

December 4, 2025

1 06 - HPairwise Stacking Hybrids (GA Models)

```
[9]: # =====
# 06 - Pairwise Stacking Hybrids (GA Models)
# Author: Nasser Qanabta
# Description:
# 1) Upload and load cleaned CKD dataset (CKD_cleaned_dataset_EN.csv)
# 2) Reproduce the same Train (70%) / Test (30%) stratified split
# 3) Upload and load GA-optimized base models:
#     - DT_GA_best_model.joblib
#     - RF_GA_best_model.joblib
#     - LR_GA_best_model.joblib
#     - SVM_GA_best_model.joblib
# 4) Build 6 pairwise stacking hybrids using:
#     - RF + DT
#     - RF + SVM
#     - RF + LR
#     - SVM + LR
#     - DT + SVM
#     - DT + LR
#     with Logistic Regression as meta-learner
# 5) Evaluate each hybrid on Train and Test:
#     Accuracy, Precision, Recall, F1, ROC-AUC + classification report
#     + Confusion matrix (Train/Test) saved as CSV and PNG
# 6) Save scores to CSV and reports to JSON
# 7) Download main results + main hybrid model
# =====

!pip install scikit-learn -q

# 1) Imports
import pandas as pd
import numpy as np
import json
import joblib
import os
```

```

from google.colab import files

from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    roc_auc_score,
    classification_report,
    confusion_matrix
)
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt

plt.style.use("seaborn-v0_8")

# -----
# 2) Upload and load cleaned CKD dataset
# -----
print("Please upload: CKD_cleaned_dataset_EN.csv")
uploaded = files.upload() # select CKD_cleaned_dataset_EN.csv

# Handle possible (1), (2) suffixes
ckd_filename = [fn for fn in uploaded if fn.
    ↳startswith("CKD_cleaned_dataset_EN")][0]
df = pd.read_csv(ckd_filename)
print("Full cleaned dataset shape:", df.shape)

# Separate features and target
target_col = "Chronic_Kidney_Disease"
X = df.drop(columns=[target_col])
y = df[target_col]

print("Features shape:", X.shape, "Target shape:", y.shape)

# -----
# 3) Train/Test split (70% / 30%) with stratification
# -----
RANDOM_STATE = 42

X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.30,
    stratify=y,

```

```

    random_state=RANDOM_STATE
)

print("\nStratified split (70/30):")
print("  X_train:", X_train.shape, "y_train:", y_train.shape)
print("  X_test :", X_test.shape, "y_test :", y_test.shape)

# -----
# 4) Upload and load GA-optimized base models
# -----
print("\nPlease upload GA-optimized base models:")
print("  - DT_GA_best_model.joblib")
print("  - RF_GA_best_model.joblib")
print("  - LR_GA_best_model.joblib")
print("  - SVM_GA_best_model.joblib")

uploaded_models = files.upload()

DT_MODEL_FILENAME = [fn for fn in uploaded_models if fn.
    ↪startswith("DT_GA_best_model")][0]
RF_MODEL_FILENAME = [fn for fn in uploaded_models if fn.
    ↪startswith("RF_GA_best_model")][0]
LR_MODEL_FILENAME = [fn for fn in uploaded_models if fn.
    ↪startswith("LR_GA_best_model")][0]
SVM_MODEL_FILENAME = [fn for fn in uploaded_models if fn.
    ↪startswith("SVM_GA_best_model")][0]

dt_ga = joblib.load(DT_MODEL_FILENAME)
rf_ga = joblib.load(RF_MODEL_FILENAME)
lr_ga = joblib.load(LR_MODEL_FILENAME)
svm_ga = joblib.load(SVM_MODEL_FILENAME)

print("\nGA-optimized base models loaded successfully:")
print(" -", DT_MODEL_FILENAME)
print(" -", RF_MODEL_FILENAME)
print(" -", LR_MODEL_FILENAME)
print(" -", SVM_MODEL_FILENAME)

# -----
# 5) Helper: evaluation + confusion matrices (CSV + PNG)
# -----
def evaluate_with_confusion(model, X, y, dataset_name="", hybrid_name=""):
    """
    Evaluate classifier on given X, y:
    - y_pred, y_prob
    - accuracy, precision, recall, f1, roc_auc
    - confusion matrix:

