rf_to_debug

December 5, 2025

[41]:
```
import pickle
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
```

[42]:
```
# Chemins vers les fichiers
train_path = "ift-3395-6390-kaggle-2-competition-fall-2025/train_data.pkl"
```

[43]:
```
# Charger les données
with open(train_path, "rb") as f:
    train_data = pickle.load(f)
```

[44]:
```
X_train = train_data["images"]
y_train = train_data["labels"].reshape(-1)
```

[45]:
```
X_tr, X_val, y_tr, y_val = train_test_split(
    X_train, y_train, test_size=0.2, random_state=42, stratify=y_train
)
```

[46]:
```
# --- Rééchantillonnage simple pour les classes 1,2,3,4 ---
classes_to_augment = [1, 2, 3, 4]
X_extra, y_extra = [], []
for cls in classes_to_augment:
    idx = np.where(y_train == cls)[0]
    X_extra.append(X_train[idx])
    y_extra.append(y_train[idx])
```

[47]:
```
X_tr = np.concatenate([X_tr] + X_extra)
y_tr = np.concatenate([y_tr] + y_extra)
```

[48]:
```
def extract_features(img_array, n_bins=8, n_circles=3):
    h, w, _ = img_array.shape
    cy, cx = h//2, w//2  # centre
    Y, X = np.ogrid[:h, :w]
    radius = np.sqrt((X - cx)**2 + (Y - cy)**2)
    max_radius = radius.max()
    features = []

    # Couleur globale et luminance
    r, g, b = img_array[:,:,0], img_array[:,:,1], img_array[:,:,2]
    features.extend([r.mean(), g.mean(), b.mean()])
    lum = (0.299*r + 0.587*g + 0.114*b).mean()
    features.append(lum)

    # Stats et histogrammes pour chaque cercle
    for i in range(n_circles):
        mask = (radius >= i*max_radius/n_circles) & (radius < (i+1)*max_radius/
  ↪n_circles)
        for ch in [r, g, b]:
            vals = ch[mask]
            if len(vals) == 0:
                vals = np.array([0])
            # stats locales
            features.extend([vals.mean(), vals.var(), vals.min(), vals.max()])
            # histogramme
            hist, _ = np.histogram(vals, bins=n_bins, range=(0,255))
            features.extend(hist / (vals.size if vals.size>0 else 1))
    return np.array(features)
```

[49]:

```
# --- Extraire features ---
X_tr = np.array([extract_features(img) for img in X_tr])
X_val = np.array([extract_features(img) for img in X_val])
```

[50]:

```
# --- RandomForest avec class_weight='balanced' ---
clf = RandomForestClassifier(n_estimators=100, class_weight='balanced',␣
 ↪random_state=42)
clf.fit(X_tr, y_tr)
```

[50]:

```
RandomForestClassifier(class_weight='balanced', random_state=42)
```

[51]:

```
# --- Évaluer sur la validation ---
pred_val = clf.predict(X_val)
print("=== Classification report sur la validation ===")
print(classification_report(y_val, pred_val, digits=3))
```

```
=== Classification report sur la validation ===
              precision    recall  f1-score   support

           0      1.000     0.722     0.838        97
           1      0.812     1.000     0.897        26
           2      0.774     1.000     0.872        41
           3      0.812     1.000     0.897        39
           4      1.000     1.000     1.000        13
```

```
    accuracy                         0.875       216
   macro avg      0.880      0.944      0.901       216
weighted avg      0.901      0.875      0.872       216
```

[52]:

```
# --- Nombre de prédictions par classe sur validation ---
unique, counts = np.unique(pred_val, return_counts=True)
print("\nNombre de prédictions par classe sur validation :")
for u, c in zip(unique, counts):
    print(f"Classe {u}: {c} images")
```

```

Nombre de prédictions par classe sur validation :
Classe 0: 70 images
Classe 1: 32 images
Classe 2: 53 images
Classe 3: 48 images
Classe 4: 13 images
```

[53]:

```
test_path  = "ift-3395-6390-kaggle-2-competition-fall-2025/test_data.pkl"


with open(test_path, "rb") as f:
    test_data = pickle.load(f)

X_test = test_data["images"]

X_test_feat  = np.array([extract_features(img) for img in X_test])
```

[54]:

```
# --- Prédire le test ---
pred_test = clf.predict(X_test_feat)
```

[55]:

```
# Nombre de prédictions par classe sur test
unique_test, counts_test = np.unique(pred_test, return_counts=True)
print("\nNombre de prédictions par classe sur test :")
for u, c in zip(unique_test, counts_test):
    print(f"Classe {u}: {c} images")

# --- Créer le CSV pour soumission ---
results_test = [{"ID": idx, "Label": int(label)} for idx, label in␣
 ↪enumerate(pred_test, start=1)]
df_test = pd.DataFrame(results_test)
df_test.to_csv("manual_classification_all_augmented_test.csv", index=False)
print("\nCSV test créé avec succès !")
```

```

Nombre de prédictions par classe sur test :
Classe 0: 179 images
Classe 1: 45 images
Classe 2: 102 images
Classe 3: 68 images
Classe 4: 6 images

CSV test créé avec succès !
```