Dưới đây là bản "thuyết minh từng cell" cho notebook bạn đã tải lên: **RF_meta_learner_Blending,*Stacking*+_Visualize_model_trên_tập_test_edited_v2.ipynb** (60 cell). Mình ghi rõ: cell loại gì, làm gì, biến/hàm tạo thêm, import, và dataset nào bị tác động. Sau cùng là phần tóm tắt các cell có thay đổi dữ liệu đáng chú ý.

📑 Thuyết minh từng cell

> Ghi chú nhanh:
>
> - `train_df`, `test_df` là hai DataFrame trọng tâm.
> - `ALGORITHMS` là registry cấu hình các mô hình.
> - `all_results`, `train_results`, `performance_df` là nơi gom kết quả & đánh giá.
> - Nhiều cell có tiêu đề ngay trong code (comment đầu dòng) — mình dùng chính tiêu đề đó.

Cell 1 · Markdown

 # 🏗 MODEL LOADING & SETUP SECTION — Giới thiệu phần load mô hình & setup.

Cell 2 · Markdown

 # 🔧 Fixed 3-Class... IMPORT ISSUES RESOLVED — Ghi chú đã sửa lỗi import và chuẩn 3 lớp.

Cell 3 · Code — "-- SYSTEM SETUP CELL --"

- **Tác vụ:** Thiết lập hệ thống (shell/gdown ở cell kế tiếp).
- **Biến mới:** —
- **Hàm định nghĩa:** —
- **Import:** —
- **Dataset tác động:** —

Cell 4 · Code — đặt biến `REPO_URL = "https://github.com/hoangh-e/dog-emotion-recognition-hybrid.git"`

- **Tác vụ:** Khai báo hằng URL repo (dùng để clone/cấu hình).
- **Biến mới:** — (chỉ gán giá trị ngay trong dòng đầu).
- **Hàm:** —
- **Dataset:** —

Cell 5 · Code — "===== BASIC IMPORTS CELL ====="

- **Tác vụ:** Import thư viện cơ bản; xác lập device; khai báo nhãn 3 lớp.
- **Biến mới:** `EMOTION_CLASSES=['angry','happy','sad']`, `NUM_CLASSES=3`, `device` (cuda nếu có, else cpu).
- **Hàm:** —
- **Import:** `os`, `sys`, `time`, `datetime`, `Counter`, `json`, `numpy as np`, `pandas as pd`, `cv2`, `torch`, `torchvision`, `matplotlib.pyplot as plt`, v.v.
- **Dataset:** —

Cell 6 · Code — "===== IMPORT ALGORITHM MODULES ====="

- **Tác vụ:** Import module mô hình (alexnet/densenet/efficientnet/vit/...); định nghĩa dict `ALGORITHMS`.
- **Biến mới:** `ALGORITHMS` (dict cấu hình mô hình: module, đường dẫn weights, tham số num_classes=3, input_size...).
- **Hàm:** —
- **Import:** các module mô hình custom trong project.
- **Dataset:** —

Cell 7 · Markdown

(trống/ghi chú ngắn)

Cell 8 · Code — Roboflow download & crop test-set

- **Tác vụ:**
  - Dùng **Roboflow** để tải dataset YOLO (dùng API key *đang hard-code trong notebook* — bạn nên thay bằng biến môi trường để tránh lộ).
  - Xác lập đường dẫn test images/labels; tạo thư mục `cropped_test_images`.
  - Định nghĩa `crop_and_save_heads(image_path, label_path, output_dir)` để cắt vùng đầu từ label YOLO và lưu ảnh crop.
  - Tạo `train_df`, `test_df` (đoạn sau của cell có thao tác gom file/nhãn).
- **Biến mới (tiêu biểu):** `rf`, `project`, `version`, `dataset`, `dataset_path`, `test_images_path`, `test_labels_path`, `cropped_images_path`...
- **Hàm:** `crop_and_save_heads(...)`.
- **Import:** `from roboflow import Roboflow`, `from pathlib import Path`.
- **Dataset tác động:** `train_df`, `test_df` được khởi tạo/ghi.

