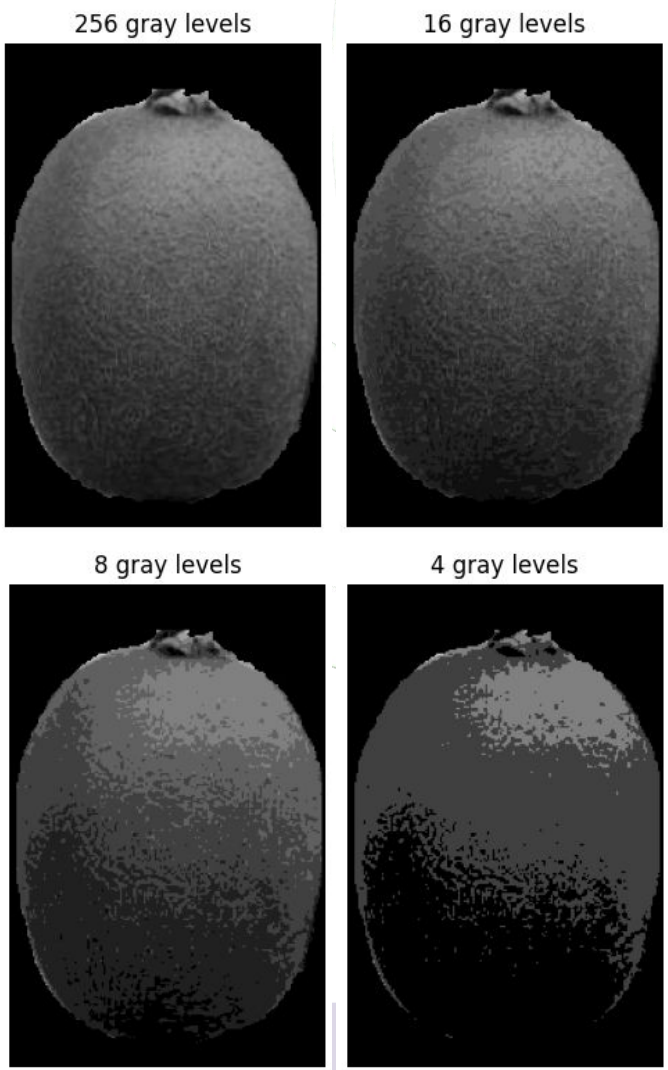
Levels	Contrast	Energy	Homogeneity
256	130.5960	0.3125	0.4443
16	0.5644	0.3587	0.8793
8	0.1775	0.4463	0.9372
4	0.0734	0.6378	0.9650



- ➔ Contrast: high → strong variation in neighboring pixels (rough texture)
- ➔ Energy: high → presence of patterns, repetitions
- ➔ Homogeneity: high → neighbors are similar, few borders

GLCM - Example

Levels	Contrast	Energy	Homogeneity
256	130.5960	0.3125	0.4443
16	0.5644	0.3587	0.8793
8	0.1775	0.4463	0.9372
4	0.0734	0.6378	0.9650



- ➔ Contrast: high → strong variation in neighboring pixels (rough texture)
- ➔ Energy: high → presence of patterns, repetitions
- ➔ Homogeneity: high → neighbors are similar, few borders
- ➔ *Note*: less levels → energy/homogeneity sharply increases → simpler, smoother textures

Feature engineering on visual data

- Texture analysis
  - ★ Spatial arrangement of pixel intensities; describes surfaces, their smoothness or roughness, etc.
  - ➔ GLCM - Grey-level co-occurrence matrix
  - ➔ Gabor filters
    - ◆ In this point of the image, is there a pattern with this *frequency* and *direction*?

