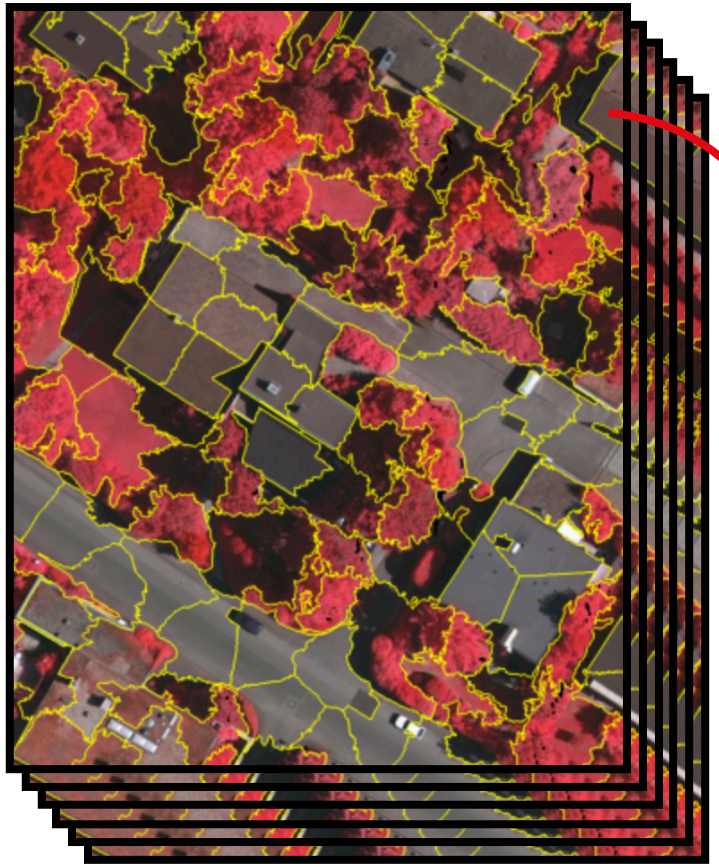
# Image classification - Part 1

Prof. Devis Tuia
Gaston Lenczner, Thien-Anh Nguyen,
Christel Chappuis

ENV-540 | October, 2024

# Extract regions and their features

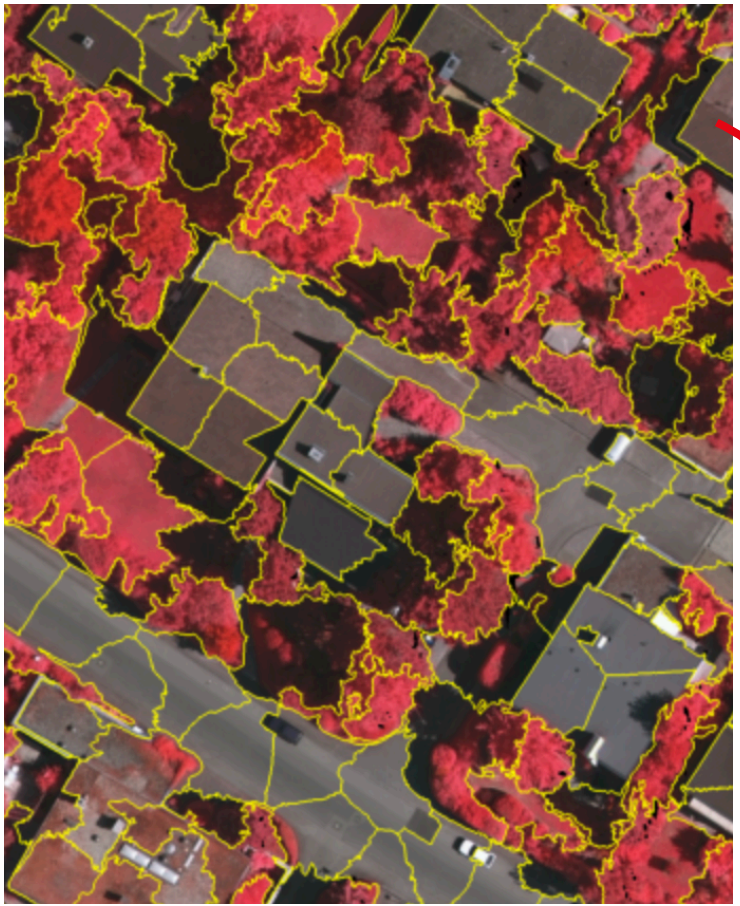




**Features**

$$\mathbf{x}_i = [\mathbf{x}_i^{\text{av}} \quad \mathbf{x}_i^{\text{std}} \quad \mathbf{x}_i^{\text{hist}}]$$

