ДОДАТОК А

Код розробленої прогами

Програмний код основного файлу recommendation_app.py та файлу додаткових функцій (utils.py) наведено у лістингах нижче.

Лістинг Б.1 – Програмний код recommendation_app.py

```
import gradio as gr
import numpy as np
import pandas as pd
import cv2
import torch
from ultralytics import YOLO
import joblib
import json
from PIL import Image
import utils

model = joblib.load("xgb_kit_recommender_v2.pkl")
encoder = joblib.load("ohe_kit_encoder_v2_final.pkl")
with open("encoder_columns_v2_final.json", "r",
encoding="utf-8") as f:
    encoder_columns = json.load(f)

yolo =
YOLO("runs/detect/kit_detector_y8m/weights/best.pt")

def explain_recommendation(row, prediction, conf_score):
    contrast_score = row["contrast_score"]
    geom_score = row["geometry_score"]
    zone = row["zone_auto"]
    cls = row["class_name"]
    area_ratio = row["rel_area"]
```

```
        explanation = f"{utils.get_icon(prediction)} {row['class_name']} - "

        # Загальний висновок
        if prediction == 1:
            explanation += "☑ Рекомендовано – добре помітно, у зоні очікування, "
            explanation += "адекватний розмір" if area_ratio > 0.01 else "може бути замалий (<1%)"
        else:
            explanation += "⚠ Сумнівно – розмір малий або зона неочікувана або погана контрасність"

        # Деталізоване пояснення
        explanation += f"\n- Контрастність: {contrast_score:.2f} "
        explanation += "(добре, >50)" if contrast_score > 50 else "(низька, <50)"

        explanation += f" - Геометрія: {geom_score:.2f} "
        explanation += "(ідеальна симетрія)" if geom_score >= 0.8 else "(може бути перекос)"

        zone_label = f"{zone} ({'очікувана' if utils.check_expected_zone(cls, zone) else 'неочікувана'})"
        explanation += f" - Зона: {zone_label}"

        if area_ratio > 0.015:
            size_comment = "великий, добре видимий"
        elif area_ratio >= 0.01:
            size_comment = "адекватний"
        else:
            size_comment = "може бути замалий (<1%)"
        explanation += f" - Розмір: {size_comment}"
```

```
            return explanation

        def summarize_predictions(all_rows):
            df = pd.DataFrame(all_rows)
            total = len(df)
            recommended = df[df["recommended"] == 1]
            recommended_count = len(recommended)

            # Перенасичення по зонах
            zone_counts = df["zone_auto"].value_counts()
            overloaded_zones = [zone for zone, count in
zone_counts.items() if count > 2]
            overload_text = " ⚠️ Перенасичення у зонах: " + ",
".join(overloaded_zones) if overloaded_zones else ""

            return f"📋 Загальна оцінка: {recommended_count} /
{total} рекомендовано.{overload_text}"

        def analyze(image, league="persha"):
            image_np = np.array(image)
            results = yolo(image_np)[0]

            boxes = results.boxes.xyxy.cpu().numpy()
            classes = results.boxes.cls.cpu().numpy().astype(int)
            confs = results.boxes.conf.cpu().numpy()
            names = results.names

            annotated = image_np.copy()
            H, W = annotated.shape[:2]
            rows = []
            explanations = []

            decor_pattern_count = 0
```

```
        for idx, (box, cls_id, conf) in enumerate(zip(boxes,
classes, confs)):
            x1, y1, x2, y2 = map(int, box)
            w, h = x2 - x1, y2 - y1
            x_center, y_center = (x1 + x2) / 2 / W, (y1 + y2)
/ 2 / H
            area = w * h / (W * H)
            class_name = names[cls_id]
            zone = utils.assign_zone(x_center, y_center)

            if class_name == "decor_pattern":
                if area < 0.02 or conf < 0.4 or
decor_pattern_count > 2:
                    continue
                decor_pattern_count += 1

            roi = image_np[y1:y2, x1:x2]
            contrast_score = utils.compute_contrast(roi)
            geometry_score = utils.compute_geometry_score(w,
h)
            expected                                        =
int(utils.check_expected_zone(class_name, zone))

            row = {
                "class_name": class_name,
                "zone_auto": zone,
                "league": league,
                "x_center": x_center,
                "y_center": y_center,
                "rel_area": area,
                "confidence": conf,
                "element_id": idx,
                "contrast_score": contrast_score,
                "geometry_score": geometry_score,
                "zone_expected_match": expected
            }
```

```
            X_cat = encoder.transform([[class_name, zone,
league]])
            X_num = np.array([[x_center, y_center, area,
conf]])

            # Створення DataFrame
            X_cat_df = pd.DataFrame(X_cat,
columns=encoder.get_feature_names_out(["class_name",
"zone_auto", "league"]))

            # Об'єднання з числовими фічами
            X_combined = pd.concat(
                [X_cat_df.reset_index(drop=True),
                 pd.DataFrame(X_num, columns=["x_center",
"y_center", "rel_area", "confidence"]).reset_index(drop=True)],
                axis=1
            )

            # Дозаповнення відсутніх колонок
            for col in encoder_columns:
                if col not in X_combined.columns:
                    X_combined[col] = 0

            X_combined = X_combined[encoder_columns]


            prediction = int(model.predict(X_combined)[0])
            row["recommended"] = prediction

            conf_score =
model.predict_proba(X_combined)[0][prediction]
            explanation = explain_recommendation(row,
prediction, conf_score)
            explanations.append(explanation)
```

```
            if row["class_name"] == "emblem_club" and
row["zone_auto"] in ["bottom", "shoulder_sleeve_right"]:
                continue
            rows.append(row)

        summary = summarize_predictions(rows)
        img_draw = utils.draw_boxes_colored(image_np.copy(),
results, rows)

        explanations_text = "\n\n".join(explanations)
        return img_draw, explanations_text, summary


demo = gr.Interface(
    fn=analyze,
    inputs=[
        gr.Image(type="pil", label="Форма клубу"),
    ],
    outputs=[
        gr.Image(type="numpy", label="Візульний аналіз"),
        gr.Textbox(label="Пояснення", lines=20),
        gr.Text(label="Загальний висновок")
    ],
    title="Аналіз футбольної форми для рекомендацій
спонсорства"
)

if __name__ == "__main__":
    demo.launch()
```