```

```

        * saved to CSV
        * saved as PNG image (no internal borders, no title)
    """
    y_pred = model.predict(X)
    y_prob = model.predict_proba(X)[: , 1]

    acc = accuracy_score(y, y_pred)
    prec = precision_score(y, y_pred, zero_division=0)
    rec = recall_score(y, y_pred, zero_division=0)
    f1 = f1_score(y, y_pred, zero_division=0)
    try:
        auc = roc_auc_score(y, y_prob)
    except ValueError:
        auc = None

    print(f"\nPerformance on {dataset_name} for {hybrid_name}:")
    print("Accuracy :", acc)
    print("Precision:", prec)
    print("Recall   :", rec)
    print("F1-score :", f1)
    if auc is not None:
        print("ROC-AUC :", auc)

    print("\nClassification report (4 decimals):")
    print(classification_report(y, y_pred, digits=4, zero_division=0))

    # Confusion matrix
    cm = confusion_matrix(y, y_pred)
    cm_df = pd.DataFrame(
        cm,
        index=[0, 1],
        columns=[0, 1]
    )

    # Safe suffix: e.g., Train_70 / Test_30
    suffix = dataset_name.replace(" ", "_").replace("(", "").replace(")", "").
    ↪replace("%", "")

    # 1) Save confusion matrix to CSV
    cm_csv_filename = f"{hybrid_name}_ConfusionMatrix_{suffix}.csv"
    cm_df.to_csv(cm_csv_filename, index=True)
    print(f"Confusion matrix CSV saved to: {cm_csv_filename}")

    # 2) Plot confusion matrix (no internal borders) and save PNG
    fig, ax = plt.subplots(figsize=(3, 3))

    im = ax.imshow(cm, interpolation="nearest", cmap=plt.cm.Blues)

```

```

fig.colorbar(im, ax=ax)

classes = [0, 1]
tick_marks = np.arange(len(classes))
ax.set_xticks(tick_marks)
ax.set_yticks(tick_marks)
ax.set_xticklabels(classes)
ax.set_yticklabels(classes)

ax.set_xlabel("Predicted label")
ax.set_ylabel("True label")

thresh = cm.max() / 2.0
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(
            j, i, format(cm[i, j], "d"),
            ha="center", va="center",
            color="white" if cm[i, j] > thresh else "black"
        )

fig.tight_layout()

cm_png_filename = f"{hybrid_name}_ConfusionMatrix_{suffix}.png"
fig.savefig(cm_png_filename, dpi=300, bbox_inches="tight")
print(f"Confusion matrix PNG saved to: {cm_png_filename}")

plt.show()

metrics = {
    "model": f"{hybrid_name}_{dataset_name}",
    "dataset": dataset_name,
    "accuracy": acc,
    "precision": prec,
    "recall": rec,
    "f1": f1,
    "roc_auc": auc
}

report = classification_report(
    y,
    y_pred,
    digits=4,
    zero_division=0,
    output_dict=True
)

```

```

    return metrics, report

# -----
# 6) Define pairwise stacking configurations
# -----
stacking_configs = [
    ("Hybrid_RF_DT_Stacking", [("rf_ga", rf_ga), ("dt_ga", dt_ga)]),
    ("Hybrid_RF_SVM_Stacking", [("rf_ga", rf_ga), ("svm_ga", svm_ga)]),
    ("Hybrid_RF_LR_Stacking", [("rf_ga", rf_ga), ("lr_ga", lr_ga)]),
    ("Hybrid_SVM_LR_Stacking", [("svm_ga", svm_ga), ("lr_ga", lr_ga)]),
    ("Hybrid_DT_SVM_Stacking", [("dt_ga", dt_ga), ("svm_ga", svm_ga)]),
    ("Hybrid_DT_LR_Stacking", [("dt_ga", dt_ga), ("lr_ga", lr_ga)]),
]

meta_learner = LogisticRegression(
    random_state=RANDOM_STATE,
    max_iter=2000,
    class_weight=None # or "balanced"
)

# -----
# 7) Train and evaluate all stacking hybrids
# -----
all_metrics = []
all_reports = {}

for hybrid_name, base_estimators in stacking_configs:
    print("\n" + "="*60)
    print(f"Training {hybrid_name}")
    print("="*60)

    stacking_clf = StackingClassifier(
        estimators=base_estimators,
        final_estimator=meta_learner,
        stack_method="auto",
        n_jobs=-1,
        passthrough=False
    )

    # Fit on TRAIN
    stacking_clf.fit(X_train, y_train)

    # Save main hybrid model (RF + DT) for later ROC and download
    if hybrid_name == "Hybrid_RF_DT_Stacking":
        model_filename = "Hybrid_RF_DT_Stacking_best_model.joblib"
        joblib.dump(stacking_clf, model_filename)
        print(f"Saved {model_filename}")