# Train set: Image features and targets for each region



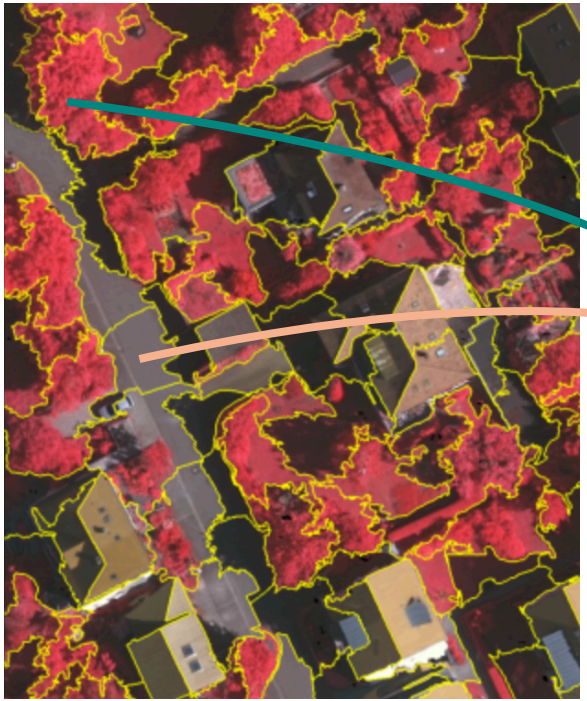
## Features

$$\mathbf{x}_i = [\mathbf{x}_i^{\text{av}} \quad \mathbf{x}_i^{\text{std}} \quad \mathbf{x}_i^{\text{hist}}]$$

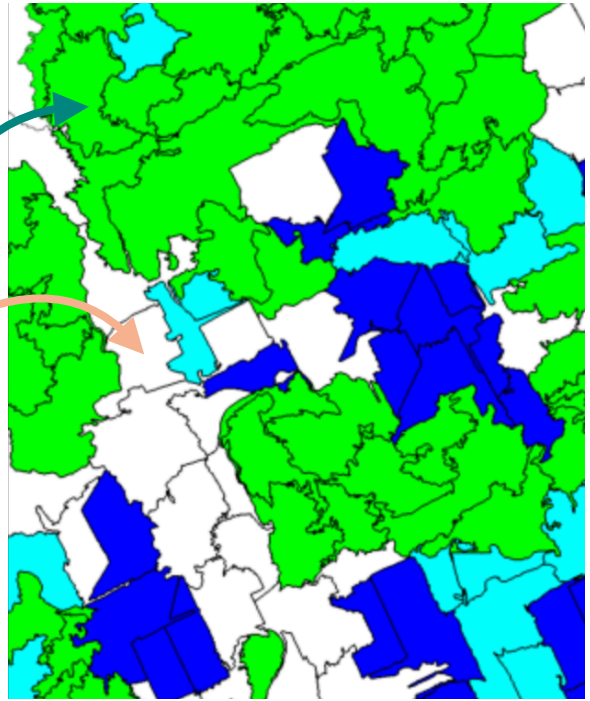
## Ground Truth / Reference / Target

0: Impervious
1 : Building
2: Lower vegetation
3: Tree

# Inference



Machine Learning Model



Impervious
Building
Low vegetation
Tree

# Example: Reading data to train a Random Forest classifier



```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
# Load data example
iris_dataset = load_iris()
features = iris_dataset["data"]
labels = iris_dataset["target"]
# Get features and targets of the train and test sets
train_features, test_features, train_labels, test_labels = train_test_split(features,
                                                                            labels,
                                                                            test_size=0.33,
                                                                            random_state=42)
```

```
# Shape of train features (2D array)
print(train_features.shape)
```

```
(100, 4)
```

```
# Shape of train labels (1D array)
print(train_labels.shape)
```

```
(100,)
```

