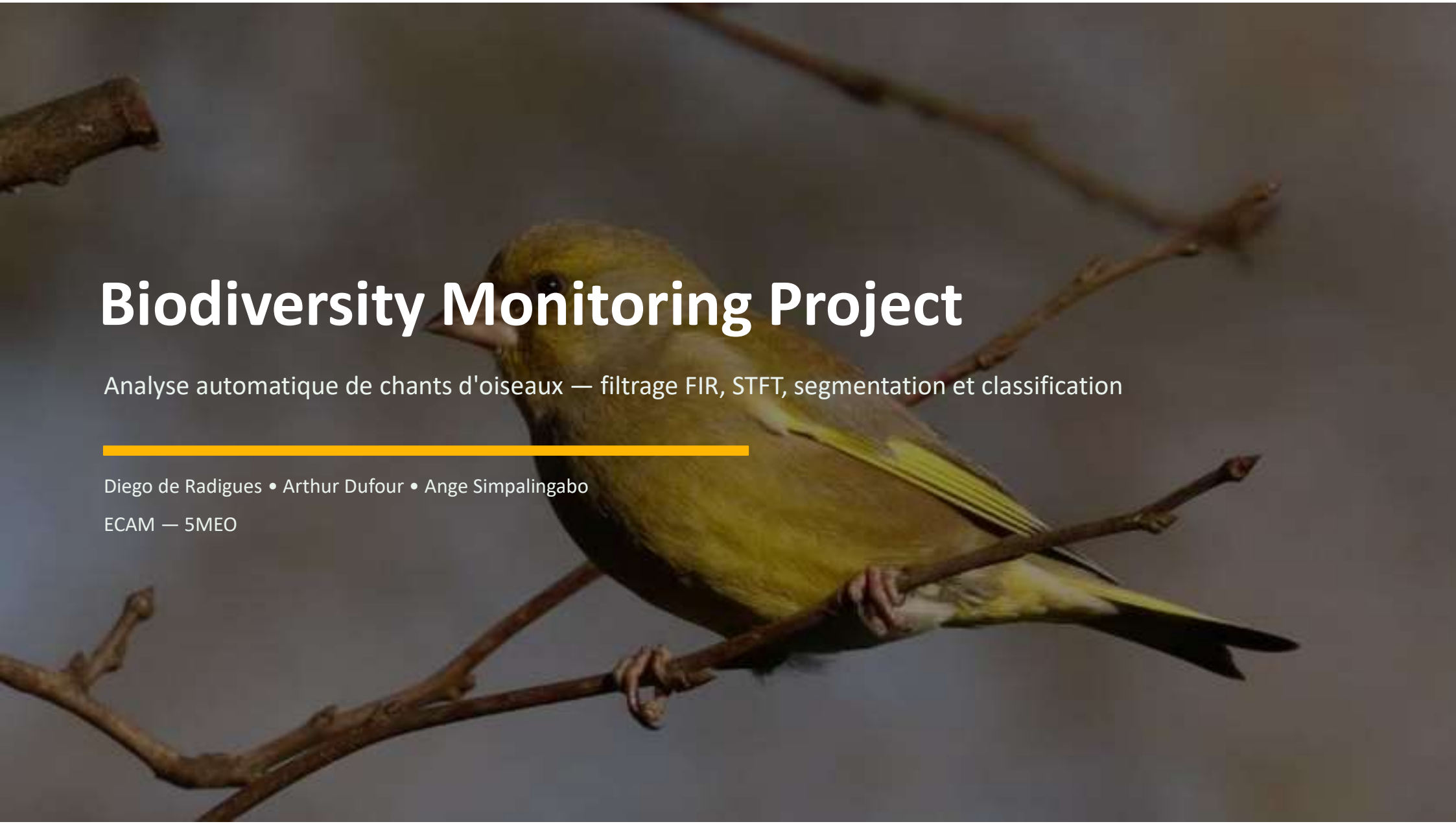


Biodiversity Monitoring Project

Analyse automatique de chants d'oiseaux — filtrage FIR, STFT, segmentation et classification

Diego de Radigues • Arthur Dufour • Ange Simpalingabo

ECAM — 5MEO

Contexte & objectifs

Objectif : construire un pipeline reproductible pour analyser des .wav de chants d'oiseaux et prédire l'espèce dominante.