```

```

# Evaluate on TRAIN
m_train, r_train = evaluate_with_confusion(
    model=stacking_clf,
    X=X_train,
    y=y_train,
    dataset_name="Train_70",
    hybrid_name=hybrid_name
)

# Evaluate on TEST
m_test, r_test = evaluate_with_confusion(
    model=stacking_clf,
    X=X_test,
    y=y_test,
    dataset_name="Test_30",
    hybrid_name=hybrid_name
)

all_metrics.extend([m_train, m_test])
all_reports[f"{hybrid_name}_TRAIN"] = r_train
all_reports[f"{hybrid_name}_TEST"] = r_test

# -----
# 8) Save metrics and reports
# -----
metrics_df = pd.DataFrame(all_metrics)
scores_filename = "Stacking_Hybrids_scores_70_30.csv"
metrics_df.to_csv(scores_filename, index=False)

reports_filename = "Stacking_Hybrids_classification_reports.json"
with open(reports_filename, "w") as f:
    json.dump(all_reports, f, indent=4)

print("\nFiles saved:")
print("  -", scores_filename)
print("  -", reports_filename)
print("  - Individual confusion matrices CSVs and PNGs for each hybrid (Train/
↳Test).")
print("  - Hybrid_RF_DT_Stacking_best_model.joblib (main hybrid model)")

# -----
# 9) Optional: download main results and main hybrid model
# -----
files.download(scores_filename)
files.download(reports_filename)
files.download("Hybrid_RF_DT_Stacking_best_model.joblib")

```

```

# -----
# 10) Optional: download all confusion matrix images
# -----
for fname in os.listdir():
    if fname.endswith(".png") and "ConfusionMatrix" in fname:
        files.download(fname)

```

Please upload: CKD_cleaned_dataset_EN.csv

<IPython.core.display.HTML object>

Saving CKD_cleaned_dataset_EN.csv to CKD_cleaned_dataset_EN (8).csv

Full cleaned dataset shape: (15051, 21)

Features shape: (15051, 20) Target shape: (15051,)

Stratified split (70/30):

X_train: (10535, 20) y_train: (10535,)

X_test : (4516, 20) y_test : (4516,)

Please upload GA-optimized base models:

- DT_GA_best_model.joblib
- RF_GA_best_model.joblib
- LR_GA_best_model.joblib
- SVM_GA_best_model.joblib

<IPython.core.display.HTML object>

Saving SVM_GA_best_model.joblib to SVM_GA_best_model (8).joblib

Saving LR_GA_best_model.joblib to LR_GA_best_model (8).joblib

Saving RF_GA_best_model.joblib to RF_GA_best_model (8).joblib

Saving DT_GA_best_model.joblib to DT_GA_best_model (8).joblib

GA-optimized base models loaded successfully:

- DT_GA_best_model (8).joblib
- RF_GA_best_model (8).joblib
- LR_GA_best_model (8).joblib
- SVM_GA_best_model (8).joblib

=====

Training Hybrid_RF_DT_Stacking

=====

Saved Hybrid_RF_DT_Stacking_best_model.joblib

Performance on Train_70 for Hybrid_RF_DT_Stacking:

Accuracy : 0.9997152349311818

Precision: 0.9998056743101438

Recall : 0.9996114241305615

F1-score : 0.9997085397843194

ROC-AUC : 0.9999997836437252

Classification report (4 decimals):