Gabor - Example

★ Searches for patterns which respond to the selected frequency and orientation

➔ E.g. veins in leafs

Gabor - Example

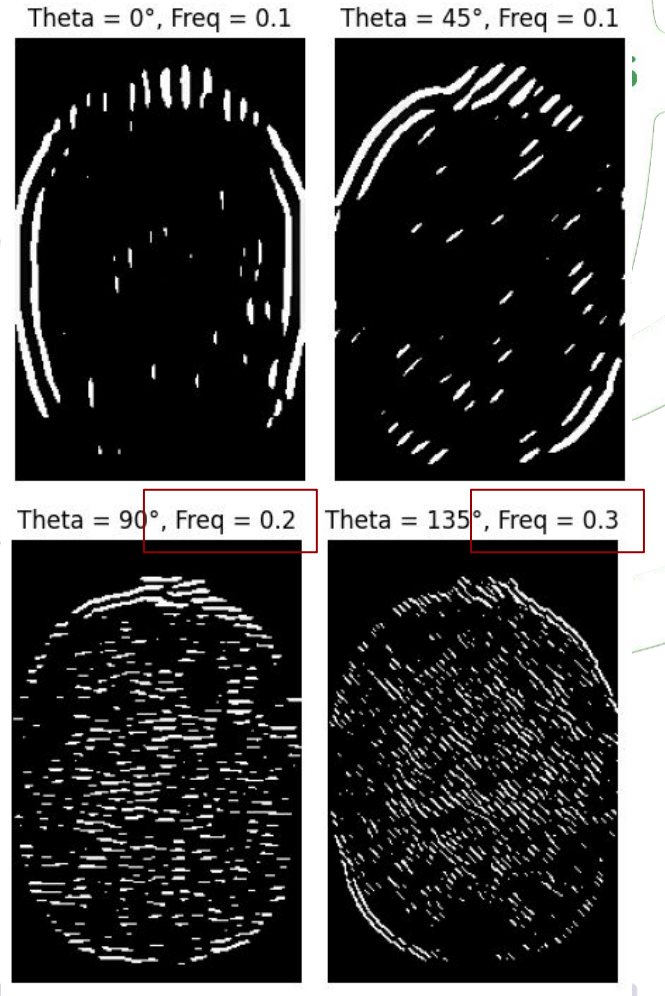
★ Searches for patterns which respond to the selected frequency and orientation

➔ E.g. veins in leafs

Gabor - Example

★ Searches for patterns which respond to the selected frequency and orientation

➔ E.g. veins in leafs



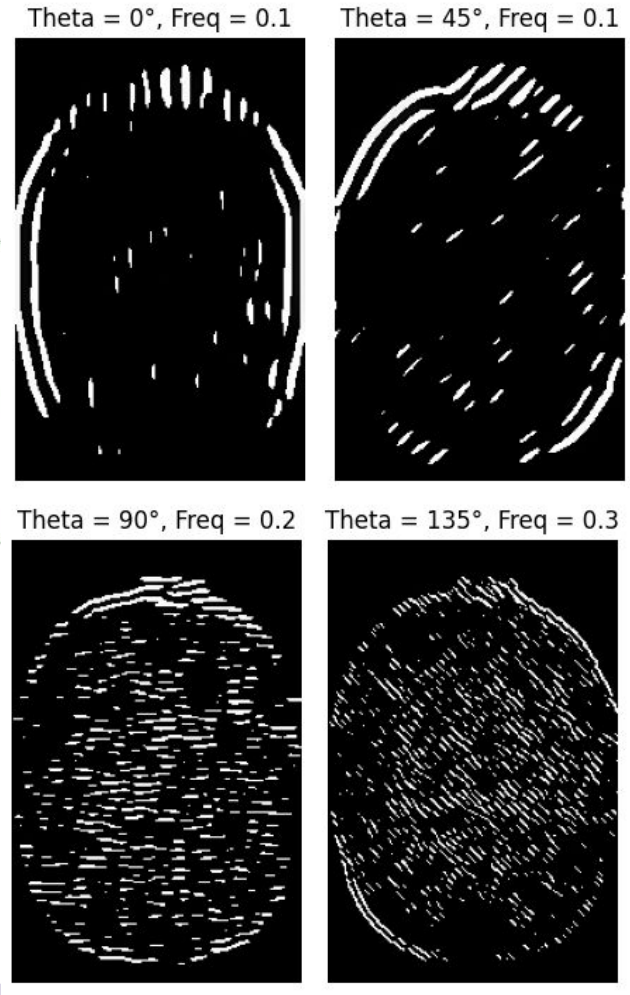
Gabor - Example

★ Searches for patterns which respond to the selected frequency and orientation

➔ E.g. veins in leafs

★ Statistical features on response, e.g.

