

jyfh12wcy

December 4, 2025

1 07 - Final Results, CKD Distribution, Model Comparison, Overfitting, and ROC

```
[17]: # =====  
# 07 - Final Results, CKD Distribution,  
#      Model Comparison, Overfitting, and ROC  
# =====  
  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
  
from google.colab import files  
from sklearn.metrics import roc_curve, auc  
from sklearn.model_selection import train_test_split  
import joblib  
import os  
  
plt.style.use("seaborn-v0_8")  
  
# -----  
# 1) Upload required CSV result files  
# -----  
print("Please upload the following files:")  
print("  - CKD_cleaned_dataset_EN.csv")  
print("  - DT_GA_best_scores_70_30.csv")  
print("  - RF_GA_best_scores_70_30.csv")  
print("  - LR_GA_best_scores_70_30.csv")  
print("  - SVM_GA_best_scores_70_30.csv")  
print("  - Stacking_Hybrids_scores_70_30.csv")  
  
uploaded = files.upload()  
  
df_ckd = pd.read_csv([fn for fn in uploaded if fn.  
    ↳startswith("CKD_cleaned_dataset_EN")][0])  
dt_scores = pd.read_csv([fn for fn in uploaded if fn.  
    ↳startswith("DT_GA_best_scores_70_30")][0])
```

```

rf_scores      = pd.read_csv([fn for fn in uploaded if fn.
    ↳startswith("RF_GA_best_scores_70_30")][0])
lr_scores      = pd.read_csv([fn for fn in uploaded if fn.
    ↳startswith("LR_GA_best_scores_70_30")][0])
svm_scores     = pd.read_csv([fn for fn in uploaded if fn.
    ↳startswith("SVM_GA_best_scores_70_30")][0])
stacking_scores = pd.read_csv([fn for fn in uploaded if fn.
    ↳startswith("Stacking_Hybrids_scores_70_30")][0])

# -----
# 2) CKD vs Non-CKD distribution
# -----
target_col = "Chronic_Kidney_Disease"
value_counts = df_ckd[target_col].value_counts().sort_index()

labels = ["Non-CKD", "CKD"]
counts = [value_counts.get(0, 0), value_counts.get(1, 0)]

plt.figure(figsize=(4, 4))
plt.bar(labels, counts, color=["#66B3FF", "#FF6666"], width=0.4)
plt.ylabel("Count")
plt.xlabel("Class")
plt.tight_layout()
plt.savefig("CKD_vs_NonCKD_Distribution.png", dpi=300, bbox_inches="tight")
plt.show()

# -----
# 3) GA models: Test table (sorted by AUC)
# -----
def ga_row(scores_df, name):
    row = scores_df[scores_df["dataset"].str.contains("Test")].iloc[0]
    return {
        "Model": name,
        "Accuracy": row["accuracy"],
        "Precision": row["precision"],
        "Recall": row["recall"],
        "F1": row["f1"],
        "AUC": row["roc_auc"],
    }

ga_rows = [
    ga_row(rf_scores, "RF_GA"),
    ga_row(svm_scores, "SVM_GA"),
    ga_row(lr_scores, "LR_GA"),
    ga_row(dt_scores, "DT_GA"),
]

```

```

ga_df = pd.DataFrame(ga_rows).sort_values("AUC", ascending=False).
    ↪reset_index(drop=True)
display(ga_df)
ga_df.to_csv("Final_GA_Models_Comparison_Test_Only.csv", index=False)

# -----
# 4) Hybrid models: Test table (sorted by AUC)
# -----
hy_test = stacking_scores[stacking_scores["model"].str.contains("Test_30")].
    ↪copy()
hy_test["Model"] = hy_test["model"].str.replace("_Test_30", "", regex=False)

hybrid_df = hy_test[["Model", "accuracy", "precision", "recall", "f1",
    ↪"roc_auc"]].rename(
    columns={
        "accuracy": "Accuracy",
        "precision": "Precision",
        "recall": "Recall",
        "f1": "F1",
        "roc_auc": "AUC",
    }
).sort_values("AUC", ascending=False).reset_index(drop=True)

display(hybrid_df)
hybrid_df.to_csv("Final_Hybrid_Stacking_Comparison_Test_Only.csv", index=False)

# -----
# 5) Helper: grouped bar chart (no title on saved image)
# -----
def plot_perf(df, filename):
    colors = {
        "Accuracy": "#FFE5CC",
        "Precision": "#FFCC99",
        "Recall": "#FFB266",
        "F1-score": "#FF9933",
        "AUC": "#00CC66",
    }
    metric_cols = {
        "Accuracy": "Accuracy",
        "Precision": "Precision",
        "Recall": "Recall",
        "F1-score": "F1",
        "AUC": "AUC",
    }
    metrics = ["Accuracy", "Precision", "Recall", "F1-score", "AUC"]
    models = df["Model"].tolist()
    x = np.arange(len(models))

