Règles générales

Objectif

Construire un classifieur d'images basé **uniquement sur la couleur** et comparer deux représentations :

- **Modèle RGB** (histogrammes R,G,B)
- **Modèle HSV** (histogrammes H,S,V)
  Puis : **tester sur une image nouvelle** et **valider** par plusieurs métriques.

Contraintes & format rendu

- **Classes** : mer, paysage (verdoyant), desert (sable/roche).
- **Rendu** : 1 notebook (ou script) + **rapport >10 pages** + quelques **captures** de votre dataset.
- **Taille d'entrée** : **128×128** (redimensionner en Python).
- **Bins fixés (obligatoire)**
  - **RGB** : R=16, G=16, B=16 bins sur [0,256) → 48 valeurs
  - **HSV** : **H=36 bins** sur [0,180) (OpenCV), **S=16**, **V=16** bins sur [0,256) → 68 valeurs
  - Tous les **histogrammes sont normalisés** (somme = 1)

Étape 1 — Collecte & organisation du dataset

Objectif. Constituer un jeu équilibré et propre.

Tâches

1. Collecter **≥ 10 images par classe**. Variantes d'éclairage et d'angles, éviter les doublons.
2. Créer l'arborescence :

data/

mer/

paysage/

desert/

3. Redimensionner toutes les images en **128×128** (le faire avec python lors du traitement).

Critères d'acceptation

- ≥ 30 images au total (équilibrées).
- Images lisibles, non floues, formats .jpg/.png. (vérifier l'extension avec python)
- Arborescence et noms de fichiers clairs. (mer/ mer1.jpg, mer2.jpg, mer3.png, etc...)

À rendre

- Dossier data/... + un paragraphe sur la collecte (5–10 lignes).

Questions (répondre dans le rapport)

- Q1. Quelles consignes avez-vous appliquées pour éviter les biais de collecte ?
- Q2. Pourquoi un **équilibrage** des classes est-il important?
- Q3. Quel type d'images de "paysage" risque de ressembler à la mer ? Comment l'éviter dès la collecte ?

Étape 2 — Exploration et visualisation

Objectif. Comprendre la distribution des couleurs.

Tâches

1. Charger **1 image par classe** et afficher :
    - image
    - histogrammes **RGB** (3 canaux *16 bins)
    - **HSV** (**H=36**, **S=16**, **V=16** bins).
    - Faire une recherche sur les bins et intégrer un paragraphe explicatif (rapport)
2. Pour chaque histogramme, normaliser (somme = 1).
3. Afficher aussi les **moyennes/écarts-types** de H, S, V pour quelques images.

Critères d'acceptation

- Figures lisibles, axes et légendes présents.
- Normalisation explicitement appliquée avant comparaison.

À rendre

- 3 figures (une par classe) : image + histos RGB & HSV.
- Tableau récapitulatif (moyenne/σ) pour H, S, V (≥ 3 images par classe).

Questions

- Q4. Quelle composante (R,G,B ; H,S,V) paraît la plus discriminante ? Justifiez avec vos courbes.
- Q5. Pourquoi comparer en HSV plutôt qu'en RGB pour des scènes naturelles ?
- Q6. Quelles limites à la couleur observez-vous déjà ?
- Q4-bis. Pourquoi fixer la même granularité de bins avant comparaison ? (recherche demandée)

- Q4-ter. Que perd-on si on passe de H=36 à H=12 bins ? Et que risque-t-on avec H=72 ?

Étape 3 — Intervalles de couleurs & vecteurs de features

Tâches

1. Déduisez des histogrammes des **plages de teinte dominantes** (ex. indicatif) :
   - Mer → H ≈ **[100, 140)**
   - Paysage → H ≈ **[50, 90)**
   - Désert → H ≈ **[10, 40)**
     (Ajustez selon vos données ; notez vos choix dans le rapport.)
2. Implémentez `extract_color_features(img)` qui renvoie :
   - **Modèle RGB** :
     - hist R(16) + G(16) + B(16) **normalisés** → 48
     - **moyenne & écart-type** par canal (6 scalaires)
       → vecteur RGB : **54** dimensions
   - **Modèle HSV** :
     - hist H(36) + S(16) + V(16) **normalisés** → 68
     - **moyenne & écart-type** H,S,V (6 scalaires)
     - **ratios de teintes** : % pixels dans vos **bandes H** mer/paysage/désert (3 scalaires)
       → vecteur HSV : **77** dimensions
3. Construisez `X_rgb, X_hsv, y` pour **toutes** les images.
4. Appliquez **StandardScaler** (indépendant pour RGB et HSV).

Critères d'acceptation

- `X_rgb.shape[0] == X_hsv.shape[0] == nb_images`, `y` aligné.
- Histogrammes **normalisés**, **scaling** appliqué avant SVM.

À rendre

- Code de `extract_color_features`.
- Impression de `X_rgb.shape`, `X_hsv.shape`, `y.shape` + aperçu de 3 vecteurs.

Questions

- Q7. Pourquoi **normaliser** les histogrammes **avant** d'empiler les features ?
- Q8. Quel compromis faites-vous avec **H=36, S=16, V=16** (précision vs bruit) ?
- Q9. Que **complètent** les moments (moyenne/σ) par rapport aux histogrammes ?

Étape 4 — Split, pipeline & entraînement SVM

Tâches

1. Split **train/test = 80/20** avec **stratification** (ou 5-fold CV).
2. Deux **pipelines** séparés :

- **RGB** : StandardScaler → (Option: PCA 0.95) → SVC(kernel='rbf')
- **HSV** : StandardScaler → (Option: PCA 0.95) → SVC(kernel='rbf')

3. **GridSearchCV** sur train uniquement :
   - C ∈ {1, 10, 100}, gamma ∈ {1e-3, 1e-2, 1e-1}
4. Évaluez sur **test** : accuracy, **matrice de confusion**, classification_report (precision/rappel/F1).