- Mean (high → orientation/frequency match the actual texture)
- Std (high → borders, sharp changes)
- Entropy (high → variable, complex patterns)

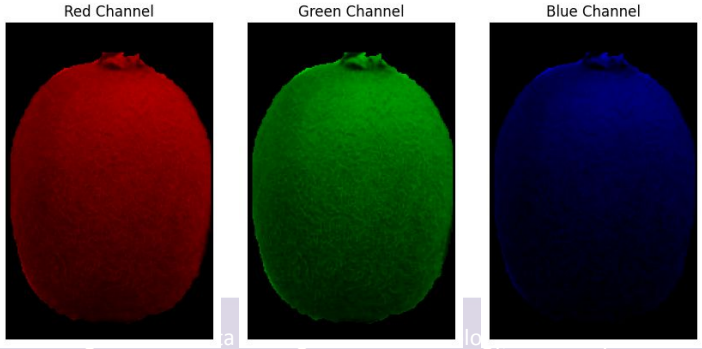


Feature engineering on visual data

- Color analysis
  - ★ Do all instances of ripe kiwis have similar colors?

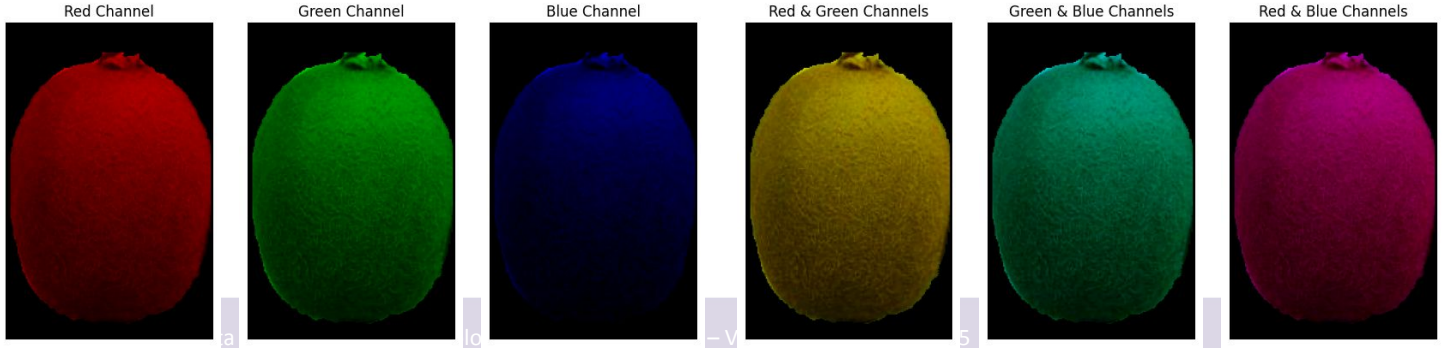
Feature engineering on visual data

- Color analysis
  - ★ Do all instances of ripe kiwis have similar colors?



Feature engineering on visual data

- Color analysis
  - ★ Do all instances of ripe kiwis have similar colors?