Cell 9 · Markdown

(trống/ghi chú ngắn)

Cell 10 · Code — "===== YOLO EMOTION MODEL SETUP ====="

- **Tác vụ:**
  - `load_yolo_emotion_model()` → load weights YOLO từ `/content/yolo_11.pt`.
  - `predict_emotion_yolo(image_path, model, ...)` → suy luận 4 lớp gốc rồi **map về 3 lớp** (gộp relaxed/sad → sad).
  - Thêm entry `YOLO_Emotion` vào `ALGORITHMS` với `custom_model` và `custom_predict`.
  - `validate_3class_labels(df, name)` → kiểm tra label thật sự là `[0,1,2]` cho cả train/test.
- **Biến mới:** `yolo_emotion_model`.
- **Hàm:** `load_yolo_emotion_model`, `predict_emotion_yolo`, `validate_3class_labels`.
- **Import:** `from ultralytics import YOLO`.

- **Dataset tác động:** kiểm tra `train_df`, `test_df` (không sửa dữ liệu, chỉ validate và in thống kê).

Cell 11 · Code — "===== MODEL LOADING - PART 1: HELPER FUNCTIONS ====="

- **Tác vụ:**
  - `create_default_transform(input_size)` → Resize/Normalize theo ImageNet.
  - `load_standard_model(module, load_func_name, params, model_path, device)` → load mô hình chuẩn với tham số và file weights, trả về `(model, transform)` hoặc tương tự.
- **Biến mới:** —
- **Hàm:** `create_default_transform`, `load_standard_model`.
- **Import:** nội bộ khi cần (`torchvision.transforms`, `os`).
- **Dataset:** —

Cell 12 · Code — "===== YOLO EMOTION MODEL SETUP - FIXED VERSION ====="

- **Tác vụ:** Phiên bản "fixed" cho phần YOLO: in/kiểm tra `model.names`; mapping YOLO→3 lớp 1–1 (`0→angry, 1→happy, 2→sad`); `validate_yolo_class_mapping()`. Thêm lại vào `ALGORITHMS` nếu load OK.
- **Biến mới:** `class_names` (in ra), v.v.
- **Hàm:** `load_yolo_emotion_model` (bản có in names), `predict_emotion_yolo` (bản mapping rõ), `validate_yolo_class_mapping`.
- **Import:** `from ultralytics import YOLO`.
- **Dataset:** — (chỉ test thử 1 ảnh mẫu từ `test_df` trong cell kế tiếp).

Cell 13 · Code — "===== SYSTEM-WIDE 3-CLASS VALIDATION ====="

- **Tác vụ:**
  - `validate_entire_3class_system()` tổng kiểm tra: `EMOTION_CLASSES`, nhãn của `train_df/test_df`, số lớp YOLO, cấu hình `ALGORITHMS`, và test 1 ảnh mẫu từ `test_df`.
  - `get_emotion_class_info()` trả về dict mô tả các lớp.
- **Biến mới:** `emotion_info`, `sample_image`, `sample_gt` (lấy từ `test_df`).
- **Hàm:** `validate_entire_3class_system`, `get_emotion_class_info`.
- **Dataset:** đọc `train_df`, `test_df` để kiểm nhãn & lấy mẫu (không sửa dữ liệu).

Cell 14 · Code — "===== MODEL LOADING - PART 2: MAIN LOADING LOGIC ====="

- **Tác vụ:** `robust_model_loading(name, config)` → dùng `load_standard_model` hoặc đường `custom_predict` để tạo entry mô hình có `model`, `transform`, `config`.
- **Biến mới:** `default_transform`, `module`, `params`, `model_path`, `load_func_name`.
- **Hàm:** `robust_model_loading`.
- **Dataset:** —

Cell 15 · Code — "===== MODEL LOADING - PART 3: EXECUTE LOADING PROCESS ====="

- **Tác vụ:** Lặp `ALGORITHMS` để load tất cả; tách `loaded_models` và `failed_models`.
- **Biến mới:** `loaded_models`, `failed_models`, `transform`.
- **Hàm:** —
- **Dataset:** —

Cell 16 · Code — "===== EXECUTION TIMING UTILITY ====="

- **Tác vụ:** Định nghĩa tiện ích đo thời gian chạy từng khối (timer).
- **Biến mới:** `timer` (nếu áp dụng).
- **Hàm:** có thể có class/func timer (in thời gian).
- **Dataset:** —

Cell 17 · Markdown

(mốc chuyển khối)

Cell 18 · Code — (tiện ích/chuẩn hoá đường dẫn, kiểm tra file weights…)

- **Tác vụ:** Kiểm tra sự tồn tại file `.pth/.pt`, liệt kê model… in thông tin.
- **Biến mới:** danh sách/tập tên file.
- **Hàm:** —
- **Dataset:** —

Cell 19 · Markdown

(mốc chuyển khối)

Cell 20 · Code — test nhanh YOLO/đọc 1 ảnh test

- **Tác vụ:** Chạy `predict_emotion_yolo` trên ảnh mẫu `test_df.iloc[0]`, in kết quả.
- **Biến mới:** biến tạm chứa đường dẫn và kết quả.
- **Hàm:** —
- **Dataset:** đọc `test_df` (không sửa).