Critères d'acceptation

- **Aucune fuite** : scaler/PCA/SVM **fit** sur train **seulement**.
- Meilleurs hyperparams affichés (pour RGB et pour HSV).

À rendre

- Meilleurs C, gamma (RGB & HSV), score validation.
- Matrices de confusion (RGB & HSV) + métriques par classe.

Questions

- Q10. Quel couple (C, gamma) retenu pour **RGB** et pour **HSV** ? Pourquoi ?
- Q11. Quelles **paires de classes** sont le plus souvent confondues ? Exemple concret si rencontré !
- Q12. Quels risques avec **C élevé** et **gamma grand** ?

Étape 5 — Baselines & contrôle

Tâches

1. **Règle Hue** : si % pixels(H ∈ bande_classe) dépasse un **seuil** → prédiction.
2. **SVM RGB-only** (48 bins + moments) vs **SVM HSV** (68 bins + moments + ratios H).
3. Comparez les scores **sur le même test**.

Critères d'acceptation

- 3 **scores** reportés : Règle Hue / SVM RGB / SVM HSV (mêmes métriques).
- Commentaire sur l'écart entre règle simple et SVM.

À rendre

- Tableau comparatif (au min : Accuracy, Macro-F1). (Expliquer Macro-F1)
- Identifier si possible, 2–3 **images pièges** (mal classées par la règle Hue mais bien par SVM, ou inversement).

Questions

- Q13. Dans quels cas la **règle Hue** dépasse le SVM ? Pourquoi ?
- Q14. Pourquoi le SVM en **HSV** est-il souvent **supérieur** au SVM en RGB ?

- Q15. Votre dataset **favorise-t-il** une classe ? Comment le contrôlez-vous ?

Étape 6 — Test les modèles sur une nouvelle image (hors dataset)

Tâches

1. Choisissez 1 **image externe** (hors dataset).
2. Appliquez **exactement** les mêmes prétraitements et extraction de **features**.
3. Prédisez avec le **meilleur modèle** (souvent HSV).
4. Affichez l'image + la **classe prédite**.

À rendre

- Image affichée + prédiction.
- Court commentaire (2 phrases) sur la cohérence du résultat.

Questions

- Q16. Pourquoi est-il essentiel de tester sur une **image inconnue** ?
- Q17. Que conclure si le résultat est **incohérent** malgré de bons scores test ?

Étape 7 — Validation multi-métriques & rapport final

Tâches

1. Calculez (sur test) : **Accuracy**, **Macro-Precision**, **Macro-Recall**, **Macro-F1** pour **RGB** et **HSV**.
2. Résumez en **tableau** + **matrices de confusion**.

Rapport (5–8 pages, structure)

1. **Objectif & classes** ; protocole de collecte
2. **Exploration couleur** ; histogrammes (bins fixés)
3. **Intervalles H** retenus ; **features RGB vs HSV**
4. **Entraînement** (pipelines, grille, meilleurs params)
5. **Résultats** (métriques, confusions, analyse)
6. **Test image nouvelle** (résultat & commentaire)
7. **Conclusion** (limites & pistes : texture/forme/segmentation)

Questions de synthèse

- Q18. Quelle **différence clé** entre RGB et HSV pour les scènes naturelles ?
- Q19. Quelle est la **leçon principale** sur le rôle de la **couleur** ?
- Q20. Quelle **amélioration prioritaire** ajouterez-vous (texture, forme, segmentation) et pourquoi ?

Mini scripts pouvant vous aider à avancer :

Histogrammes (bins fixés)

```
# RGB 16 bins
r = cv2.calcHist([img],[2],None,[16],[0,256]); r = r / r.sum()
g = cv2.calcHist([img],[1],None,[16],[0,256]); g = g / g.sum()
b = cv2.calcHist([img],[0],None,[16],[0,256]); b = b / b.sum()
# HSV H=36, S=16, V=16
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
h = cv2.calcHist([hsv],[0],None,[36],[0,180]); h = h / h.sum()
s = cv2.calcHist([hsv],[1],None,[16],[0,256]); s = s / s.sum()
v = cv2.calcHist([hsv],[2],None,[16],[0,256]); v = v / v.sum()
```

Ratios de teinte (ex. mer/paysage/désert)

```
H = hsv[:,:,0]  # 0..179
def ratio_range(H, lo, hi):  # [lo = lower bound, hi = higher bound)
        mask = (H >= lo) & (H < hi)
        return mask.sum() / H.size
rat_mer    = ratio_range(H, 100, 140)  # pixels bleus (mer)
rat_paysage = ratio_range(H,  50,  90) # pixels verts (végétation)
rat_desert  = ratio_range(H,  10,  40) # pixels jaunes/bruns (sable)
```

Pipelines SVM (RGB & HSV)

```
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV, train_test_split
pipe = Pipeline([ (
"scaler", StandardScaler()),
  # ("pca", PCA(0.95)),  # optionnel
  ("svm", SVC(kernel="rbf"))
])
param_grid = {"svm__C":[1,10,100], "svm__gamma":[1e-3,1e-2,1e-1]}
grid_rgb = GridSearchCV(pipe, param_grid, cv=5, n_jobs=-1)
grid_hsv = GridSearchCV(pipe, param_grid, cv=5, n_jobs=-1)
```