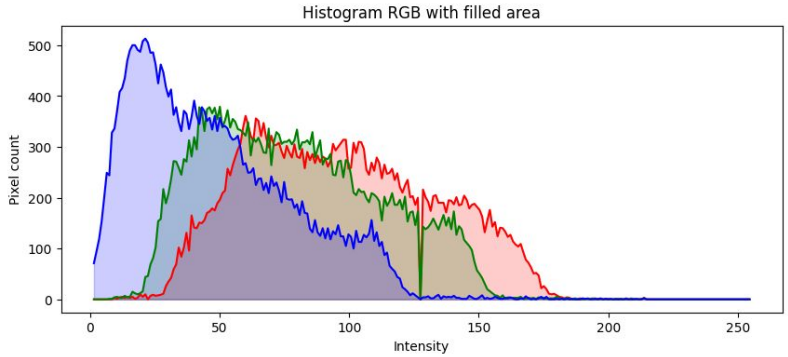


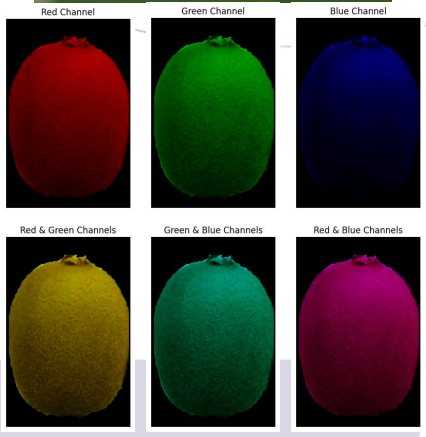
Feature engineering on visual data

- Color analysis

  ★ Do all instances of ripe kiwis have similar colors?

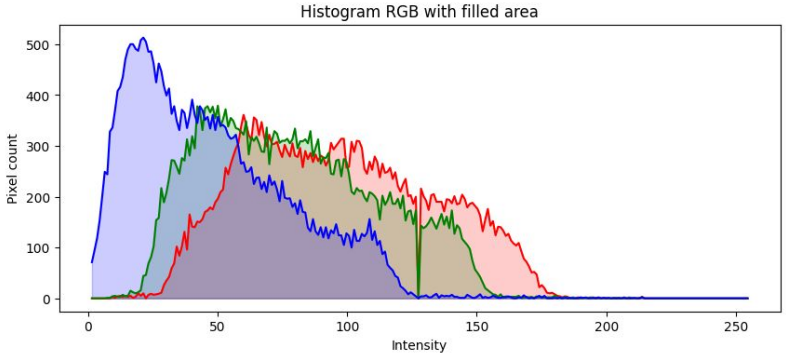
  ➔ Color histograms



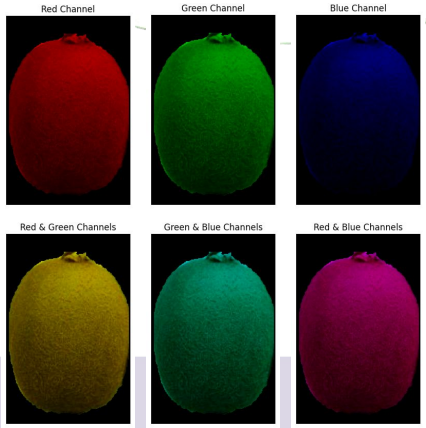


Feature engineering on visual data

- Color analysis
  - ★ Do all instances of ripe kiwis have similar colors?
  - ➔ Color histograms

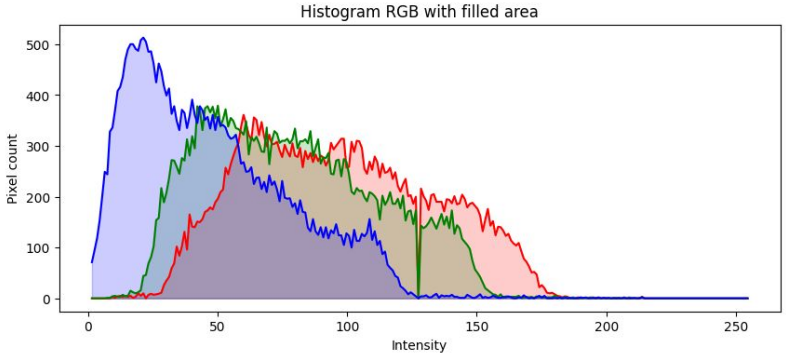


  - ➔ Color histogram moments
    - Histograms are distributions right?
    - So: mean, variance, skewness, and kurtosis of the color channels

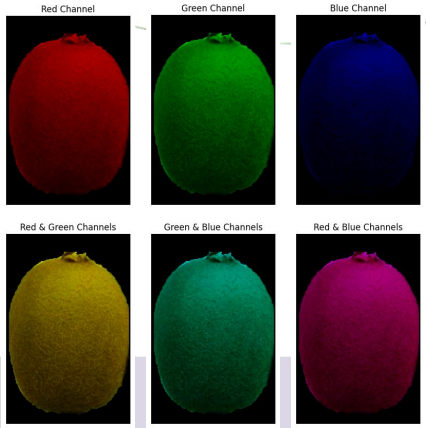


Feature engineering on visual data

- Color analysis
  - ★ Do all instances of ripe kiwis have similar colors?
  - ➔ Color histograms



More about “color” this afternoon!