```

```

bar_w = 0.12

plt.figure(figsize=(10, 5))
for i, m in enumerate(metrics):
    vals = df[metric_cols[m]].values
    plt.bar(
        x + i * bar_w,
        vals,
        width=bar_w,
        label=m,
        color=colors[m],
        edgecolor="black",
        linewidth=0.7,
    )

plt.xticks(x + (len(metrics) - 1) * bar_w / 2, models, rotation=10)
plt.ylim(0.97, 1.01)
plt.ylabel("Score")
plt.legend(
    loc="upper center",
    bbox_to_anchor=(0.5, 1.15),
    ncol=len(metrics),
    frameon=False,
)
plt.grid(axis="y", linestyle="--", alpha=0.5)
plt.tight_layout()
plt.savefig(filename, dpi=300, bbox_inches="tight")
plt.show()

# -----
# 6) Plot GA and Hybrid performance
# -----
plot_perf(ga_df, "GA_Models_Performance_Test_30.png")
plot_perf(hybrid_df, "Hybrid_Models_Performance_Test_30.png")

# -----
# 7) Overfitting GA (Train/Test accuracy, sorted by AUC)
# -----
def ga_overfit(scores_df, name):
    tr = scores_df[scores_df["dataset"].str.contains("Train")].iloc[0]
    te = scores_df[scores_df["dataset"].str.contains("Test")].iloc[0]
    return {
        "Model": name,
        "Accuracy_Train": tr["accuracy"],
        "Accuracy_Test": te["accuracy"],
        "AUC": te["roc_auc"],
    }

```

```

ga_over = pd.DataFrame([
    ga_overfit(rf_scores, "RF_GA"),
    ga_overfit(svm_scores, "SVM_GA"),
    ga_overfit(lr_scores, "LR_GA"),
    ga_overfit(dt_scores, "DT_GA"),
]).sort_values("AUC", ascending=False).reset_index(drop=True)

ga_over.to_csv("GA_Overfitting_View_Accuracy_Train_Test.csv", index=False)
display(ga_over)

plt.figure(figsize=(8, 5))
x = np.arange(len(ga_over["Model"]))
w = 0.35
plt.bar(x - w/2, ga_over["Accuracy_Train"], w, label="Train Accuracy",
        color="#99CCFF", edgecolor="black", linewidth=0.7)
plt.bar(x + w/2, ga_over["Accuracy_Test"], w, label="Test Accuracy",
        color="#FF9999", edgecolor="black", linewidth=0.7)
plt.xticks(x, ga_over["Model"], rotation=10)
plt.ylabel("Accuracy")
plt.ylim(0.97, 1.01)
plt.legend(loc="upper center", bbox_to_anchor=(0.5, 1.12), ncol=2,
          frameon=False)
plt.grid(axis="y", linestyle="--", alpha=0.5)
plt.tight_layout()
plt.savefig("GA_Overfitting_Accuracy_Train_Test.png", dpi=300,
          bbox_inches="tight")
plt.show()

# -----
# 8) Overfitting Hybrid (Train/Test accuracy, sorted by AUC)
# -----
hy_tr = stacking_scores[stacking_scores["model"].str.contains("Train_70")].
    copy()
hy_tr["Model"] = hy_tr["model"].str.replace("_Train_70", "", regex=False)

hy_te = stacking_scores[stacking_scores["model"].str.contains("Test_30")].copy()
hy_te["Model"] = hy_te["model"].str.replace("_Test_30", "", regex=False)

hy_over = hy_tr.merge(
    hy_te[["Model", "accuracy", "roc_auc"]].rename(
        columns={"accuracy": "Accuracy_Test", "roc_auc": "AUC"}
    ),
    on="Model",
    how="inner",
)