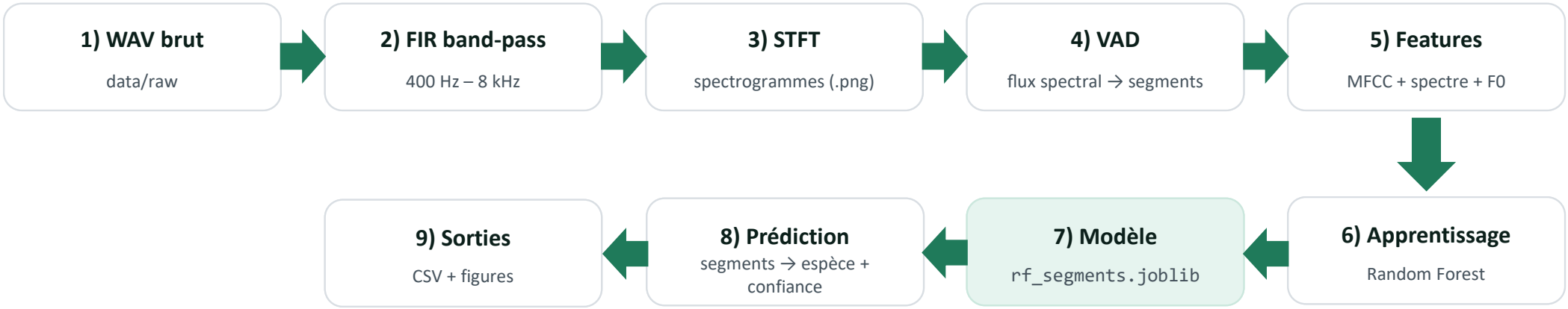
Ce que fait le pipeline

- Chargement automatique des enregistrements (.wav)
- Filtrage FIR passe-bande (400 Hz – 8 kHz)
- Spectrogrammes STFT et sauvegarde des figures
- Détection d'activité (VAD) par flux spectral → segments
- Extraction de features par segment (MFCC, stats spectrales, F0)
- Classification par segments (Random Forest) + prédiction

Livrables

- segments.csv (onset/offset)
- features.csv (features agrégées)
- figures : spectrogrammes & segments
- rf_segments.joblib (modèle entraîné)
- scripts “prêts à lancer”

Pipeline global



Principaux fichiers produits

Audio filtré : `data/processed/*_bandpass.wav`

Spectrogrammes : `assets/figures/*_spectrogram.png`

Segments : `experiments/exp1/segments.csv`

Features : `experiments/exp1/features.csv`

Modèle + résultats : `rf_segments.joblib` + `predictions_segments.csv` + `confusion_matrix.png`

Prétraitement : filtrage FIR + STFT

Pourquoi filtrer ?

- Réduire les basses fréquences (vent, trafic, manipulations)
- Conserver la bande informative pour les chants (≈ kHz)
- Choix FIR + filtfilt : phase globale nulle (pas de décalage)

Paramètres (config.py)

```
TARGET_SR = 22050 Hz
BANDPASS_LOW = 400 Hz
BANDPASS_HIGH = 8000 Hz
FIR_NUMTAPS = 1025
STFT_N_FFT = 1024
STFT_HOP_LENGTH = 256
STFT_WINDOW = hann
```

Extrait : STFT + figure

```
S_db, freqs, times = compute_spectrogram(
  y_filt, sr=sr,
  n_fft=1024, hop_length=256,
  window="hann"
)

save_spectrogram_figure(
  S_db, sr=sr,
  hop_length=256,
  out_path=fig_out_path
)
```

STFT : DFT sur fenêtres
→ représentation temps-fréquence

Détection d'activité : flux spectral

Idée

- Calculer |STFT| (magnitude) frame par frame
- Flux spectral = variation entre frames (augmentations positives)
- Seuillage adaptatif : médiane + k·σ
- Conversion en segments (t_onset, t_offset) + fusion des pauses courtes