  - ➔ Color histogram moments
    - Histograms are distributions right?
    - So: mean, variance, skewness, and kurtosis of the color channels

Feature engineering on visual data

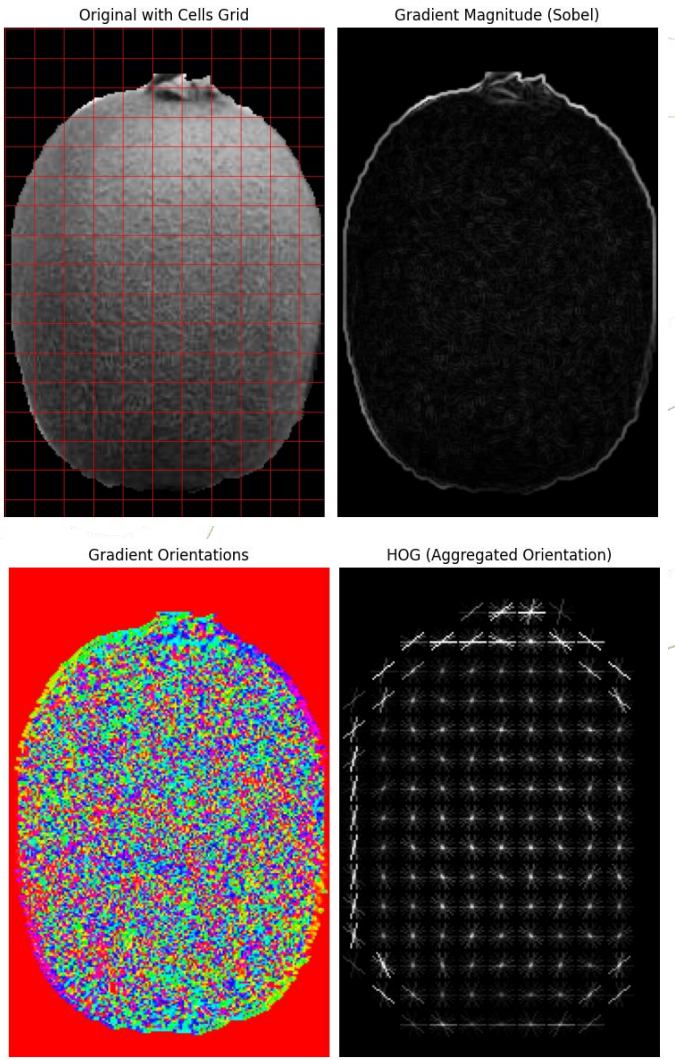
- Local feature descriptors
  - ★ Distinctive information from specific regions or *keypoints*

Feature engineering on visual data

- Local feature descriptors
  - ★ Distinctive information from specific regions or *keypoints*
  - ➔ Histogram of Oriented Gradients (HOG)
    - ◆ Divides the image into *uniformly spaced* **subregions** and computes **local gradient** information

HOG - Example step-by-step

1. Divides the image into small subregions
   a. e.g. 8x8 or 16x16 pixels
2. It computes gradient magnitudes
   a. gradient=change in color intensity of neighboring pixels
3. Within each subregion, histogram of pixel-level gradient information
   a. For each pixel: **magnitude** and **orientation** of the gradient (i.e. change in color intensity of neighboring pixels)
   b. These info put into (9-binned) histogram
4. Cells aggregated into blocks (e.g. 2x2) and normalized → avg orientation/magnitude info
   a. Each block → 36-D vector