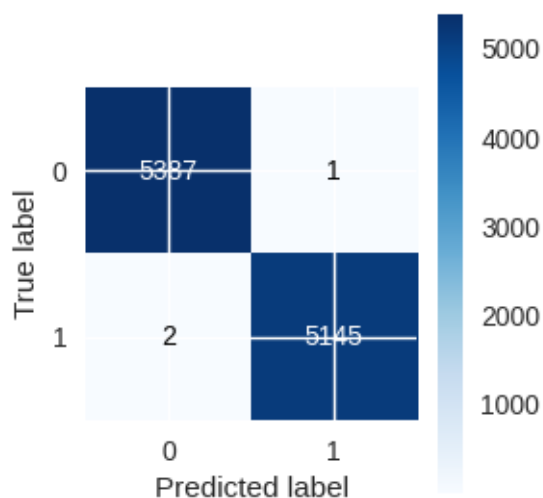
	precision	recall	f1-score	support
0	0.9996	0.9998	0.9997	5388
1	0.9998	0.9996	0.9997	5147
accuracy			0.9997	10535
macro avg	0.9997	0.9997	0.9997	10535
weighted avg	0.9997	0.9997	0.9997	10535

Confusion matrix CSV saved to:

Hybrid_RF_DT_Stacking_ConfusionMatrix_Train_70.csv

Confusion matrix PNG saved to:

Hybrid_RF_DT_Stacking_ConfusionMatrix_Train_70.png



Performance on Test_30 for Hybrid_RF_DT_Stacking:

Accuracy : 0.9991142604074402

Precision: 0.99909338168631

Recall : 0.99909338168631

F1-score : 0.99909338168631

ROC-AUC : 0.9999984300981581

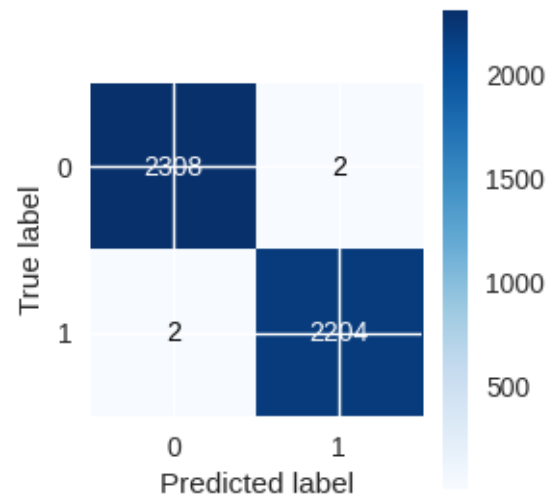
Classification report (4 decimals):

	precision	recall	f1-score	support
0	0.9991	0.9991	0.9991	2310
1	0.9991	0.9991	0.9991	2206

accuracy			0.9991	4516
macro avg	0.9991	0.9991	0.9991	4516
weighted avg	0.9991	0.9991	0.9991	4516

Confusion matrix CSV saved to: Hybrid_RF_DT_Stacking_ConfusionMatrix_Test_30.csv

Confusion matrix PNG saved to: Hybrid_RF_DT_Stacking_ConfusionMatrix_Test_30.png



=====
Training Hybrid_RF_SVM_Stacking
=====

Performance on Train_70 for Hybrid_RF_SVM_Stacking:

Accuracy : 0.9994304698623635

Precision: 1.0

Recall : 0.9988342723916844

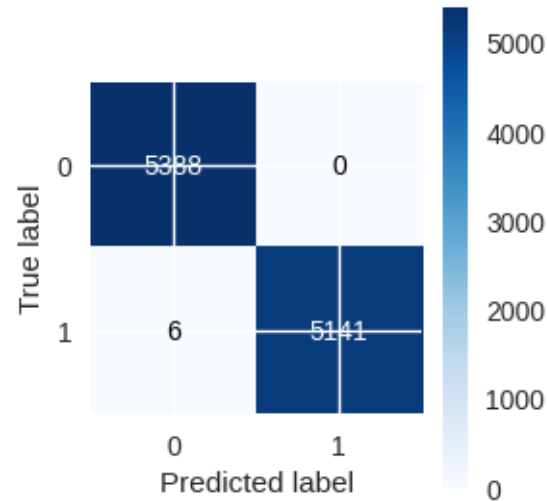
F1-score : 0.9994167962674961

ROC-AUC : 0.9999996754655879

Classification report (4 decimals):

	precision	recall	f1-score	support
0	0.9989	1.0000	0.9994	5388
1	1.0000	0.9988	0.9994	5147
accuracy			0.9994	10535
macro avg	0.9994	0.9994	0.9994	10535
weighted avg	0.9994	0.9994	0.9994	10535