```

```

hy_over = hy_over[["Model", "accuracy", "Accuracy_Test", "AUC"]].rename(
    columns={"accuracy": "Accuracy_Train"})
).sort_values("AUC", ascending=False).reset_index(drop=True)

hy_over.to_csv("Hybrid_Overfitting_View_Accuracy_Train_Test.csv", index=False)
display(hy_over)

plt.figure(figsize=(8, 5))
x = np.arange(len(hy_over["Model"]))
w = 0.35
plt.bar(x - w/2, hy_over["Accuracy_Train"], w, label="Train Accuracy",
        color="#99CCFF", edgecolor="black", linewidth=0.7)
plt.bar(x + w/2, hy_over["Accuracy_Test"], w, label="Test Accuracy",
        color="#FF9999", edgecolor="black", linewidth=0.7)
plt.xticks(x, hy_over["Model"], rotation=10)
plt.ylabel("Accuracy")
plt.ylim(0.97, 1.01)
plt.legend(loc="upper center", bbox_to_anchor=(0.5, 1.12), ncol=2,
           frameon=False)
plt.grid(axis="y", linestyle="--", alpha=0.5)
plt.tight_layout()
plt.savefig("Hybrid_Overfitting_Accuracy_Train_Test.png", dpi=300,
           bbox_inches="tight")
plt.show()

# -----
# 9) ROC for Hybrid_RF_DT_Stacking (Test only)
# -----
print("\nFor ROC, please upload:")
print("    - CKD_cleaned_dataset_EN.csv")
print("    - Hybrid_RF_DT_Stacking_best_model.joblib")

uploaded_roc = files.upload()

if any(fn.startswith("CKD_cleaned_dataset_EN") for fn in uploaded_roc) and \
    any(fn.startswith("Hybrid_RF_DT_Stacking_best_model") for fn in
       uploaded_roc):

    ckd_file = [fn for fn in uploaded_roc if fn.
                 startswith("CKD_cleaned_dataset_EN")][0]
    model_file = [fn for fn in uploaded_roc if fn.
                  startswith("Hybrid_RF_DT_Stacking_best_model")][0]

    df_full = pd.read_csv(ckd_file)
    X_full = df_full.drop(columns=[target_col])
    y_full = df_full[target_col]

```

```

X_train_f, X_test_f, y_train_f, y_test_f = train_test_split(
    X_full, y_full, test_size=0.30, stratify=y_full, random_state=42
)

hybrid_model = joblib.load(model_file)
y_test_proba = hybrid_model.predict_proba(X_test_f)[: , 1]

fpr_test, tpr_test, _ = roc_curve(y_test_f, y_test_proba)
roc_auc_test = auc(fpr_test, tpr_test)

plt.figure(figsize=(6, 6))
plt.plot(fpr_test, tpr_test,
         label=f"Test ROC (AUC = {roc_auc_test:.3f})",
         color="#009933", linewidth=2.0)
plt.plot([0, 1], [0, 1], "k--", label="Chance")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.tight_layout()
plt.savefig("Hybrid_RF_DT_Stacking_ROC_Test_30.png",
           dpi=300, bbox_inches="tight")
plt.show()
else:
    print("ROC was skipped (required files not uploaded or names mismatched).")

# -----
# 10) Download all PNG images
# -----
imgs = [f for f in os.listdir() if f.lower().endswith(".png")]
print("\nDownloading image files:")
for f in sorted(imgs):
    print(" -", f)
    files.download(f)

```

Please upload the following files:

- CKD_cleaned_dataset_EN.csv
- DT_GA_best_scores_70_30.csv
- RF_GA_best_scores_70_30.csv
- LR_GA_best_scores_70_30.csv
- SVM_GA_best_scores_70_30.csv
- Stacking_Hybrids_scores_70_30.csv