# Example: Training a Random Forest classifier

## Training a classifier

```
from sklearn.ensemble import RandomForestClassifier

# Normalize features
mean_array = np.mean(train_features, axis=0)
std_array = np.std(train_features, axis=0)
norm_train_features = (train_features - mean_array) / std_array

# Create random forest classifier
classifier = RandomForestClassifier(random_state=10)

# Fit model
classifier.fit(norm_train_features, train_labels)
```

## Prediction

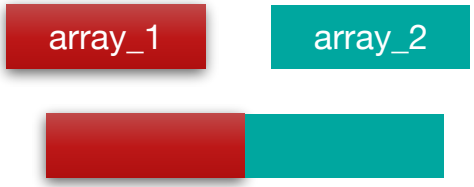
```
# Normalize features
norm_test_features = (test_features - mean_array) / std_array

# Predict labels for the test set
label_predictions = classifier.predict(norm_test_features)
```

# Exercise



- Read the provided PDF file and Jupyter Notebook with detailed instructions
- Tasks:
    1. Install dependencies, if needed
    2. Extract regions and their features
    3. Create a training dataset, with the following classes
        - 0: Impervious
        - 1: Building
        - 2: Low vegetation
        - 3: Tree
    4. Normalize features
    5. Train Random Forest classifier
    6. Predict classification maps
    7. Visualize predictions
    8. Answer the questions of the PDF file of instructions

# Concatenate arrays



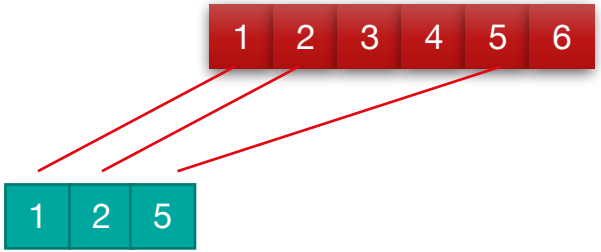

```
import numpy as np

array_1 = np.array([10, 70 , 60])
array_2 = np.array([50, 150 , 200])

list_of_arrays = [array_1, array_2]
print(list_of_arrays)
```

```
[array([10, 70, 60]), array([ 50, 150, 200])]
```

```
merged_array = np.concatenate(list_of_arrays)

print(merged_array)
```

```
[ 10  70  60  50 150 200]
```

# randomly split train test

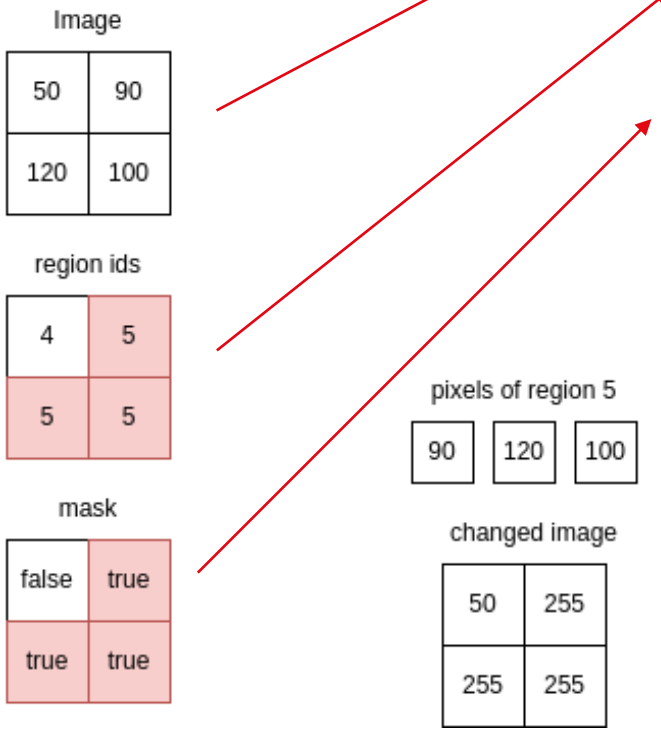



```
train_image_numbers = np.random.choice([1, 2, 3, 4, 5, 6],
                                       size=3,
                                       replace=False)
```


```
test_image_numbers = []
for i in all_image_numbers:
    if i not in train_image_numbers:
        test_image_numbers.append(i)
```

```
# alternative one liner
test_image_numbers = [i for i in all_image_numbers
                      if i not in train_image_numbers]
```