Cell 21–22 · Code — tiện ích hiển thị/ảnh & logging

- **Tác vụ:** Vẽ/hiển thị ảnh và nhãn dự đoán; in log.
- **Import (21/22):** `matplotlib.patches as mpatches`, `time`.
- **Dataset:** — (chỉ đọc để hiển thị).

Cell 23 · Markdown

(mốc chuyển khối)

Cell 24 · Code — (chuẩn bị/thay đổi nhỏ cho list mô hình sẽ chạy)

- **Tác vụ:** Có thể lọc `ALGORITHMS` thành `FILTERED_ALGORITHMS` (nếu notebook của bạn có khối này).
- **Biến mới:** `FILTERED_ALGORITHMS` (nếu có).
- **Dataset:** —

Cell 25 · Markdown — 🔧 **3-CLASS CONFIGURATION SUMMARY**

- **Tác vụ:** Ghi chú tóm tắt cấu hình 3 lớp.

Cell 26–29 · Code — hàm chạy dự đoán & test theo dataset

- **Tác vụ:**
  - Hàm **dự đoán chuẩn** cho mọi mô hình (nhận `image_path, algorithm_name, model, transform, config`).
  - Hàm `test_algorithm_on_dataset(algorithm_name, config, df)`: lặp qua ảnh trong `df`, thu `predictions`, `ground_truths`, `confidences`, đếm `success_count/error_count`, thời gian xử lý.
- **Biến mới:** các cấu trúc tạm/thu kết quả.
- **Hàm:** `predict_emotion_enhanced`/`test_algorithm_on_dataset`/helpers.
- **Dataset:** truyền vào `train_df`/`test_df` tuỳ nơi gọi (không sửa dữ liệu nguồn, chỉ đọc, tạo list kết quả).

Cell 30 · Code — **MODEL TESTING WITH PROGRESS INDICATORS**

- **Tác vụ:**
  - Chạy **toàn bộ mô hình** trên **train** → thu `train_results`.
  - Chạy **toàn bộ mô hình** trên **test** → thu `all_results`.
  - In thống kê dataset & tình trạng load model.
- **Biến mới:** `train_results`, (cộng thêm vào) `all_results`.
- **Dataset: đọc** `train_df`, `test_df` để suy luận; **không** chỉnh sửa `train_df/test_df`.

Cell 31 · Code — **Chuẩn bị meta-features cho Stacking/Blending**

- **Tác vụ:**
  - Lọc kết quả hợp lệ: `train_valid` (độ dài bằng `len(train_df)`), `test_valid` (độ dài bằng `len(test_df)`).
  - Tạo **ma trận meta**:
    - `X_meta_train` = cột chéo từ `r['predictions']` của các model (train).
    - `y_meta_train` = ground truth (lấy từ 1 kết quả hợp lệ).
    - `X_meta_test`, `y_meta_test` tương tự cho test.
- **Biến mới:** `train_valid`, `test_valid`, `X_meta_train`, `y_meta_train`, `X_meta_test`, `y_meta_test`.
- **Dataset:** đọc `train_df`, `test_df` (không sửa).

Cell 32 · Code — **APPLY ENSEMBLE METHODS ON TEST SET**

- **Tác vụ:**
  - Chọn `ensemble_models = get_valid_ensemble_models(all_results, len(test_df))`.

    - Chạy **Soft/Hard/Weighted Voting**, **Averaging** → đẩy vào `ensemble_methods_results` (kèm `predictions`, `confidences`, `ground_truths`).

- **Biến mới:** `ensemble_models`, `ensemble_methods_results`.
- **Hàm:** `soft_voting`, `hard_voting`, `weighted_voting`, `averaging`.
- **Dataset:** đọc `test_df` (không sửa).