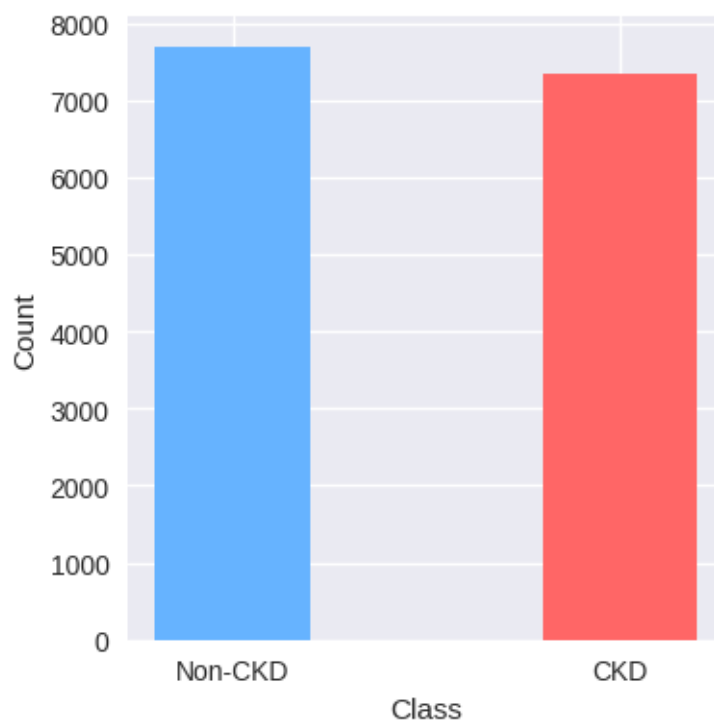
<IPython.core.display.HTML object>

Saving Stacking_Hybrids_scores_70_30.csv to Stacking_Hybrids_scores_70_30 (12).csv

Saving SVM_GA_best_scores_70_30.csv to SVM_GA_best_scores_70_30 (14).csv

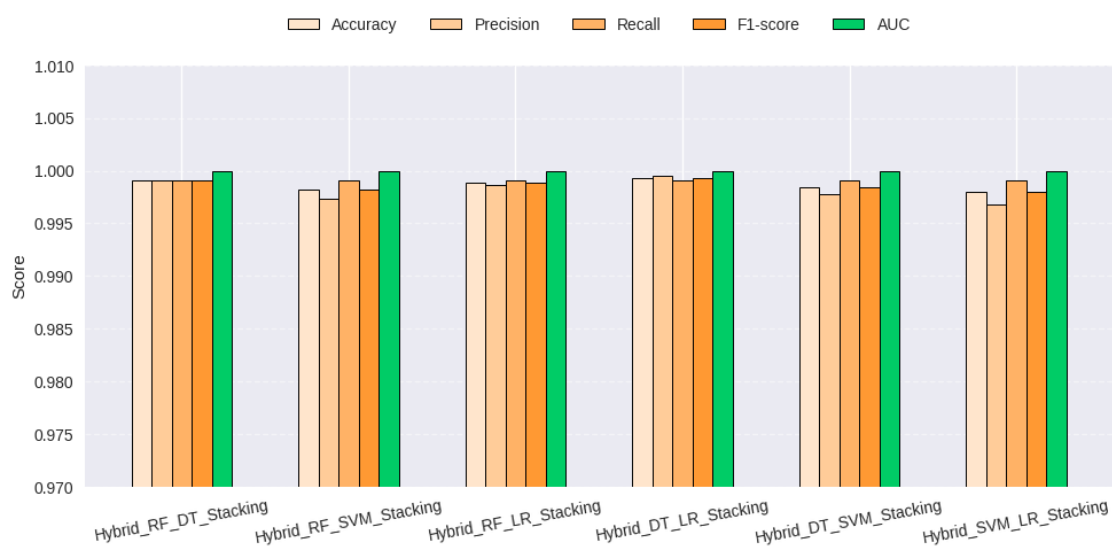
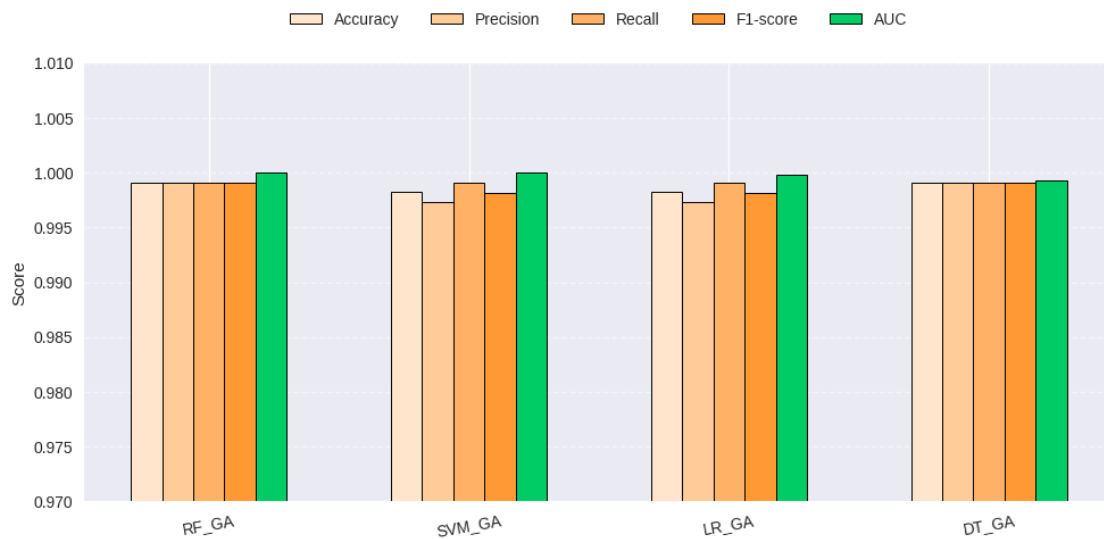
Saving LR_GA_best_scores_70_30.csv to LR_GA_best_scores_70_30 (14).csv