Feature engineering on visual data

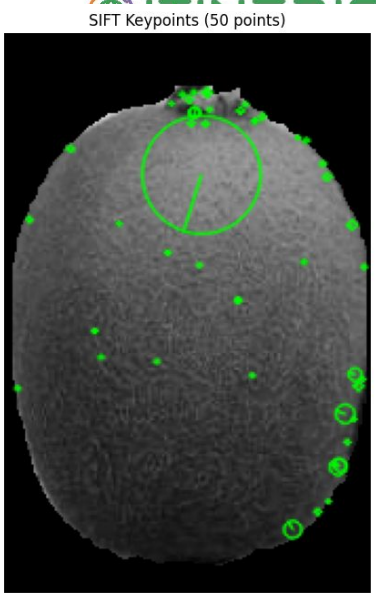
- Local feature descriptors
  - ★ Distinctive information from specific regions or *keypoints*
  - ➔ Histogram of Oriented Gradients (HOG)
    - ◆ Divides the image into *uniformly spaced* **subregions** and computes **local gradient** information
    - ◆ Final descriptor becomes **huge**

Feature engineering on visual data

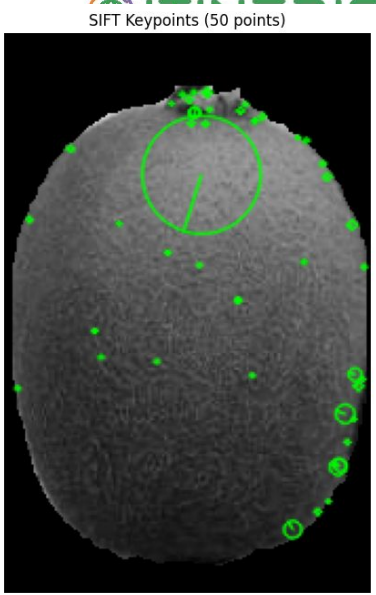
- Local feature descriptors
  - ★ Distinctive information from specific regions or *keypoints*
  - ➔ Scale-Invariant Feature Transform (SIFT)
    - ◆ Keypoints which capture “details” of the image

Feature engineering on visual data

- Local feature descriptors
  - ★ Distinctive information from specific regions or *keypoints*
  - ➔ Scale-Invariant Feature Transform (SIFT)
    - ◆ **Keypoints** which capture “details” of the image
      - “Details” do **not** change even though the image is *rotated*, *scaled*, or *subject to illumination change*
      - Often: borders, edges of the object, points with strong variation across all directions



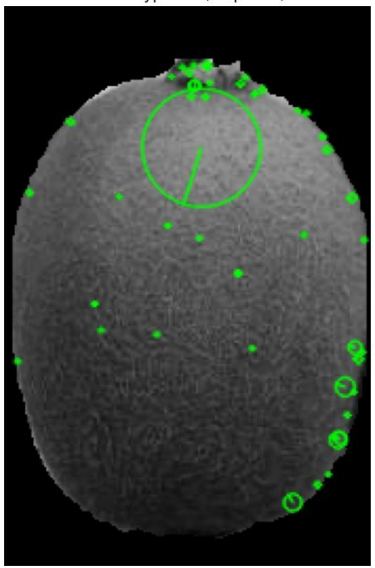
Feature engineering on visual data



- Local feature descriptors
  - ★ Distinctive information from specific regions or *keypoints*
  - ➔ Scale-Invariant Feature Transform (SIFT)
    - ◆ **Keypoints** which capture “details” of the image
      - “Details” do **not** change even though the image is *rotated*, *scaled*, or *subject to illumination change*
      - Often: borders, edges of the object, points with strong variation across all directions
    - ◆ For each keypoint, a 128-D *descriptor* is computed
      - Picks patch around the keypoint
      - Divided into subregions (4x4…)
      - Within each, HOG-like (…x8)