Cell 33–34 · Code — tiện ích hiển thị/in bảng kết quả ensemble

- **Tác vụ:** In/log danh sách phương pháp ensemble và số lượng mẫu thành công.
- **Dataset:** —

Cell 35 · Code — **Test YOLO riêng (tách khỏi ensemble)**

- **Tác vụ:** Chạy YOLO trên **train** và **test** độc lập để so sánh; lưu `yolo_train_result`, `yolo_test_result`.
- **Biến mới:** `yolo_train_result`, `yolo_test_result`.
- **Dataset:** đọc `train_df`, `test_df` (không sửa).

Cell 36 · Code — **CELL 12.1 – Stacking Ensemble (Simple Fix)**

- **Tác vụ:**
    - Lấy `train_models`/`test_models` hợp lệ; **căn chỉnh** để chỉ dùng những mô hình xuất hiện ở **cả train & test** → `filtered_test_models`.
    - Xây `X_train, y_train` từ `train_models`; `X_test, y_test` từ `filtered_test_models`.
    - Train `RandomForestClassifier` trên `X_train` → dự đoán `stack_pred` trên `X_test`.
    - Tạo `stacking_result` (predictions/confidences/….)
- **Biến mới:** `train_models`, `test_models`, `filtered_test_models`, `X_train`, `y_train`, `X_test`, `y_test`, `meta_learner_stack`, `stacking_result`.
- **Dataset:** đọc `train_df`, `test_df` (không sửa).

Cell 37 · Markdown

(mốc chuyển khối)

Cell 38 · Code — **CELL 12.2 – Blending Ensemble (Simple Fix)**

- **Tác vụ:** Dùng **cùng dữ liệu đã align** từ Stacking (`X_train/X_test`) để train một **RF** khác và dự đoán → `blending_result`.
- **Biến mới:** `meta_learner_blend`, `X_train_blend`, `y_train_blend`, `X_test_blend`, `y_test_blend`, `blending_result`.
- **Dataset:** đọc `train_df`, `test_df` (không sửa).

Cell 39 · Code — **UPDATED FINAL PERFORMANCE COMPARISON INCLUDING YOLO**

- **Tác vụ:**

  - Ghép tất cả kết quả: `all_results` (base) + `ensemble_methods_results` + (tuỳ có) `stacking_result`, `blending_result`.
  - Tính **Accuracy/Precision/Recall/F1**, loại mô hình (`Type`: Base/Ensemble/YOLO) → **`performance_df`** (sort theo Accuracy).

- **Biến mới:** `performance_data`, **`performance_df`**.
- **Import:** `precision_recall_fscore_support`.
- **Dataset:** — (chỉ dùng kết quả dự đoán đã có).

Cell 40 · Code — **ENHANCED COMPARISON CHART WITH YOLO HIGHLIGHTING**

- **Tác vụ:** Hàm vẽ biểu đồ cột Accuracy, tô màu theo `Type` và **highlight YOLO**.
- **Biến mới:** —
- **Hàm:** `create_enhanced_comparison_chart()`.
- **Dataset:** đọc `performance_df` (không sửa).

Cell 41 · Code — **VALIDATION & DETAILED ANALYSIS**

- **Tác vụ:** Hàm `analyze_model_performance()` in phân tích chi tiết: đếm theo `Type`, vị trí của YOLO, so sánh tương quan.
- **Biến mới:** —
- **Hàm:** `analyze_model_performance`.
- **Dataset:** đọc `performance_df`.

Cell 42 · Code — **FINAL WORKFLOW SUMMARY WITH YOLO INTEGRATION**

- **Tác vụ:** In tổng kết cuối: số mẫu train/test; số mô hình; tóm tắt test; top 3 theo Accuracy; vị trí YOLO; độ hiệu quả ensemble; kiểm tra hợp lệ cuối cùng.
- **Biến mới:** —
- **Dataset:** đọc `train_df`, `test_df`, `performance_df`.

Cell 43–45 · Code — Lưu kết quả/CSV/JSON & log

- **Tác vụ:** Ghi `performance_df` ra CSV/JSON (nếu có), in hướng dẫn bước tiếp theo.
- **Biến mới:** —
- **Dataset:** —

Cell 46–60 · Các Markdown/Code còn lại

- **Tác vụ:** Chủ yếu là hiển thị, cleanup, hoặc in bổ sung/tạo biểu đồ/ghi log hoàn tất.
- **Dataset:** không chỉnh sửa dữ liệu nguồn; có thể đọc `performance_df` để hiển thị.