Confusion matrix CSV saved to:
Hybrid_RF_SVM_Stacking_ConfusionMatrix_Train_70.csv
Confusion matrix PNG saved to:
Hybrid_RF_SVM_Stacking_ConfusionMatrix_Train_70.png

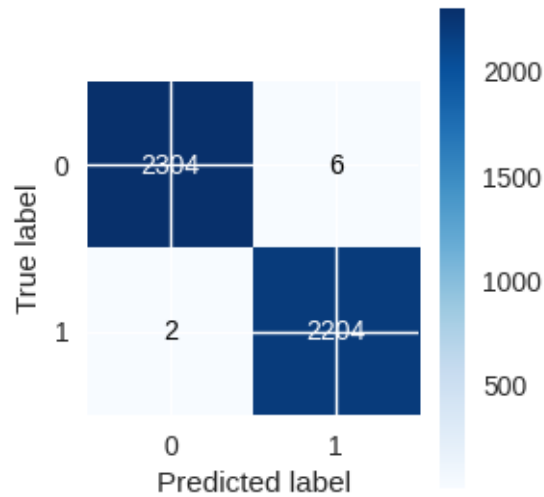


Performance on Test_30 for Hybrid_RF_SVM_Stacking:
Accuracy : 0.9982285208148804
Precision: 0.9972850678733032
Recall : 0.99909338168631
F1-score : 0.9981884057971014
ROC-AUC : 0.999996663958586

Classification report (4 decimals):

	precision	recall	f1-score	support
0	0.9991	0.9974	0.9983	2310
1	0.9973	0.9991	0.9982	2206
accuracy			0.9982	4516
macro avg	0.9982	0.9982	0.9982	4516
weighted avg	0.9982	0.9982	0.9982	4516

Confusion matrix CSV saved to:
Hybrid_RF_SVM_Stacking_ConfusionMatrix_Test_30.csv
Confusion matrix PNG saved to:
Hybrid_RF_SVM_Stacking_ConfusionMatrix_Test_30.png



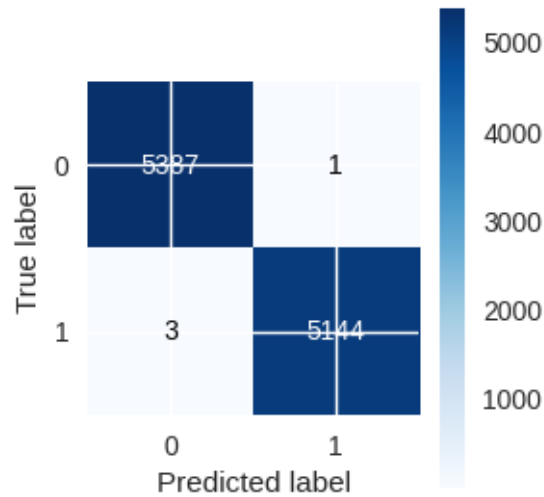
```
=====
Training Hybrid_RF_LR_Stacking
=====
```

```
Performance on Train_70 for Hybrid_RF_LR_Stacking:
Accuracy : 0.9996203132415757
Precision: 0.9998056365403304
Recall    : 0.9994171361958423
F1-score  : 0.9996113486202876
ROC-AUC   : 0.9999998918218626
```

```
Classification report (4 decimals):
```

	precision	recall	f1-score	support
0	0.9994	0.9998	0.9996	5388
1	0.9998	0.9994	0.9996	5147
accuracy			0.9996	10535
macro avg	0.9996	0.9996	0.9996	10535
weighted avg	0.9996	0.9996	0.9996	10535

```
Confusion matrix CSV saved to:
Hybrid_RF_LR_Stacking_ConfusionMatrix_Train_70.csv
Confusion matrix PNG saved to:
Hybrid_RF_LR_Stacking_ConfusionMatrix_Train_70.png
```



Performance on Test_30 for Hybrid_RF_LR_Stacking:

Accuracy : 0.9988928255093003

Precision: 0.9986406887177164

Recall : 0.99909338168631

F1-score : 0.9988669839111716

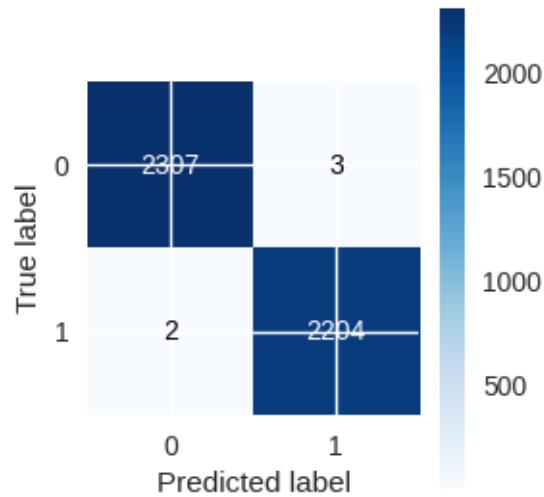
ROC-AUC : 0.9999952902944743

Classification report (4 decimals):

	precision	recall	f1-score	support
0	0.9991	0.9987	0.9989	2310
1	0.9986	0.9991	0.9989	2206
accuracy			0.9989	4516
macro avg	0.9989	0.9989	0.9989	4516
weighted avg	0.9989	0.9989	0.9989	4516

Confusion matrix CSV saved to: Hybrid_RF_LR_Stacking_ConfusionMatrix_Test_30.csv

Confusion matrix PNG saved to: Hybrid_RF_LR_Stacking_ConfusionMatrix_Test_30.png



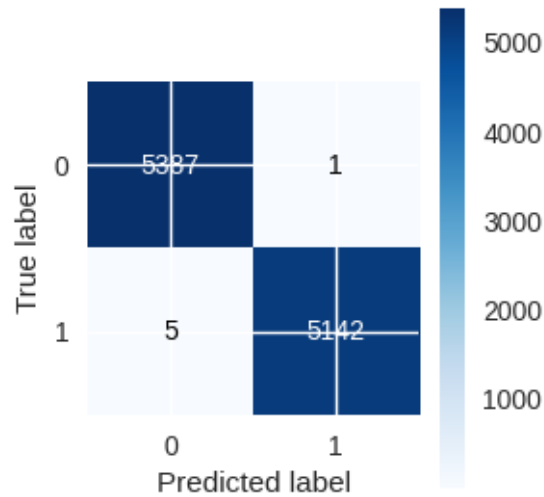
```
=====
Training Hybrid_SVM_LR_Stacking
=====
```

```
Performance on Train_70 for Hybrid_SVM_LR_Stacking:
Accuracy : 0.9994304698623635
Precision: 0.9998055609566401
Recall   : 0.9990285603264037
F1-score : 0.9994169096209913
ROC-AUC  : 0.9999992788124176
```

```
Classification report (4 decimals):
```

	precision	recall	f1-score	support
0	0.9991	0.9998	0.9994	5388
1	0.9998	0.9990	0.9994	5147
accuracy			0.9994	10535
macro avg	0.9994	0.9994	0.9994	10535
weighted avg	0.9994	0.9994	0.9994	10535

```
Confusion matrix CSV saved to:
Hybrid_SVM_LR_Stacking_ConfusionMatrix_Train_70.csv
Confusion matrix PNG saved to:
Hybrid_SVM_LR_Stacking_ConfusionMatrix_Train_70.png
```



Performance on Test_30 for Hybrid_SVM_LR_Stacking:

Accuracy : 0.9980070859167405

Precision: 0.9968340117593849

Recall : 0.99909338168631

F1-score : 0.9979624179307222

ROC-AUC : 0.9999271958020824

Classification report (4 decimals):