Paramètres clés

k (sensibilité) • min_duration_s • merge_gap_s

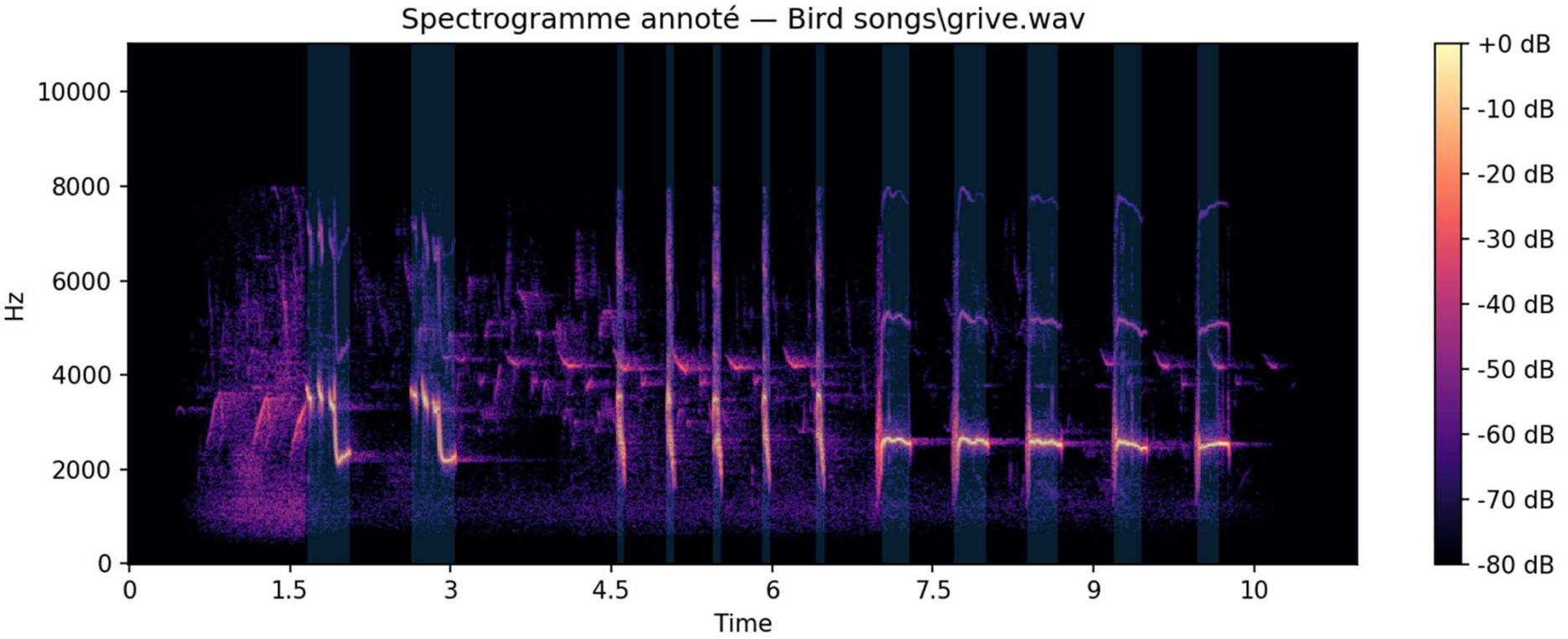
→ à régler selon le bruit & les espèces

Pseudo-code

```
S = abs(stft(y))
flux = sqrt(sum(max(diff(S),0)^2))
th = median(flux) + k*std(flux)
activity = flux > th
segments = runs_to_segments(activity)
segments = merge_close(segments, gap=0.10)
segments = drop_short(segments, min_dur=0.08)
```

Sortie : segments.csv (onset/offset)

Exemple : spectrogramme annoté + segments



Bandes verticales = segments détectés (VAD). Utilisées ensuite pour extraire les features par segment.

Extraction de features (par segment)

Features calculées

MFCC	20 coefficients + stats (mean/std/min/max)
Spectre	centroid • bandwidth rolloff95 • flatness • entropie
F0	YIN : f0_mean/std/min/max + ratio voiced
Métadonnées	file • segment_id t_onset/t_offset • durée

Sortie : features.csv

Exemples segments (console)

```
=== Fichier : Bird songs\grive.wav ===
  - segment 0 : 1.660s -> 2.055s (durée ~ 0.395s)
  - segment 1 : 2.635s -> 3.042s (durée ~ 0.406s)
  - segment 2 : 4.551s -> 4.621s (durée ~ 0.070s)
  - segment 3 : 5.016s -> 5.085s (durée ~ 0.070s)
  - segment 4 : 5.445s -> 5.526s (durée ~ 0.081s)
  - segment 5 : 5.909s -> 5.979s (durée ~ 0.070s)
  - segment 6 : 6.409s -> 6.490s (durée ~ 0.081s)
  - segment 7 : 7.024s -> 7.279s (durée ~ 0.255s)
  - segment 8 : 7.697s -> 7.999s (durée ~ 0.302s)
  - segment 9 : 8.382s -> 8.673s (durée ~ 0.290s)
  - segment 10 : 9.195s -> 9.451s (durée ~ 0.255s)
  - segment 11 : 9.973s -> 10.170s (durée ~ 0.197s)
```

Classification par segments (Random Forest)

Modèle

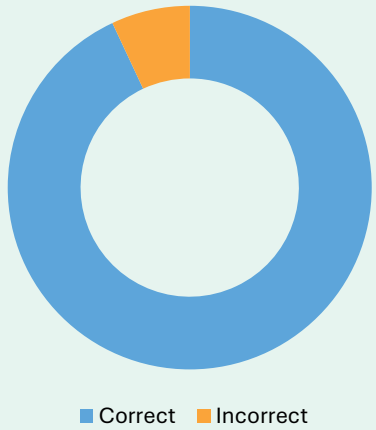
- Pipeline : StandardScaler → RandomForestClassifier
- Entraînement sur les segments (features.csv)
- Labels : espèce = nom du fichier (Path(stem))
- n_estimators = 300, random_state = 42

```
clf = Pipeline(
  steps=[
    ("scaler", StandardScaler()),
    ("rf", RandomForestClassifier(
      n_estimators=300,
      random_state=42,
      n_jobs=-1
    )),
  ]
)
```