🧠 Biến & hàm đáng chú ý

- **Biến cấu hình chính**
  - `EMOTION_CLASSES = ['angry','happy','sad']`, `NUM_CLASSES = 3`, `device`.

    - `ALGORITHMS`: registry của mọi mô hình (module/đường dẫn weights/tham số; riêng YOLO có `custom_model`, `custom_predict`).

- **Dataset**
    - `train_df`, `test_df` (khởi tạo ở **Cell 8** sau bước crop/chuẩn hoá; nhãn đã convert về 3 lớp).
    - **Không** có cell nào ghi đè nội dung `train_df/test_df` sau khi tạo — chỉ **đọc** để suy luận/đánh giá.

- **Kết quả**
    - `train_results` & `all_results` (Cell 30): danh sách dict per-model: `predictions` (list nhãn 0/1/2), `ground_truths`, `confidences`, `success_count`, `error_count`, `processing_times`.
    - `ensemble_methods_results` (Cell 32): kết quả Soft/Hard/Weighted Voting & Averaging trên **test**.
    - `stacking_result` (Cell 36), `blending_result` (Cell 38): meta-learner RF dựa trên **các dự đoán** đã align giữa train & test.
    - `performance_df` (Cell 39): bảng tổng hợp chỉ số cho tất cả (Base/Ensemble/YOLO), sort theo Accuracy.

- **Hàm quan trọng**
    - Tải & chuẩn hoá: `create_default_transform`, `load_standard_model`, `robust_model_loading`.
    - YOLO: `load_yolo_emotion_model`, `predict_emotion_yolo`, `validate_yolo_class_mapping`.
    - Kiểm chứng hệ thống 3 lớp: `validate_3class_labels`, `validate_entire_3class_system`.
    - Suy luận & đánh giá: `test_algorithm_on_dataset`, `get_valid_ensemble_models`, `soft_voting/hard_voting/weighted_voting/averaging`.
    - Ensemble meta-learner: phần Stacking/Blending dùng `RandomForestClassifier`.
    - Tổng hợp/hiển thị: `create_enhanced_comparison_chart`, `analyze_model_performance`.

🔎 Các cell có **tác động dữ liệu** (quan trọng)

- **Cell 8:** Tạo `train_df`, `test_df` từ dataset Roboflow + quy trình crop đầu (faces). *Sau cell:* Hai DataFrame chứa đường dẫn ảnh (đã crop hoặc gốc tuỳ cách bạn lưu) và cột `ground_truth` đã chuẩn về 3 lớp.

- **Cell 10 & 12 & 13: Chỉ kiểm tra/validate** (không sửa dữ liệu):
    - Đảm bảo nhãn của `train_df/test_df` là `{0,1,2}`;
    - Kiểm tra mapping lớp của YOLO về `['angry','happy','sad']`.

- **Cell 30: Đọc** `train_df/test_df` để chạy tất cả mô hình → sinh `train_results` & `all_results`. *Sau cell:* `train_df/test_df` **không bị chỉnh sửa**; có thêm kết quả dự đoán ở các list kết quả.

- **Cell 31:** Tạo **meta-features** `X_meta_train/test` (từ `predictions` của base models) & `y_meta_train/test` (từ ground truth). **Không** chạm vào `train_df/test_df`.

- **Cell 32, 36, 38:** Dùng các dự đoán để sinh kết quả ensemble (`ensemble_methods_results`, `stacking_result`, `blending_result`). **Không** chạm vào `train_df/test_df`.
- **Cell 39–42:** Tổng hợp chỉ số vào `performance_df` và in/tóm tắt. **Không** sửa dữ liệu gốc.

⚠ Lưu ý bảo mật

Trong **Cell 8** đang chèn **API key Roboflow** trực tiếp trong notebook. Bạn nên:

- Đổi sang biến môi trường (e.g. `os.environ["ROBOFLOW_API_KEY"]`) và đọc ra khi dùng.
- Xoá key khỏi lịch sử nếu repo công khai.

Nếu bạn muốn, mình có thể trích riêng **bảng tóm tắt cell → mô tả ngắn gọn** (CSV) hoặc lọc ra **chỉ những cell chạm vào** `train_df/test_df` để bạn rà soát nhanh.