Saving RF_GA_best_scores_70_30.csv to RF_GA_best_scores_70_30 (14).csv
 Saving DT_GA_best_scores_70_30.csv to DT_GA_best_scores_70_30 (14).csv
 Saving CKD_cleaned_dataset_EN.csv to CKD_cleaned_dataset_EN (14).csv



	Model	Accuracy	Precision	Recall	F1	AUC
0	RF_GA	0.999114	0.999093	0.999093	0.999093	0.999998
1	SVM_GA	0.998229	0.997285	0.999093	0.998188	0.999971
2	LR_GA	0.998229	0.997285	0.999093	0.998188	0.999771
3	DT_GA	0.999114	0.999093	0.999093	0.999093	0.999330

	Model	Accuracy	Precision	Recall	F1	AUC
0	Hybrid_RF_DT_Stacking	0.999114	0.999093	0.999093	0.999093	0.999998
1	Hybrid_RF_SVM_Stacking	0.998229	0.997285	0.999093	0.998188	0.999997
2	Hybrid_RF_LR_Stacking	0.998893	0.998641	0.999093	0.998867	0.999995
3	Hybrid_DT_LR_Stacking	0.999336	0.999546	0.999093	0.999320	0.999992
4	Hybrid_DT_SVM_Stacking	0.998450	0.997737	0.999093	0.998414	0.999974
5	Hybrid_SVM_LR_Stacking	0.998007	0.996834	0.999093	0.997962	0.999927



	Model	Accuracy_Train	Accuracy_Test	AUC
0	RF_GA	0.999620	0.999114	0.999998
1	SVM_GA	0.999241	0.998229	0.999971
2	LR_GA	0.999525	0.998229	0.999771
3	DT_GA	0.999620	0.999114	0.999330



	Model	Accuracy_Train	Accuracy_Test	AUC
0	Hybrid_RF_DT_Stacking	0.999715	0.999114	0.999998
1	Hybrid_RF_SVM_Stacking	0.999430	0.998229	0.999997
2	Hybrid_RF_LR_Stacking	0.999620	0.998893	0.999995
3	Hybrid_DT_LR_Stacking	0.999810	0.999336	0.999992
4	Hybrid_DT_SVM_Stacking	0.999336	0.998450	0.999974
5	Hybrid_SVM_LR_Stacking	0.999430	0.998007	0.999927



For ROC, please upload:

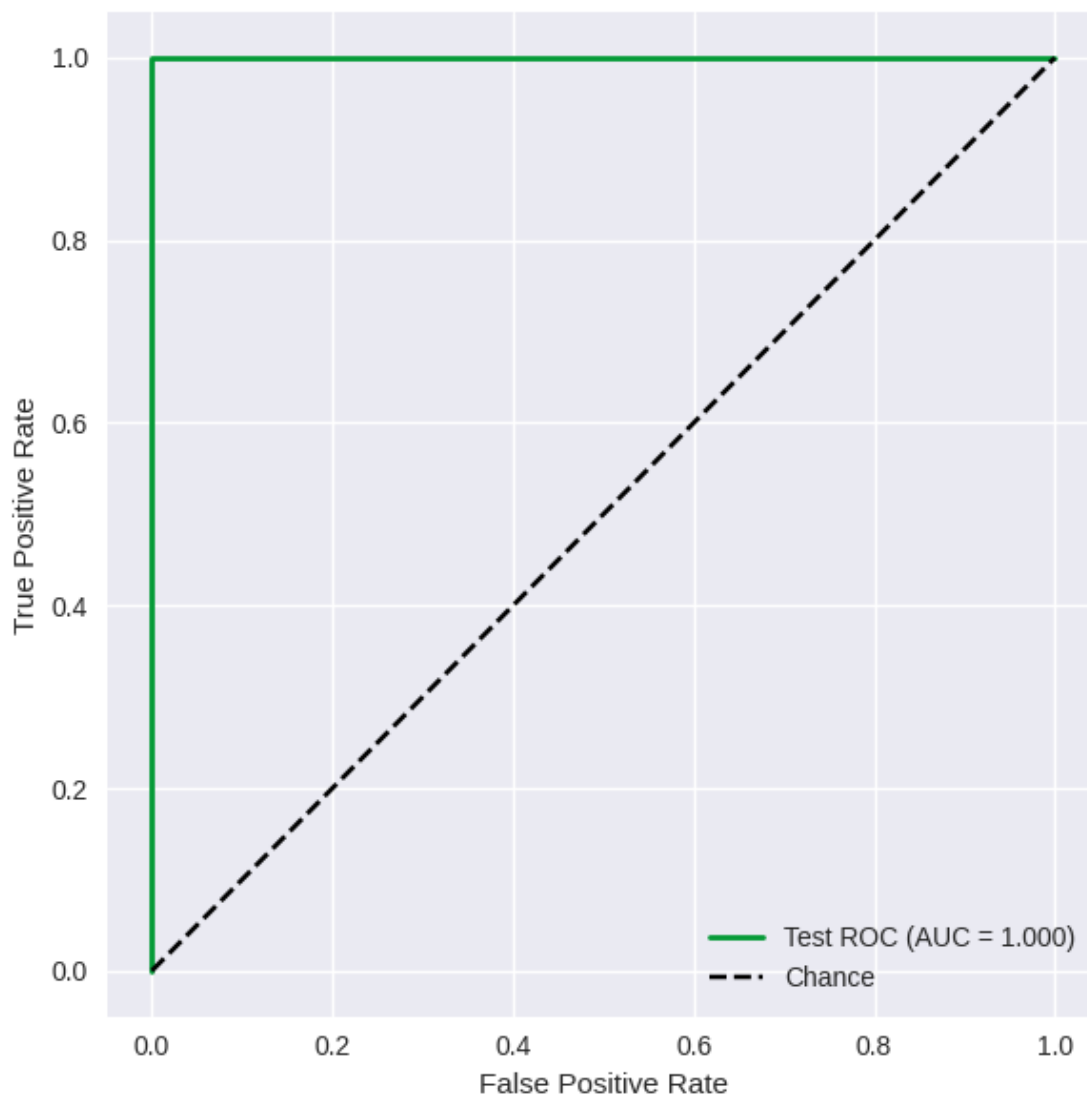
- CKD_cleaned_dataset_EN.csv
- Hybrid_RF_DT_Stacking_best_model.joblib

<IPython.core.display.HTML object>

Saving Hybrid_RF_DT_Stacking_best_model.joblib to

Hybrid_RF_DT_Stacking_best_model (2).joblib

Saving CKD_cleaned_dataset_EN.csv to CKD_cleaned_dataset_EN (15).csv



Downloading image files:

- CKD_vs_NonCKD_Distribution.png

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

- GA_Models_Performance_Test_30.png

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

- GA_Overfitting_Accuracy_Train_Test.png

<IPython.core.display.Javascript object>

```
<IPython.core.display.Javascript object>
- Hybrid_Models_Performance_Test_30.png
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
- Hybrid_Overfitting_Accuracy_Train_Test.png
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
- Hybrid_RF_DT_Stacking_ROC_Test_30.png
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
```