Résultats (test)

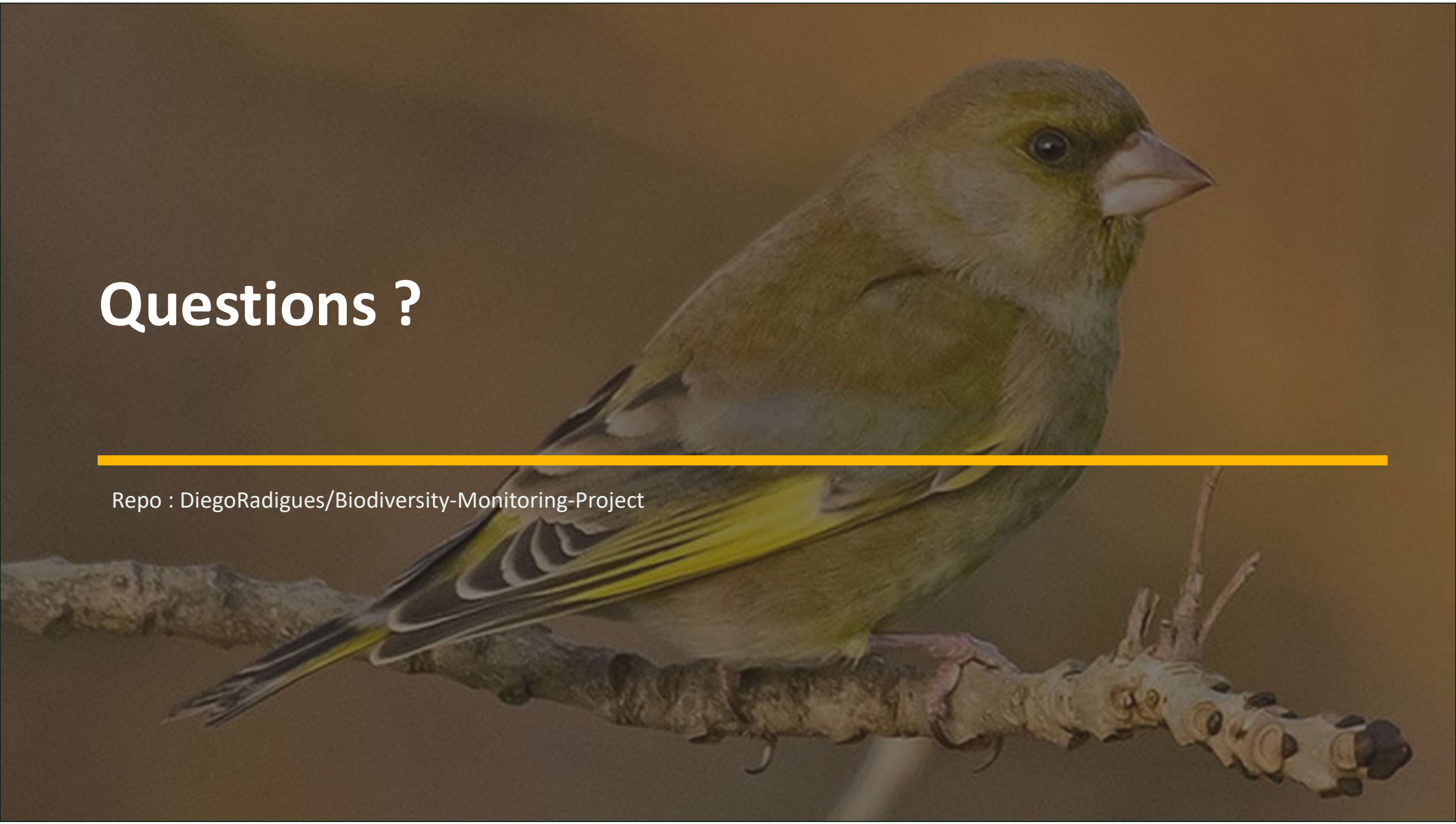
Accuracy ≈ 0.93

Macro F1 ≈ 0.91



Observations

- Peu de données → gros risque de sur-apprentissage
- Quelques confusions sur classes rares



Questions ?

Repo : DiegoRadigues/Biodiversity-Monitoring-Project