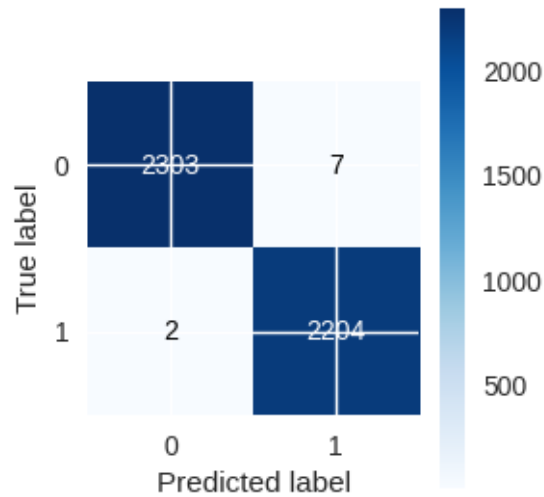
	precision	recall	f1-score	support
0	0.9991	0.9970	0.9980	2310
1	0.9968	0.9991	0.9980	2206
accuracy			0.9980	4516
macro avg	0.9980	0.9980	0.9980	4516
weighted avg	0.9980	0.9980	0.9980	4516

Confusion matrix CSV saved to:

Hybrid_SVM_LR_Stacking_ConfusionMatrix_Test_30.csv

Confusion matrix PNG saved to:

Hybrid_SVM_LR_Stacking_ConfusionMatrix_Test_30.png



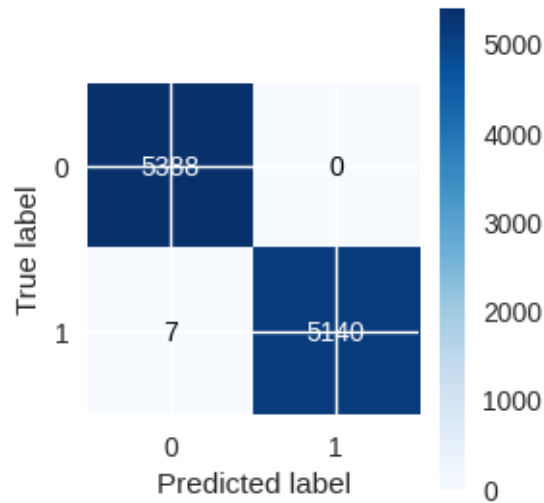
```
=====
Training Hybrid_DT_SVM_Stacking
=====
```

```
Performance on Train_70 for Hybrid_DT_SVM_Stacking:
Accuracy : 0.9993355481727575
Precision: 1.0
Recall   : 0.9986399844569652
F1-score : 0.9993195295032565
ROC-AUC  : 0.9999996033468296
```

```
Classification report (4 decimals):
```

	precision	recall	f1-score	support
0	0.9987	1.0000	0.9994	5388
1	1.0000	0.9986	0.9993	5147
accuracy			0.9993	10535
macro avg	0.9994	0.9993	0.9993	10535
weighted avg	0.9993	0.9993	0.9993	10535

```
Confusion matrix CSV saved to:
Hybrid_DT_SVM_Stacking_ConfusionMatrix_Train_70.csv
Confusion matrix PNG saved to:
Hybrid_DT_SVM_Stacking_ConfusionMatrix_Train_70.png
```



Performance on Test_30 for Hybrid_DT_SVM_Stacking:

Accuracy : 0.9984499557130204

Precision: 0.9977365323675872

Recall : 0.99909338168631

F1-score : 0.9984144960362401

ROC-AUC : 0.9999742928573391

Classification report (4 decimals):

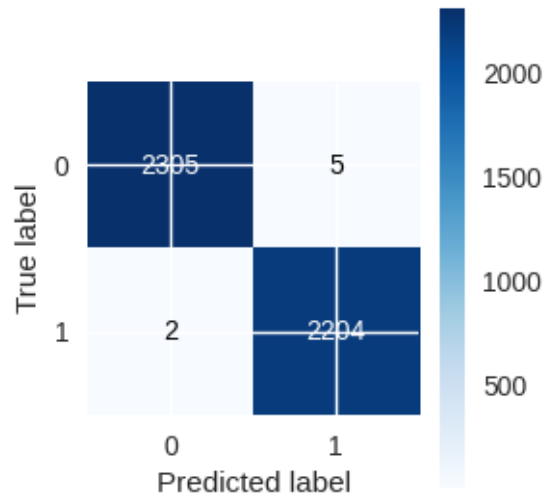
	precision	recall	f1-score	support
0	0.9991	0.9978	0.9985	2310
1	0.9977	0.9991	0.9984	2206
accuracy			0.9984	4516
macro avg	0.9984	0.9985	0.9984	4516
weighted avg	0.9985	0.9984	0.9984	4516