# Change Values at selected locations




```
image = np.array([[  50,  90],
                  [ 120, 100]])

regions = np.array([[4, 5],
                    [5, 5]])
```

```
mask_region_5 = regions == 5
print(mask_region_5)
```

```
[[False  True]
 [ True  True]]
```

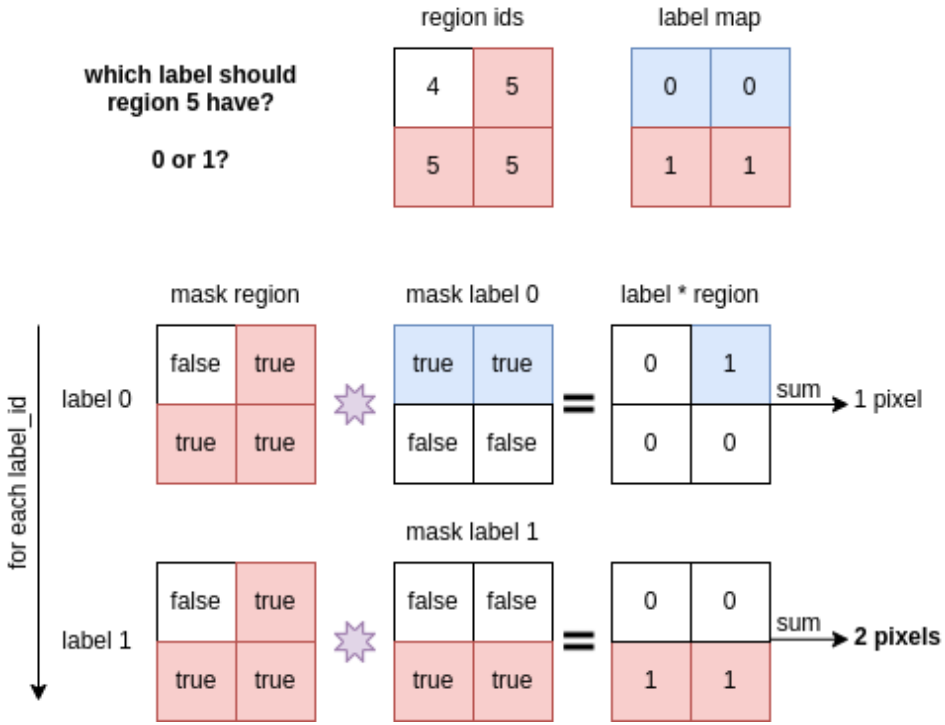
```
# Select pixels of the region 5
print(image[mask_region_5])
```

```
[ 90 120 100]
```

```
# Change value of pixels of the region 5
image[mask_region_5] = 255

print(image)
```

```
[[ 50 255]
 [255 255]]
```

# Find label for region




```
regions = np.array([[4, 5],
                    [5, 5]])

label_map = np.array([[0, 0],
                      [1, 1]])
```

```
# Computing the best label for region 5
mask_region_5 = regions == 5
num_labels = 2

intersection_per_label = []
for label_id in range(num_labels):
    mask_label = label_map == label_id
    # Compute intersection of each region with each label
    intersection = np.sum(mask_region_5 * mask_label)

    intersection_per_label.append(intersection)

intersection_per_label = np.array(intersection_per_label)
print(intersection_per_label)
```

```
[1 2]
```

```
# Obtain the index of the label with largest intersection
selected_label = np.argmax(intersection_per_label)
print(selected_label)
```

```
1
```