Лістинг Б.2 – Програмний код utils.py

```
import cv2
import numpy as np
from skimage import color
```

```
    # Геометрична зона розташування
    def assign_zone(x_center, y_center):
        if y_center < 0.2:
            return "neck"
        elif y_center < 0.35:
            return "upper_chest"
        elif y_center < 0.5:
            return "mid_chest"
        elif y_center < 0.65:
            return "lower_chest"
        elif y_center >= 0.65:
            return "bottom"
        else:
            return "unknown"

    # Обчислення контрастності (ΔE в просторі Lab)
    def compute_contrast(roi):
        try:
            roi_lab = color.rgb2lab(roi.astype(np.uint8))
            center = roi_lab[roi_lab.shape[0] // 2,
roi_lab.shape[1] // 2]
            deltaE = np.linalg.norm(roi_lab - center, axis=2)
            return np.mean(deltaE)
        except Exception:
            return 0.0

    # Геометрія (симетричність) – співвідношення ширини до
висоти
    def compute_geometry_score(w, h):
        if w == 0 or h == 0:
            return 0.0
        ratio = min(w, h) / max(w, h)
        return round(ratio, 2)

    # Очікувана зона для класу
    def check_expected_zone(class_name, zone):
```

```
        expected = {
            "brand_main": ["mid_chest", "upper_chest"],
            "brand_secondary": ["mid_chest", "upper_chest"],
            "brand_shoulder":         ["shoulder_sleeve_left",
"shoulder_sleeve_right"],
            "decor_maker":
["upper_chest","shoulder_sleeve_left",
"shoulder_sleeve_right"],
            "decor_pattern":                          ["bottom",
"shoulder_sleeve_left", "shoulder_sleeve_right"],
            "decor_symbol": ["upper_chest", "neck"],
            "emblem_club": ["upper_chest", "mid_chest"],
            "emblem_federation": ["upper_chest", "neck"]
        }
        return zone in expected.get(class_name, [])

    # Іконка для prediction
    def get_icon(pred):
        return "☑ Рекомендовано" if pred == 1 else "⚠
Сумнівно"

    # Малювання кольорових боксів
    def draw_boxes_colored(image, results, rows):
        colors = {
            1: (0, 255, 0),    # зелений – рекомендовано
            0: (0, 165, 255)  # помаранчевий – сумнівно
        }

        boxes = results.boxes.xyxy.cpu().numpy().astype(int)
        for idx, (box, row) in enumerate(zip(boxes, rows)):
            x1, y1, x2, y2 = box
            label = row['class_name']
            pred = row["recommended"]
            color = colors.get(pred, (128, 128, 128))
```

```
            cv2.rectangle(image, (x1, y1), (x2, y2), color, 2)
            cv2.putText(image, label, (x1, y1 - 5),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.45, color,
1, cv2.LINE_AA)
        return image
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	75 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Нетеса К.В.			Рекомендаційна система вибору спонсорів для футбольних клубів	Шифр групи	Код напр./спец.
Перевірив		Дейнеко А.О.				ІТШІ-21-3	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Золотухін О.В.					