Confusion matrix CSV saved to:

Hybrid_DT_SVM_Stacking_ConfusionMatrix_Test_30.csv

Confusion matrix PNG saved to:

Hybrid_DT_SVM_Stacking_ConfusionMatrix_Test_30.png



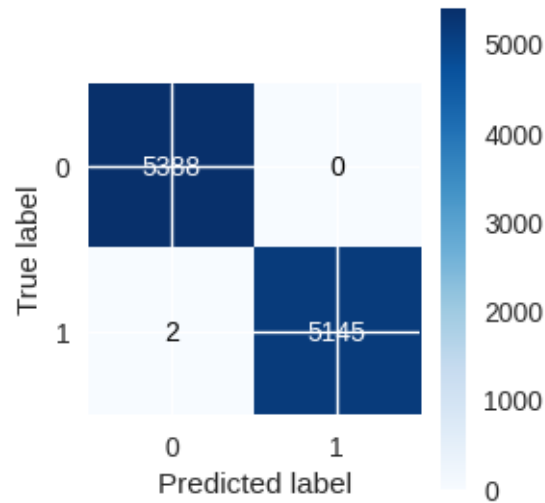
```
=====
Training Hybrid_DT_LR_Stacking
=====
```

```
Performance on Train_70 for Hybrid_DT_LR_Stacking:
Accuracy : 0.9998101566207879
Precision: 1.0
Recall   : 0.9996114241305615
F1-score : 0.9998056743101438
ROC-AUC  : 0.9999997836437252
```

```
Classification report (4 decimals):
```

	precision	recall	f1-score	support
0	0.9996	1.0000	0.9998	5388
1	1.0000	0.9996	0.9998	5147
accuracy			0.9998	10535
macro avg	0.9998	0.9998	0.9998	10535
weighted avg	0.9998	0.9998	0.9998	10535

```
Confusion matrix CSV saved to:
Hybrid_DT_LR_Stacking_ConfusionMatrix_Train_70.csv
Confusion matrix PNG saved to:
Hybrid_DT_LR_Stacking_ConfusionMatrix_Train_70.png
```



Performance on Test_30 for Hybrid_DT_LR_Stacking:

Accuracy : 0.9993356953055802

Precision: 0.999546485260771

Recall : 0.99909338168631

F1-score : 0.9993198821128996

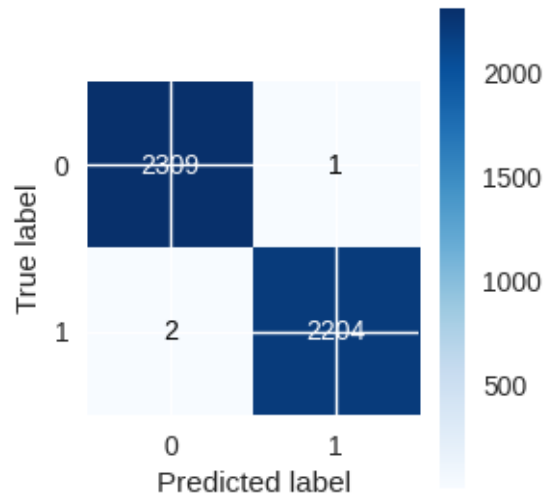
ROC-AUC : 0.9999923467285209

Classification report (4 decimals):

	precision	recall	f1-score	support
0	0.9991	0.9996	0.9994	2310
1	0.9995	0.9991	0.9993	2206
accuracy			0.9993	4516
macro avg	0.9993	0.9993	0.9993	4516
weighted avg	0.9993	0.9993	0.9993	4516

Confusion matrix CSV saved to: Hybrid_DT_LR_Stacking_ConfusionMatrix_Test_30.csv

Confusion matrix PNG saved to: Hybrid_DT_LR_Stacking_ConfusionMatrix_Test_30.png



Files saved:

- Stacking_Hybrids_scores_70_30.csv
- Stacking_Hybrids_classification_reports.json
- Individual confusion matrices CSVs and PNGs for each hybrid (Train/Test).
- Hybrid_RF_DT_Stacking_best_model.joblib (main hybrid model)

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>