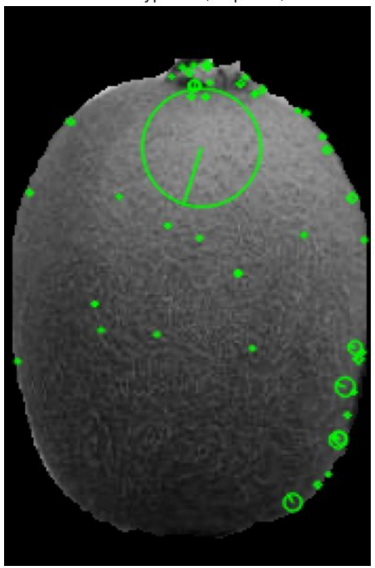
Feature engineering on visual data

- Local feature descriptors
  - ★ Distinctive information from specific regions or *keypoints*
  - ➔ Histogram of Oriented Gradients (HOG)
    - ◆ Works on the **whole** image (*global* info)
  - ➔ Scale-Invariant Feature Transform (SIFT)
    - ◆ Works on **keypoints** (*local* info)



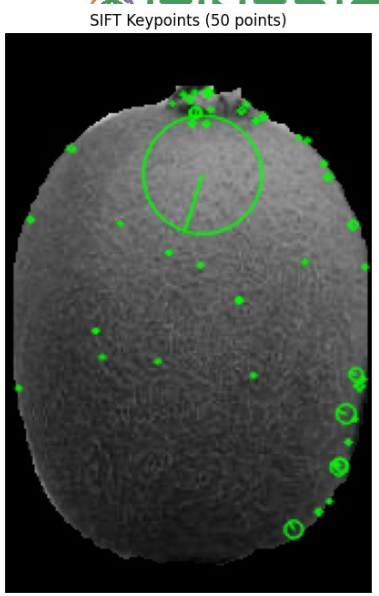
Feature engineering on visual data

- Local feature descriptors
  - ★ Distinctive information from specific regions or *keypoints*
  - ➔ Histogram of Oriented Gradients (HOG)
    - Works on the **whole** image (*global* info)
    - Not really invariant to scale/rotation
  - ➔ Scale-Invariant Feature Transform (SIFT)
    - Works on **keypoints** (*local* info)
    - Scale/rotation/illumination invariant



Feature engineering on visual data

- Local feature descriptors
  - ★ Distinctive information from specific regions or *keypoints*
  - ➔ Histogram of Oriented Gradients (HOG)
    - Works on the **whole** image (*global* info)
    - Not really invariant to scale/rotation
    - 36-D per block *but #blocks scales with image size*
  - ➔ Scale-Invariant Feature Transform (SIFT)
    - Works on **keypoints** (*local* info)
    - Scale/rotation/illumination invariant
    - “Fixed”-size (128D per keypoint)



Limitations of feature engineering

- Relies on domain expert
  - Eccentricity might be useful for kiwi ripeness detection, HOG might be pointless
- Does not generalize well
  - Useful features for kiwi likely useless for pears or apples
  - (i.e., feature engineering needs to be done again)
- Requires quite some effort
  - Need to analyze and manually try different techniques, different features, etc
- Some nuances might be invisible to these methods
  - Very peculiar aspects of the image that the human eye does not perceive (and thus does not design a technique for)
  - Deep Learning might come in help

Just a remark

- While **Deep Learning** is becoming a staple method for *automating* the features extraction process, feature engineering can still play an important role when dealing with small datasets or when having expert insights into “what” we should put our focus on

Conclusion

Recap:

- Feature engineering extracts useful numerical information from raw data
- Tabular data
  - Statistical features: mean/median aggregations, trends in windows, ...
  - Derived features: combination of variables usually expert-driven
- Images
  - Shape features: contour and region-based
  - Texture features: GLCM, Gabor
  - Color features: Histograms and derived features
  - Local features: HOG, SIFT

Conclusion

Recap:

- Feature engineering extracts useful numerical information from raw data
- Tabular data
  - Statistical features: mean/median aggregations, trends in windows, …
  - Derived features: combination of variables usually expert-driven
- Images
  - Shape features: contour and region-based
  - Texture features: GLCM, Gabor
  - Color features: Histograms and derived features
  - Local features: HOG, SIFT
- How are these info used?
  - **Concatenated** → they become like **observations** in tables
  - Then, fed to ML models → [more on ML models tomorrow!]

THANKS!

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani