Código Modificado en PDF

```
# ■ CLASIFICACIÓN AVANZADA DE CÁNCER DE PECHO — Versión Optimizada
# (Código completo modificado)
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score, classification_report, confusion_matrix,
    roc_curve, auc, roc_auc_score, f1_score
)

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import warnings

warnings.filterwarnings("ignore")
plt.style.use("seaborn-v0_8-darkgrid")
sns.set_palette("husl")

def evaluar_modelo(nombre, modelo, X_train, X_test, y_train, y_test):
    modelo.fit(X_train, y_train)
    y_pred = modelo.predict(X_test)
    y_proba = modelo.predict_proba(X_test)[:, 1]

    accuracy = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    auc_score = roc_auc_score(y_test, y_proba)
    cv = cross_val_score(modelo, X_train, y_train, cv=5)

    return {
        "pred": y_pred,
        "proba": y_proba,
        "accuracy": accuracy,
        "f1": f1,
        "auc": auc_score,
        "cv_mean": cv.mean(),
        "cv_std": cv.std(),
    }

def imprimir_encabezado(titulo):
    print("=" * 80)
    print(" " * ((80 - len(titulo)) // 2) + titulo)
    print("=" * 80)

# Carga de datos
datos = load_breast_cancer()
X, y = datos.data, datos.target

df = pd.DataFrame(X, columns=datos.feature_names)
df["target"] = y

X_train, X_test, y_train, y_test = train_test_split(
    X, y, stratify=y, test_size=0.2, random_state=42
)

scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s = scaler.transform(X_test)

modelos = {
    "KNN": KNeighborsClassifier(5),
    "Random Forest": RandomForestClassifier(n_estimators=200, random_state=42),
    "SVM RBF": SVC(kernel="rbf", probability=True, random_state=42),
    "Logistic Regression": LogisticRegression(max_iter=2000, random_state=42),
```

```
    "Gradient Boosting": GradientBoostingClassifier(random_state=42)
}

resultados = {}

for nombre, modelo in modelos.items():
    resultados[nombre] = evaluar_modelo(nombre, modelo, X_train_s, X_test_s, y_train, y_test)

mejor = max(resultados, key=lambda m: resultados[m]["accuracy"])
```